WOAH: An Obstacle Avoidance Technique for High Speed Path Following

Nat Tuck, Michael McGuinness, and Fred Martin*
University of Massachusetts Lowell, 1 University Avenue, Lowell, MA, USA 01854

ABSTRACT

This paper presents WOAH, a method for real-time mobile robot path following and obstacle avoidance. WOAH provides reactive speed and turn instructions based on obstacle information sensed by a laser range finder. Unlike many previous techniques, this method allows a robot to move quickly past obstacles that are not directly in its path, avoiding slowdowns in path following encountered by previous obstacle avoidance techniques.

Keywords: robotics, laser rangefinder, obstacle avoidance, reactive, path following

1. INTRODUCTION

Consider the problem of a robot autonomously driving to a goal point on flat terrain. Assume the robot always knows its position and what direction it's facing. If there are no obstacles, this is simple: the robot can just drive towards the goal until it gets there. When obstacles are added, this technique must be modified slightly. As long as none of the obstacles form traps, or configurations where the robot would need to move away from the goal in order to get there, it is sufficient for the robot to simply drive in the direction that will allow it to make the most progress towards the goal without hitting any obstacles.

Eliminating traps is a well understood problem. The combination of a mapping and pathfinding technique, such as an occupancy grid and A* search,[1] can be used to create a sequence of intermediate waypoints such that no traps exist between adjacent waypoints. Once such a path has been generated, the problem of reaching the goal has been reduced to the problem of traveling to the next waypoint.

In order to solve this problem, we have developed WOAH, a Working Obstacle Avoidance Heuristic. WOAH determines, based on instantaneous sensor readings, a direction the robot should turn, how rapid the turn should be, and how quickly the robot can safely move forward. In this paper, we present the algorithm behind WOAH and show that WOAH provides excellent performance compared to previous techniques for a four wheeled differential drive robot with one forward mounted LIDAR scanner .

1.1 Inspiration

The development of WOAH was inspired by the AUVSI Intelligent Ground Vehicles Competition Navigation Challenge. In this challenge, a competing robot must navigate autonomously through an unknown set of static obstacles to nine GPS waypoints on a 55 by 65 meter course. Whichever robot hits the most waypoints the fastest wins.

In the 2009 IGVC competition we used Nearness Diagram[2] for obstacle avoidance. This worked reasonably well, but for the 2010 competition we wanted better performance. After evaluating various options, we concluded that the existing solutions were too timid to provide the performance that we wanted. An early version of the algorithm we present in this paper was used on our two entries at the 2010 IGVC, achieving 3rd and 11th place in a field of 22.

*{ntuck, mmcguinn, fredm}@cs.uml.edu

2. RELATED WORK

2.1 Vector Field Histogram (VFH)

One approach to robot obstacle avoidance that appears relatively early in the literature is the use of potential fields. The idea is that if you sum a field that pulls the robot towards its goal with fields that push it away from obstacles, the result will be a direction that the robot can move to reach the goal without hitting anything. This approach is well represented by the Vector Field Histogram (VFH)[3] family of algorithms, of which VFH+[4] is a modern example.

This approach has two basic weaknesses. First, the combination of fields can produce local minima where the force from the goal is exactly balanced by the forces from obstacles, which will cause the robot to get stuck. Second, the nature of the technique will cause the robot to veer away from obstacles even when there is no risk of collision, which will result in the robot taking sub-optimal paths. Both of these weaknesses can be worked around to some extent by properly selecting configuration constants, but finding a set of constants that will work well in general is difficult.

2.2 Nearness Diagram (ND)

Nearness Diagram[2] (ND) is an approach to obstacle avoidance that was developed to provide better performance in cluttered environments, where other methods like potential fields tend to run into problems. ND operates by dividing the situations it expects to encounter into five specific cases and providing appropriate behavior for each of these cases. This results in reasonable performance in a variety of different situations, including narrow corridors and cluttered areas. Smooth Nearness Diagram, or SND, is a newer development based on ND designed to produce less jerky movement. SND has been simplified from its predecessors to operate on a single law of motion as well as a single non-physical configuration parameter.

2.3 Open Path

Open Path[5] is a procedure for processing laser range data to find directions that are open paths for the robot to travel along. The robot then moves in the direction with the largest open area. This algorithm is not a path following solution as described, but the analysis of LIDAR data is very similar to what is described in this paper such that many of the optimization techniques used in Open Path should be directly applicable to WOAH.

2.4 Search-based Approaches

For high budget projects where more complicated sensor packages are a feasible option, a common approach is to solve obstacle avoidance by treating finding the best local path as a search problem. For example, the Carnegie Mellon University entries in the DARPA Grand Challenge[6] ran A* search at 20Hz on a set of probable local paths. This technique required multiple LIDAR units and a dedicated processor to run the search, which puts it well out of the budget range of many projects.

