Velocity occupancy space: autonomous navigation in an uncertain, dynamic environment

Rachael Bis*

Department of Mechanical Engineering,
University of Michigan,
2277 G.G. Brown Building,
2350 Hayward Street,
Ann Arbor, MI 48109-2125, USA
Fax: (734) 647-3170
E-mail: rachbis@umich.edu
*Corresponding author

Huei Peng

G036 Lay Automotive Laboratory,
Department of Mechanical Engineering,
University of Michigan,
Ann Arbor, MI 48109-2133, USA
Fax: (734) 764-4256
E-mail: hpeng@umich.edu

A. Galip Ulsoy

Department of Mechanical Engineering,
University of Michigan,
2266 G.G. Brown Building,
2350 Hayward Street,
Ann Arbor, MI 48109-2125, USA
Fax: (734) 647-3170
E-mail: ulsoy@umich.edu

Abstract: In order to autonomously navigate in an unknown environment, a robotic vehicle must be able to sense obstacles, determine their velocities, and select a collision-free path that will lead quickly to a goal. However, the perceived location and motion of the obstacles will be uncertain due to the limited accuracy of the robot's sensors. Thus, it is necessary to develop a system that can avoid moving obstacles using uncertain sensor data. The method proposed here is based on an occupancy grid – which has been used to avoid stationary obstacles in an uncertain environment – in conjunction with velocity obstacles – which allow a robot to avoid well-known moving obstacles. The combination of these techniques leads to Velocity Occupancy Space (VOS): a search space which allows the robot to avoid moving obstacles and navigate efficiently to a goal using uncertain sensor data. The proposed method is validated by numerous simulation trials.

Keywords: velocity occupancy space; velocity obstacle; autonomous navigation; moving obstacle avoidance; uncertain sensor data; occupancy grid; robot motion planning.

Reference to this paper should be made as follows: Bis, R., Peng, H. and Ulsoy, A.G. (2012) 'Velocity occupancy space: autonomous navigation in an uncertain, dynamic environment', *Int. J. Vehicle Autonomous Systems*, Vol. 10, Nos. 1/2, pp.41–66.

Biographical notes: Rachael Bis received her BS (2005) and MS (2006) Degrees in Mechanical Engineering at Virginia Polytechnic Institute and State University. She is currently a PhD candidate at the University of Michigan, Ann Arbor where she is involved in research with the university's Ground Robotics Reliability Centre. Her research interests include autonomous robotics for military, civilian and agriculture applications. She is a student member of the ASME Dynamic Systems and Controls Division a recipient of a NSF Graduate Research Fellowship.

Huei Peng received the PhD Degree in Mechanical Engineering from the University of California, Berkeley, in 1992. He is currently a Professor with the Department of Mechanical Engineering, University of Michigan, Ann Arbor. His research interests include adaptive control and optimal control, with an emphasis on their application to vehicular and transportation systems. He has been an active member of SAE and the ASME Dynamic System and Control Division.

Galip Ulsoy is the C.D. Mote, Jr. Distinguished University Professor of Mechanical Engineering and the W.C. Ford Professor of Manufacturing at University of Michigan, Ann Arbor. He received the PhD from University of California at Berkeley (1979), the MS from Cornell University (1975), and the BS from Swarthmore College (1973). His research interests are in the dynamics and control of mechanical systems. He has received numerous awards, including the 2008 Rufus T. Oldenburger Medal from ASME. He is a member of the National Academy of Engineering and is a Fellow of ASME, SME and IFAC.

1 Introduction

The ability of a robotic vehicle to safely and autonomously navigate among stationary and moving obstacles (including pedestrians and vehicles) in an unknown setting is essential to the vehicle's operation in most environments. Until autonomous vehicles can be assured of not colliding with humans (and other moving and stationary obstacles) as they navigate towards their destination, they cannot be widely and generally employed. Therefore, a system is needed which can control the movement of an autonomous robot and allow it to avoid moving obstacles as it performs a task (e.g., reaching a target).

A moving obstacle avoidance system must be able to work within the practical constraints of a realistic robotic vehicle that could conceivably be developed for commercial applications. One of the most restrictive of these constraints is the cost limitations on the sensor(s) and on-board computer that can be used. This will affect both the quality and the quantity of the data available to the robot and require algorithms that can both compensate for such shortcomings and are computationally inexpensive.

Consequently, this paper proposes a new algorithm for safe autonomous vehicle operation in an unknown environment in the presence of moving obstacles using uncertain sensor data.

1.1 Review of related literature

The field of robot navigation (particularly the problems of obstacle avoidance and path planning) has been well researched over the years. Various types of stationary obstacle avoidance algorithms have been developed and successfully implemented for a variety of applications, a number of examples of which are described by Latombe (1991). Global path planners, such as those reviewed by Siegwart and Nourbakhsh (2004), allow a robot to navigate along a predefined path in a known environment. These types of planners assume that the robot's environment is either stationary or that the planner has complete knowledge about the movement of all obstacles. As neither type of environment is very common in the real world, local obstacle avoidance algorithms, such as those in Thrun et al. (2005), or path adaption algorithms, such as path-velocity decomposition (Kant and Zucker, 1986) or gradient-based path re-planning (Konolige, 2000), are often integrated into global planners to allow for some degree of reactive behaviour.

The method that we have developed, VOS, falls in between a high level global planner and a merely reactive obstacle avoidance system. Like a reactive system, VOS allows the robot to choose velocities that will avoid moving obstacles about which it has no prior knowledge. However, VOS also incorporates rudimentary path planning that takes the location of the robot's goal into consideration when selecting the velocities that will allow the robot to avoid any obstacles. Finally, VOS performs significantly longer-term planning than most low-level planners: the velocities that it selects will usually allow the robot to avoid all observable obstacles for as long as the obstacles maintain a near constant velocity, instead of just for the next time step.

The VOS method incorporates the use of the *certainty grid* concept to avoid moving obstacles using uncertain sensor data, as introduced by Moravec and Elfes (1985), together with the *velocity obstacle* method of moving obstacle avoidance developed by Fiorini and Shiller (1998). Certainty grids allow a robot to navigate in a stationary, cluttered environment using only data from a realistically uncertain range sensor (i.e., laser range finder, sonar). The certainty grid deals reliably with the high error rate often found with lower cost range sensors by utilising the frequent updates of data that are produced by these types of sensors. Specifically, this method works by decomposing the robot's environment into discrete cells and assigning each cell a relative probability of occupancy based on the number of times that a sensor has detected an obstacle in that cell. However, certainty and occupancy grids, and their many variations, are not able to actively avoid moving obstacles or safely navigate in an environment with moving obstacles due to their inability to track obstacles and the certainty grid's inherent non-time-varying representation.

Conversely, the velocity obstacle concept, developed by Fiorini and Shiller (1998) and expanded upon by Shiller et al. (2001) and Large et al. (2005) allows the robot to avoid collisions with moving obstacles in a time-varying environment using a first-order method of motion planning. This method uses information about the obstacles' locations and velocities to compute 'velocity obstacles', which are a set of robot velocities which will lead to a collision between the robot and an obstacle at some future point in time. As long as the robot does not move with a velocity that falls within a velocity

obstacle, it will not collide with that obstacle. However, while the velocity obstacle concept allows for the avoidance of moving obstacles, in its original form it requires complete knowledge of the obstacles' dimensions, locations and velocities – information that is typically not available from a low cost, range finding sensor.

The concept of combining an occupancy grid representation of an uncertain environment with velocity obstacles that was initially explored with VOS in Bis et al. (2009) and is being further developed in this paper was recently also developed by Fulgenzi et al. (2007) and Fulgenzi (2009). The authors of these papers use a Bayesian Occupancy Filter (BOF) to represent obstacles and estimate their velocities in an unknown and uncertain environment and they use Probabilistic Velocity Obstacles (PVOs) to find safe robot velocities. While they address the same problem that we are attempting to solve using VOS, namely forming velocity obstacles from uncertain sensor data, the specific methods that they use are significantly different from the techniques that we are exploring, though both approaches have their own strengths. The BOF allows for more accurate obstacle tracking and velocity estimation but it also greatly restricts the recognisable obstacle velocities and/or is significantly more computationally complex than the VOS method developed in Bis et al. (2009). More discussion on the differences between our method and BOFs and other, more common, probabilistic methods will be provided in Section 2.1 of this paper.

1.2 Purpose and scope

The purpose of this paper is to present a new method for safe autonomous vehicle navigation in an unknown environment in the presence of moving obstacles using uncertain sensor data. This method, termed VOS, combines the sensor error and uncertainty representation of certainty grid occupancy space with the velocity obstacle representation of moving obstacles. In addition, VOS allows for active velocity selection which will enable the robot to navigate efficiently and autonomously, as well as perform obstacle avoidance, while moving toward the desired destination.

