[BACK TO INDEX]

Proceedings of the *1999 ANS Conference on Robotics and Remote Systems*, Pittsburgh, PA, April '99

DOUBLE-VFH: RELIABLE OBSTACLE AVOIDANCE FOR LARGE, NON-POINT, OMNI-DIRECTIONAL MOBILE ROBOTS

Hong Yang[1], Johann Borenstein[1], David Wehe[2]
[1] Department of Mechanical Engineering and Applied Mechanics
[2] Department of Nuclear Engineering and Radiological Sciences
The University of Michigan
Ann Arbor, MI 48109-1101,USA
yanghong@umich.edu, johannb@umich.edu*, dkw@umich.edu
*Corresponding author

ABSTRACT

This paper presents a method for reliable real-time obstacle avoidance for a tele-autonomous omnidirectional mobile robot, called "OmniMate." The OmniMate is a large, rectangular-footprint platform with a flat loading deck measuring 184 ×92 cm (72×36").

Implemented with a shared control architecture, the OmniMate can operate in a so-called *tele-autonomous* mode that allows the operator to prescribe a general direction of motion even without visual contact. If the OmniMate encounters an obstacle, it autonomously avoids collision with that obstacle while trying to match the prescribed direction as closely as possible. Tele-autonomous operation is based on the newly developed *Double-VFH* (DVFH) method, which is a modified version of our earlier Vector Field Histogram (VFH) method. The DVFH method accounts for the rectangular footprint of the OmniMate by calculating simultaneously two *vector field histograms* centered at two points along the longitudinal axis of the OmniMate, based on the environment information dynamically derived from a ring of 32 ultrasonic sensors. Experimental results of the OmniMate traversing a densely cluttered obstacle courses at an average speed of 0.2 - 0.3 m/sec and a maximum speed of 0.5 m/sec are also presented.

1. INTRODUCTION

The OmniMate, shown in Figure 1, is a rectangular-shaped mobile platform measuring 184×92 cm (72×36") with three Degrees-of-Freedom of motion. The OmniMate is made from two "trucks" that are connected by a so-called compliant linkage and a flat loading deck. Each truck can rotate relative to the loading deck and the compliant linkage allows linear relative motion between the two trucks. The complete description of the kinematic design and control of the OmniMate is given in [Borenstein and Evans, 1997].

Figure 1: OmniMate with ultrasonic sensors for fully omnidirectional obstacle avoidance.

The *tele-autonomous* system of the OmniMate, which combines autonomous obstacle avoidance with tele-operation, has a *shared control* architecture [Borenstein and Koren, 1990]. In the shard control system of the OmniMate a complex task is divided between the human operator and the mobile robot. There are two levels of control which are smoothly integrated: The global control task of guiding the

robot to a target is performed by the operator, in the external loop, as shown in Figure 2, while the local control task of guiding the OmniMate around unexpected obstacles is performed in the internal loop by the robot's onboard computer.

The sampling time of the internal loop is T_I = 100 ms for the current system, but a faster sampling time of 50 ms or better could be easily attained through some optimization. The sampling period of the outer loop is set to $T_o = 2T_I$; this value is not critical.

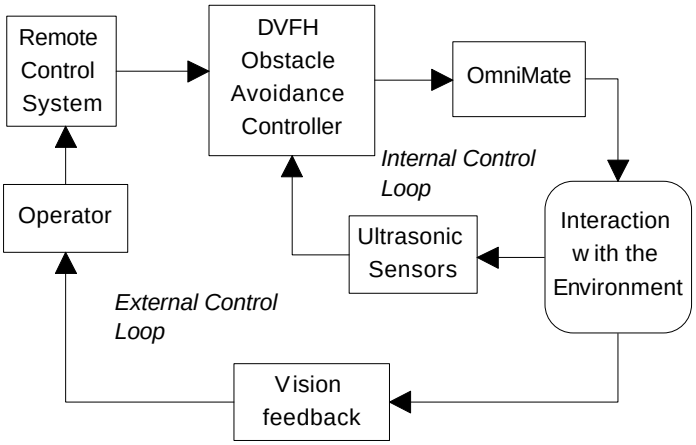

Figure 2: Shared control architecture of our tele-autonomous system.

The central part of this tele-autonomous system is the autonomous obstacle avoidance algorithm. Earlier obstacle avoidance methods developed at the University of Michigan, such as the *Virtual Force Field (VFF)* method and the *Vector Field Histogram* (VFH) method, have already been used for tele-autonomous control [Borenstein and Koren, 1990]. But neither method can be readily applied to the OmniMate because of that platform's elongated, rectangular shape, which is also referred to as a *non-point* feature. It is quite difficult to provide a non-point vehicle with adequate all-around protection from collisions with unexpected obstacles.