3. DESCRIPTION

3.1 How WOAH Works

WOAH operates reactively to calculate instantaneous speed and turn rate given the current available sensor readings. The intent is to select the speed and turn rate that will cause the robot to make the fastest progress towards the goal point without hitting anything. WOAH consists of a mathematical function mapping sensor inputs and desired location to appropriate the forward and rotation speeds, as shown in the Scheme code below. This procedure is run frequently, ten times per second in our testing, and the resulting forward and rotation speeds are used to calculate the motor commands to be applied until the next iteration.

First, WOAH makes sure the robot is facing generally towards the next waypoint. In order to make progress towards the waypoint, the angle to the goal must be less than 90 degrees. If the angle is greater than that, the robot will turn in place towards the waypoint.

If the robot is facing within 90 degrees of the next waypoint, the next step is to find the direction that will allow the robot to make the most rapid progress towards that goal. WOAH finds this direction by searching the space scanned by the LIDAR for the best open corridor. A corridor is a rectangle projecting from the front of the robot that is wide enough for the robot to safely travel along (the width of the robot plus a safety margin, as illustrated in Figure 1). WOAH considers each corridor distinguishable by the LIDAR: one for each LIDAR sample, except for those samples too close to the edge of the LIDAR arc to be the center of a full corridor. For each corridor, the amount of progress that can be made along that corridor is calculated by finding the nearest LIDAR reading in that corridor.

To find the nearest LIDAR reading in a corridor, we take the minimum of readings that fall within the corridor. There are three possibilities for each ray:

1. The ray is within the corridor rectangle for its entire range. In this case this sample is always a candidate for nearest reading.
2. The ray exits the rectangle on the left. In this case the sample is only a candidate for nearest reading if it occurs before the ray exits the rectangle.
3. The ray exits the rectangle on the right. This is symmetric with case 2.

The following formula will calculate the intersection distance $d(w, \alpha, \theta)$ in all three cases. The arguments to this function are the width of the corridor (w), the angle of the center of the corridor (α), and the angle of the laser ray (θ). If the ray exits the rectangle on either side, this gives the appropriate distance. If it does not exit the rectangle, it will give a very large positive value. This means that any sample distances greater than $d(w, \alpha, \theta)$ do not occur within the corridor.

$$d(w,\alpha,\theta) = \begin{cases} \frac{w \sin(\frac{\pi}{2} - \alpha)}{2 \sin(\alpha - \theta)} & \theta \le \alpha \\ d(w, -\alpha, -\theta) & \theta > \alpha \end{cases}$$

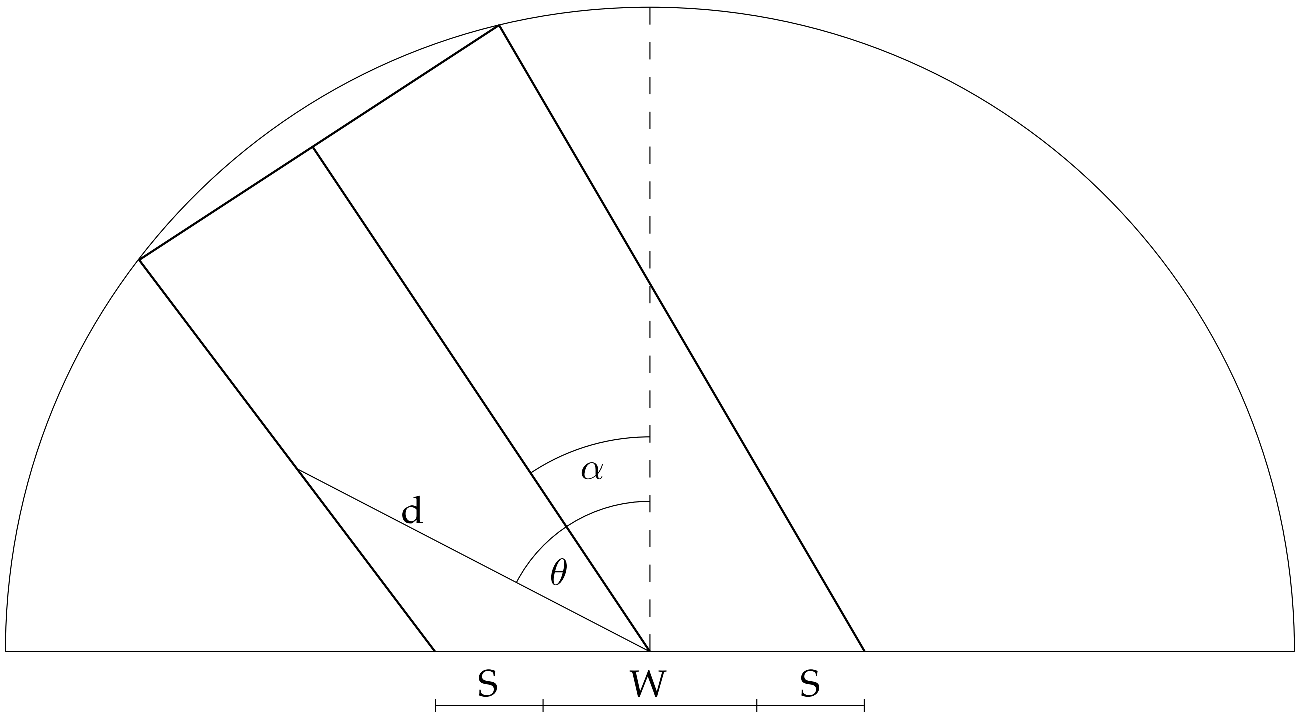

Figure 1. Geometry of a Corridor.