Our goal is to design a system that is inexpensive yet highly versatile; suitable for both military and civilian applications in structured and unstructured environments. As such, we do not make any assumptions about the types of obstacles that we are likely to encounter and assume that we have almost no prior knowledge of the environment (we do not have a map, nor do we attempt to build one). There are two assumptions that we made about the environment and the vehicle. First, we assume that the local environment is relatively flat so that we can approximate our environment as two dimensional. Second, this method assumes that the vehicle is holonomic and moves with a bounded linear velocity but that it can instantaneously change velocity (within a bounded range). These assumptions can be easily adapted to allow VOS to operate on a non-holonomic experimental vehicle with realistic kinodynamic constraints (Bis et al., 2010). In addition, it should be noted that the algorithm operates best when the obstacles maintain a constant velocity; however the fast update rate and velocity uncertainty factor allow the algorithm to work effectively with obstacles that change velocity as demonstrated by the simulations in Section 5 and Figure 14.

In Section 2 of this paper, we will show how the uncertain location and velocity of obstacles, and the robot's goal, can be represented in velocity space. Then, in Section 3, we will populate VOS with repulsive and attractive values to allow for robot navigation based on weighted environmental influences. In Section 4, we describe the selection of

weights via an optimisation processes. The results of simulations will be presented in the Section 5 and finally, we will give our conclusions and plans for future work in Section 6.

1.3 Background

Details are provided here on related previous research, including a discussion of how they will be utilised in this paper as well as discussion of our implementation and extensions of these methods.

1.3.1 Configuration space and timing

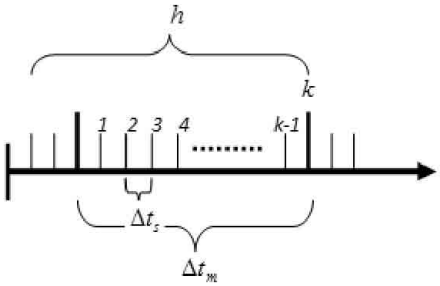
As previously mentioned, cell decomposition and certainty grids have been used to allow a robot to navigate and avoid stationary obstacles using uncertain sensor data. Using a distance-finding sensor, such as a Laser Range Finder (LRF), the robot can determine the approximate angle and distance which the obstacle is from the robot. The accuracy and precision of the data collected is correlated to the quality of the sensor in use. As such, when using a low cost sensor a high error rate is unavoidable and a certainty grid can be employed in order to account for data errors. Moravec and Elfes (1985), Borenstein and Koren (1989) use a certainty grid, which gives each cell a certainty value that indicates the confidence that the cell is occupied, in order to represent the uncertainty and error inherent in the sensor measurements.

Two separate time steps are used in this derivation, a motor time step, t_m, and a sensor time step, t_s, which are related according to

$$\Delta t_m = k \cdot \Delta t_s \tag{1}$$

where k is an integer and $k > 1$. The time steps are related in this way as it is assumed that a large (e.g., $k \approx 10$) number of sensor measurements will be read for every motor command produced by the algorithm. The number of sensor time steps in a motor time step (k) is independent of the number of sensor time step observations (h) used to calculate the VOS of each motor time step. However, the value of h must be at least $2\Delta t_s$ to allow for obstacle association across sensor time steps, and experimentation has shown that an h value between $k/2$ and $3k/2$ tends to produce the best results. Figure 1 shows a graphical representation of the relationship between time steps.

Figure 1 Relative time steps, h and k independent



The Cartesian grids are summed for h sensor time steps (see Figure 2) in order to compute the weighted Cartesian occupancy space grid (see Figure 3), used for the

subsequent velocity calculations. Before summation, the obstacles are 'grown' by the dimensions of the robot, which thereafter allows the robot to be treated as a point. This is a common practice for occupancy space navigation techniques; see Murphy (2000). The grids are summed using the equation.

$$O_c\left(x_i(t_s), y_i(t_s)\right) = \sum_{\tau=t_s-h\cdot\Delta t_s}^{t_s} \left(\frac{1}{\beta\cdot(ts-\tau)\cdot\left\|\vec{v}_r\right\|+1} \right) O_c\left(x_i(\tau), y_i(\tau)\right) \tag{2}$$

where $\|\vec{v}_r\|$ is the magnitude of the robot's velocity during the current motor time step, and $\beta \geq 0$ is a user defined variable that regulates how much influence the time-lag and robot velocity should have on the sensor measurement from each previous time step. In this way the most recent sensor measurement is given its full weight while previous measurements have reduced weights based on the time elapsed and the velocity of the robot; we have found that a β value of 1.5 produces good results. These terms help both to reduce the error in the position estimate of moving obstacles as well as compensate for error in the robot's movements. Basic coordinate transformations, detailed by Bis et al. (2009), are used to correct for the change in position and orientation of the robot so that the occupancy value of each discrete element in configuration occupancy space at each time step, $O_c\left(x_i(\tau), y_i(\tau)\right)$, will refer to the same location.

Figure 2 Cartesian grid for several time steps (see online version for colours)

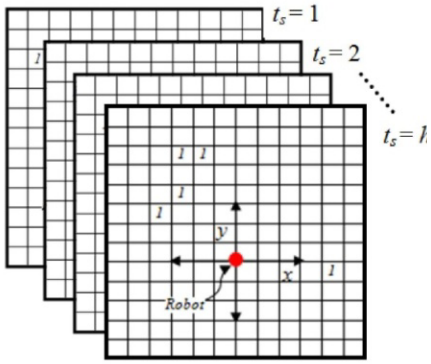
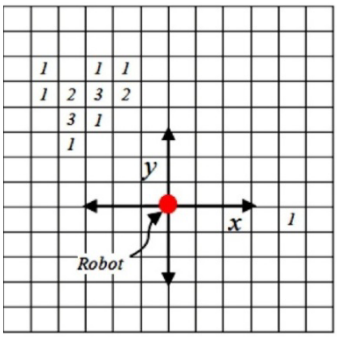


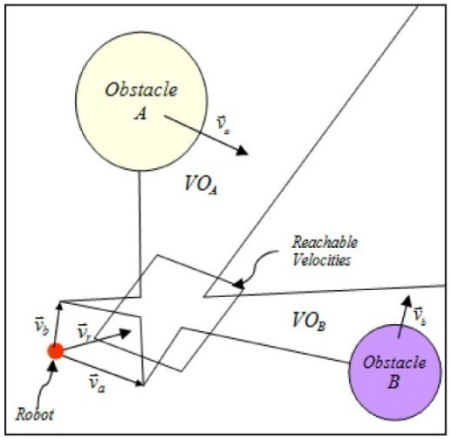
Figure 3 Summed Cartesian grid indicating range detection over a past horizon (see online version for colours)



1.3.2 Background on velocity obstacles

As previously mentioned, the concept of a velocity obstacle, as detailed by Fiorini and Shiller (1998), Large et al. (2005) and Shiller et al. (2001), has been used in the development of VOS. Under the original velocity obstacle concept, all linear robot velocities that will lead to a collision between the robot and an obstacle, *Obstacle A* in Figure 4, are considered to be part of the velocity obstacle, VO_A, of that obstacle. As long as the tip of the robot's velocity vector, $\vec{v}_r$, does not fall within VO_A and the obstacle's linear velocity, $\vec{v}_A$ remains constant, then the robot will avoid a collision with *Obstacle A*. The velocity obstacles' of multiple obstacles, *Obstacles A* and *B*, can be combined along with the set of dynamically possible robot velocities, or *Reachable Velocities*, in order to find a safe and dynamically feasible velocity for the robot.

Figure 4 Robot and velocity obstacles of Obstacles A and B (see online version for colours)



Source: Adapted from Fiorini and Shiller (1998)

2 Representing obstacles and the goal in VOS

In order to produce the VOS for a robot based on uncertain sensor data, the approximate location and velocity of each obstacle and the goal in configuration space must be determined. Using this information, the velocity obstacles, a set of velocities which will lead to a collision between the robot and an obstacle, can be found.

2.1 Centre of certainty

Sensor data from a laser range finder is collected in the form of local, polar coordinates of obstacles. It is then converted into global, Cartesian coordinates based on the robot's perceived location and used to build a discrete occupancy grid. The specifications of the LRF that we plan to use for experimental testing (the Hokuyo UTM-30LX; ±30 mm accuracy and 0.25° resolution) are used when converting the scan data into the occupancy grid in order to account for sensor error. This is accomplished by increasing the occupancy value of both the occupancy grid elements that have been determined to be

occupied as well as neighbouring elements that are within the range of error. Occupied obstacle elements are clustered together and the approximate location of each obstacle is found using a variation of the centre of mass equation, termed the Centre of Certainty, C_j:

$$C_j\left(x(t_s), y(t_s)\right) = \frac{\sum_{i=0}^{n_j}(x_i, y_i)O_c\left(x_i(t_s), y_i(t_s)\right)}{\sum_{i=0}^{n_j}O_c\left(x_i(t_s), y_i(t_s)\right)} \tag{3}$$

where (x_i, y_i) is the location of the element i which has the occupancy value, $O_c(x_i(t_s), y_i(t_s))$, at time t_s. It should be noted that the obstacles are numbered as $j = 1, 2, \ldots, N_o$, where N_o is the number of obstacles that the robot detects throughout the simulation. In addition, each obstacle, j, consists of n_j elements, numbered as $i = 1, 2, \ldots, n_j$. The centre of certainty equation uses the number of times that an obstacle is detected in an element of occupancy space as a measure of how certain it is that the element is actually occupied. This data is used to create a weighted average and locate the approximate centre of the obstacle.

