THE JOURNAL OF NAVIGATION (2012), **65**, 265–279.
doi:10.1017/S0373463311000725

TelOpTrak: Heuristics-enhanced Indoor Location Tracking for Tele-operated Robots

Johann Borenstein[1], Russ Miller[2] and Adam Borrell[3]

[1](*Department of Mechanical Engineering, University of Michigan, MI, USA*)
[2](*Department of Natural Resources and Environment, University of Michigan, MI, USA*)
[3](*Boston Dynamics, Waltham, MA, USA*)
(E-mail: johannb@umich.edu)

With most tele-operated robots the operator's only feedback is the view from an 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. An additional plot of the robot's trajectory would be helpful for the operator and is sometimes provided, based on GPS. However, indoors where GPS is unavailable, tracking has to rely on dead-reckoning, which is too inaccurate to be useful. Our proposed TelOpTrak method corrects dead-reckoning errors even when only odometry and a low-cost (and thus, high-drift) MEMS-class gyro are available on the robot. TelOpTrak corrects gyro drift by exploiting the structured nature of most buildings, but without having to directly sense building features. This paper explains the TelOpTrak method and provides comprehensive experimental results.

Earlier versions of this paper (Borenstein et al., 2010a), (Borenstein et al., 2010b) were presented at two conferences. The main difference between the earlier conference papers and the present manuscript is that the latter is more comprehensive, more up-to-date, and it presents an entirely new set of experimental results, including results of a live demonstration at the 2010 Robotics Rodeo event at Ft. Benning, USA.

KEY WORDS

1. Tele-operated. 2. Location. 3. Heuristic.

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) offer a second window, in which the trajectory of the robot is plotted. However, this window is typically shown only if Global Positioning System (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 (Borenstein and Feng, 1996). Odometry only works well if there is no wheel slippage. If there is much slippage, such as when skid-steer or 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) (Borenstein, 2001). 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 (De Agostino et al., 2010). Fibre optic gyros produce errors at much lower rates, in the order of a few degrees per hour, 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 (Tardós et al., 2002; Levinson and Thrun, 2010). 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 (Nistér et al., 2006; Johnson et al., 2008) 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 also allows for the simultaneous correlation of multiple attributes of features, such as shape, colour, and brightness. However, visual odometry is 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 TelOpTrak. TelOpTrak 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 sense 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) (Borenstein and Ojeda, 2009), with which TelOpTrak shares some elements of approach.

In other earlier work we developed the so-called Heuristic Drift Elimination (HDE) method for tracking the position of persons walking inside buildings (Borenstein and Ojeda, 2010). HDE is a significant improvement over HDR because it actually reduces heading errors to near-zero at steady state. This is accomplished while walking

along straight corridors, even if heading errors were incurred previously, during transients or arbitrary motion. The core component of the HDE method is what we call the 'heuristics engine', which is identical in both applications of the HDE method: the human walker application (Borenstein and Ojeda, 2010) and the tele-operated robot TelOpTrak application of this present paper. The fact that the exact same heuristics engine is applicable with only minor differences to two fundamentally different applications is a testimony to the robustness of the underlying approach. For completeness, we repeat a description of the heuristics engine in Section 2 of this paper.

The remainder of this paper is structured as follows. In Section 2 we describe the relevant background and explain the heuristic approach. Section 3 describes our hardware system designed for a generic skid-steer robot, while Section 4 provides extensive experimental results.

2. HEURISTIC DRIFT ELIMINATION.

This section presents our earlier-developed HDE method applied to tele-operated robots. This discussion was presented originally in our two earlier conference papers and is duplicated here for completeness (Borenstein et al., 2010a), (Borenstein et al., 2010b). New in this present paper is the additional discussion of optional refinements, so-called attenuators, in Section 0. Also new in this paper is a comprehensive set of new experimental results obtained from implementation of the system on a 'Superdroid' All-Wheel Drive (AWD) robot.

The TelOpTrak method almost completely eliminates heading errors due to gyro drift and other slow-changing gyro errors. In suitable indoor environments, TelOpTrak 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.*

TelOpTrak works in environments in which the number of possible heading angles is 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 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. We call environments that conform to these architectural properties *conforming* environments.