Given the length of a clear corridor in front of the robot, the possible progress that can be made by following that corridor can be calculated by comparing the current distance from the robot to the waypoint to the distance from the end of the corridor to the waypoint. The width of a corridor is the width of the robot plus a user specified safety margin. The best corridor is selected by picking the corridor that makes the most progress towards the goal. WOAH will produce a turn speed that will turn the robot towards this corridor.

Once the turn speed has been determined, the next step is to calculate the robot's forward speed. Since we are facing less than 90 degrees from the goal, we know that the robot will make progress by going forward. Therefore, we would like the robot to go full speed unless that would cause it to hit an obstacle. WOAH picks a speed by considering an arc consisting of the intersections of all the corridors from the corridor at its current facing to the corridor at the best direction determined earlier. The speed chosen is either the maximum speed or the speed that will cause the robot to move to the user specified safety margin from the nearest obstacle in the user specified minimum impact time.

In addition to safety margin and minimum impact time, WOAH has three other non-physical parameters. Turn intensity is used to generate a scaling factor for turn speed using the expression:

$$(\frac{|\alpha|}{\pi})^{1/TI}$$

In effect, the larger turn intensity is, the larger the turn rate will be. This is primarily for dealing with variable control loop times and oscillatory behavior; a value of 1.7 works well with our 10Hz control loop. Turn resistance causes WOAH to favor corridors near its current facing over ones further away, using a similar scaling method and the expression:

$$(\cos|goal\alpha - \alpha|)^{TR}$$

Where $\cos|goal\alpha - \alpha|$ is the normalized value for progress towards a goal at some alpha. Its primary function is to reduce fighting over similar quality corridors by introducing a cost to turning. The final parameter, extra margin, extends the safety margin around the robot slightly when selecting corridors. This alleviates slow sharp turns due to quantization error.

3.2 Scheme Code

We have included the Scheme implementation of WOAH below. The **d** and **min-dist** functions correspond to the d function defined above and the minimum of all such d's in a corridor. The **forward-speed** function computes the maximum safe speed in a given corridor. If the goal-flag is set, it will approach the goal slowly as to meet it rather than running through it like a waypoint. **turn-speed** determines how fast the robot should turn to meet a corridor using the formula described above. The **goal-progress** function is used by the **find-direction** function to select the corridor that makes the best progress towards the goal. Finally, the **woah-forward** and **woah-backward** functions are called by the main **woah** function when the requested angle is in front of and behind the robot, respectively. It returns the recommended translational and rotation velocity commands.

```
(define (d w α θ)
  (if (> θ α)
     (d w (* -1.0 α) (* -1.0 θ))
     (*
      (/ (sin (- (/ π 2.0) α))
         (sin (- α θ)))
      (/ w 2.0))))


(define (min-dist w α-left α-right ranges)
  (define (corridor-distance θ-range-pair)
    (let ((θ (car θ-range-pair))
          (range (cdr θ-range-pair)))
      (cond
       ((< θ α-right)
        (if (< range (d w α-right θ))
            range
            +inf.0))
       ((and (> θ α-right) (< θ α-left))
        range)
       (else ;(> θ α-left)
        (if (< range (d w α-left θ))
            range
            +inf.0)))))
  (min-list (map corridor-distance ranges)))
```

```
(define (forward-speed w α goal-distance goal-flag ranges)
  (let* ((α-left  (max 0.0 α))
         (α-right (min 0.0 α))
         (safe-dist   (- (min-dist w α-left α-right ranges)
                         (* SAFETY_MARGIN 2)))
         (distance    (if goal-flag
                          (min goal-distance safe-dist)
                          safe-dist)))
    (min MAX_SPEED (/ distance MIN_IMPACT_TIME))))


(define (turn-speed α)
  (* (sign α)
     MAX_TURN_SPEED
     (expt (* (abs (/ α π)) 2.0) (/ 1.0 TURN_INTENSITY))))


(define (goal-progress w goal-α goal-distance α ranges)
  (* (min goal-distance (min-dist (+ EXTRA_MARGIN w) α α ranges))
     (real-part (expt (cos (abs (- goal-α α))) TURN_RESISTANCE))))


(define (find-direction w goal-α goal-distance ranges)
  (define (find-better-α α α-progress-pair)
    (let* ((old-progress (cdr α-progress-pair))
           (progress     (goal-progress w goal-α goal-distance α ranges)))
      (if (> progress old-progress)
          (cons α progress)
          α-progress-pair)))
  (let* ((angles (map car ranges))
         (best-pair (fold-left find-better-α (cons 0.0 -inf.0) angles))
         (best-α    (car best-pair))
         (best-prog (cdr best-pair)))
    best-α))


(define (woah-forward goal-α goal-distance goal-flag ranges)
  (let* ((w (+ ROBOT_WIDTH SAFETY_MARGIN))
         (α (find-direction w goal-α goal-distance ranges))
         (speed (forward-speed w α goal-distance goal-flag ranges))
         (turn  (turn-speed α)))
    (cons speed turn)))


(define (woah-backward goal-α)
  (let ((forward-speed 0.0)
        (turn-speed    (* (sign goal-α) MAX_TURN_SPEED)))
    (cons forward-speed turn-speed)))


(define (woah goal-α goal-distance goal-flag ranges)
  (if (< (abs goal-α) (/ π 2.0))
      (woah-forward goal-α goal-distance goal-flag ranges)
      (woah-backward goal-α)))
```