The Configuration Space Approach [Lozano-Perez, 1987], works well with relatively small, circular-footprint mobile robots but it is too conservative to deal with larger, non-circular platforms because it doesn't take into account the robot's orientation. The Voronoi Diagram [Barraquand, 1992], applies global path planning techniques to optimize the orientation of an irregular shaped mobile robot relative to the obstacles in the environment, but this method is not quite suitable for real-time application in dynamic environments.

The solution presented in this paper is called the Double-VFH (DVHF) method, and it is designed to work with rectangular footprint platforms with omnidirectional motion capabilities, like the OmniMate. The DVFH method is based on the *vector field histogram* (VFH) method, which was originally developed by Borenstein and Koren [1992] for circular-footprint robots. A review of that method and some other related methods is given in Section 2. Then, in Section 3, we introduce the new Double-VFH (DVHF) method in detail. The complete experimental system is described in Section 4 and experimental results are presented in Section 5.

2. EARLIER WORK

In earlier research at the University of Michigan several related obstacle avoidance methods were developed. In this section we will briefly review these earlier methods in order to document the evolution toward the DVFH method, which is the focus of this paper.

2.1 The Virtual Force Field (VFF) Method

Borenstein and Koren [1989] developed one of the earliest real-time obstacle avoidance methods, called the *virtual force field* (VFF) method. The VFF method worked in real-time and with actual sensory data, allowing a mobile robot to traverse an obstacle course at average speeds of 0.4-0.6 m/s.

The VFF method is specifically designed to accommodate and compensate for inaccurate range readings from ultrasonic or other sensors. To do so, the VFF method uses a two-dimensional Cartesian grid, called the *histogram grid* ***C,*** to represent data from ultrasonic (or other) range sensors. Each cell *(i, j)* in the *histogram grid* holds a *certainty value* (CV) $c_{i,j}$ that represents the confidence of the algorithm in the existence of an obstacle at that location. This representation was derived from the *certainty grid* concept originally developed by Moravec and Elfes, [1985]. In the *histogram grid*, CVs are incremented when the range reading from an ultrasonic sensor indicates the presence of an object at that cell.

Combining the *histogram grid* (as the world model) with the potential field concept, the VFF method allows to immediately use real-time sensor information to generate repulsive force fields. Figure 3 illustrates this approach: As the vehicle moves, a square "*window*" accompanies it, overlying a region of ***C***. We call this region the "*active region*" (denoted as ***C****), and cells that momentarily belong to the *active region* are called "*active cells*" (denoted as $c^*_{i,j}$). In the original implementation the size of the window is 33×33 cells (with a cell size of 10×10 cm), and the window is always centered about the robot's position.

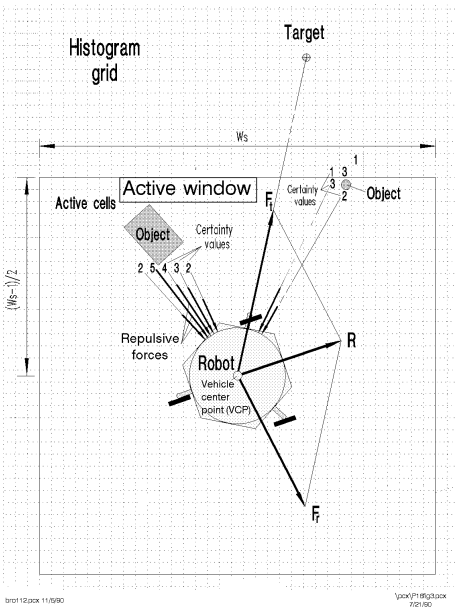

Figure 3: The Virtual Force Field (VFF) concept: Occupied cells exert repulsive forces onto the robot, while the target applies an attractive force.

Each *active cell* exerts a virtual repulsive force $\boldsymbol{F}_{i,j}$ toward the robot. The magnitude of this force is proportional to $c^*_{i,j}$ and inversely proportional to d^n, where d is the distance between the cell and the center of the vehicle, and n is a positive number (usually, n=2). All virtual repulsive forces add up to yield the resultant repulsive force $\boldsymbol{F}_r$. Simultaneously, a virtual attractive force $\boldsymbol{F}_t$ of constant magnitude is applied to the vehicle, "pulling" it toward the target. Summation of $\boldsymbol{F}_r$ and $\boldsymbol{F}_t$ yields the *resultant force vector* ***R***. The direction of ***R*** is used as the reference for the robot's steering command.

