Proceedings of the SPIE Defense, Security + Sensing;
Unmanned Systems Technology XII, Conference DS117: Unmanned, Robotic, and Layered Systems. Orlando, FL, April 5-9, 2010

Heuristics-enhanced Dead-reckoning (HEDR) for Accurate Position Tracking of Tele-operated UGVs

Johann Borenstein*, Adam Borrell, Russ Miller, David Thomas

All authors are with the University of Michigan, Dept of Mechanical Engineering,
2260 Hayward St, Ann Arbor, MI 48109.

* johannb@umich.edu, ph.: 734-763-1560

ABSTRACT

This paper introduces a new approach for precision indoor tracking of tele-operated robots, called "Heuristics-Enhanced Dead-reckoning" (HEDR). HEDR does not rely on GPS, or external references; it uses odometry and a low-cost MEMS-based gyro. Our method corrects heading errors incurred by the high drift rate of the gyro by exploiting the structured nature of most indoor environments, but without having to directly measure features of the environment. The only operator feedback offered by most tele-operated robots is the view from a low to the ground onboard camera. Live video lets the operator observe the robot's immediate surroundings, but does not establish the orientation or whereabouts of the robot in its environment. Mentally keeping track of the robot's trajectory is difficult, and operators easily become disoriented. Our goal is to provide the tele-operator with a map view of the robot's current location and heading, as well as its previous trajectory, similar to the information provided by an automotive GPS navigation system. This frees tele-operators to focus on controlling the robot and achieving other mission goals, and provides the precise location of the robot if it becomes disabled and needs to be recovered.

Keywords: Heuristic, Dead-reckoning, Odometry, Mobile Robot, Gyro, GPS-denied, Tele-operated, Tracking

1 INTRODUCTION

Most tele-operated robots offer the remote operator just one kind of visual feedback: the view from an onboard camera. These video pictures do not allow the operator to establish the orientation or whereabouts of the robot in its environment. Some Operator Control Units (OCUs), such as that for iRobot's PackBot, offer a second window, in which the trajectory of the robot is plotted. However, this window is shown only if GPS is available. Indoors, where GPS is not available, it is necessary to employ other vehicle tracking methods to supply the tele-operator with a view of the robot's trajectory.

The most widely used method for tracking the position of unmanned ground vehicles (UGV) in GPS-denied environments is odometry, that is, the counting of fractional revolutions of wheels on the left and right side of the UGV[1]. Odometry only works well if there is no wheel slippage. If there is much slippage, such as when tracked vehicles turn, then odometry alone is not useful. A common approach for tracked vehicles is to combine odometry with at least one gyroscope or with a complete inertial measurement unit (IMU)[2]. 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[3]. Fiber optic gyros produce errors at much lower rates, on the order of a few degrees per hour[2], but these gyros cost thousands of dollars.

Other position tracking methods exist that use pre-existing recognizable features in the environment as absolute references. One such method, which uses directional range-finding sensors or cameras is Simultaneous Localization And Mapping, or SLAM[4]. In SLAM, prominent features, such as planes and corners, are extracted from sensor data and used as landmarks to build a map of the vehicles environment. Changes in the range and bearing of the landmarks as measured by the vehicle over time are used to estimate the vehicle's trajectory. One major challenge with SLAM is the

problem of landmark association, as individual sensor readings must be correctly associated with new landmarks, or re-associated with their corresponding, previously observed landmarks. Only landmarks that are static with respect to the environment can be used to infer the motion of the vehicle, so SLAM systems must also be able to identify and ignore other moving objects.

A purely vision-based method similar to SLAM is visual odometry[5,6], where cameras are used to perform dead reckoning more directly. Features, such as corners, are tracked in successive video frames and used to estimate the motion of the camera and vehicle in the period between the frames. Feature association is less of a problem for visual odometry, as the changes from image to image tend to be minor, and there is no need to associate a re-observed feature with its previous encounters. Visual odometry is, however, computationally intensive, as video must be processed in real-time to extract and track many features. Environmental conditions such as darkness or smoke may also interfere with video quality.

In this paper we propose a substantially different approach, called "Heuristics-Enhanced Dead-reckoning" (HEDR). HEDR uses odometry and a low-cost MEMS-based gyro for indoor tracking. Our method corrects heading errors incurred by the high drift rate of the gyro by exploiting the structured nature of most indoor environments, but without having to directly measure features of the environment. In earlier work we developed a much less powerful but more widely applicable heuristic-enhanced dead-reckoning method for vehicles, called Heuristic Drift Reduction (HDR)[7].