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

In compliant environments, TelOpTrak detects when motion matches one of the four dominant directions and gradually corrects gyro output. During turns, TelOpTrak suspends its corrective action. While TelOpTrak is suspended, drift causes new heading errors, but once TelOpTrak resumes, it effectively eliminates accumulated heading errors because it gradually nudges headings toward alignment with the

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

closest dominant direction. When motion is mostly compliant, TelOpTrak 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.

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_d. Relative heading can be computed from:

$$\psi_i = \psi_{i-1} + \omega_{meas,i} T \tag{1}$$

where:

ψ_i is the computed heading after interval i, in [°].
$\omega_{meas,i}$ is the true rate of turn plus the unknown drift error $\varepsilon_{d,i}$, in [°/s].
T is the sampling time in [s].

In the following sections we explain how the TelOpTrak algorithm:

- Reduces driving in any of the nominally four dominant directions to the functional equivalent of driving in a direction of zero degrees.
- Models drift, 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.

As explained, the TelOpTrak algorithm assumes that a building has four dominant directions, Ψ. Dominant directions are spaced at 90-degree intervals called the *dominant direction interval, Δ.* A further assumption is that most corridors and inside walls in a 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 TelOpTrak algorithm to work well, this latter assumption does not have to hold true all of the time.

2.3. *The Heuristics Engine*. The core component of the TelOpTrak algorithm is the heuristics engine, which functions essentially like a feedback control system, as illustrated by Figure 1.

We start the explanation of this feedback control system with the signal from the gyro, ω_{meas}, which is modelled 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. When driving straight, $\omega_{meas} = \varepsilon_d$, so the only output from the gyro is drift, ε_d.

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

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

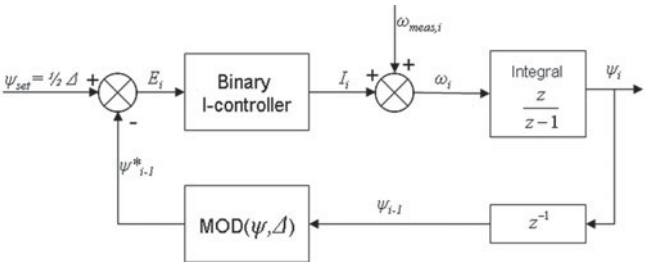

Figure 1. The core component of the TelOpTrak algorithm, called "heuristics engine," is modelled as a feedback control system. The block labelled "Binary I controller" is explained in the narrative.

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.

The block labelled "MOD(θ,Δ)" applies the MOD function, which is defined as:

$$\mathrm{MOD}(n, d) = n - d\mathrm{INT}(n/d) \tag{2}$$

MOD(ψ,Δ) maps ψ onto a direction that lies between 0 and Δ. With the help of the MOD function the heuristics engine performs a simple test to see if a momentary heading angle is immediately to the right or left of *any* dominant direction:

$$\psi_i^* = \mathrm{MOD}(\psi_{i-1}, \Delta) \tag{3}$$

where:

ψ_i^* is the 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:

$$E_i = \Delta/2 - \psi_i^* \tag{4}$$

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 small, fixed increment, i_c. If E is negative (i.e., the 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 the 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 robot 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 Equation (4), 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

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

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:

$$I_i = \begin{cases} I_{i-1} - i_c & \text{for } E < 0 \\ I_{i-1} & \text{for } E = 0 \\ I_{i-1} + i_c & \text{for } E > 0 \end{cases} \tag{5a}$$

where:

i_c is a fixed increment, also considered the gain of the binary I-Controller in units of degrees.

An alternative way of writing Eq. (5a) is:

$$I_i = I_{i-1} - \text{SIGN}\left(\psi^*_{i-1} - \frac{\Delta}{2}\right) i_c \tag{5b}$$

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:

$$\omega_i = \omega_{meas,i} + I_i \tag{6}$$

where:

ω_i is the corrected rate of rotation [°/s].

Substituting Equation (3) and Equation (5b) in Equation (6) yields:

$$\omega_i = \omega_{meas,i} + I_{i-1} - i_c \text{SIGN}\left(\text{MOD}(\psi_{i-1}, \Delta) - \frac{\Delta}{2}\right) \tag{7}$$

While Equation (7) represents the complete TelOpTrak algorithm in principle, in practice an additional software component is required, as explained next.