In the course of our experimental work with the VFF algorithm we found that the potential field approach caused oscillations in the presence of obstacles, especially in narrow passages.

2.2 The Vector Field Histogram (VFH)

To overcome these problems, Borenstein and Koren [1991] developed the *vector field histogram* (VFH) method. The VFH method builds the *histogram grid* the same way the VFF method does. However, the VFH method then introduces an *intermediate data-representation*, called the *polar histogram*. The *polar histogram* retains the statistical information of the *histogram grid* (to compensate for the inaccuracies of the ultrasonic sensors), but reduces the amount of data that needs to be handled in real-time. This way, the VFH algorithm produces a sufficiently detailed spatial representation of the robot's environment for travel among densely cluttered obstacles, without compromising the system's real-time performance.

2.2.1 Creating the Polar Histogram

The *polar histogram H* is an array comprising 72 elements; each element represents a 5°-sector of the robot's surroundings. During each sampling interval, the *active region* of the *histogram grid C** is mapped onto *H* as shown in Figure 4, resulting in 72 values that can be interpreted as the *instantaneous polar obstacle density* around the robot. Figure 5 shows two representations of the same sample polar histogram. In Figure 5a the polar histogram is plotted as a bar chart, with bars of different heights indicating the *polar obstacle density* in different directions. In Figure 5b is polar histogram is overlaying the *histogram grid* from which it was created by the mapping process of Figure 4. The size of each blob in Figure 5b represents the certainty value of the corresponding cell in the histogram grid.

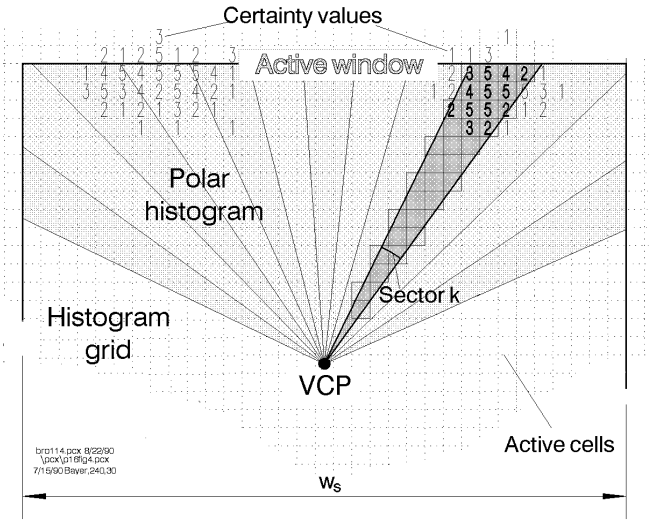

Figure 4: Mapping the histogram grid onto the polar histogram.

2.2.2 Computing the Steering Control

After the *polar histogram* has been constructed, the VFH algorithm computes the required steering direction for the robot. As can be seen in Figure 5, a *polar histogram* typically has peaks (sectors with high *obstacle density*), and valleys (sectors with low *obstacle density*). Any valley with *obstacle densities* below threshold is a candidate for travel. Since there are usually several candidate-valleys, the algorithm selects the one that most closely matches the direction to the target.

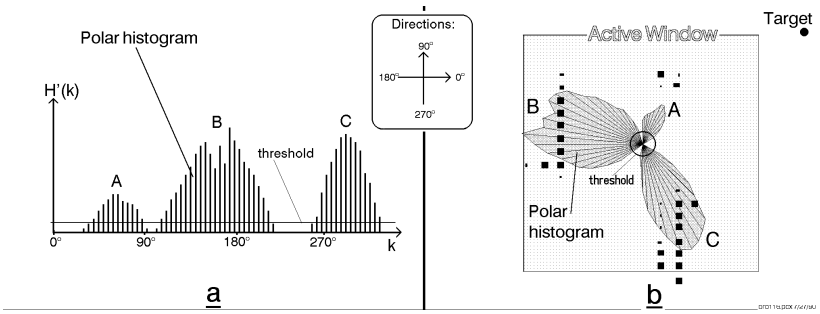

Figure 5: Snapshot of a momentary polar histogram for a sample obstacle course with three obstacles A, B, and C.