The remainder of this paper is structured as follows. In Section 2 we describe the relevant background and provide a broad description of our heuristic approach. A detailed explanation of the HEDR method is provided in Section 3. Section 4 then describes our hardware system designed for the PackBot, while Section 5 provides extensive experimental results.

2 HEURISTIC DRIFT ELIMINATION

This paper introduces our Heuristics Enhanced Dead-reckoning (HEDR) method for effectively eliminating 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)."

2.4 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:

$$\mathrm{MOD}(n, d) = n - d\ \mathrm{INT}(n/d) \qquad (4)$$

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

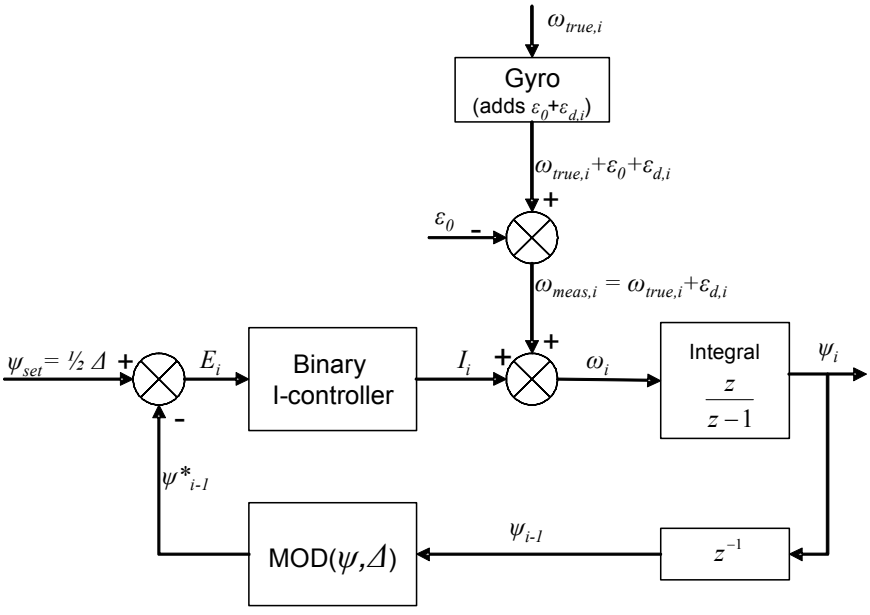


Figure 1: The HEDR algorithm viewed as a feedback control system. The binary I-controller and the block labeled 'MOD' are explained in the narrative.

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.

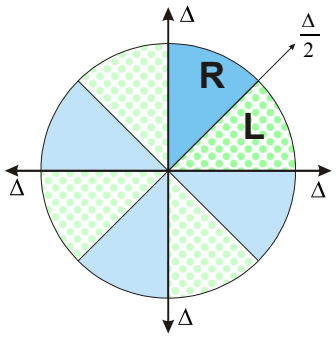


Figure 2: Angle mapping with the MOD function.

In the proposed 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: Significance of θ* = 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

2.5 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.

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

where

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

ψ^*_i is then compared to the fixed set point, ψ_{set} = Δ/2, resulting in an error signal

$$E_i = \Delta/2 - \psi^*_i \quad (6)$$

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:

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

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 but are omitted in this paper because of space limitations.

3 THE EXPERIMENTAL SYSTEM

We chose to test and validate the HEDR algorithm on an iRobot PackBot, a tele-operated tracked robot already in common use. Our PackBot is equipped with a rotating boom-arm with a pan/tilt camera head mounted on the end. PackBots have encoders on their track motors and the capability to accept data from additional sensor payloads. With the addition of a gyroscope the HEDR algorithm could run directly on a PackBot's on-board computer, but for simplicity we constructed a separate HEDR test system to operate in parallel with the stock remote control system, as shown in Figure 3. This separate system has the advantage of being transferrable to any tele-operated robot.

