2010 NDIA GROUND VEHICLE SYSTEMS ENGINEERING and TECHNOLOGY SYMPOSIUM

MODELING & SIMULATION, TESTING AND VALIDATION (MSTV) MINI-SYMPOSIUM

AUGUST 17-19 DEARBORN, MICHIGAN

HEURISTICS-ENHANCED DEAD-RECKONING FOR IMPROVED SITUATION AWARENESS WITH TELE-OPERATED ROBOTS

Johann Borenstein, D.Sc.
Dept. of Mechanical Engineering
The University of Michigan
Ann Arbor, MI

Adam Borrell
Dept. of Mechanical Engineering*
The University of Michigan
Ann Arbor, MI

Russ Miller
Dept. of Mechanical Engineering
The University of Michigan
Ann Arbor, MI

David Thomas
Dept. of Mechanical Engineering
The University of Michigan
Ann Arbor, MI

ABSTRACT

This paper presents a practical and easy to implement method for tracking the position of tele-operated Unmanned Ground Vehicles (UGVs) inside buildings, where GPS is unavailable. In conventional dead-reckoning systems, which typically use odometry combined with a single-axis gyro or an Inertial Measurement Unit (IMU), heading errors grow without bound. For that reason, tracking the position of tele-operated UGVs for more than a few minutes becomes unfeasible. Our method, called Heuristics-Enhanced Dead-reckoning (HEDR), overcomes this problem by completely eliminating heading errors at steady state in tele-operated missions of unlimited duration. As a result, HEDR allows the plotting of very accurate trajectories on the Operator Console Unit (OCU). When overlaid over an aerial photo of a building, the real-time trajectory display gives the operator crucial information about position and heading of the UGV relative to the building. This feature offers the operator much improved situation awareness that is not provided by the conventional video feed from onboard cameras.

1 INTRODUCTION

Tele-operators of small Unmanned Ground Vehicles (UGVs) usually have to rely on the video stream from an onboard camera as feedback. The video pictures alone hardly ever allow the

* Adam Borrell was with the University of Michigan when this work was conducted. He is now with Boston Dynamics, Boston, MA.

operator to establish the orientation or whereabouts of the UGV in its environment. Some Operator Console Units (OCUs), such as that for iRobot's Packbot, offer a second window on which the trajectory of the UGV is plotted, but only if GPS is available. Indoors, where GPS is always unavailable, this screen is blank, presumably because the onboard dead-reckoning degrades over time and becomes too inaccurate to be useful after just a few minutes of driving [1].

Much research has focused on improving dead-reckoning capabilities for unmanned vehicles. For example, a common approach especially for tracked vehicles is to combine odometry with at least one gyroscope [2] or with a complete inertial measurement unit (IMU) [3]. The accuracy of these methods depends to a large degree on the quality of the gyro. Inexpensive (e.g., MEMS-based) gyros tend to produce heading errors at a rate of up to tens of degrees per minute. Fiber optic gyros produce errors at much lower rates, on the order of a few degrees per hour, but these gyros cost thousands of dollars.

To overcome this problem, we developed a heuristics-based method that tracks tele-operated UGVs with great accuracy, even with low cost MEMS gyros. Indeed, our method, called Heuristics-enhanced Dead-reckoning (HEDR), completely eliminates heading errors due to gyro drift when used inside buildings. HEDR produces a trajectory that has zero heading errors at steady state and, consequently, extremely small position errors (on the order of <1% of distance traveled).

HEDR exploits the fact that driving in structured, indoor environments mostly happens along straight corridors or walls, and that most corridors and walls intersect perpendicularly. We call the directions of corridors and other indoor travel paths that meet these requirements "dominant directions."

The foremost strength of the HEDR method is that it is tolerant to deviations from the heuristic assumptions. For example, zigzagging down a corridor, as happens often when tele-operators are disoriented, does not impede the effectiveness of HEDR, as long as the corridor is generally aligned with a dominant direction. Whenever HEDR detects that the UGV is not driving along a dominant direction, HEDR suspends its corrective action and accrues drift and thus heading and position errors—just as any other gyro-based dead-reckoning system does. However, once the driving in a dominant direction resumes, drift is again eliminated and heading errors gradually return to zero.

2 HEURISTICS-ENHANCED DEAD-RECKONING (HEDR)

This section is a duplication of work already published in an earlier paper (Proceedings of the SPIE Defense, Security + Sensing Symposium, 2010 [3]). We duplicate that section here for completeness and for the reader's convenience.