2.2.3 Tele-autonomous control

If the vehicle is to follow the joystick commands of a remote human operator, then this can be done with a very simple modification. Instead of selecting a candidate valley based on the direction to the target (see last sentence in the proceeding Section 2.2.2) a candidate valley is selected that is closest to the operator-prescribed direction.

2.3 The Combined Vector Field (CVF) Method

As a first attempt at avoiding obstacles with large, rectangular-shaped mobile robots Borenstein and Raschke [1991] developed the *combined vector field* (CVF) method, which combines the VFF and VFH method. With this method, the principal steering direction of the non-point robot is determined by the VFH algorithm, since stable motion and better spatial resolution are the strength of the VFH method; while the VFF algorithm applies virtual forces as a corrective measure to account for the robot's dimension.

The general approach of the CVF method is as follows:

1. Computing the principal steering direction with the VFH method – The task of the VFH component is to find the openings through which the robot should travel to approach the target. As illustrated in Borenstein and Raschke [1991], the VFH algorithm is applied at a point CP to determine the principal steering direction. A vector $\boldsymbol{F}_{\mathrm{VFH}}$ is computed and applied in that direction. CP is located near the center point of the vehicle on the longitudinal axis.
2. Local protection with the VFF method – The VFF method is applied to each one of the n act-on points on the periphery of the robot. This is done for each act-on point by adding up all individual repulsive forces $\boldsymbol{F}_{\mathrm{ij}}$ from filled cells in the histogram grid, yielding n repulsive force $\boldsymbol{F}_{\mathrm{VFF.}}$ The force fields that act on each act-on point are very steep. They are generated by $\boldsymbol{F}_{\mathrm{ij}} \propto 1/d^5$. So, the effective range is very short in order to avoid the oscillation effect of VFF.
3. Computing the corrected steering vector – The kinematic control program of the OmniMate is designed to accept translation velocities and orientation commands separately. The translation command is the summation $\boldsymbol{V} = a\boldsymbol{F}_{\mathrm{VFH}} + b\boldsymbol{F}_{\mathrm{VFF}}$, where the coefficients *a* and *b* are determined empirically so as to avoid the oscillatory behavior usually associated with potential field control.

The orientation command is the sum of moments $\boldsymbol{W} = c\boldsymbol{W}_{\mathrm{VFH}} + d\boldsymbol{W}_{\mathrm{VFF}}$, where $\boldsymbol{W}_{\mathrm{VFH}}$ is computed from the forces on each so-called act-on point, and $\boldsymbol{W}_{\mathrm{VFF}}$ is the angular difference between the vehicle's current orientation and the VFH ou t-put. Coefficients *c* and *d* help avoid any situation where the robot straddles an opening and generates conflicting moments from the act-on points to either side.

Although the CVF method used only steep force profiles with short-range effects in its VFF components, the oscillation tendency of this potential fields-based method was still observed in environments with densely spaced obstacles.

3. THE DOUBLE-VFH (DVFH) METHOD

This paper introduces the Double-VFH (DVFH) method, a modification of the original VFH method for the elongated, rectangular-shaped OmniMate mobile robot with three DOF. DVFH method easily solves the oscillatory tendency, straddling-of-an-obstacle and Ladder-in-door problem of previous CVF method [Holt et al., 1996].

As explained in Section 1, the OmniMate is based on two trucks, which are kinematically simple differential-drive vehicles. The DVFH algorithm treats these two trucks identically, that is, either one may lead or follow. Two basic behaviors need to be defined:

- Truck A serves as *Leader*, Truck B serves as *Follower*;
- Truck B serves as *Leader*, Truck A serves as *Follower*;

The motion mode generated by the DVFH method is similar to a *Follow-the-Leader* approach if no obstacle is encountered. However, there is an important distinction between these two apparently similar modes: While the *Follow-the-Leader* mode always results in the rear truck following precisely along the trajectory of the front truck, in the DVFH mode, the two trucks determine their own direction of travel in parallel and independently. This is done according to the following general approach:

1) A separate polar histogram is built around the two *center points* O_A and O_B, which are located along the longitudinal axes of the robot (as shown in Figure 6). The optimal location of O_A and O_B depends on the dimension of the robot and its nominal travelling speed, and these positions should be determined experimentally.
2) Using the original VFH method as described in Sect. 2.2.2, candidate direction vectors $\boldsymbol{V}_A$ and $\boldsymbol{V}_B$ are calculated separately for each polar histogram centered at O_A and O_B.
3) A *Cost-Based Arbitration* scheme (discussed below) is applied to resolve conflicts between the separately determined $\boldsymbol{V}_A$ and $\boldsymbol{V}_B$ in order to find a set of realizable directions, which result in the desired trajectory.
4) Using kinematic analysis and synthesis, steering directions for each truck are then derived, and motion control commands for all four drive wheels are generated.