2.4. *Low-Pass Filter and De-Lagging*. In practice, noise in the gyro signal substantially reduces the effectiveness of the TelOpTrak algorithm. To alleviate this problem, it is necessary to apply a low-pass filter to the raw gyro data. This can be done easily in software:

$$\omega'_i = \frac{\omega_{meas,i} T_i + \tau \omega'_{i-1}}{T_i + \tau} \tag{8}$$

where:

T_i is the sampling time. $T = 0{\cdot}1$ s in the TelOpTrak system.

ω' is the low-pass filtered rate of turn.

τ is the low-pass filter time constant.

We determined experimentally that a low-pass filter with a time constant of $\tau = 200$ s (i.e., a cut-off frequency of $f_{c^\circ} = 1/\tau = 0{\cdot}005$ Hz) is effective in our system. However, such a low cut-off frequency introduces substantial lag. If the low-pass filtered ω was used to compute heading and subsequently the x-y position, the trajectory would look overly smooth, and sharp turns would be misrepresented by moderately curving ones.

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

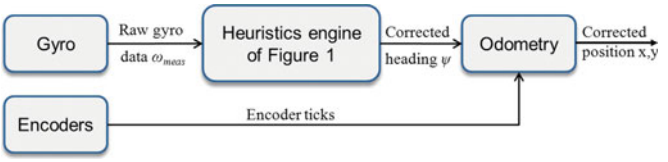

Figure 2. Block diagram of the complete TelOpTrak system.

To overcome this problem, we make use of the fact that the effect of a digitally implemented low-pass filter is fully reversible, simply by inverting Equation (8):

$$\omega_{d,i} = \omega_i + \frac{\tau}{T_i}(\omega_i - \omega_{i-1}) \tag{9}$$

where $\omega_{d,i}$ is the de-lagged rate of turn [°/s].

Figure 2 illustrates the four processing steps in the TelOpTrak algorithm:

1. Apply Eq. (8) to low-pass filter the raw gyro data.
2. Apply the core TelOpTrak algorithm Equation (7), but using the low-pass filtered ω' of Equation (8) instead of $\omega_{meas,i}$.
3. De-lag the computed ω_i of Equation (7) by applying Equation (9). After this step, the resulting ω has no lag and is essentially drift-free at steady state.
4. Additionally, compute the corrected heading, ψ_i, by rewriting Equation (1):

$$\psi_i = \psi_0 + \psi_{i-1} + \omega_{d,i} T_i \tag{10}$$

where ψ_0 – initial heading at the beginning of the drive [°].

Step 4 is needed since ψ_{i-1} is required in Equation (7).

Figure 2 shows a block diagram of the TelOpTrak system. TelOpTrak does not require mathematical models of the robot, the gyro, or the environment, and it requires the tuning of only two parameters: τ and i_c.

The caveats are:

- The method works only with tele-operated robots driving through mostly conforming buildings. Conceivably, the method will also work with autonomous robots that have obstacle avoidance and wall following capabilities, but we did not test such a configuration.
- Much of the driving must be compliant (i.e., along dominant directions).

When these conditions are not met, then the algorithm with eventually fail by 'snapping' to a wrong dominant direction.

TelOpTrak can be used with gyros of different quality levels. The results will differ only insofar as that with a high-quality gyro, the gain i_c can be set to a small value. The effect is greater robustness in the face of non-compliant driving. In practice, we have driven our robots non-compliantly for ten minutes and more with a $300 gyro. Indeed, in one particular implementation of TelOpTrak on a differential-drive robot, we used no gyro at all. Instead, rate of turn was estimated from the output of the wheel encoders.

2.5. *Additional Refinements – Attenuators.* The TelOpTrak method as described up to this point is fully functional and the experimental results presented in Section 4

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

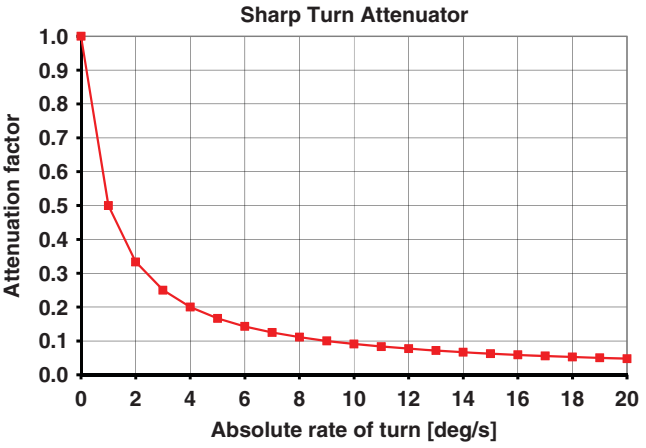

Figure 3. One possible profile for the sharp turn attenuator.

are based on the implementation as discussed so far. However, depending on the specifics of the application, it may be desirable to introduce additional refinements. In this section we discuss some possible refinements and a common mechanism for implementing them.