The HEDR method effectively eliminates gyro errors due to drift and other slow-changing errors. In suitable indoor environments, HEDR maintains zero heading errors in drives of unlimited duration, at steady state. However, these desirable performance characteristics are achieved only in environments that match certain heuristic assumptions, discussed next.

2.1 The Heuristic Assumptions

HEDR works in environments, in which possible heading angles are limited. For example, in man-made structures most corridors are straight and either parallel or orthogonal to each other

and to the peripheral walls. We will call the typical directions of walls and corridors the "dominant" directions of the building. In the huge majority of buildings there are only four dominant directions. One can easily verify this observation by looking at aerial or satellite photos of city and rural buildings: with very few exceptions, residential and business buildings have rectangular footprints, suggesting that most corridors and walls inside follow four dominant directions. Among the rare exceptions, we found empirically that the most frequent one is that of corridors angled at 45 degrees to others. In order to account for this exception, our heuristic assumptions actually allow for eight dominant directions, spaced at 45-degree intervals. Informally, we estimate that well over 99 percent of all man-made structures have four or eight dominant directions. Prominent exceptions are the Pentagon, some architectural landmarks such as theaters, opera houses, and some large hotel complexes, and indoor sports arenas. In these structures, as well as in many tunnels and all natural caves, HEDR cannot be used.

We call driving that complies with the heuristic assumptions (i.e., driving along a dominant direction) "compliant" driving. The strength of HEDR lies in the fact that it applies corrections only gradually, when it believes driving to be compliant, and it reduces or suspends its corrections when driving is not compliant. While prolonged non-compliant motion may render HEDR ineffective, the method is nonetheless very robust in the face of short non-compliance. For example, HEDR will easily tolerate the crossing of a large hall (e.g., in a mall or warehouse) at an angle other that 90°.

In all other, "normal" environments, HEDR detects when motion matches one of the eight dominant directions and gradually corrects gyro output so that the combined effect of drift and HEDR correction is such that the computed heading of the tele-operated vehicle matches the closest dominant direction. When the vehicle turns, HEDR suspends its corrective action. While HEDR is suspended, drift causes new heading errors, but once HEDR resumes, it effectively eliminates accrued heading errors because it gradually forces headings to be aligned with the closest dominant directions. When motion is mostly compliant, HEDR assures zero heading errors in drives of unlimited duration at steady state. Steady state is usually reached within a few seconds of compliant motion after turning. With heading errors eliminated, position errors remain *orders of magnitude* smaller than with conventional dead-reckoning. The resulting small position errors make it possible to track the position of tele-operated vehicles accurately and reliably over extended periods of time.

HEDR works generally with any dead-reckoning system that uses one or more gyros for measuring rate of yaw to compute the vehicle's heading. Moreover, HEDR is extremely simple and can be implemented in ~20 lines of program code and on low-cost microcontrollers.

In summary, HEDR applies two simple but highly effective heuristic assumptions:

1) Most travel inside buildings happens along straight lines, called "dominant directions." Dominant directions are defined by the typically rectangular footprint of man-made structures. Corridors, walls, or traffic patterns typically force traffic to follow dominant directions. Zigzagging within a corridor adds a little noise, but has otherwise no negative effect on HEDR.

2) There are nominally four dominant directions in a building, spaced at 90-degree intervals. However, as we will see in the following section, it is practical to broaden the heuristic assumptions and define eight dominant directions, spaced at 45-degree intervals.

2.2 General heading estimation

Tele-operated vehicles are often equipped with single z-axis gyros or even inertial measurement units (IMUs) for dead-reckoning. When driving straight forward, the output of the z-axes gyro should be exactly zero. However, due to drift the actual output is off by some small value ε.

Due to the drift error ε, in each sampling interval the rate of rotation computed based on the z-axis gyro is

$$\omega_{raw} = \omega_{true} + \varepsilon_0 + \varepsilon_d \quad (1)$$

where

ω_{raw} – Rate of rotation measurement. This is the direct output of the gyro.

ω_{true} – True rate of rotation. In reality ω_{true} is not known or measured. We only know that when driving straight forward, $\omega_{true} = 0$.

ε_0 – Static bias drift, measured immediately prior to a drive, while the vehicle is standing still.

ε_d – Bias drift. This is the difference between the static bias drift ε_0 and the unknown slow-changing drift component.

Immediately prior to each drive and with the gyro held completely motionless, the static bias drift ε_0 is measured by averaging T_{bias} worth of gyro data. The value of T_{bias} depends on the characteristics of the gyro and a detailed discussion of this subject is beyond the scope of this paper.