Obstacle association between consecutive sensor time steps is performed as the obstacles in configuration space are composites of observations from multiple previous sensor time steps. Therefore, each obstacle at each sensor time step will fall (at least partially) within the bounds of its location at the previous sensor time step, usually allowing for non-ambiguous data association for a very low computational cost. However, this technique cannot accurately distinguish between two or more obstacles that are extremely close together (as they would appear to merge). We compensate for this both with a velocity uncertainty factor (equation (8), Section 3.1.1) which will cause the robot to be more cautious in a combined obstacle situation, and with a high data update rate that will allow the obstacles to be correctly distinguished and tracked as soon as they have moved apart.

The velocity of the centre of certainty of obstacle $j, \dot{C}_j$, can be estimated by calculating how far the centre of certainty of the obstacle moves between sensor time steps using simple differencing techniques. Simulation tests have shown that the centre of certainty obstacle velocity estimation method is between 22% and 30% more accurate at estimating the obstacle's velocity in the dominant direction of movement when compared to using the centre of the obstacle's bounding box (or basic optical flow techniques). However, basic optical flow velocity estimation methods are about 5% more accurate in the obstacle's non-dominant movement direction.

Because the configuration space is discrete, rounding errors are produced when finding the estimated velocities, especially when a lower resolution configuration space grid is used. To compensate for this, the velocities are smoothed by averaging the obstacle's velocities over a number of sensor time steps, h, to find the obstacle velocity at motor time step, t_m,

$$\dot{C}_j(\dot{x}, \dot{y}, t_m) = \frac{\sum_{\tau=t_m-h\cdot\Delta t_s}^{t_m}\dot{C}_j(\dot{x}, \dot{y}, \tau)}{h}. \tag{4}$$

This method finds the centre of the side(s) of the obstacle presented to the robot, not the physical centre of the obstacle. For obstacles with a large aspect ratio, this will produce some velocity error when the obstacle turns or the robot circles the obstacle and a new side is presented to the robot. However, using the obstacle's centre of certainty (instead

of the centre of the obstacle's observed physical dimensions) to estimate its velocity and averaging the estimated velocities over multiple sensor time steps decreases the error in the estimated obstacle velocity used for the formulation of VOS. Initial low-speed experimental tests with a large aspect ratio obstacle moving across the sensors field of view (along a line perpendicular to the robot, so that the size of the obstacle appeared to change) showed that the error produced from obstacles with large aspect ratios was typically less than 5% of the actual obstacle velocity (this was found using the Hokuyo UTM-30LX, and a configuration space resolution of 0.2 m). This method is similar to an approach explored by Fuerstenberg et al. (2003), but we retain the ability to operate in any environment by not making any assumptions about the properties of the obstacles.

For ease of notation, the velocity of obstacle element i will be referred to as $(\dot{x}_i(t_m), \dot{y}_i(t_m))$, where

$$\forall(\dot{x}_i(t_m), \dot{y}_i(t_m)) \quad (\text{if } i \in j \Rightarrow (\dot{x}_i(t_m), \dot{y}_i(t_m)) = \dot{C}_j(\dot{x}(t_m), \dot{y}(t_m))). \tag{5}$$

Our method differs significantly from the more commonly used probability based data association methods, such as those found in Almeida and Araujo (2008), Benenson et al. (2008) or Schulz et al. (2003), that are used to populate occupancy grids and track obstacles. While these methods will almost always produce more accurate results in terms of locating and tracking obstacles (especially occluded obstacles), most data association techniques are very time consuming and computationally expensive, especially as the number of obstacles that they are tracking increases.

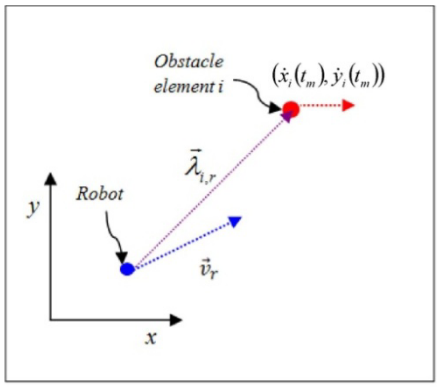
The crude, yet fast, obstacle tracking and velocity estimation method used in the VOS algorithm takes only around 8 ms to update the occupancy grid and estimate the obstacle locations and velocities (running in parallel with the rest of the algorithm, described in Section 3, on a 2.53 GHz laptop). As this is faster than the scan rate of our LRF (Hokuyo UTM-30LX, 40 Hz) the algorithm is able to make use of all available sensor data. In addition, we compensate for the early loss of accuracy by giving our velocity weighting algorithm the ability to compensate for error and obstacle unpredictability (see Section 3). The main difference between our algorithm and other methods is that we have shifted most of the computational load from the map building stage to the velocity selection stage. The benefits of this shift are that VOS can make use of much more sensor data, accumulate less odometry error between sensor readings and recognise and respond more quickly to unanticipated events.

2.2 Obstacles and the goal in velocity space

Using the location and approximate velocity that was previously calculated for the obstacle, the obstacle's location in VOS can be determined, and from this location the velocity obstacle (i.e., the set of robot velocities that will lead to a collision between the robot and obstacle) can be found.

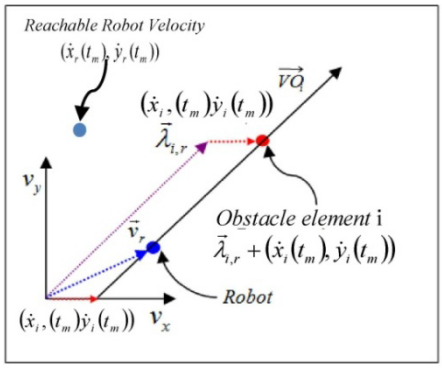
Figure 5 shows the position and velocity, $\vec{v}_r$, of the robot and the position of element i, of obstacle j, and the vector, $\vec{\lambda}_{i,r}$, between the robot and the obstacle element. The centre of certainty velocity of the obstacle that element i belongs to is $(\dot{x}_i, \dot{y}_i, t_m)$. While we have clustered the occupied elements into obstacles in order to determine their velocity, we still use the individual elements from configuration space to populate velocity space so that their occupancy certainty values can be directly utilised.

Figure 5 Configuration space representation of the robot and an obstacle (see online version for colours)



In Figure 6, the robot and one obstacle are shown in velocity space. The robot is located at its velocity, $\vec{v}_r$, and the obstacle is located at the sum of the obstacle's centre of certainty velocity, $(\dot{x}_i(t_m), \dot{y}_i(t_m))$, and the vector between the robot and the obstacle in configuration space, $\vec{\lambda}_{i,r}(\dot{x}(t_m), \dot{y}(t_m))$. In other words, the obstacle is located at the velocity that the robot would need to assume in order to collide with the obstacle in one motor time step, which takes into account both the obstacle's distance from the robot as well as the obstacle's own velocity. The vector originating at $(\dot{x}_i(t_m), \dot{y}_i(t_m))$ and intersecting, $\vec{\lambda}_{i,r}(\dot{x}_i(t_m), \dot{y}_i(t_m)) + (\dot{x}_i(t_m), \dot{y}_i(t_m))$ in velocity space, is the set of collision causing velocities that makes up the velocity obstacle, $\overrightarrow{VO_i}$. Any of these velocities will cause the robot to collide (at some point in time) with the obstacle, assuming constant obstacle velocity.

Figure 6 Velocity space representation of the robot velocity, $\vec{v}_r$, and the velocity obstacle, $\overrightarrow{VO_i}$ (see online version for colours)



In this paper, it is assumed that the relative position and velocity of the goal are known. As such, locating and tracking the centre of certainty is not needed to find the *velocity goal*, $\overrightarrow{VG_i}$, or the set of robot velocities which will lead the robot to the goal, in the same manner that the velocity obstacle was found. However, if a sensor was employed that could distinguish the goal from surrounding obstacles, then the same technique used for

the obstacles could be used to locate the goal, track it and determine its location in velocity space.

3 Populating Velocity Occupancy Space

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. VOS consists of weighted elements that correspond to possible linear robot velocities. While the velocity obstacle shown in Figure 6 is in continuous space, VOS itself is discrete with a user defined resolution. If any part of a velocity obstacle will fall into a discrete element of VOS, the velocity represented by that element is considered to lead to a collision. Therefore, the higher the resolution the more precisely the robot will be able to select velocities and avoid obstacles, however the increased resolution also increases the computational time needed to populate VOS. For most of the simulations in this paper the velocity space resolution was 0.1 m/s which was less than the error rate of the motors of the robot on which experimental verification of the algorithm will be performed.