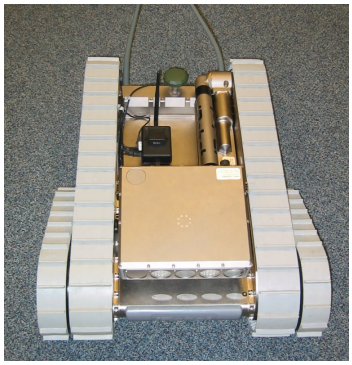
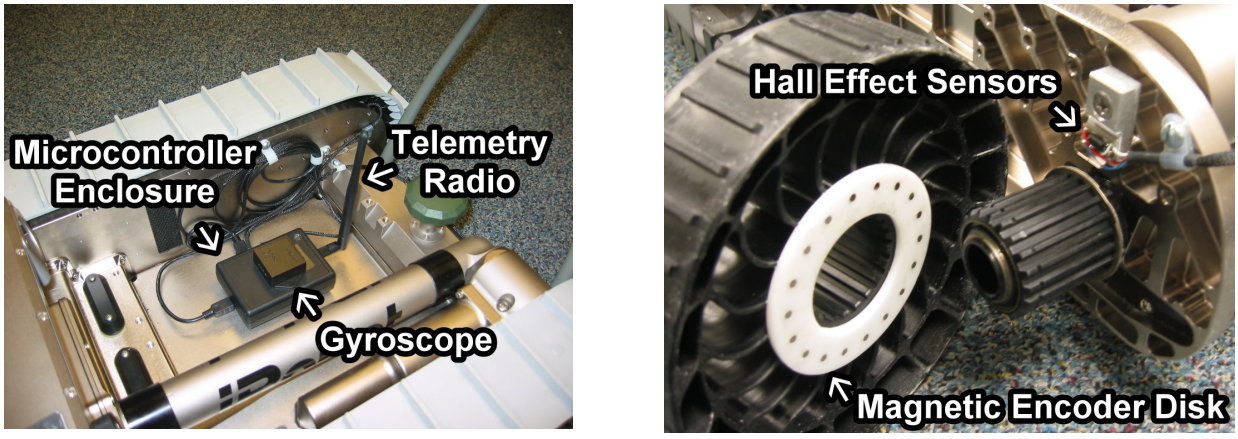


Figure 3: iRobot PackBot outfitted as an HEDR test vehicle.

Our test system consists of a low-cost ($300) MEMS gyroscope (the CruizCore XG1010 made by Microfinity[8]), a serial radio, and an 8-bit microcontroller on a custom PCB contained in an enclosure, as shown in Figure 4a. For odometry, we mounted custom magnetic encoders inside the PackBot's track sprockets. Each encoder consists of a pair of latching hall-effect sensors mounted to the PackBot chassis next to a rear sprocket, and a laser-cut plastic encoder disk sized to fit inside the sprocket. (Figure 4b). Each encoder disk has 18 press-fit neodymium magnets arranged in alternating north-south orientation. As the sprocket turns alternating magnetic poles are passed over the hall-effect sensors generating quadrature signals. While encoder data is available from the PackBot's internal motor encoders, we wanted to build an entirely self-contained system that can be added to virtually any robot, not just the iRobot PackBot, within as little as an hour. The HEDR algorithm performed well in spite of the relatively low resolution of the external encoders.

The system can be powered from a USB host port, a battery pack, or a 5V power supply. The microcontroller directly samples and decodes quadrature encoder signals from the left and right tracks, as well as the rate of rotation measured by the gyro. 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 a laptop where plotting software shows a live trajectory plot to the operator. Figure 5 shows an artist's rendition of how this plot would appear if displayed on the PackBot OCU in place of the GPS map.



(a) (b)

Figure 4: (a) The HEDR system mounted on the PackBot test vehicle; (b) Custom magnetic encoder fitted to a track drive sprocket.