During the drive, ε_0 is subtracted from every reading of ω_{raw}:

$$\omega_{meas} = \omega_{raw} - \varepsilon_0 = \omega_{true} + \varepsilon_d \quad (2)$$

Then, the new heading ψ_i is computed

$$\psi_i = \psi_{i-1} + \omega_{meas,i} T \quad (3)$$

where

ψ_i – Computed heading after interval i, in [°].

$\omega_{meas,i}$ – ω after removing static bias drift, in [°/sec].

T – sampling time in [sec].

When driving straight forward, $\omega_{true} = 0$, and $\psi = \varepsilon_d T$. If we further assume that the driving happens along the first of the four or eight dominant directions, which we define as being aligned with 0°, then ψ_i should be zero. If it is not, then it must be so because of drift, ε_d. Moreover, since drift is a slow-changing phenomenon, the sign of ε_d and thus also of ψ will stay the same over many successive intervals. We therefore hypothesized that it might be possible to track and estimate ε_d by examining the sign of ψ. While this hypothesis might seem somewhat optimistic, the HEDR algorithm does exactly that and more. In the following sections we explain how the HEDR algorithm:

- Reduces driving in any of the nominally four or more dominant directions to the functional equivalent of driving in a direction of zero degrees.
- Models ε_d as a disturbance in a feedback control system

- Estimates the magnitude of this disturbance by examining the content of the accumulator in the I-controller of that feedback control circuit.
- Remains largely insensitive to additional, large-amplitude disturbances of short duration.

2.3 The Basic HEDR Method

As explained, the HEDR algorithm assumes that a building has four or eight dominant directions, Ψ. In order to keep the discussion in this section intuitive, we will explain the HEDR method for four dominant directions. In that case, dominant directions are spaced at 90-degree intervals and we call this interval the "dominant direction interval," Δ.

A further assumption is that most corridors and inside walls in a rectangular-footprint building run parallel to its dominant directions. If this assumption is true, then one can further assume that most driving in such buildings is also done along dominant directions. For the HEDR algorithm to work well, this latter assumption does not have to hold true all of the time.

The basic HEDR algorithm functions essentially like a feedback control system. This is different from most other measuring systems, where signals pass from the sensor to the instrument's output in open-loop fashion. Figure 1 shows a block diagram of the feedback control system for the HEDR algorithm. Before we explain the overall function of this feedback control system, it is necessary to define in detail the function of the block labeled "MOD(θ,Δ)."

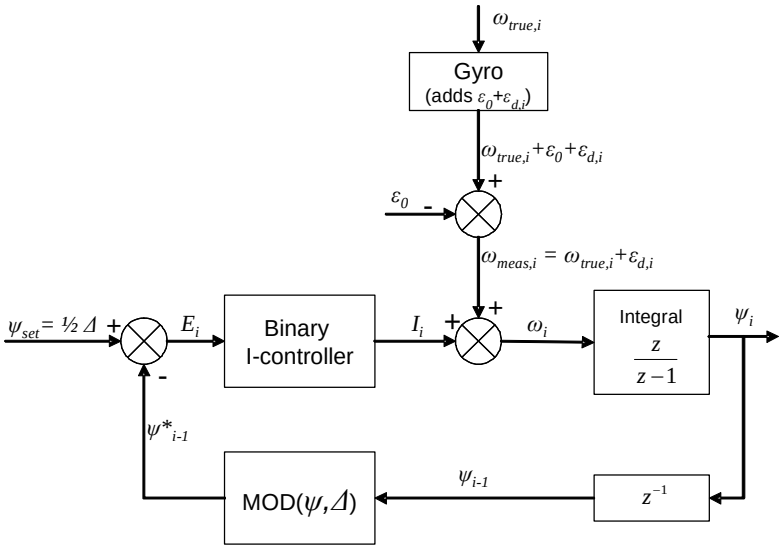

Figure 1: The HEDR algorithm viewed as a feedback control system. The binary I-controller and the block labeled 'MOD' are explained in the narrative.

Use of the MOD function

A "MOD" function is available in different programming environments, but we found that its definition varies, especially with regard to treating negative numbers. In the context of this paper we use the implementation found in Microsoft Excel. In Excel, the MOD function has two arguments, (n, d). MOD(n, d) returns the remainder after n is divided by d, and the result has the same sign as d. In programming environments, in which the MOD function performs differently, the Excel version can be emulated by this formula:

$$\text{MOD}(n, d) = n - d\ \text{INT}(n/d) \tag{4}$$

Where INT(r) is a function that rounds a real number r down to the nearest integer. For example: INT(-0.3) = -1.