The weight of each element in VOS is based on two sets of factors. The first set forms a repulsive weight, based on the possibility that this velocity might lead the robot to a collision with an obstacle. The second set is based on how quickly and directly a velocity will lead the robot to its goal.

3.1 Repulsive weights

The repulsive weighting value of each element of a velocity obstacle is influenced by a number of variables that determine how much of a threat an obstacle is and to what degree it should be avoided over other obstacles. The repulsive value, R, of each element is defined by the equation

$$R = W_R \left[D_R A_R \ (W_{AR}) \cdot \left(\frac{W_{TTC}}{TTC} + \frac{1}{CD} \right) \cdot E_{Oc} \right] \tag{6}$$

where the weights (W_R, W_{TTC}, and W_{AR}) are defined and optimised 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 other terms in equation (6) are detailed as follows.

The term E_{Oc} is the occupancy value, $O_C(x_i(t_m), y_i(t_m))$, for the obstacle element with which each robot velocity will lead to a collision. This value is the maximum value of the occupancy values of the various obstacles to which this velocity will lead.

The other terms are variables related to the robot's state and environment and include D_R, which is the repulsive direction term, A_R, which is the repulsive angular term, TTC, which is a measure of the time to collision and CD which is the Cartesian distance between the robot and the obstacle.

3.1.1 Angle and direction equations

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. First, the velocity represented by the element must cause the robot to move with a negative speed relative to the obstacle, as defined by:

$$D_R = \begin{cases} 1 & \text{if } \left(\dfrac{\vec{\lambda}_{i,r}(\dot{y}(t_m)}{(\dot{y}_r(t_m) - \dot{y}_i(t_m) \cdot (1 \pm V_U))} \wedge \dfrac{\vec{\lambda}_{i,r}(\dot{x}(t_m))}{\dot{x}_r(t_m) - \dot{x}_i(t_m) \cdot (1 \pm V_U)} \right) \geq 0 \\ 0 & \text{othewise} \end{cases} \tag{7}$$

where $\vec{\lambda}_{i,r}(\dot{y}(t_m))$ and $\vec{\lambda}_{i,r}(\dot{x}(t_m))$ are the relative displacement vectors between the robot and the obstacle element *i* (see Figure 5), $\dot{y}_r(t_m)$ 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. If a value of zero is found in equation (7), then the velocity is not part of a velocity obstacle and will therefore have no repulsive weight. The velocity uncertainty, V_U, represents how uncertain the estimate of the magnitude of the obstacle's velocity is, and is found from the equation

$$V_U = \min(|(\dot{x}_i(t_m), \dot{y}_i(t_m)) - (\dot{x}_i(t_{m-1}), \dot{y}_i(t_{m-1}))|, \max|(\dot{x}_r(t_m), \dot{y}_r(t_m))|). \tag{8}$$

This uncertainty factor, V_U is used to increase the range of obstacle velocities that are avoided. If the currently measured velocity is the same as what was measured on the previous time step, then only the estimated obstacle velocity is avoided. However, as the prediction and the observation differ, the range of obstacle velocities that are assumed to be hazardous also increases. The upper bound on this term is the robot's maximum velocity, as the robot cannot be guaranteed of avoiding an obstacle moving more quickly than the robot is capable of. This term also compensates for quickly accelerating vehicles that may move unpredictably.

In equation (7), and in later equations, the plus minus sign, ± is used in order to signify a continuous, inclusive OR operator. For example, in the first part of equation (7) the terms $\dot{y}_i(t_m) \cdot (1 \pm V_U)$ indicate that if any value equal to or between $\dot{y}_i(t_m) \cdot (1 - V_U)$ and $\dot{y}_i(t_m) \cdot (1 + V_U)$ will cause

$$\left(\frac{\vec{\lambda}_{i,r}(\dot{y}(t_m))}{\dot{y}_r(t_m) - \dot{y}_i(t_m) \cdot 1 \pm V_U} \right) \geq 0,$$

then that value is used and the first part of the equation is considered to be greater than zero. This term will be frequently used when we are increasing the range of velocities that are considered occupied or dangerous in order to deal with error or uncertainty.

The second criterion is that the element's velocity must move the robot into a collision course with the obstacle as defined by the relative angles between the robot's and obstacle's positions and velocities. The equation

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

defines the angle of collision where V_U has the same role as before, only now it increases the range of angles instead of the radial direction, based on the predictability of the obstacle's velocity. The weighted angular term, W_{AR}, allows a more or less conservative range of velocity angles that are assumed to lead to a collision to be defined based on the situation (a W_{AR} value of one will not affect the range). Finally, the angular proximity, P_A, is defined by

$$P(A) = \left(\frac{sr(t_m) - \| (x_i(t_m), y_i(t_m)) - (x_r(t_m), y_r(t_m)) \|}{sr(t_m)}\right)^2 \cdot \frac{\pi}{2} \quad (10)$$

where the term $sr(t_m)$ is the extent of the robot's sensor range at the current time step. Using this equation in conjunction with equation (9), the angular proximity term will increase the range of angles considered to be occupied by a single obstacle element by 0 rad (0°) for obstacles that are at the limit of the sensor range to $\frac{\pi}{2}$ rad (90°) for obstacles that are adjacent to the robot. This is in addition to the velocity uncertainty and weighted angular terms. This helps the robot to account for additional obstacles that might be hidden behind a closer, occluding obstacle. If one of these undetected obstacles were to unexpectedly move out from behind an occluding obstacle that was very near to the robot a collision would very likely result. However, if the occluding obstacle was farther away from the robot, then the robot would have time to detect the newly revealed obstacle and respond appropriately.

3.1.2 Time to collision and Cartesian distance

The time to collision, *TTC*, and the Cartesian distance, *CD*, terms are variables which measure the physical relationship between the robot and an obstacle. If a velocity does not meet the angle and direction requirements, as described above, then the *TTC* and *CD* are not calculated, as the overall repulsive weight, *R*, is already set to zero (equation (6)). Therefore, every velocity value for which *TTC* and *CD* are computed is assumed to lead to a collision. This limitation on the set of robot velocities examined greatly reduces the complexity of these equations. The weighting term, W_{TTC}, is used as a ratio between the two variables. It represents how important it is to avoid velocities that will lead to a collision as opposed to velocities that lead to an obstacle that is close to the robot. These priorities can be radically different for an obstacle that is close to the robot, but moving in the opposite direction.

As commonly defined, the time to collision term, *TTC* measures the amount of time that it will take the robot to collide with an obstacle for each element's velocity value, $(\dot{x}_r(t_m), \dot{y}_r(t_m))$, assuming that both the robot and the obstacle maintain a constant velocity. For our derivation, the equation for the *TTC* is

$$TTC = \begin{cases} \dfrac{\left\|\vec{\lambda}_{i,r}\ (\dot{x}(t_m),\dot{y}(t_m))\right\|}{\left\|(\dot{x}_r(t_m),\dot{y}_r(t_m))-(\dot{x}_i(t_m),\dot{y}_i(t_m))\right\|} & \left\|(\dot{x}_i(t_m),\dot{y}_i(t_m))\right\| \le \left\|\dot{x}_r(t_m),\dot{y}_r(t_m)\right\| \\ & \le \left\|\vec{\lambda}_{i,r}+(\dot{x}_i(t_m),\dot{y}(t_m))\right\| \\ \dfrac{\left\|\vec{\lambda}_{i,r}(\dot{x}(t_m),\dot{y}(t_m))\right\|}{(\dot{x}_r(t_m),\dot{y}_r(t_m))} & \left\|\vec{\lambda}_{i,r}+(\dot{x}_i(t_m),\dot{y}_i(t_m))\ \right\| < \left\|\dot{x}_r(t_m),\dot{y}_r(t_m)\right\| \end{cases} . \tag{11}$$

If the robot velocity in question falls between the obstacle's velocity and the sum of the obstacle's velocity and $\vec{\lambda}_{i,r}(\dot{x}(t_m),\dot{y}(t_m))$ then the *TTC* value is calculated. If the robot velocity is greater than this sum, then that velocity will cause a collision in less than one time step (i.e., before the robot has a chance to respond) so these velocities are given the smallest possible value (which makes equation (6) highly repulsive).

The Cartesian distance,

$$CD = (x_i(t_m),y_i(t_m))-(x_r(t_m),y_r(t_m))^2 \tag{12}$$

is a simple measure of how far away the obstacle is from the robot. Obstacles that are close by, and therefore present a more immediate threat, are given higher repulsive weightings than obstacles which are farther away.