4. PERFORMANCE

WOAH was tested in simulation along with two other obstacle avoidance algorithms: SND and VFH+. We used the Player/Stage[7] system for control and simulation, an environment that makes it easy to switch between simulation and hardware with minimal, if any, changes. Both VFH+ and SND have existing implementations in Player, which we used with minor modifications. WOAH is implemented by using GNU Guile[†] to run the Scheme code in our Player client program.

The first two sets of tests were performed over 144 maps like the one shown in Figure 2. Each map has a unique permutation of the small obstacles in the North-West and South-East portions of the map. The simulated robot is rectangular and has differential drive. Two main sets of tests were run using these maps: static and dynamic. Static tests use a precomputed path for the robot while dynamic tests re-calculate the path as the

[†]http://www.gnu.org/software/guile, version 1.9

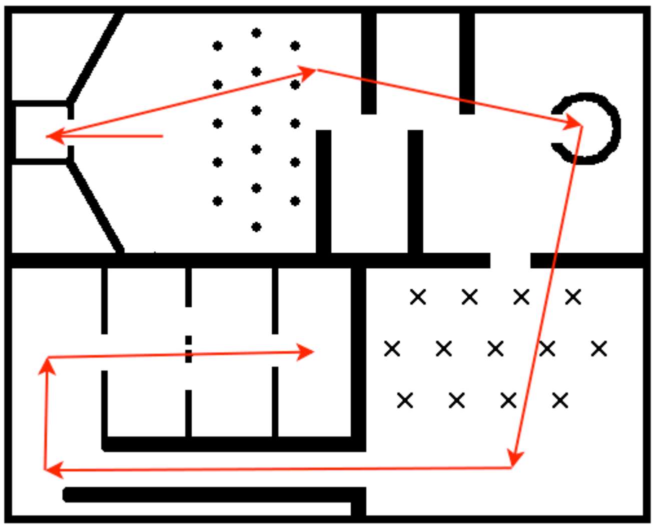

Figure 2. Test Map. Arrows indicate overall waypoints.

robot runs using a map built from sensor data. The last set of simulation tests were run using a test map similar to the IGVC navigation challenge. Finally, we ran a small number of tests on our robot, Stark. All tests assume a robot radius of 0.75 meters for pathfinding with an actual robot radius of about 0.63 meters (half the longest diagonal).

4.1 Static Testing

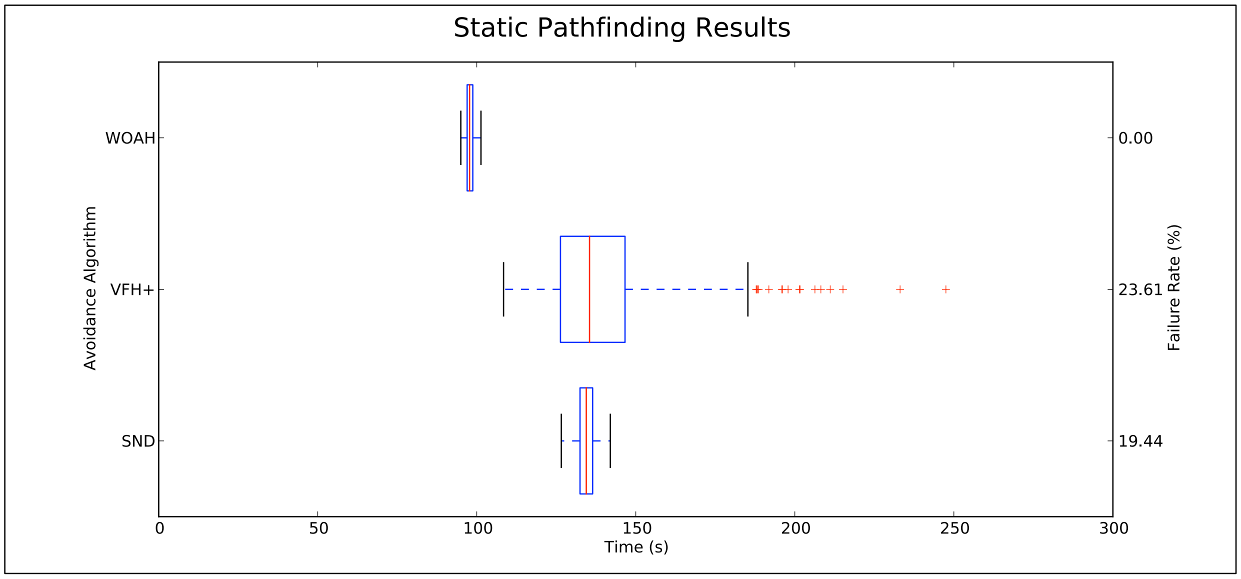

Figure 3. Plot of static test data; crosses mark points outside 1.5 inter-quartile range. Lower runtimes are better. Results show that WOAH performed most consistently and the best; VFH+ was most erratic.