Our general approach for refinements makes use of adaptive gain functions. With these functions, the nominal gain i_c is reduced when the system detects certain adverse conditions. This is accomplished by multiplying i_c with so-called attenuators in every iteration. The attenuators can change between zero and one. If no adverse conditions exist, then the attenuators are all set to '1', thereby leaving the gain unchanged. If one of the attenuators is set to '0', then TelOpTrak corrections are effectively suspended during that interval. For the TelOpTrak system we identified three candidate attenuators: (1) the sharp turn attenuator, (2) the speed attenuator, and (3) the absolute heading difference attenuator.

As an example, we discuss the function of the sharp-turn attenuator. We know for sure that whenever the robot is turning, TelOpTrak cannot correct heading errors. To prevent TelOpTrak from 'correcting' heading incorrectly while turning, the rate of turn (ω_i) is monitored and attenuated as shown in Figure 3 according to this function:

$$A_{1,i} = \frac{1}{1 + c_1|\omega_i|} \tag{11}$$

where:

c_1 is the sharp-turn attenuation constant
A_1 is the sharp turn attenuator

It is apparent from Equation (11) that $0° < A_1 \leqslant 1$ for all possible rates of turn. Other functions are possible, as long as they keep A_1 between 0 and 1, in some inverse (but not necessarily linear proportion) to ω_i.

Another attenuator that produced some improvements when we tested it is the speed attenuator, A_2. It is proportional (but not necessarily linearly) to the robot's speed. The assumption is that the robot is moving relatively fast when driving down a

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

Figure 4. Our Superdroid ATR skid-steer robot instrumented for tele-operation.

corridor, and relatively slowly when performing an object handling or inspection task. The speed attenuator can also be implemented as a simple threshold, th_2, where $A_2 = 0$ for speeds below th_2 and $A_2 = 1$ for speeds above th_2.

The third candidate attenuator is probably only useful when working with a higher-end gyro. This so-called 'heading-difference attenuator' examines the difference between the current heading and the nearest dominant direction. The rationale is that whenever that difference is large (e.g., greater than, say, 20°), it must be because the robot is indeed not driving along a dominant direction and therefore TelOpTrak corrections should be reduced or suspended. This is a reasonable approach for high-end gyros, because true errors with those gyros (provided TelOpTrak is running) will always remain small. However, with low-quality gyros, drift-induced errors can indeed be very large, for example, due to extended non-compliant driving. It is therefore not a good idea to suspend TelOpTrak when the heading difference is large; that difference may indeed be the result of an error that should be corrected.

Any number of attenuators can be applied by modifying Equation (7):

$$\omega_i = \omega_{meas,i} + I_{i-1} - A_1 A_2 i_c \mathrm{SIGN}\left(\mathrm{MOD}(\psi_{i-1}, \Delta) - \frac{\Delta}{2}\right) \tag{12}$$

The 'cost' of implementing one or more attenuators is the additional number of parameters that can be tuned. We found that the benefits provided by the attenuators was marginal, and therefore we did not use any of them in the final system that produced the experimental results reported in Section 4.

3. THE EXPERIMENTAL SYSTEM.

The TelOpTrak algorithm was tested on a skid-steer 'Superdroid' All Terrain Robot (ATR), instrumented with our custom-designed control electronics (see Figure 4). The ATR chassis features a welded aluminium structure with four Direct Current (DC) gear motors, which drive encoder-equipped wheel axles via roller chains.

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

The TelOpTrak algorithm runs on an 8-bit microcontroller that is mounted on a custom Printed Circuit Board (PCB). The PCB accepts encoder feedback from the front left and right wheel encoders, samples the output of a low-cost MEMS gyroscope (Cruizcore XG1010 made by Microinfinity), and interfaces with a 900-MHz mesh network digi-Xtend serial radio. The microcontroller applies the TelOpTrak algorithm to the gyro data, and uses encoder data and vehicle-specific parameters (encoder resolution, effective wheel diameter, etc.) to calculate the TelOpTrak-corrected position and heading of the robot 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.