3.2 Attractive weights

As previously mentioned, only velocity space elements that are part of a velocity obstacle – and will therefore lead to a collision between the robot and the obstacle – are given any repulsive weighting. All other elements are assumed to represent safe robot velocities. However, all elements in VOS 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 value for each VOS element is found from the equation

$$A = [W_{VD}\cdot VD + VC + W_A\cdot A_A] \tag{13}$$

where the weights, W_{VD} and W_A, are defined based on the robot's objectives. The velocity difference, *VD*, term is found from the equation

$$VD = \frac{\left\|(\dot{x}_r(t_m),\dot{y}_r(t_m))-\kappa\right\|}{2\cdot\left\|\max\ (\dot{x}_r(t_m),\dot{y}_r(t_m))-\min(\dot{x}_r(t_m),\dot{y}_r(t_m))\right\|}-1 \tag{14}$$

where the κ is defined as

$$\kappa = \begin{cases} \max(\dot{x}_r(t_m),\dot{y}_r(t_m)) & \text{if} \quad \max\ (\dot{x}_r(t_m),\dot{y}_r(t_m)) < \left(\vec{\lambda}_{G,r}+(\dot{x}_G(t_m),\dot{y}_G(t_m))\right) \\ \vec{\lambda}_{G,r}+(\dot{x}_G(t_m),\dot{y}_G(t_m)) & \text{if} \quad \min\ (\dot{x}_r(t_m),\dot{y}_r(t_m)) \le \left(\vec{\lambda}_{G,r}+(\dot{x}_G(t_m),\dot{y}_G(t_m))\right) \\ & \quad \le \max\ (\dot{x}_r(t_m),\dot{y}_r(t_m)) \\ \min\ (\dot{x}_r(t_m),\dot{y}_r(t_m)) & \text{if} \quad \left(\vec{\lambda}_{G,r}+(\dot{x}_G(t_m),\dot{y}_G(t_m))\right) < \min\ (\dot{x}_r(t_m),\dot{y}_r(t_m)) \end{cases} \tag{15}$$

where $\left(\vec{\lambda}_{G,r}+\dot{x}_G(t_m),\dot{y}_G(t_m)\right)$ is the location of the goal in velocity space, and $\max(\dot{x}_r(t_m),\dot{y}_r(t_m))$ and $\min(\dot{x}_r(t_m),\dot{y}_r(t_m))$ are the maximum and minimum velocities that the robot can reach during this time step. The κ term is used in the velocity difference equation so that a consistent weighting can be maintained no matter how far the goal is from the robot. In other words, it is desirable that the velocity which will lead most quickly to the goal will always have the same attractive weighting no matter how far the goal actually is from the robot.

The next term, velocity change, *VC*, is given by the equation

$$VC=\frac{\left\|(\dot{x}_r(t_{m-1}),\dot{y}_r(t_{m-1}))-(\dot{x}_r(t_m),\dot{y}_r(t_m))\right\|}{\max(\dot{x}_r(t_m),\dot{y}_r(t_m)-\min(\dot{x}_r(t_m),\dot{y}_r(t_m)}-1. \tag{16}$$

The purpose of this term is to discourage frequent accelerations and decelerations.

Finally the cosine of the angle between the goal's location in velocity space and the velocity element in question, is found from

$$\alpha=\cos\left[\tan^{-1}\left(\frac{\vec{\lambda}_{G,r}(\dot{y}(t_m))+\dot{y}_G(t_m)}{\vec{\lambda}_{G,r}(\dot{x}(t_m))+\dot{x}_G(t_m)}\right)-\tan^{-1}\left(\frac{\dot{y}_r(t_m)}{\dot{x}_r(t_m)}\right)\right] \tag{17}$$

and the attractive angle term, A_A, is set as the negative of the cosine of that angle or zero, if the angle is large enough that the velocity would no longer be leading in the direction of the goal

$$A_A=\begin{cases}-\alpha & \text{if } \left|\tan^{-1}\left(\frac{\vec{\lambda}_{G,r}(\dot{y}(t_m))+\dot{y}_G(t_m)}{\vec{\lambda}_{G,r}(\dot{x}(t_m))+\dot{x}_G(t_m)}\right)-\tan^{-1}\left(\frac{\dot{y}_r(t_m)}{\dot{x}_r(t_m)}\right)\right|\le\frac{\pi}{2}\\ 0 & \text{if } \left|\tan^{-1}\left(\frac{\vec{\lambda}_{G,r}(\dot{y}(t_m))+\dot{y}_G(t_m))}{\vec{\lambda}_{G,r}(\dot{x}(t_m))+\dot{x}_G(t_m)}\right)-\tan^{-1}\left(\frac{\dot{y}_r(t_m)}{\dot{x}_r(t_m)}\right)\right|>\frac{\pi}{2}\end{cases}. \tag{18}$$

Similar to the κ term in equation (14), the contents of the first inverse tangent term in equation (17) are replaced with the maximum or minimum reachable robot velocity ($\max(\dot{x}_r(t_m),\dot{y}_r(t_m))$ and $\min(\dot{x}_r(t_m),\dot{y}_r(t_m))$) if the goal's location in velocity space is outside of these bounds. This prevents the preferred angle from decreasing too much to encourage circumnavigation of an obstacle if the robot is far from the goal.

The equations shown in this section form the basis of VOS. Other logic was included to avoid numerical contingencies, such as division by zero, in the actual program, but is omitted here for brevity.

3.3 Velocity selection and navigation

For the simulation, negative values are used to represent how attractive an element of VOS is, *A* from equation (13), and positive values to represent how repulsive the element is, *R* from equation (6). This allows the attractive and repulsive values of a single element to be summed so that the final value of a single element in VOS can be influenced by multiple factors. The value of each element indicates the desirability of that robot velocity and the element with the lowest value (i.e., the most desirable velocity) can be found with a simple gradient search.

The weights in equations (6) and (13) can be adjusted both to influence how the various terms should rank relative to each other, as well as to govern the interplay between the attractive and repulsive values. These weights are pre-calculated by the process outlined in Section 4.

After the robot velocity has been chosen for a motor time step, the process described in the previous two sections is repeated so as to allow the robot to continuously adjust its velocity in order to deal with non-constant obstacle velocities and newly detected obstacles. The processes described in Section 2 (building the occupancy space grid and estimating obstacle locations and velocities) and Section 3 (populating VOS and selecting the next velocity) are performed in parallel. As shown in Figure 1, information from multiple sensor sweeps is used to form VOS and find each consecutive robot velocity.

4 Optimisation of weights

The value of each element in VOS is defined by the sum of equations (6) and (13):

$$\begin{aligned} \mathrm{VOS}(\dot{x}_r(\dot{t}_m), \dot{y}_r(t_m)) &= R + A \\ &= W_R \left[D_R \cdot A_R(W_{AR}) \cdot \left(\frac{W_{TCC}}{TTC} + \frac{1}{CD} \right) \cdot E_{Oc} \right] \\ &\quad + [W_{VD} \cdot VD + VC + W_A \cdot A_A]. \end{aligned} \tag{19}$$

Along with defining the value of the individual terms based on the physical properties of the system and environment, the individual weights (W terms) must be determined in order to effectively prioritise the various aspects of obstacle avoidance and goal finding. Initially, these weights were hand-tuned based on empirical knowledge and observation of the system (Bis et al., 2009). In this section a combination of an exhaustive search and an optimisation process, which produced significantly better weights, is described.

4.1 Evaluation criteria

Four evaluation metrics were used in order to judge the quality of the path that the robot followed given each set of weights. During each time step the position of the robot, the robot's velocity and the relative position of each obstacle to the robot was recorded. The magnitude of the robot's change in position (the distance that it travelled during the simulation), change in velocity and the square of the inverse of the closest obstacle's proximity to the robot were each summed for every time step and used for the first three evaluation metrics: distance travelled, acceleration and obstacle proximity. In addition, the number of time steps required for the robot to reach the goal and two binary values that indicated if a collision occurred during the scenario and if the robot was successful at reaching the goal were also used for evaluation metrics.

The number of collisions and the number of times that the robot successfully reaches that goal are the most important measures of the algorithm's performance. However, the other evaluation metrics, shown in Table 1, were also recorded in order to compare the quality of the paths that the robot chooses using each set of weights.

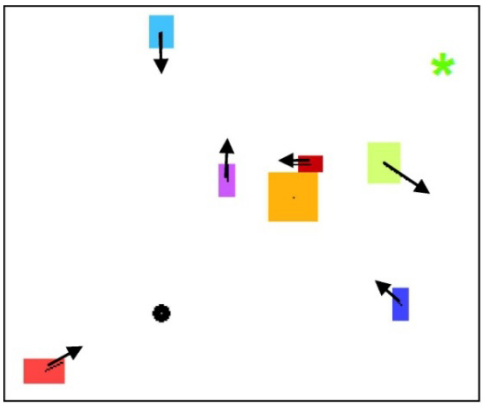
Table 1 Evaluation metrics

Evaluation metrics	*Equations*
Obstacle proximity	$\frac{1}{N}\cdot\sum_{n=1}^{N}\sum_{\tau=1}^{Tn}\frac{1}{\sum_{j}^{J}\left\|C_r(t)-C_{O\,j}(t)\right\|^2}$
Change in velocity	$\frac{1}{N}\cdot\sum_{n=1}^{N}\sum_{\tau=1}^{Tn}\left\|V_r(\tau)-V_r(\tau-1)\right\|$
Distance travelled	$\frac{1}{N}\cdot\sum_{n=1}^{N}\sum_{\tau=1}^{Tn}\left\|C_r(\tau)-C_r(\tau-1)\right\|$
Time	$\frac{1}{N}\cdot\sum_{n=1}^{N}T_n$

N: Number of iterations; T_n: Number of time steps in iteration n; J: Number of obstacles; C_r: Robot position; C_o: Obstacle position; V_r: Robot velocity.