This procedure is visualized in Figure 7 and will be explained in detail in the following sections.

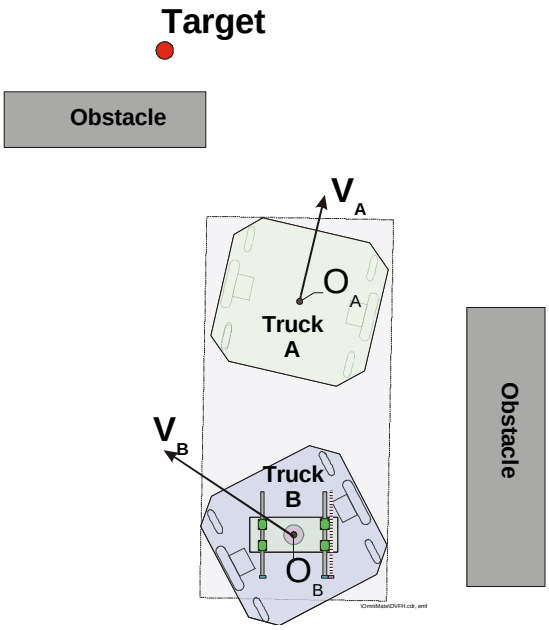

Figure 6: DVFH selects directions for two center points located along the longitudinal axis of the robot.

3.1 Cost-based Arbitration

When separate directions for travel are computed independently for each of the two trucks, it is quite likely that these directions conflict (i.e., when each truck. has a different translational speed component along the longitudinal vehicle axis). We resolve such a conflict using a process called *cost-based arbitration*.

As an example consider a situation where truck A functions as the leader. Truck A should have high priority and it should determine the principal steering direction for the whole robot, while imposing certain constraints on truck B.

Truck B, which is forced to be the follower when truck A leads, should act with a tendency to align itself with the orientation of the robot. This minimizes the conflict between A and B and helps recover from any deviation from the nominal extension of the compliant linkage in the OmniMate.

The *cost function* $W(\Delta\alpha_1, \Delta\alpha_2, \Delta\alpha_3)$ of the cost-based arbitration process is given as:

$$W = k_1\Delta\alpha + k_2\Delta\alpha_2 + k_3\Delta\alpha_3 \qquad (1)$$

Where:

$\Delta\alpha_1$ — difference between the candidate steering direction of the leading truck and the current orientation of the robot.

$\Delta\alpha_2$ — difference between the candidate steering direction of the following truck and the current orientation of the robot.

$\Delta\alpha_3$ — difference between the candidate steering direction of the robot center point and the previous one from the last sampling interval.

k_1, k_2, k_3 — weighting coefficients.

This cost function is used by the DVFH algorithm to determine for each truck its motion mode: either *leader* or *follower*. The larger the first term in the cost function *W*, the larger the tendency for this truck to switch from leader mode to follower mode. The larger the second term in the cost function, the larger the tendency to maintain the current motion mode unchanged. The third term determines the tendency for the robot to deviate from the original path when an obstacle is encountered. The larger this term, the larger the tendency to change the motion mode of two trucks.

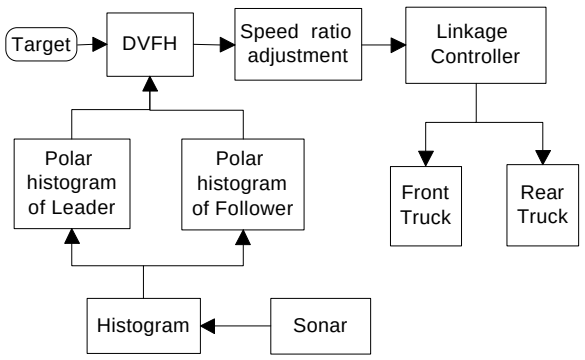

Figure 7: Block diagram of the DVFH obstacle avoidance system for the OmniMate.

The motion mode decision-making procedure is shown in Figure **8**. Based on the odometry and ultrasonic sensor information, the direction to the target and the orientation of the vehicle are determined. Also, the candidate steering direction vectors $\boldsymbol{V}_A$ and $\boldsymbol{V}_B$ of points O_A and O_B are pre-calculated. W_A, the cost value if truck A serves as leader and W_B, the cost value if truck B serves as leader are calculated by the cost function. By comparing W_A and W_B, the motion mode and priority levels are determined for both truck A and B.