When n and d are angles in the Cartesian coordinate system used throughout this paper, then MOD(n,d) performs a highly useful function: it maps an angle of any magnitude n onto a sector that is bounded on one side by the positive x-axis and that has a central angle of d. As an example, consider the two sub-sectors labeled 'R' and 'L' in Figure 2. Together, these two sub-

sectors form a sector with a central angle of d = 90° that coincides with the first quadrant of a coordinate system. The function MOD(n,90°) maps any angle n, which may be greater than 360° or negative, onto Sectors R and L. Similarly, MOD(n,45°) maps any angle n onto Sector L.

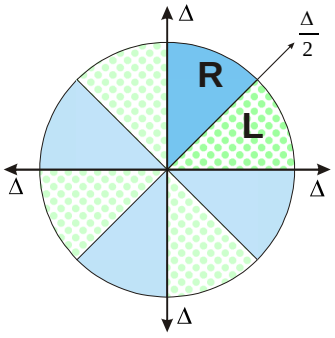

Figure 2: Angle mapping with the MOD function.

In the feedback control system of the HEDR algorithm, block MOD(θ,Δ) with Δ = 90° maps any heading angle onto either Sub-sectors R or L of Figure 2. The functional significance of applying MOD(θ,Δ=90°) is this: In a building with the four dominant directions Ψ = 90°, 180°, 270°, and 360° (= 0°) any momentary heading direction θ that is immediately to the right of any of these four dominant directions will be mapped into Sub-sector R. Similarly, any heading direction that was immediately to the left of any of these four dominant directions will be mapped into Sub-sector L. With "immediately to the right" we mean angles that are between Ψ and Ψ-Δ/2 (the solid blue sectors in Figure 2), while "immediately to left" means between Ψ and Ψ+Δ/2 (the dotted green sectors in Figure 2).

As an example, consider a momentary heading of θ = -25°, which is immediately to the right of the dominant direction Ψ = 0°. The result of applying the MOD function is MOD(-25°,90°) = 65°. 65° is also immediately to the right of a dominant direction, namely Ψ = 90°. If the vehicle turned three full revolutions in counter-clockwise direction, then the vehicle's heading should still be immediately to the right of a dominant direction. In the HEDR system, the vehicle's new heading would be represented as θ = -25° + 3×360° = 1055°. The MOD function maps the new heading right back to θ* = MOD(1055°,90°) = 65° which is, as before, immediately to the right of a dominant direction. Table I illustrates how with the help of the MOD function we can perform a simple test to see if a momentary heading angle is immediately to the right or left of *any* dominant direction.

Table I: Relationship of θ* to MOD(θ,Δ) with regard to dominant directions.

θ*	Significance
> Δ/2	*θ immediately to the right of a dominant direction Ψ*
= Δ/2	*θ perfectly in-between adjacent dominant directions Ψ*
< Δ/2	*θ immediately to the left of a dominant direction Ψ*
= 0	*θ perfectly aligned with a dominant direction Ψ*

2.3.1 The HEDR Feedback Control System

We start the explanation of the feedback control system with the signal from the gyro, which is modeled as a disturbance in the block diagram of Figure 1. For the purpose of explaining the feedback control system, let us assume that the vehicle is driving straight ahead and in a dominant direction. We will discuss how the algorithm handles cases when this assumption is not true later. When driving straight, ω_{true}= 0, so the only output from the gyro, after subtracting the static bias drift ε_0, is drift, ε_d.

This signal is added to the output of the binary I-controller, which will be explained later in this section. Initially, the output of the I-controller is zero, so ε_d is passed through to a numeric integrator, which computes the relative change of heading, ψ_i.

After the first iteration, when $i>1$, the control loop can be closed by submitting the previous value of ψ, ψ_{i-1}, to the MOD function. The label 'z^{-1}' in the feedback loop is the common notation for a pure delay of one sampling interval. As explained before, MOD(ψ,Δ) maps θ onto a direction that lies between 0 and Δ.

$$\psi^*_i = \text{MOD}(\psi_{i-1},\Delta) \tag{5}$$

where

ψ^*_i – Mapped heading that lies between 0 and Δ, in degrees.