A random scenario generator was also used in order to produce a broad range of situations in which the algorithm's ability to successfully guide a robot could be tested. The scenario generator produced a number of obstacles (between one and eight) with a range of velocities and starting positions, as well as different positions for the goal. Impossible scenarios (e.g., if the scenario started with a collision) were removed. An example of the initial conditions of one of these scenarios is shown in Figure 7. While Figure 7 (and later figures) shows an overhead view of the robot and obstacles, for all of the simulations the robot only had access to the noisy LRF data that would have been produced from the environment.

Figure 7 Initial conditions of a sample scenario. Figure contains the robot (circle), obstacles (rectangles with velocity vectors) and the goal (asterisk) (see online version for colours)



The random scenarios used for the optimisation, as well as the example simulation shown throughout the paper, have the following parameters: configuration space resolution = 0.2 m velocity space resolution = 0.1 m/s, $k = 10$, $h = 7$, $\beta = 1.5$ and the sensor range = 20 m.

The performance measures for 10 different scenarios were used for the optimisation process so that the results of the optimisation would be appropriate for a more general environment, instead of being over-specific for a single scenario. In addition, the same set of 10 scenarios was used throughout the optimisation processes so that the results could be accurately compared.

4.2 Optimisation

A coarse exhaustive search was performed in order to find a selection of better initial design variables that could avoid some of the less desirable local minima. The course exhaustive search showed that there were a significant number of non-optimal local minima and also some discontinuities within the design variable search space. However, using some of the better sets of variables from the exhaustive search as the initial set of design variables, the optimisation process produced improved results. For validation, the results of the optimisation process were tested on a set of one thousand scenarios, from the random obstacle scenario generator, in order to verify that the VOS algorithm would operate acceptably for almost any scenario.

Optimisation was performed using MATLAB's Fgoalattain function. This function uses sequential quadratic programming to reduce a set of nonlinear functions to below a given goal level. We used it in the following manner:

$$\underbrace{\text{minimax}}_{x,\gamma}\ \gamma \ \text{ such that} \begin{cases} \boldsymbol{F}(\boldsymbol{x}) - \text{weight} \cdot \gamma \le \boldsymbol{F}_{\boldsymbol{g}}(x) \\ \boldsymbol{h}(\boldsymbol{x}) = 0 \end{cases} \tag{20}$$

where the design variables, $\boldsymbol{x}$, were the weights: $\{W_R, W_{TTC}, W_{AR}, W_{VD}, W_A\}$. The equality constraints, $\boldsymbol{h}(\boldsymbol{x})$, were set so that the number of collisions and the number of times that the robot was unsuccessful at reaching the goal had to equal zero. The hand tuned weights and the optimised weights are compared in Table 2.

Table 2 Coefficients / design variable values used for velocity element weighting

		Hand-tuned weights	*Optimised weights*
Repulsive weights	Repulsive weight (WR)	1.0	0.4
	Time to collision (WTTC)	3.5	3.5
	Angular range (W_{AR})	1.0	1.0
Attractive weights	Velocity distance (WVD)	2.7	2.2
	Angle (WA)	0.3	1.2

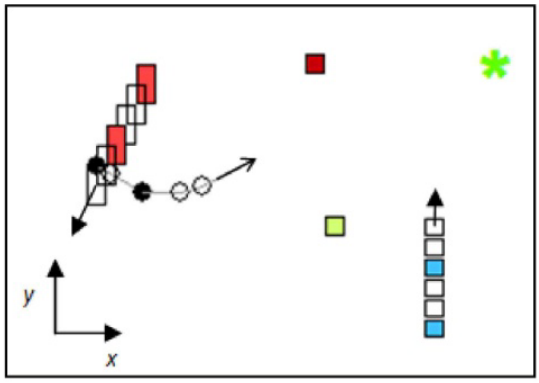
The optimisation process did not neatly converge to a global minimum that was the optimal set of weights for any situation. Our inability to find a global optimum is probably the result of two aspects of the system. First, this is a five-dimensional design problem; the function is non-convex and has some discontinuities. As such, finding the global minimum, even for a set of 10 scenarios, is an extremely challenging and time consuming optimisation process. Second, some of the solutions obtained using optimisation appear to have been overly designed for the 10 design scenarios that were used for the optimisation process, as they failed to cause the VOS algorithm to work acceptably for a broader range of scenarios. Ideally, a much larger set of scenarios should

be used for both the exhaustive search and the optimisation, however, even using just 10 scenarios made for an extremely time consuming processes (weeks of dedicated CPU time on a laptop computer), so optimising with a more complete set of scenarios would not be practical.

5 Results

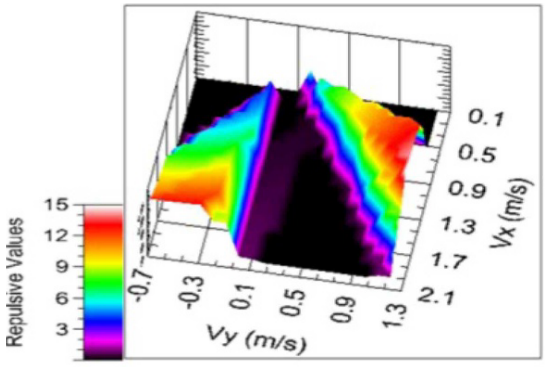
An example of the VOS algorithm run with a fairly simple scenario is shown in Figures 8–11. In Figure 8, the robot is avoiding two moving and two stationary obstacles. The simulation covers the first six motor time steps. The robot is initially stationary, so that it can make an initial estimate as to the obstacle velocities.

Figure 8 First six simulation steps of an example scenario (see online version for colours)



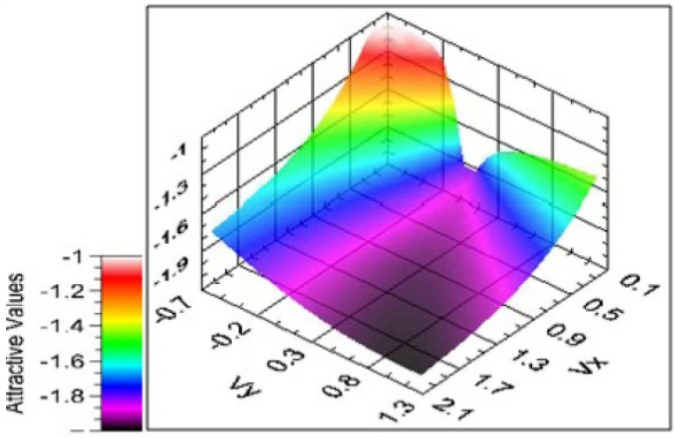
The VOS representation of the obstacles in Figure 8 is shown in Figure 9, where the velocity values (on the *x*- and *y*-axes) that will lead to a collision can be seen as the cones of repulsive values (positive values on the z-axis) in VOS.

Figure 9 VOS populated with repulsive values (see online version for colours)



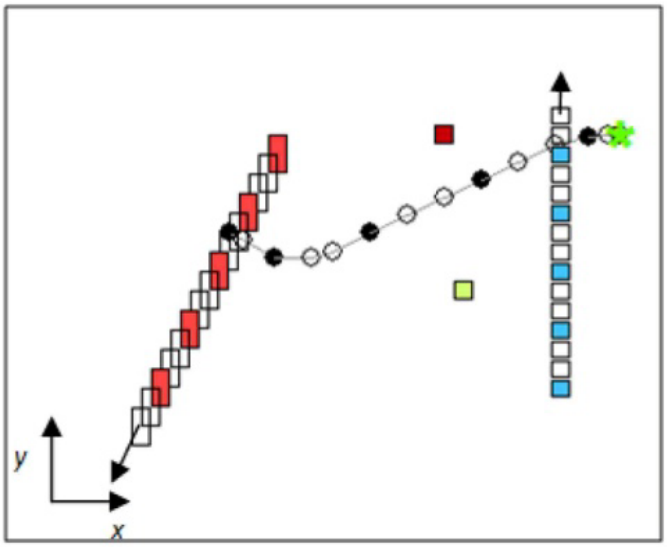
All of the VOS elements are given attractive values (negative values on the *z*-axis) in proportion to how effectively each velocity will lead the robot to the goal. In Figure 10, VOS is shown populated with attractive values based on the scenario in Figure 8.

Figure 10 VOS populated with attractive values (see online version for colours)



After 15 time steps the robot has successfully reached the goal while avoiding all obstacles, as shown in Figure 11.