When traveling among densely cluttered obstacles there are situations where Trucks A and B might switch roles of *leader* and *follower* rapidly and repeatedly. This problem is overcome by implementing a hysteresis function based on two thresholds, P_A and P_B (with $P_A < P_B$):

The resulting decision strategies are summarized below:

- If $W_A < W_B$ and $|W_A - W_B| < P_A$, then truck A will change to or remain as leader.
- If $W_A > W_B$ and $|W_A - W_B| > P_B$, then truck B will change to or remain as leader.
- In any other case both trucks remain in their current motion mode.
- Whenever one truck changes its motion mode, the other will make the change according to it.

After the motion mode is decided, the leader can select a candidate steering direction autonomously (as explained in Section 2.2.2). At the same time the follower must submit to restrictions on its choice of steering directions and speeds.

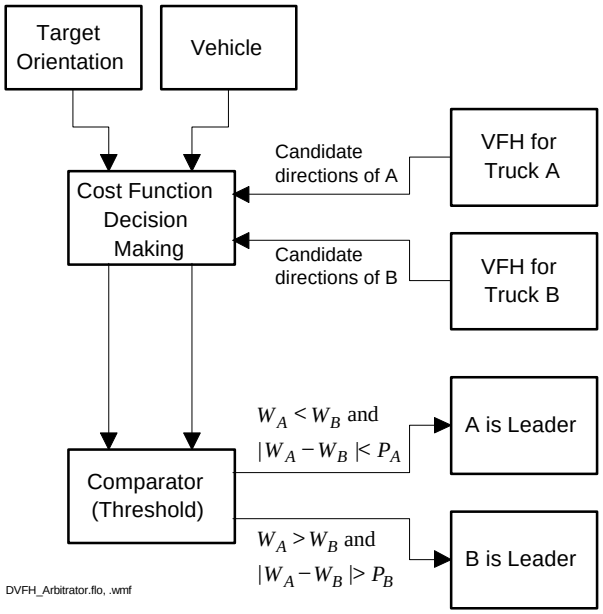

Figure 8: Motion mode decision making strategy

3.2 Instantaneous trajectory generation and steering control

After the steering velocity vectors $\boldsymbol{V}_A$ and $\boldsymbol{V}_B$, which are centered at the points O_A and O_B, are calculated by the DVFH method, the motion control commands for each wheel on both trucks can be derived based on the concept of *the instantaneous center of rotation* (ICR). Because of the Cost-based Arbitration $\boldsymbol{V}_A$ and $\boldsymbol{V}_B$ always have the same magnitude of the translation velocity component along the longitudinal axis of the vehicle.

In a fast sampled system, smooth motion can be approximated by re-computing an ICR during each sampling interval. The ICR is constructed as the intersection between the two normals to the steering directions $\boldsymbol{V}_A$ and $\boldsymbol{V}_B$, and every point on the vehicle should rotate around the ICR at the specific speed at this instant, as illustrated in Figure 9, Point A and B are the center points of front and rear truck respectively. Then, the orientations of two differentially driven wheels of both trucks are set normal to the two position vectors *A_ICR* and *B_ICR*. These orientations of the drive-wheels will cause both trucks to rotate around the ICR and, consequently, points A and B will (momentarily) move in the required steering direction, which form the trajectory of the vehicle.

The truck orientations are calculated based on the location of the ICR. Both angles are specified as positive in the counter-clockwise (ccw) direction, with 0° corresponding to straightforward alignment of a truck and the long i-tudinal axis of the vehicle. α specifies the front truck orientation, and β specifies the rear truck orientation. L is the distance between point A and B, and x_{ICR} and y_{ICR} is the relative coordinate of the ICR specified with respect to the center point O of the robot frame.

$$a = -\tan^{-1}\left(\frac{L/2 - y_{ICR}}{x_{ICR}}\right) \qquad (2a)$$

$$b = \tan^{-1}\left(\frac{L/2 + y_{ICR}}{x_{ICR}}\right) \qquad (2b)$$

The wheel velocities are computed as tangential velocities to an arc from ICR to the centers of four wheels. The relative location of the ICR usually changes between sampling intervals, requiring instantaneous velocity changes for all drive motors.