Static tests were run following a static precomputed path created using the A* algorithm with full map knowledge. This test was run nine times per algorithm per test map.

The results of static testing are summarized by the box plot in Figure 3 and table in Figure 4. Navigating using static paths is often prone to failure because of its fixed nature; robots that veer off path in the course of obstacle avoidance can get stuck returning to the path, among other difficulties. Unsurprisingly then, VFH+ and SND experienced a fairly high percentage of failures (defined as taking at least five minutes to complete the

	WOAH	VFH+	SND
Mean (s)	97.88	138.16	134.45
Median (s)	97.74	135.45	134.43
Std Deviation (s)	1.29	17.02	2.90
Failures	0	306	252

Figure 4. Summary of static test data.

course), about 19% and 23% respectively. Surprisingly however, WOAH experienced exactly zero failures in its 1296 test runs. Performance-wise, WOAH and SND were very consistent with WOAH completing the course on average about 37 seconds faster. VFH+ was comparatively poor in terms of consistency though it in some cases preformed nearly as well as WOAH.

4.2 Dynamic Testing

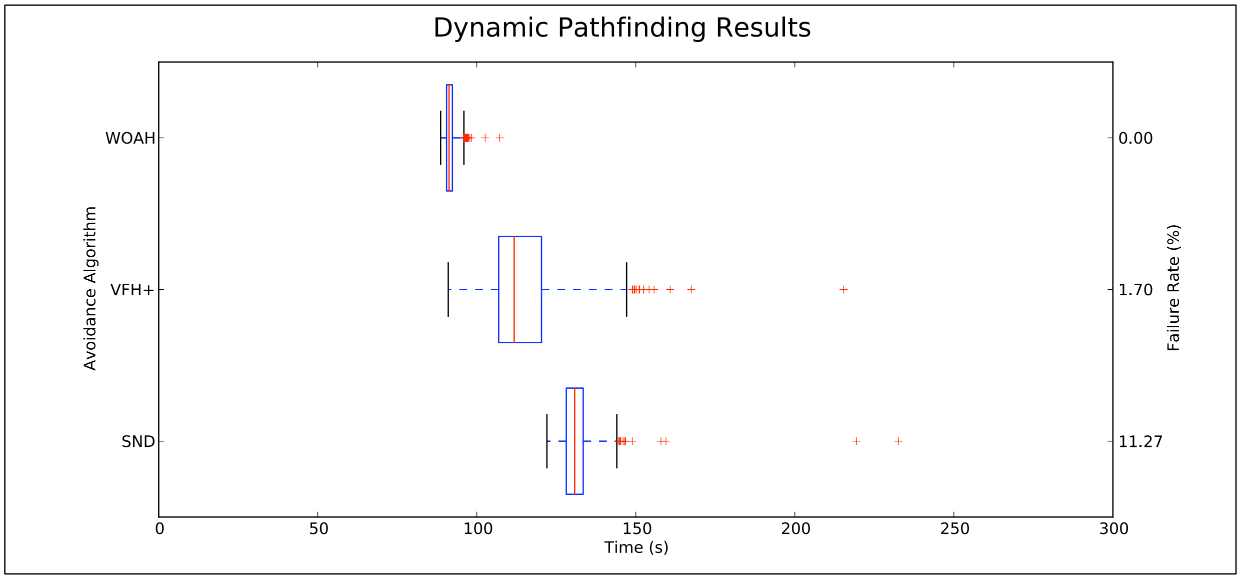

Figure 5. Plot of dynamic test data. Results show that WOAH performed the best and most consistently.

	WOAH	VFH+	SND
Mean (s)	91.67	114.73	131.25
Median (s)	91.29	111.76	130.78
Std Deviation (s)	1.71	11.70	5.82
Failures	0	22	146

Figure 6. Summary of dynamic test data.

Dynamic tests were run using dynamic pathfinding again using A* but this time using information discovered by the robot in real time. Like the previous one, this test was run nine times per algorithm per test map.

The results of dynamic testing are summarized by the box plot in Figure 5 and table in Figure 6. Dynamic pathfinding generally avoids the problem of getting off track as described above, though it is still possible to get stuck in bad states when the pathfinder and avoidance algorithm disagree. Both VFH+ and SND experienced significantly less failures in this test. All three algorithms performed better in terms of average times in this test. This can be explained as the dynamic method allows for imperfect path following, and none of the three algorithms can follow static paths perfectly. All three also had a higher rate of outliers in this test. This can be explained by the dynamic map discovery; although the map layout was designed so discovery plays a minimal role, it still adds another failure case to the pathfinding. The overall performance pattern of each algorithm was roughly the same as it was with static pathfinding, with WOAH being very consistent, followed by SND and finally VFH+. VFH+ however, gained substantially more performance than the other algorithms; VFH+ appears to be more desirable with dynamic pathfinding than it is with static paths.