Figure 11 Simulation results after 15 time steps (see online version for colours)

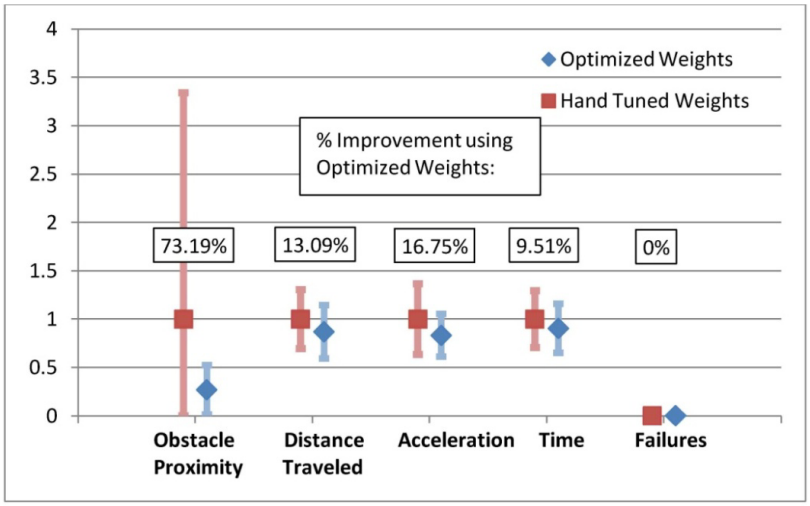


The set of weights that resulted from the optimisation process are shown in Table 2, and results from the algorithm run with these weights are shown in Figures 12 and 13.

In Figure 12, the performance of the algorithm using the original hand-tuned weights vs. the performance with the optimised weights is shown for the set of 10 design scenarios which were used to perform the optimisation. In Figure 13, the performance of the algorithm with the different sets of weights is again compared, but this time for the 1000 randomly generated scenarios, which were used to validate the results of the optimisation. The values of the evaluation metrics were normalised independently for the two sets of scenarios against the values found using the hand tuned weights; the lower the value of the evaluation metrics, the better the performance of the algorithm. Neither set of weights caused a failure (either due to a robot collision or the inability of the robot to reach the goal within a specified amount of time) of the simulation for the set of 10 design scenarios. For the 1000 validation scenarios, there were nine failures (0.9%) for the hand-tuned weights and four failures (0.4%) for the optimised weights. The simulations in which failures did occur where usually situations that even a human driver would have had difficulty successfully navigating. For example, one of the failures using the optimised weights occurred when a couple of obstacles converged almost immediately on the robot. An omniscient agent would have been able to find a successful path; however the algorithm had very little time to collect velocity data on the surrounding obstacles and there were a very limited number of velocity choices that would have allowed the robot to successfully avoid all of the obstacles.

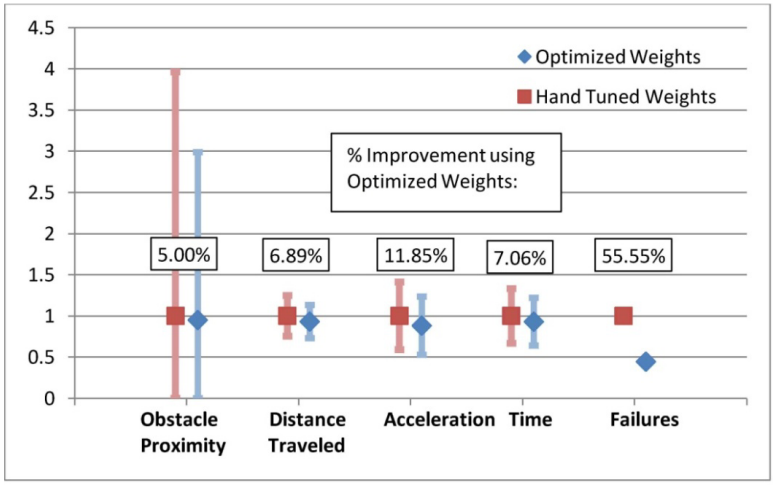
The optimisation process significantly improved the algorithm's performance for the design scenarios that were used in the optimisation. While the overall improvement for the validation scenarios was not as large, it was still statistically significantly for three of the evaluation metrics: the distance travelled, acceleration, time ($p < 0.005$, on a two-tailed, paired t-test) and for the number of simulation failures ($p < 0.025$).

Figure 12 Comparison of normalised evaluation metrics between hand tuned and optimised weights for 10 design scenarios (one sigma error bars) (see online version for colours)



While the obstacle proximity evaluation metric did not see a significant improvement using the optimised weights, the true improvement may be covered up by the improvement in the collision failure rate. The values of the evaluation metrics for a simulation in which a failure occurred where not included in the statistics. Therefore, improvements in the performance of the algorithm with the optimised weights which allowed the robot to avoid a collision (presumably by decreasing the robot's proximity to the obstacles) while the hand tuned weights led to a collision would not influence the final value of the obstacle proximity for either set of weights. In other words, the simulations where the obstacle proximity metric would have been the worst (due to a collision) for the hand tuned weights, were removed from the analysis, which may have artificially improved the value of the obstacle proximity for the hand tuned weights when compared to the optimised weights.

Figure 13 Comparison of normalised evaluation metrics between hand tuned and optimised weights for 1000 scenarios (one sigma error bars) (see online version for colours)



The VOS algorithm works most effectively when the obstacles maintain a constant velocity and the optimisation was performed using constant velocity obstacles. However, the algorithm was also tested using obstacles that could randomly change their velocity. Using the optimised weights and the 1000 validation scenarios (with obstacle's that had a 20% chance of changing their velocity every time step), eight simulation failures where recorded (0.8% failure rate). There was also no statistically significant difference between three of the four evaluation metrics (obstacle proximity, distance and time) between the scenarios where the obstacles all had constant velocities and the scenarios where the velocities randomly changed.

Figure 14 shows a scenario where the algorithm allows a robot to navigate around stationary and moving obstacles (that are subject to random velocity changes) and reach a goal. Figures 15–17 show sequential segments of the robot navigation in this scenario. In Figure 15, the robot starts to accelerate in the positive y-direction to avoid *Obstacles A* and *B,* and then circles around *Obstacle A* as *A* changes velocity. In Figure 16, the robot is unable to move between the two stationary obstacles, due to the presence of *Obstacle C,* so the robot circles around the stationary obstacles while staying out of the

path of *Obstacle B* (as *Obstacle B* is moving at the robot's maximum velocity). Finally, in Figure 17, the robot follows *Obstacle D* towards the goal, but when *Obstacle D* changes direction, the robot chooses more conservative velocities and comes to almost a complete stop because the obstacle is no longer moving at the predicated velocity.

Videos of additional simulations can be viewed at http://sites.google.com/site/rachaelbis/research/simulations-and-videos.

Figure 14 Simulation results with four moving and two stationary obstacles for 32 time steps (see online version for colours)

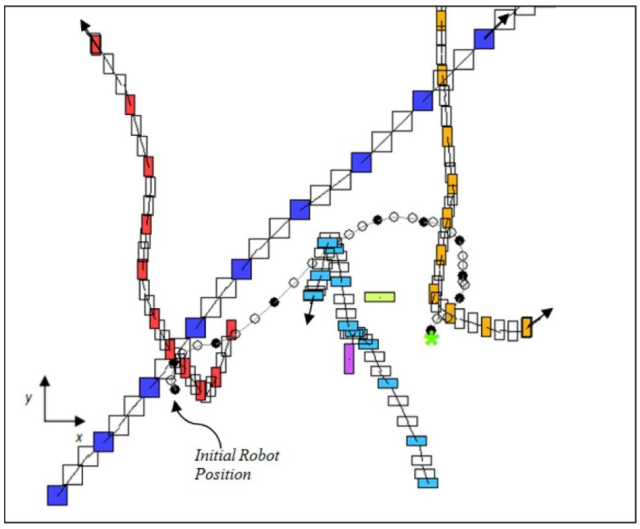


Figure 15 First seven motor time steps (see online version for colours)

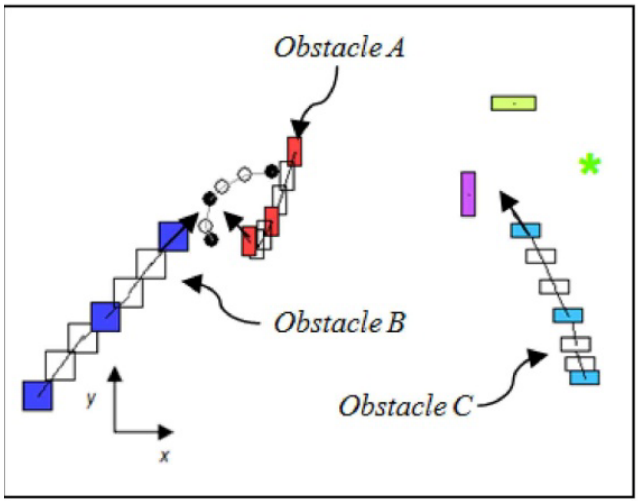


Figure 16 Motor time steps 7 through 19 (see online version for colours)