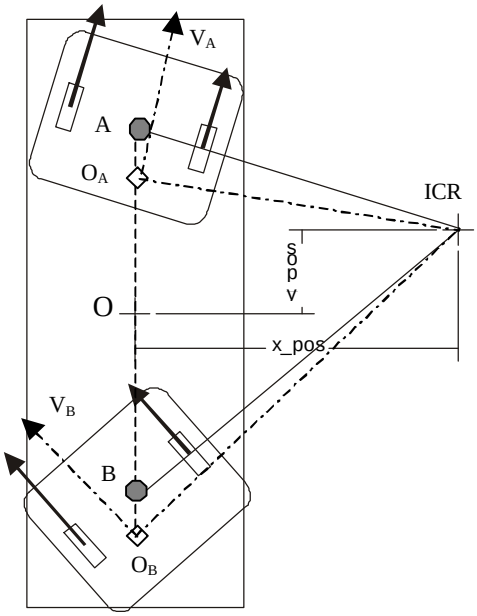

Figure 9: Deriving the motion control command according to the instantaneous center of rotation (ICR).

3.3 Special considerations

1. One special case exists in so-called trap situations that arise when obstacles block the leader's forward direction or when a sharp direction change is needed for the follower in order to clear a nearby obstacle. In this case the cost function *W* is calculated again and compared with the threshold, and the robot will have the tendency to switch the motion mode and drive backward to find a new candidate direction toward the target under the control of the newly decided leader. In this undesirable case a more efficient high-level path planner is needed to solve the local minimum problem.
2. Another special case is an environment with very densely spaced obstacles. In such environments, because of the dynamic restriction of the system, it is often necessary to reduce the overall speed of the platform to let the trucks assume new orientations.
3. The robot's maximum speed V_{max} can be set at the beginning of a run. The robot tries to maintain this speed during the run unless forced by the DVFH algorithm to a lower instantaneous speed *V*, which is determined in each sampling interval as follows:

   Let θ be the desired rotation angle of the truck relative to the axis of the robot. A large value of θ means a large obstacle lies ahead or an obstacle is very close to the robot. Either case is likely to require a drastic change in direction. Let ω be the rate of this rotation angle. θ_{max} and ω_{max} are the maximum allowable steering angle and rotation rate respectively for the mobile robot. Then the reduced translation speed *V* is:

$$V = V_{max} \cdot \left(1 - K_1 \cdot \frac{q}{q_{max}}\right) \cdot \left(1 - K_2 \cdot \frac{w}{w_{max}}\right) + V_{min} \qquad (3)$$

Where K_1 and K_2 are empirically determined coefficients for speed reduction. The $(1 - \omega/\omega_{max})$ term reduces the speed of the robot in anticipation of a steering angle change. Note that V is prevented from going to zero by setting a lower limit i.e. $V \geq V_{min}$.

4. EXPERIMENTAL RESULTS

The obstacle avoidance system of the OmniMate comprises a ring of 32 Polaroid ultrasonic sensors, an electronic interface board equipped with four 68HC11 micro-controllers and software that runs on the OmniMate onboard PC. The sonar ring is mounted underneath the loading deck and along the periphery of the OmniMate (as was shown in Figure 1). The electronic interface board implements the earlier developed method for Error Eliminating Rapid Ultrasonic Firing (EERUF) [Borenstein and Koren, 1995]. The EERUF method allows each sonar to detect and reject readings caused by crosstalk or other ultrasonic noise in the environment. This feature allows for much faster firing rates than those used in conventional sonar-based obstacle avoidance systems, yet provides more reliable and nearly noise-free data. The obstacle avoidance system for the OmniMate provides fully directional protection in low- to medium-cluttered environments at a maximum speed of 0.4 m/sec.

The *DVFH* obstacle avoidance method proposed in this paper was tested on the OmniMate in the Mobile Robotics Lab of the University of Michigan. In typical runs through densely cluttered environments the OmniMate would travel at a maximum speed of 0.4 m/sec. The average speed in such runs was lower, about 0.2 - 0.3 m/sec, because the robot slowed down whenever it came close to obstacles. Higher maximum speeds were possible in less densely cluttered environments.

5. CONCLUSIONS

This paper presented the DVFH obstacle avoidance method for the tele-autonomous operation of OmniMate robot. The DVFH method was extensively tested on the OmniMate and found to provide smooth and robust control. The DVFH algorithm is computational efficient and insensitive to misreading, it allows continuous motion of the mobile robot without stopping for obstacles. The DVFH-controlled mobile robot traverses very densely cluttered obstacle courses and is able to pass through narrow openings without oscillation as compared with the *CVF* method.