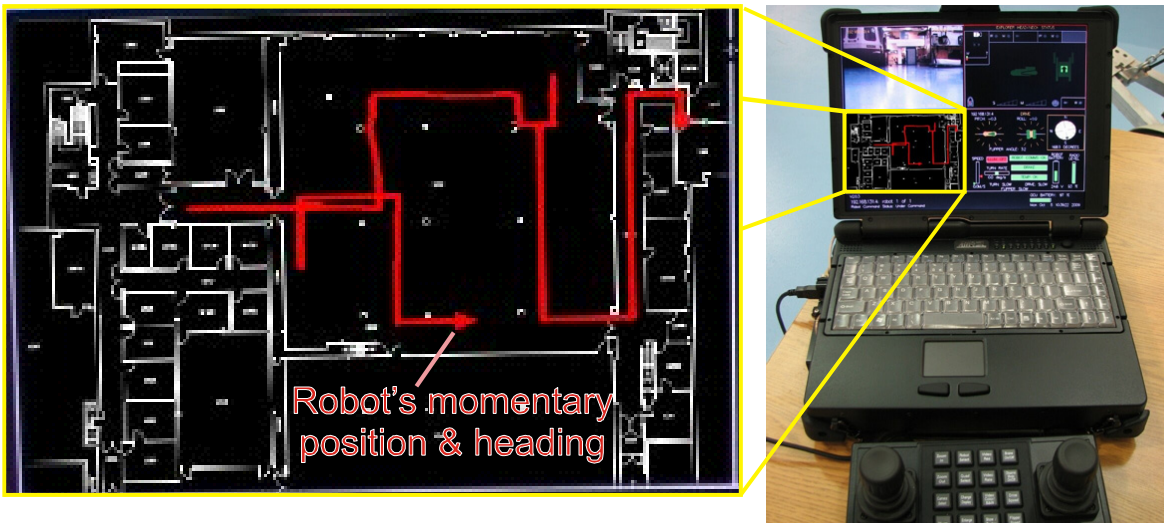


Figure 5: Artist's rendition of a PackBot OCU displaying a live HEDR trajectory in bottom left pane. Current production PackBot OCU's don't have the capability of showing robot trajectories in GPS-denied environments.

4 EXPERIMENTAL RESULTS

In order to illustrate the accuracy of position tracking with the HEDR algorithm, Figure 6 shows the trajectory of a 22-minute/1000-meter drive of a tele-operated, tracked mobile robot (not a PackBot) overlaid over a floor plan of the building in which the drive took place. As is apparent from the parallel trajectory segments of multiple traverses of the same corridors, heading errors are zero at steady state. In a real military scenario, floor plans are likely not available, the trajectory could likely be overlaid over a satellite or aerial photo of the building.

In a more rigorous set of experiments, this time with the PackBot equipped with the "quick-mount" encoder system described in the proceeding section, we performed six indoor runs under tele-operator control. In all six 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 true ground truth for position in each run. We also didn't see the need for that effort since we are providing plotted results for each run.

In all runs a tele-operator controlled the PackBot remotely, using a typical iRobot OCU and using only the onboard camera for feedback. 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 or when the video feedback signal is degraded due to obstructions. 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.

Each run took 20-25 minutes. In Runs 1, 2, and 3 the robot was driven around the corridors and lab space without a specific mission. In Run 4 we emulated a mission of handling an unexploded ordinance. To emulate this task, we marked a 2 m × 2 m square on the floor and placed a cardboard box over one of the corners of the square. The operator then drove the robot along corridors, as before, but then steered the robot such that it would push the box around the marked square. This activity required repeated engagement and disengagement of the robot with the box to push it in the right direction. Of the overall 20 minutes of drive time, 5 minutes were spent on the box pushing task.

Figure 6: Trajectory of a tele-operated tracked UGV 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.

In Run 5 we emulated the task of inspecting a suspicious object. For that task, an experimenter had written messages in marker on the four vertical inside surfaces of a box. During the run, the tele-operator had to maneuver the robot such that he could actually read the messages with the onboard, on-arm camera, and verify all four messages with the experimenter before moving on.

Run 6 was the most challenging for both the operator and the HEDR system. As in Run 4, the tele-operator had to push a box along a marked path, but in this run the path formed the letters UM (for University of Michigan). This activity required repeated engagement and disengagement of the robot from the box, as well as pushing the box in non-dominant directions for brief periods of time. Of the overall 20 minutes of drive time, 10 minutes were spent on the box pushing task.

Table II summarizes the specifications and results for all six runs.

Table II: Specifications and results for the six PackBot runs.