ψ^*_i is then compared to the fixed set point, $\psi_{set} = \Delta/2$, resulting in an error signal

$$E_i = \Delta/2 - \psi^*_i \tag{6}$$

This brings us to the binary I-controller. Unlike conventional integral (I) or proportional-integral (PI) controllers, the binary I-controller is designed not to respond at all to the magnitude of E. Rather, it only responds to the sign of E. If E is positive (i.e., heading points to the left of a dominant direction), then a counter (called "Integrator" or "I") is incremented by a fixed small increment, i_c. If E is negative (i.e., heading points to the right of the dominant direction), then I is decremented by i_c. In this fashion, and although the controller does not respond to the magnitude of E immediately, *repeated* instances of E having the same sign will result in repeated increments or decrements of I by i_c.

The reason for using a binary I-controller is that the ideal condition $\Psi^* = 0°$ (i.e., Ψ^* being perfectly aligned with one of the dominant directions) is rarely met. Indeed, Ψ^* can differ from zero by tens of degrees, for example, when the vehicle is turning. In that case a conventional I-controller would not work well, since it would respond strongly to the large value of E, even though large E are not necessarily an indication for a large amount of drift. The proposed binary I-controller, on the other hand, is insensitive to the magnitude of E. Rather, the controller reacts, slowly, to E having the same sign *persistently*.

As established by Eq. (6), if $\psi^*>\psi_{set}$ then ψ^* is immediately to the right of Ψ, and if $\psi^*< \psi_{set}$ then ψ^* is immediately to the left of Ψ. During straight-line driving along a dominant direction Ψ, a heading to the right of Ψ suggests that the only possible source for this error, ε_d, had a negative value. To counteract this error, the binary I-controller adds the small increment, i_c to the Integrator. Conversely, if $\psi^*<\psi_{set}$, then the Integrator is decremented by i_c.

We can now formulate the binary I-controller

$$I_i = \begin{cases} I_{i-1} - i_c & \text{for} \quad E < 0 \\ I_{i-1} & \text{for} \quad E = 0 \\ I_{i-1} + i_c & \text{for} \quad E > 0 \end{cases} \tag{7a}$$

where

i_c – fixed increment, also considered the gain of the binary I-controller in units of degrees

An alternative way of writing Eq. (7a) is

$$I_i = I_{i-1} - \text{SIGN}(\psi^*_{i-1} - \frac{\Delta}{2}) i_c \tag{7b}$$

where SIGN() is a programming function that determines the sign of a number. SIGN(x) returns '1' if x is positive, '0' if $x = 0$, and '-1' if x is negative.

The next element in the control loop adds the controller output to the raw measurement

$$\omega_i = \omega_{true,i} + \varepsilon_{d,i} + I_i \tag{8}$$

where

ω_i – Corrected rate of rotation [°/sec].

If $I \cong -\varepsilon_d$, as we assume for now to be the case under ideal conditions, in steady state, and because of the I-controller in the feedback control system, then by substituting $I \cong -\varepsilon_d$ in Eq. (8) we would get $\omega_i \cong \omega_{true,i}$. This result would be desirable, since the unknown slow-changing drift component is removed. In practice, though, neither ω_{true} nor ε_d are known. Instead, we only know the measured $\omega_{meas,i} = \omega_{true,i} + \varepsilon_{d,i}$.

We rewrite Eq. (8) accordingly:

$$\omega_i = \omega_{meas,i} + I_i \tag{9}$$

Substituting Eq. (5) and Eq. (7b) in Eq. (9) yields:

$$\omega_i = \omega_{meas,i} + I_{i-1} - i_c \mathrm{SIGN}\left((MOD(\psi_{i-1}, \Delta) - \frac{\Delta}{2}) \right) \tag{10}$$

Equation (10) represents the complete HEDR algorithm. When applied to the output of a z-axis gyro, $\omega_{meas,i}$, of a tele-operated vehicle, the algorithm will effectively remove drift and other slow-changing errors. An additional benefit is that the momentary value of I is an accurate estimate of these errors. The only tunable parameter in the algorithm is i_c. A good starting point for tuning i_c is at about ten times the estimated magnitude of the drift rate, in deg/sec.

Additional refinements are possible and were used in our system but are omitted in this paper because of space limitations.

3 THE EXPERIMENTAL SYSTEM

We chose to test and validate the HEDR algorithm on a simple commercial off-the-shelf COTS skid-steer tracked platform available from Lynxmotion [4]. We outfitted each track with a DC gear-head drive motor with an optical encoder, and connected each to a separate drive output of a serial motor controller. To complement the motor encoders we added a low-cost ($300) MEMS gyroscope to the platform, the CruizCore XG1010 made by Microfinity [5]. For visual feedback to a remote operator, we selected a COTS 802.11n wireless webcam capable of transmitting Real Time Streaming Protocol (RTSP) video and audio over an infrastructure or ad-hoc wireless network. We mounted the camera on a hobby-servo driven pan/tilt platform, controlled by a serial servo controller. Throughout this paper, we refer to this platform as "Trackey" (Figure 3).