In order to provide tele-operation capability similar to that of small tactical robots in military and law enforcement use, a forward looking, wide angle, VGA-resolution Internet Protocol (IP) camera was mounted near the rear of the robot. The mounting height and angle of the camera give a good view of a vertical arc between the horizon and the ground approximately 25 cm ahead of the front wheels, which are also in the field of view to aid the operator in avoiding obstacles. Although they provide relatively long range, the bandwidth of the serial radios is limited to less than 20 kbps in our usage, which is inadequate to provide video frame rates for tele-operation. For that reason we added a high-power 802·11 g WiFi access point to the robot, to transmit the video from the onboard camera.

4. EXPERIMENTAL RESULTS. We implemented the TelOpTrak algorithm as explained in Section 2 on the Superdroid ATR of Section 3 and performed six runs under controlled conditions. In the first five runs the robot was driven mostly in a compliant manner (i.e., driving mostly straight along corridors that intersected at 90°). However, from run to run we increased the challenges for the TelOpTrak algorithm by performing more and more intentionally non-compliant manoeuvres. These manoeuvres included zig-zagging, driving around in tight circles, and avoiding imaginary or real obstacles. In the sixth run the operator drove *entirely* non-compliantly, that is, in completely arbitrary patterns and while avoiding arbitrarily placed obstructions. In that run the heuristic assumptions were almost never true.

Results of all six runs are shown in Figure 5. In each run the robot started at a position labelled (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 conventional dead-reckoning and based on TelOpTrak with the actual final position (0, 0). The difference between the two is called the Return Position Error (RPE) and the RPEs for all six runs are listed in Table 1. 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. Therefore, for the more structured Runs #1 through #5 we also constructed ground truth for heading from the known direction of the corridors. This allows a quantitatively more meaningful comparison of the heading errors as shown in Table 1. Heading errors were not computed for the non-compliant Run 6 since no ground truth was available or could be constructed for this intentionally erratic run.

In all runs the tele-operator controlled the robot remotely and without line of sight, using only video from the onboard camera for feedback. We emphasize this fact because under these conditions, tele-operated driving is much more erratic than