Run #	Total duration [minutes]	Total travel distance [meters]	Return position error (RPE) [meters]	Relative RPE as percentage of distance traveled [%]
1	23	460	2.37	0.52%
2	21	522	1.73	0.33%
3	23	638	1.03	0.16%
4	22	591	0.94	0.15%
5	21	565	1.91	0.34%
6	23	442	2.83	0.63%
Average	22	536	1.80	0.36%

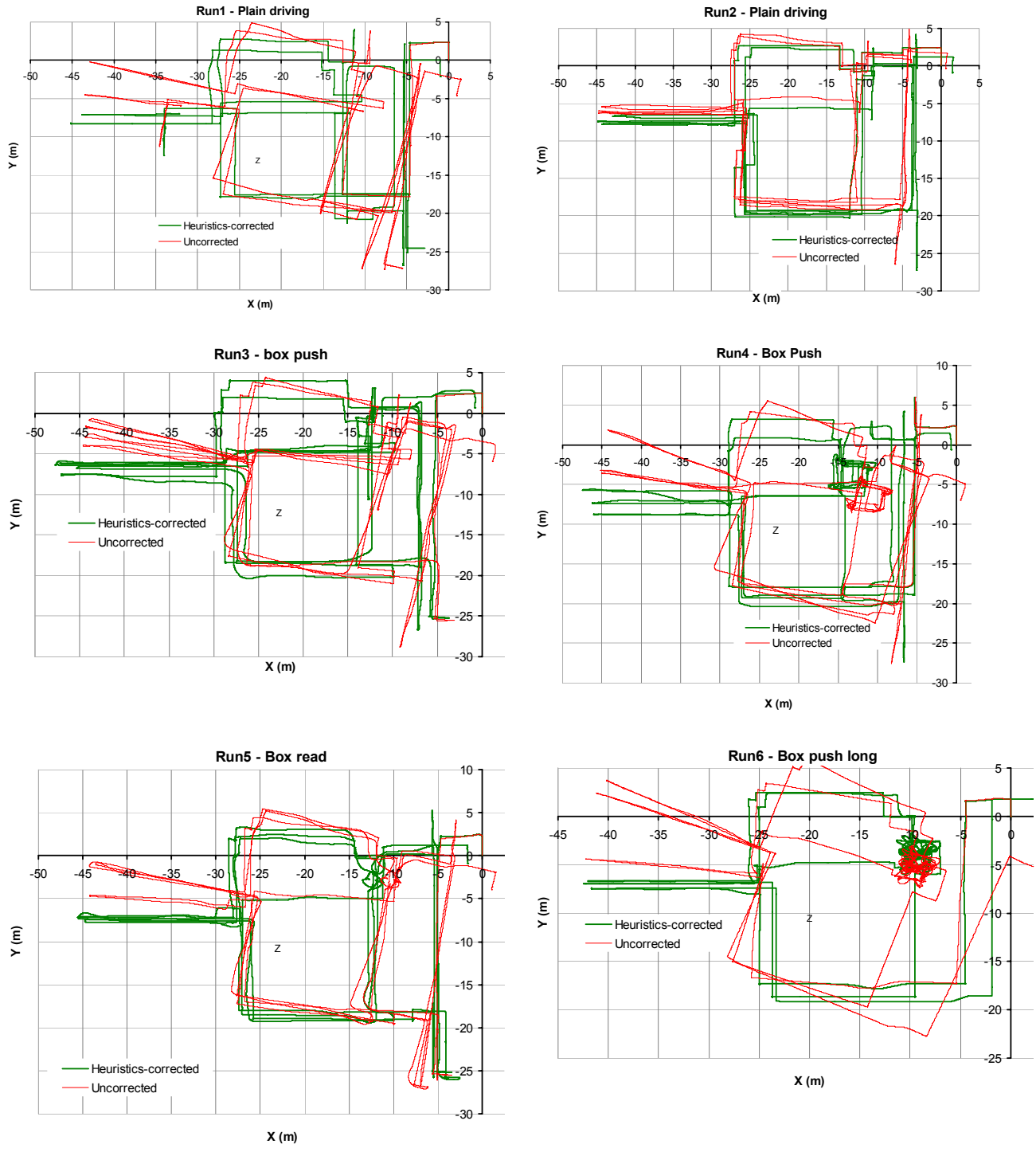


Figure 7: Six runs with a PackBot with conventional (thin red line) and heuristics-enhanced (thick green line) dead-reckoning. Runs varied in duration between 21-23 minutes. In Runs 3 through 6 we challenged the HEDR algorithm by having the robot perform complex tasks other than just driving along corridors.