To complete the robot side of the experimental system we mounted a custom PCB on the platform, including a serial radio and an 8-bit microcontroller. The motor controllers draw power directly from an on-board NiMH battery pack, while the custom PCB is powered through a 5V switching DC/DC converter. The microcontroller directly samples and decodes quadrature encoder signals from the left and right tracks using hardware interrupt pins. The microcontroller also receives serial packets from the gyro and processes them to extract the measured rate of rotation. The microcontroller applies the HEDR algorithm to the encoder and gyro data, and uses

parameters of the vehicle (encoder resolution, effective track sprocket diameter, etc.) to calculate the HEDR-corrected position and heading of the vehicle in real time. This information is transmitted over the serial radio to the Operator Console Unit (OCU).

The OCU, shown in Figure 4) consists of a small tablet-style laptop mounted on a sheet-plastic plate. We mounted two analog joysticks (similar to those used in videogame controllers) to the sides of the plate, and another custom microcontroller board with a serial radio and a USB to serial adapter to the back of the plate. This microcontroller receives the position packets transmitted by the robot over the serial radio link, and passes them on to the laptop through the USB to serial adapter. Robot trajectory is displayed in real time by our custom plotting software. Streaming video from the robot is received using the laptop's internal wireless card and displayed beside the trajectory plot. The OCU microcontroller also samples the positions of the joysticks using a Digital to Analog Converter (DAC), and generates and transmits control packets to the robot. These packets are received by the robot's serial radio, and processed by its microcontroller into commands for the motor and servo controllers. One joystick controls the motion of the robot, while the other aims the pan/tilt camera platform.

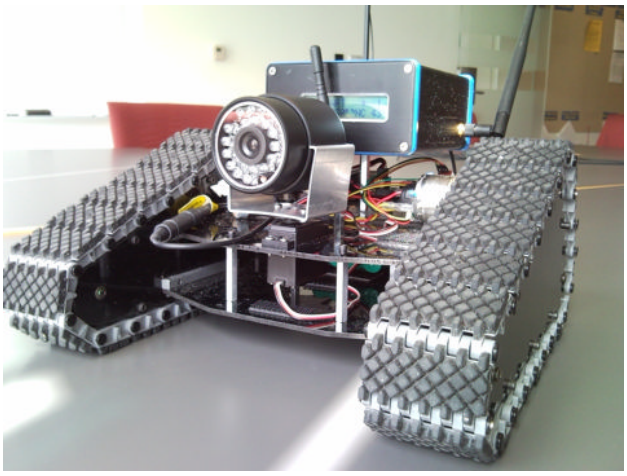

Figure 3: Our small, tracked robot Trackey served as the test platform for this project.

Figure 4: Operator Console Unit with tablet style laptop, dual joysticks, and serial radio (mounted underneath) for transmitting joystick commands.

4 EXPERIMENTAL RESULTS

In order to test the effectiveness and accuracy of the HEDR system, we performed five tele-operated test runs with Trackey in various indoor environments. In all five runs a tele-operator controlled Trackey remotely, using the view of the onboard camera and the HEDR-based trajectory plot for feedback. A photo of the operator's console during a run is shown in Figure 5.

At no time did the operator have a direct line of sight to the UGV. We emphasize this fact because under these conditions, tele-operated driving is much more erratic than driving with a direct line of sight. Also, the tele-operator had to cope with real reductions in frame rate that are typical in tele-operated robots once the robot gets further away from the operator, as well as with degraded video quality. With frame rates as low as one frame per second, turning often results in significant overshoot. Under these conditions, the HEDR system is fully challenged since some of the driving is zigzagging and otherwise not very straight, even in straight corridors.

In all five runs the robot started at a position labeled (0,0) and at the end of the run stopped at that exact same position. At the stopping position we compared the computed final position based on dead-reckoning and based on HEDR with the actual final position (0,0). The difference between the two is called the Return Position Error (RPE). The RPE is not a great indicator for the accuracy of a tracking system, since it is quite possible to have large heading errors but very small RPEs. In the case here, however, it would have been too tedious to try and compile ground truth for position in each run. We also did not see the need for that effort since we are providing plotted results for the five runs in Figure 6 .