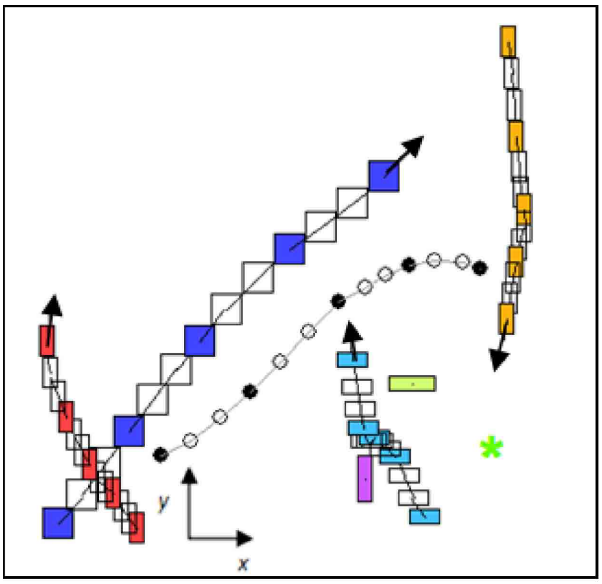
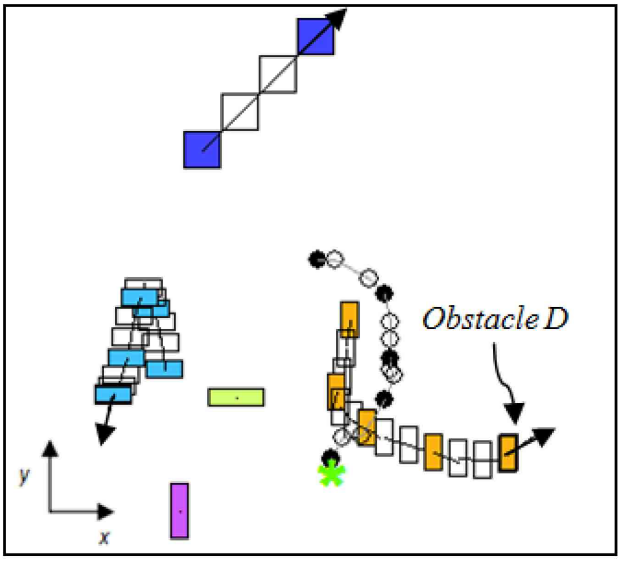


Figure 17 Motor time steps 19 through 32 (see online version for colours)



6 Summary and conclusions

In this paper VOS, a new navigation algorithm that allows a robot to operate using only a range finding sensor with uncertainty in an unknown environment and successfully avoid stationary and moving obstacles while navigating towards a goal, has been developed and presented. This method uses the uncertain obstacle representation of occupancy space to

estimate the location of each obstacle and approximate each obstacle's velocity over multiple time steps. This basic obstacle information is converted into velocity obstacle form and used to calculate variables that describe the benefits or detriments of each possible robot velocity. We optimised the relative weight that each of these variables should have, in comparison to the other variables, and used the weighted variable values to form VOS. From this space, the robot can find a velocity that is both safe and that will lead it towards the goal, if such a velocity exists. While the choice of velocity using the optimised weights may not always be ideal, we have shown that, in general, the weights will allow the robot to avoid a collision and reach its destination in the vast majority of situations.

VOS is not meant as a complete and independent robot navigation system. It is designed to perform low level robot navigation and obstacle avoidance. While it can operate independently in many types of environments, it would not be suitable to plan long, complex paths. As such, VOS would ideally be integrated with other algorithms to allow for higher level planning and navigation. For example, integration with an algorithm such as SLAMMOT (Wang et al., 2007) would allow the robot to respond quickly to avoid moving and stationary obstacles (while still choosing desirable, goal approaching velocities) using VOS, but also build a map of and recognise an environment so that it could make more complex navigation decisions. A higher-level algorithm could break down a long, but desirable, route to a destination and provide VOS with more direct, intermediate goals.

6.1 Future work

In the future, we would like to adapt VOS so that it can be directly integrated with different types of higher-level planners, as previously mentioned. We are also investigating the use of different weights for different types of specific environments, such as structured roads and uncluttered areas. During navigation, the robot could identify what type of environment it was in (structured, cluttered, etc) and then select the weights that had been optimised for such a situation.

We are also working to show the effectiveness of including additional sensor information into the VOS framework. Additional sensors, such as infrared or visible light cameras, can be used to identify different types of obstacles. If vulnerable obstacles, such as pedestrians, were identified the velocities that led to a collision with these vulnerable obstacle could be assigned more repulsive weights so that the avoidance of these obstacles would be prioritised. In addition, the algorithm is currently designed for an omni-directional robot. As the majority of robots are non-holonomic, we are extending the algorithm so that it is applicable for robots with more general movement and velocity restrictions. Finally, work is currently underway to experimentally implement and evaluate the VOS algorithm on a robotic platform.

Acknowledgements

This research was supported in part by the Ground Robotics Reliability Centre (GRRC) at the University of Michigan, with funding from government contract DoD-DoA W56H2V-04-2-0001 through the Joint Center for Robotics. The authors would also like to thank Elizabeth Boettler for her assistance.

References

Almeida, J. and Araujo, R. (2008) 'Tracking multiple moving objects in a dynamic environment for autonomous navigation', *10th IEEE International Workshop on Advanced Motion Control, (AMC '08)*, pp.21–26.

Benenson, R., Petti, S., Fraichard, T. and Parent, M. (2008) 'Towards urban driverless vehicles', *International Journal of Vehicle Autonomous Systems*, Vol. 6, Nos. 1–2, pp.4–23.

Bis, R., Peng, H. and Ulsoy, G. (2009) 'Velocity occupancy space: robot navigation and moving obstacle avoidance with sensor uncertainty', *Proceedings of the 2009 Dynamic Systems and Controls Conference*, October, Hollywood, CA, pp.363–370.

Bis, R., Peng, H. and Ulsoy, G. (2010) 'Velocity occupancy space for differential drive vehicles', *Proceedings of the 2010 Dynamic Systems and Controls Conference*, September, Cambirdge, MA, pp.249–256.

Borenstein, J. and Koren, Y. (1989) 'Real-time obstacle avoidance for fast mobile robots', *IEEE Transactions on Systems Man and Cybernetics*, Vol. 19, No. 5, pp.1179–1187.

Fiorini, P. and Shiller, Z. (1998) 'Motion planning in dynamic environments using velocity obstacles', *International Journal of Robotics Research*, Vol. 17, No. 7, pp.760–772.

Fuerstenberg, K.C., Linzmeier, D.T. and Dietmayer, K.C.J. (2003) 'Pedestrian recognition and tracking of vehicles using a vehicle based multilayer laserscanner', *Proceedings of the 10th World Congress on Intelligent Transport Systems (ITS)*, Madrid, Spain, pp.31–35.

Fulgenzi, C. (2009) *Autonomous Navigation in Dynamic Uncertain Environment using Probabilistic Models of Perception and Collision Risk Prediction*, PhD Thesis, Institut National Polytechnique de Grenoble, Rhne-Alpes, France.

Fulgenzi, C., Spalanzani, A. and Laugier, C. (2007) 'Dynamic obstacle avoidance in uncertain environment combining PVOs and Occupancy Grid', *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, pp.1610–1616.

Kant, K. and Zucker, S. (1986) 'Toward efficient trajectory planning: the path-velocity decomposition', *International Journal of Robotics Research*, Vol. 5, No. 3, pp.72–89.

Konolige, K. (2000) 'A gradient method for real time robot control', *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, pp.639–646.

Large, F., Laugier, C. and Shiller, Z. (2005) 'Navigation among moving obstacles using the NLVO: principles and applications to intelligent vehicles', *Autonomous Robots*, Vol. 19, No. 2, pp.159–171.

Latombe, J.C. (1991) *Robot Motion Planning*, Kluwer, Boston.

Murphy, R.R. (2000) *Introduction to AI Robotics*, The MIT Press, Cambridge.

Moravec, H. and Elfes, A. (1985) 'High resolution maps from wide angle sonar', *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, pp.116–121.

Schulz, D., Burgard, W., Fox, D. and Cremers, A.B. (2003) 'People tracking with mobile robots using sample-based joint probabilistic data association filters', *The International Journal of Robotics Research*, Vol. 22, No. 2, pp.99–116.

Shiller, Z., Large, F. and Sekhavat, S. (2001) 'Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories', *Proceedings of 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp.3716–3721.

Siegwart, R. and Nourbakhsh, I. (2004) *Introduction to Autonomous Mobile Robots*, The MIT Press, Cambridge.

Thrun, S., Burgard, W. and Fox, D. (2005) *Probabilistic Robotics*, The MIT Press, Cambridge.

Wang, C.C., Thorpe, C., Thrun, S., Hebert, M. and Durrant-Whyte, H. (2007) 'Simultaneous localization, mapping and moving object tracking', *International Journal of Robotics Research*, Vol. 26, No. 9, pp.889–916.