5 CONCLUSIONS

This paper introduced the "Heuristics-Enhanced Dead-reckoning" (HEDR) algorithm for precision indoor tracking of tele-operated robots. HEDR does not rely on GPS or external references; it uses odometry, a low-cost MEMS-based gyroscope, and heuristic assumptions about the structured nature of most indoor environments. Features of the environment are not directly measured by the system; instead they are inferred from the motion of the robot under the direction of the tele-operator viewing live video from an onboard camera or cameras.

HEDR is applicable in structured indoor environments, in which much (but not all) of the travel occurs along what we call "dominant directions." Most buildings have rectangular footprints, with four dominant directions that are typically parallel to the outside walls and offset by 90° from each other. Among the rare exceptions to this assumption, the most common one is that of buildings that have eight dominant directions, offset by 45°. HEDR is designed to cope with this exception and handles easily dominant directions that are offset by 45°.

The HEDR algorithm itself is relatively simple, and can be implemented in ~20 lines of program code running either on a mobile robot's onboard computer, or on a separate low-cost microcontroller. It can make use of encoders built into a mobile robot, or use our low-resolution "quick mount" magnetic encoders, which can easily be attached to wheels or sprockets on the robot. A tele-operated mobile robot, such as an iRobot PackBot, can run the HEDR algorithm natively, or be retrofitted with an add-on HEDR system operating in parallel with the otherwise stock robot and OCU.

In buildings or other environments that meet the dominant direction criteria, 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.

The only operator feedback offered by most tele-operated robots is the view from a low-to-the-ground onboard camera. Mentally keeping track of the robot's trajectory is difficult and as a result, operators easily become disoriented and lose track of the robot's position and heading relative to the building. The HEDR system alleviates this problem by providing the tele-operator with a map view of the robot's current location and heading, and its previous trajectory, similar to the information provided by an automotive GPS navigation system. This frees tele-operators to focus on controlling the robot and achieving other mission goals, and provides the precise location of the robot if it becomes disabled and needs to be recovered.

A short video demonstration of the HEDR algorithm running on a small tracked robot is linked to Figure 8.

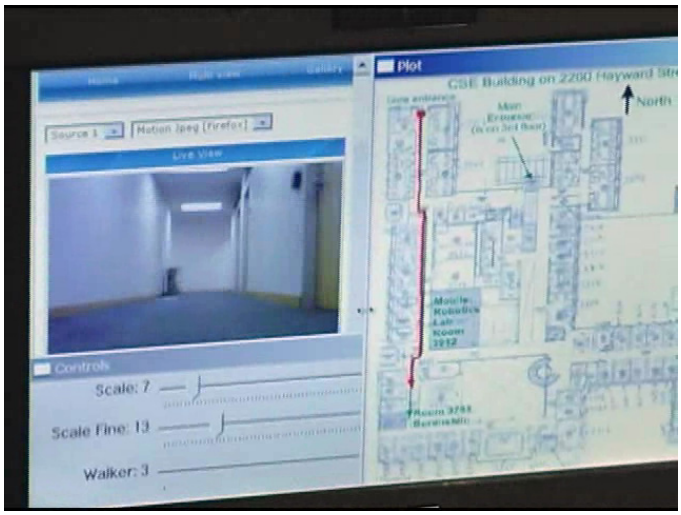


Figure 8: Video 1, HEDR algorithm video demonstration on a small tracked robot http://dx.doi.org/doi.number.goes.here

This video clip can also be downloaded from:
http://www.engin.umich.edu/research/mrl/video/Trackey_TelOp_1mbps.wmv

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. The PackBot used in our experiments was generously donated by iRobot in support of the GRRC.

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] Microstain, Spec-sheet, "3DM-GX1: MicroStrain AHRS Orientation Sensor," http://www.microstrain.com/3dm-gx1.aspx , (2009)

[4] Tardós, J., Neira, J., Newman, P. and Leonard, J., "Robust Mapping and Localization in Indoor Environments using Sonar Data." *International Journal of Robotics Research*, 21(4), 311-330 (2002).

[5] Hogg, R. W., et al., "Algorithms and Sensors for Small Robot Path Following." *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington DC, USA, 3850-3857 (2002).

[6] Nister, D., Naroditsky, O., and Bergen, J., "Visual odometry." *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* , 1, I-652-I-659 (2004).

[7] Borenstein, J., and Ojeda, L., "Heuristic Reduction of Gyro Drift in Vehicle Tracking Applications." *International Journal of Vehicle Information and Communication Systems,* 2(1/2), 78–98 (2009).

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