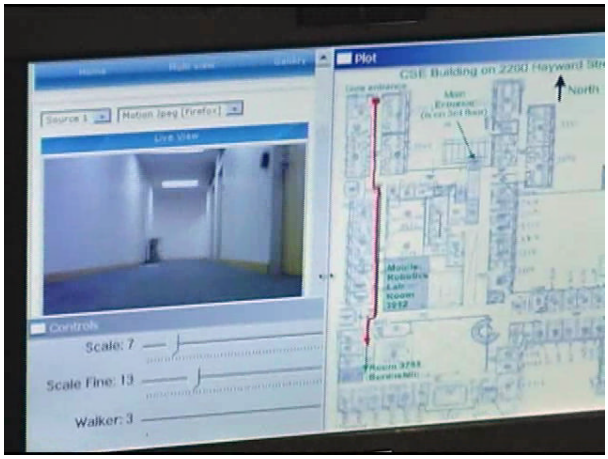

Figure 5: The display of the Operator Console Unit (OCU) during a tele-operated drive. The left-hand pane shows the video picture from the robot. The right-hand pane shows the trajectory of the robot as it develops in real-time.

The five runs took between 27 and 50 minutes to complete. Run 1 is about 20 minutes longer than the others because it includes a 20-minute segment of complete standstill at an arbitrary angle. With that we wanted to emulate the conditions of persistent stare, which is of interest for security applications. Table II shows a summary of the specific conditions of each run, along with some numeric results.

Table II: Specifications and results for the five Trackey runs.

Run #	Total duration [minutes]	Total travel distance [meters]	Return position error (RPE) [meters]	Relative RPE as percentage of distance traveled [%]
1	50.0	1,260	3.43	0.27%
2	30.9	1,147	2.12	0.33%
3	32.6	1,181	2.37	0.20%
4	31.0	1,147	2.12	0.18%
5	27.3	880	6.02	0.68%
Average	**34.4**	**1,123**	**3.21**	**0.33%**

It is of interest to note that in our earlier paper [3], which provides experimental results for HEDR implemented on a Packbot, the Average Relative RPE for six runs under more challenging conditions was 0.36% – almost exactly the same as in the series of experiments described here for the Trackey platform.

In order to show the absolute accuracy of HEDR-derived trajectories, we overlaid the trajectory of a 20-minute drive over the floor plan of the building in which the drive took place, in Figure 7. In a realistic military scenario, floor plans are likely not available, but the trajectory could be overlaid over a satellite or aerial photo of the building.