However, the DVFH method as described in this paper provides only one mode of motion with obstacle avoidance, which, in practice, is similar to a follow-the-leader approach. Yet, the OmniMate is capable of full 3 DOF motion, including sideways crabbing or diagonal motion. We are currently developing other versions of DVHF that will provide full obstacle avoidance for other types of motion.

Acknowledgements

This material is based upon work supported by DOE under Award No. DE-FG04-86NE37969.

6. REFERENCES

1. Barraquand, J., Langloi, B., and Latombe, J.C., 1992, “Numerical Potential Field Techniques for Robot Path Pla n-ning.” *IEEE Trans. on Systems, Man, and Cybernetics*, Vol.22, No.2, pp. 224-241.
2. 15 Borenstein, J. and Koren, Y., 1990, “Tele-autonomous Guidance for Mobile Robots.” *IEEE Transactions on Systems, Man, and Cybernetics, Special issue on unmanned systems and vehicles*, Vol. 20, No. 6, Nov/Dec, pp. 1437-1443.
3. Borenstein, J. and Koren, Y., 1991, “The Vector Field Histogram Fast Obstacle Avoidance for Mobile Robots.” *IEEE Journal of Robotics and Automation*, Vol. 7, No. 3, pp. 278-288.
4. Borenstein, J. and Raschke, U., 1992, “Real-time Obstacle Avoidance for Non-Point Mobile Robots.” *SME Trans. on Robotics Research*, Vol. 2, pp. 2.1-2.10.
5. Borenstein, J. and Koren, Y., 1995, “Error Eliminating Rapid Ultrasonic Firing for Mobile Robot Obstacle Avoidance.” *IEEE Trans. on Robotics and Automation*, Feb., Vol. 11, No. 1, pp. 132-138.

6. Borenstein, J., 1995, “Control and Kinematic Design of Multi-Degree-of-Freedom Mobile Robots with Compliant Linkage.” *IEEE Trans. on Robotics and Automation*, Vol. 11, No. 1, pp.21-35.

7. Borenstein, J. and Koren, Y, 1995, “Error Eliminating Rapid Ultrasonic Firing for Mobile Robot Obstacle Avoidance.” *IEEE Trans. on Robotics and Automation*, February, Vol. 11, No. 1, pp. 132-138.

8. Borenstein, J. and Evans, J., 1997, “The OmniMate Mobile Robot – Design, Implementation, and Experimental R e-sults.” *Proc. of the 1997 IEEE Int. Conference on Robotics and Automation*, Albuquerque, NM, Apr. 21-27, pp. 3505-3510.

9. Elfes, A., 1987, “Sonar-Based Real-World Mapping and Navigation.” *IEEE Trans. on Robotics and Automation*, RA-3, No. 3, pp. 249-265.

10. Holt, B., Borenstein, J., Koren, Y., and Wehe, D.K., 1996, “OmniNav: Obstacle Avoidance for Large, Non-circular, Omnidirectional Mobile Robots.” *Robotics and Manufacturing*, Vol. 6 (ISRAM 1996), Montpellier, France, May 27-30, pp. 311-317.

11. Gourley, C. and Trivedi, M., 1994, “Sensor Based Obstacle Avoidance and Mapping for Fast Mobile Robots.” *Proc. of the 1994 IEEE Int. Conference on Robotics and Automation*, San Diego, CA, May 8-13, pp. 1306-1311.

12. Gourley, C. et. al. 1994, “An Integrated Mobile Robotic System for Environmental Mapping and Object Localization.” *Proc. of the SPIE Symposium on Optical Eng. Aerospace Sensing*, Orlando, FL.

13. Desai, J. and Wang, C.C., 1996, “Motion Planning for Multiple Mobile Manipulators.” *Proc. of the 1996 IEEE Int. Conference on Robotics and Automation*, Minneapolis, MN.

14. Lozano-Perez, 1987, “A simple Motion Planning Algorithm for General Robot Manipulators.” *IEEE Trans. on Robotics and Automation*. Vol. RA-3, No.3, pp. 224-238

15. Moravec, H. P. and Elfes, A., 1985, “High Resolution Maps from Wide Angle Sonar.” *Proc. of the IEEE Conference on Robotics and Automation*, Washington, D.C., pp. 116-121.

16. Samson, C., 1993, “Time-varying Feedback Stabilization of Car-like Wheeled Mobile Robots.” *The International Journal of Robotics Research*, Vol.12, No. 1, pp. 55-64.