4.3 Stress Testing

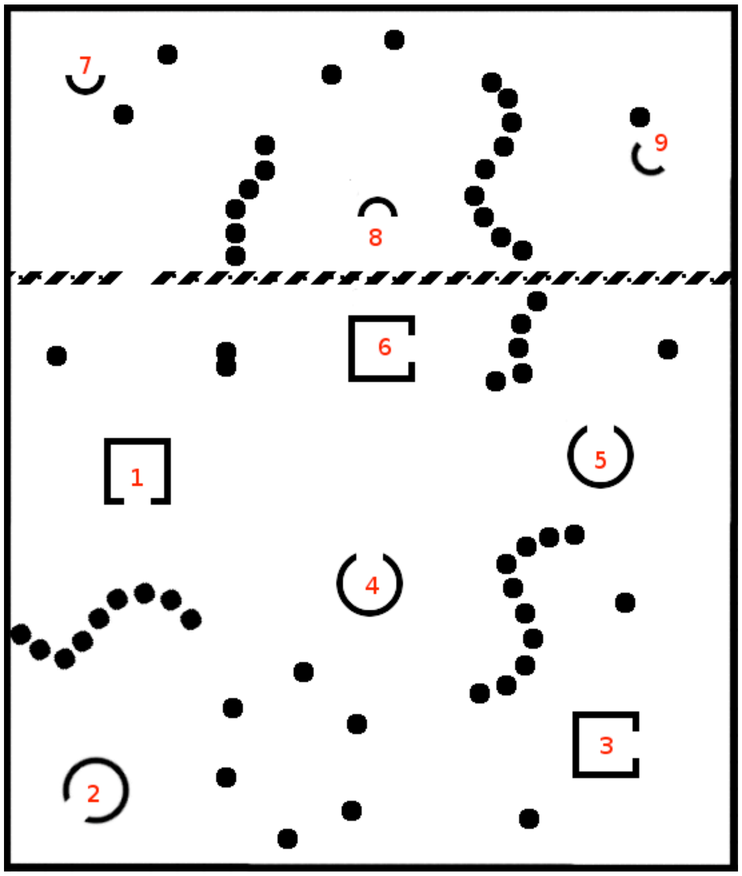

Figure 7. Stress Test Map based on IGVC navigation contest. Each number denotes a waypoint.

The last set of simulation tests were designed to stress test each algorithm in a more realistic simulation. The map was adapted from one of those we use to test for the IGVC navigation challenge as discussed previously. The resulting map is shown in Figure 7. The robot starts at the center of the map, and proceeds to each numbered waypoint in order using the same dynamic pathfinding as the dynamic test above. Note that only one transition, from waypoint five to six, allows the robot to reach the waypoint without having to navigate around the enclosing box or circle. Localization is handled using an average of two simulated GPS units, each with up to a meter of drift, mirroring the setup on our physical robot. Each run had a timeout of 900 seconds with an expected runtime of less than 300 seconds.

Each algorithm was tested 50 times on this map. The results of these tests are summarized by the box plot in Figure 8 and a table in Figure 9. Results are similar to those obtained in the previous test in terms of WOAH and VFH+, with the former holding a small lead in terms of speed, consistency. WOAH encountered some failures here for the first time in our testing, though they were minimal compared to VFH+ and SND. SND performed quite badly, with a failure rate of 82%. In many cases we observed SND becoming permanently stuck after reaching only one or two waypoints.