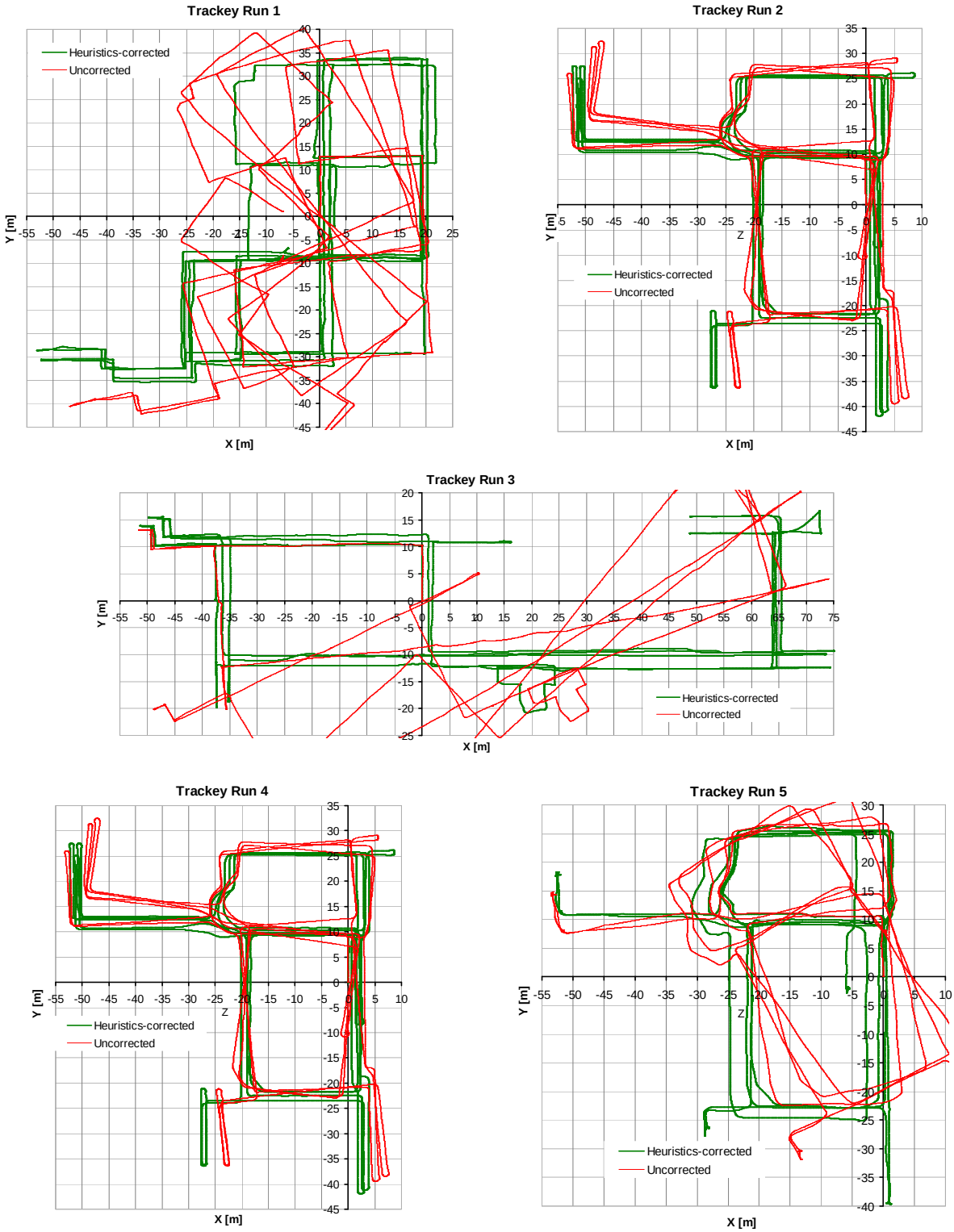

Figure 6: Five indoor runs with Trackey. Thin red lines: conventional dead-reckoning; (thick green line heuristics-enhanced dead-reckoning. Runs varied in duration between 27 to 50 minutes and in travel distance from 880 to 1,260 meters.

Figure 7: Trackey's trajectory after being driven through a large building, overlaid over a floor plan. In real applications, the trajectory would be shown overlaid over an aerial image (if available) or a blank background.

5 CONCLUSIONS

In this paper we explained the "Heuristics-Enhanced Dead-reckoning" (HEDR) algorithm for indoor tracking of tele-operated robots. We implemented HEDR in the Transferable Indoor Position Tracking (TIPT) system.

In five carefully monitored indoor experiments we demonstrated that HEDR eliminates heading error caused by gyro drift and other slow-changing sources of error, effectively maintaining zero heading errors in drives of unlimited duration at steady state. As a direct result, HEDR significantly reduces position errors, as accumulated heading errors are almost always the primary source of position errors in a dead-reckoning system.

From the very small Return Position Errors (RPEs), which are all well below 1% of distance traveled, one can derive several insights:

1. The HEDR method is indeed successful at containing heading errors, and doing so is helpful for containing position errors.

2. Once heading errors are effectively eliminated, position errors are very small even on tracked vehicles—at least while traveling on benign, smooth terrain and when using a gyro for estimating rates of turn.

Acknowledgements

This work was supported by The University of Michigan's Ground Robotics Reliability Center (GRRC), with funding provided by TARDEC, and with support from the U.S. Dept. of Energy under Award No. DE FG52 2004NA25587.

6 REFERENCES

[1] Borenstein, J. and Feng, L., "Measurement and Correction of Systematic Odometry Errors in Mobile Robots." *IEEE Transactions on Robotics and Automation*, 12(6), 869-880 (1996).

[2] Chung, H., Ojeda, L. and Borenstein, J., "Accurate Mobile Robot Dead-reckoning With a Precision-calibrated Fiber Optic Gyroscope." *IEEE Transactions on Robotics and Automation*, 17(1), 80-84 (2001).

[3] Borenstein, J., Miller, R., Borrell, A., and Thomas, D., 2010, "Heuristics-enhanced Dead-reckoning (HEDR) for Accurate Position Tracking of Tele-operated UGVs." *Proceedings of the SPIE Defense, Security + Sensing; Unmanned Systems Technology XII, Conference DS117: Unmanned, Robotic, and Layered Systems*. Orlando, FL, April 5-9, 2010.

[4] Lynxmotion, http://www.lynxmotion.com/

[5] Microfinity, Spec Sheet, http://www.cruizcore.com.