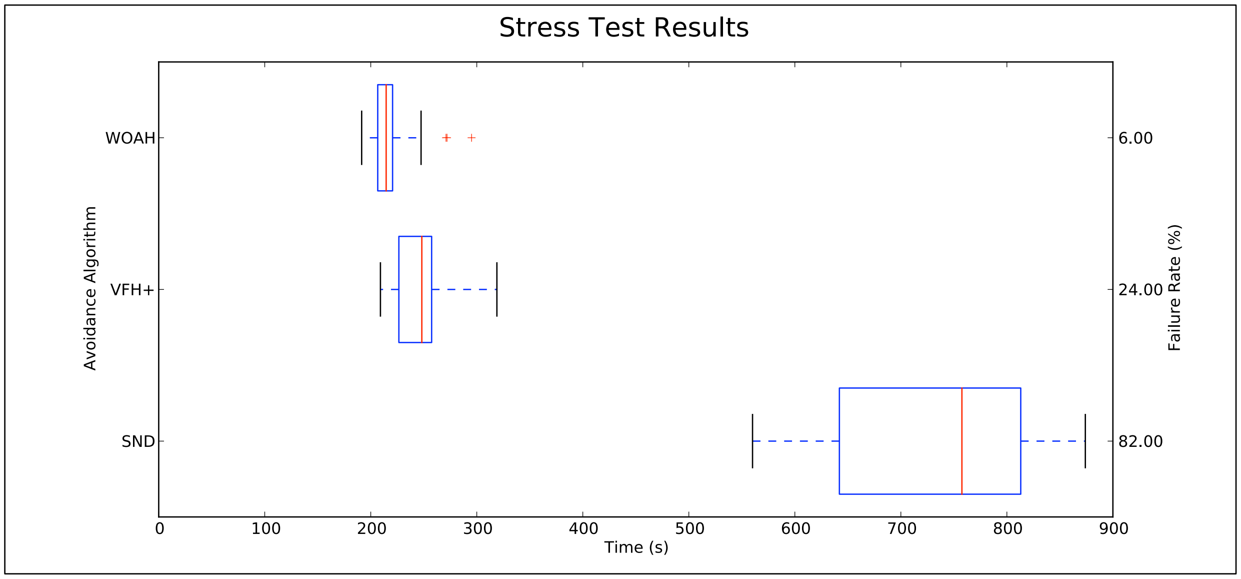

Figure 8. Plot of stress test data. Results show that WOAH perfomed the best and most consistently. SND peformed notably poorly.

	WOAH	VFH+	SND
Mean (s)	218.93	245.60	734.80
Median (s)	214.53	248.17	757.59
Std Deviation (s)	19.63	23.09	96.78
Failures	3	12	41

Figure 9. Summary of stress test data.

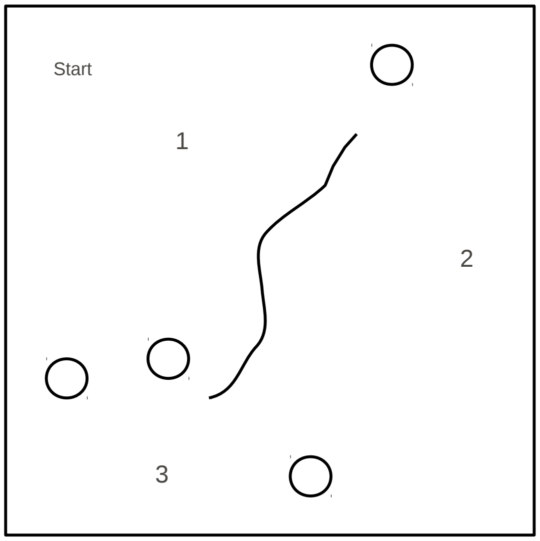

Figure 10. Sketch of the physical test course. The circles represent trash barrels and the line a construction fence. Each number represents a waypoint.

Figure 11. Our robot, Stark.

4.4 Physical Testing

We ran a small series of tests outside using our robot Stark. Stark is a rectangular differential drive robot with two non-differential GPS units and a digital compass for localization. During testing, a significant cloud cover caused GPS readings that were even less accurate than those in the above stress test. This led to the robot spending significant time circling waypoints, and this dominated run times. A sketch of the course layout can be seen in Figure 10 and a photograph of Stark on the course in Figure 11. Each run consisted of a path from the start following the waypoint sequence 1-2-3-1-3-2-1 using dynamic pathfinding. The course was run six times per algorithm.

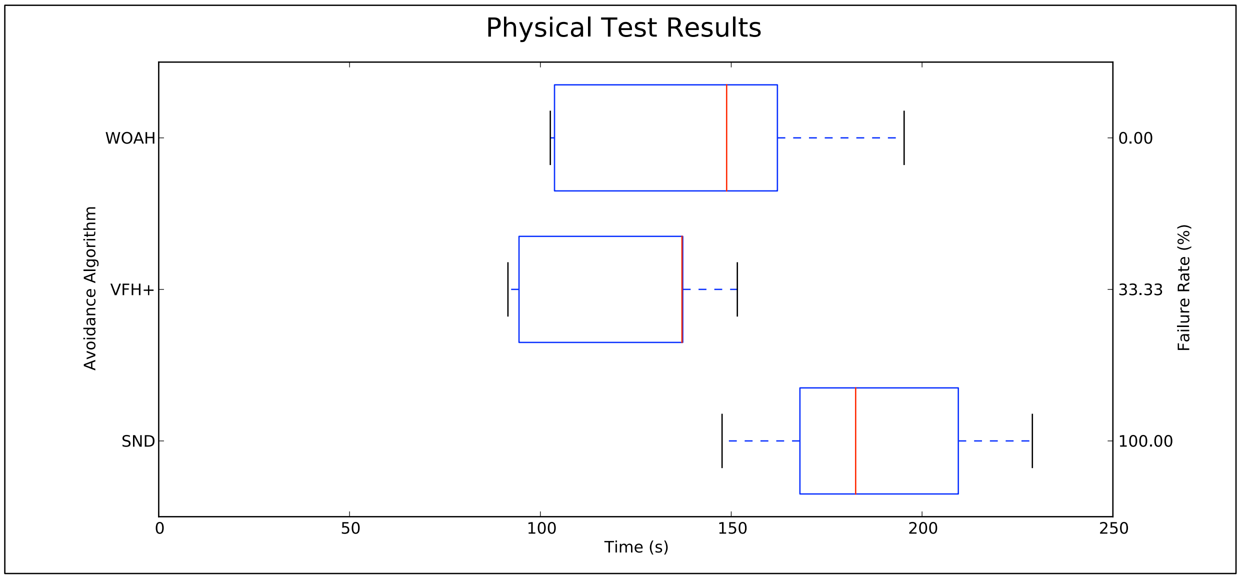

Figure 12. Plot of physical test data. Results show VFH+ and WOAH performed better than SND.

	WOAH	VFH+	SND
Mean (s)	137.73	118.48	186.30
Median (s)	131.35	118.05	181.90
Std Deviation (s)	37.39	26.40	29.07
Failures	0	2	6

Figure 13. Physical test data.

The results of this testing are summarized by the box plot in Figure 12 and table in Figure 13. We consider failures in this case to be any run where the robot struck an obstacle or needed outside intervention to continue. Additionally, runs that are considered failures are still included in the data set for completeness considering the low number of runs. VFH+ actually outperformed WOAH in terms of raw speed, but on several occasions impacted obstacles. SND consistently needed intervention to continue its runs, like the many times it got permanently stuck in simulation testing, and was significantly slower even considering the time wasted on circling GPS error. WOAH, though slower than VFH+, did not hit any obstacles on its runs. Reliable obstacle avoidance is a common application requirement for mobile robots. Overall the test was a bit inconclusive with the time distortion due to GPS inaccuracy and the low number of runs, but at least demonstrated that WOAH can run well on a physical robot.

4.5 Ease of Configuration

During the process of setting up and configuring the tests, the ease of configuration emerged as a critical property of an algorithm. SND was the easiest to configure with only a single non-physical parameter for avoid distance. Some effort was put into reducing parameters during the initial formulation of WOAH, though it was not a focus of our work. The result is five non-physical parameters which we were able to tune without much pain. The Player implementation of VFH+ on the other hand, was much more difficult to configure. As mentioned above, we had 15 different non-physical parameters to contend with, some of which the documentation failed to clearly explain.

5. FUTURE WORK

Future work concerning WOAH has a number of potential objectives. The production of an optimized implementation, possibly using optimization techniques from OpenPath, would allow it to run on resource constrained platforms. WOAH is currently designed specifically with a four-wheel differential drive vehicle in mind; redesigning it to work with additional drive systems would be worthwhile pursuit. Generalizing the sensory input to WOAH so that other sensor configurations can be used would allow WOAH to be used by far more robotic platforms and incorporate additional sensor information. Finally, modifying WOAH to incorporate a dynamic obstacle model would allow WOAH to operate in more dynamic environments.

6. CONCLUSION

We have presented WOAH, a technique for reactive local obstacle avoidance in primarily static environments. WOAH leverages basic assumptions about its goals and angular range data to move quickly and reliably through an environment. In simulation, WOAH follows paths given by a global planner more quickly and more reliably than both SND and VFH+. Our physical testing demonstrated that WOAH works properly on an actual robot. More testing is needed to confirm the performance benefit seen in simulation.

REFERENCES

1. Thrun, S., "Robotic mapping : a survey," in [*Exploring artificial intelligence in the new millennium*], 1 –37 (2002).
2. Minguez, J. and Montano, L., "Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios," *Robotics and Automation, IEEE Transactions on* **20**, 45 – 59 (Feb. 2004).
3. Borenstein, J. and Koren, Y., "The vector field histogram-fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on* **7**, 278 –288 (June 1991).
4. Ulrich, I. and Borenstein, J., "Vfh+: reliable obstacle avoidance for fast mobile robots," in [*Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*], **2**, 1572 –1577 (May 1998).
5. Von Wahlde, R., Wiedenman, N., Brown, W. A., and Viqueira, C., "An open-path obstacle avoidance algorithm using scanning laser range data," tech. rep., Army Research Lab Aberdeen Proving Ground (Feb. 2009).
6. Urmson, C., Anhalt, J., Bartz, D., Clark, M., Galatali, T., Gutierrez, A., Harbaugh, S., Johnston, J., Kato, H., Koon, P., Messner, W., Miller, N., Mosher, A., Peterson, K., Ragusa, C., Ray, D., Smith, B., Snider, J., Spiker, S., Struble, J., Ziglar, J., and Whittaker, W., "A robust approach to high-speed navigation for unrehearsed desert terrain," in [*The 2005 DARPA Grand Challenge*], Buehler, M., Iagnemma, K., and Singh, S., eds., *Springer Tracts in Advanced Robotics* **36**, 45–102, Springer Berlin / Heidelberg (2007).
7. Gerkey, B., Vaughan, R. T., and Howard, A., "The player/stage project: Tools for multi-robot and distributed sensor systems," in [*Proceedings of the 11th International Conference on Advanced Robotics*], 317 –323 (June 2003).