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

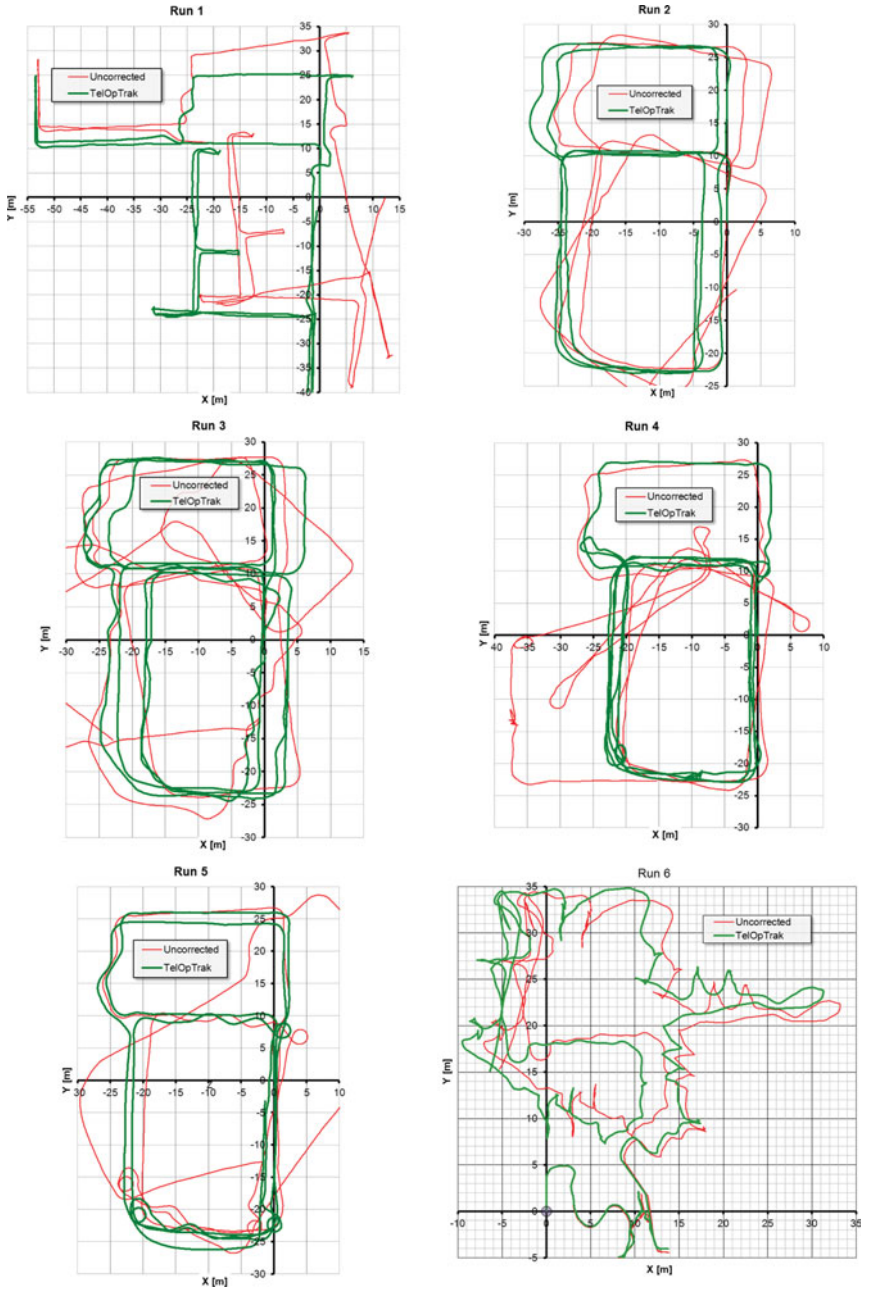

Figure 5. Six runs with TelOpTrak on the Superdroid under controlled conditions. Thin red curves show the result of conventional dead-reckoning, while thick green lines show the result with TelOpTrak. Runs varied in duration between 10–14 minutes. Run 6 was performed inside one very large room and driving was intentionally erratic and arbitrary. Quantitative results for these six runs are provided in Table 1.

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

Table 1. Specifications and results for the six Superdroid runs.

Specifications			Return Position Error (RPE)				Average Heading Error		
			Without TelOpTrak		With TelOpTrak				
Run #	Total duration [min]	Total travel distance [m]	Absolute [m]	Relative to distance travelled [%]	Absolute [m]	Relative to distance travelled [%]	Without TelOpTrak [°]	With TelOpTrak [°]	Relative reduction of heading errors with TelOpTrak
1	11·9	477	12·3	2·6	2·88	0·60	5·60	1·44	3·9-fold
2	11·4	433	10·4	2·4	4·49	1·04	10·7	1·48	5·9-fold
3	13·4	629	27·5	4·4	3·93	0·62	35·4	2·21	16-fold
4	11·2	497	27·4	5·5	1·58	0·32	28·4	1·81	16-fold
5	10·3	468	12·8	2·7	3·44	0·74	21·0	1·59	13-fold
6	12·0	263	16·2	6·1	16·4	0·60	N/A	N/A	
Average*	11·6	501	18·1	3·5	3·26	0·64	20·2	1·71	13-fold

* Note: Run 6 is not included in the averages since it was intentionally performed with unrealistic, extremely irregular motion. The intent was to show that even in such extreme cases, TelOpTrak performance is not worse than performance without TelOpTrak.

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 resulted in significant overshoot. Under these conditions, the TelOpTrak system is fully challenged since some of the driving is zigzagging and otherwise not very straight, even in straight corridors and when we attempted to drive compliantly.

A more realistic experiment was performed at the 2010 Robotics Rodeo at Fort Benning, in September 2010. In this run our Superdroid robot was driven inside one building of a Military Operations on Urban Terrain (MOUT) site. The tele-operator and the audience were located in a nearby building. The tele-operator simulated a reconnaissance mission by driving the robot through the corridors and some of the rooms of the building. Only video from the onboard camera and the TelOpTrak-generated trajectory plots were available to the operator as feedback.

We note the presence of an audience because the tele-operator was simultaneously operating the robot and narrating the experiment for the audience, a condition that resulted in stress and distractions and led to several consequential operator errors. In the remainder of this section we describe this experiment and the unintended operator errors in some detail because operator errors are an integral part of tele-operated systems and they highlight the utility of the TelOpTrak system.

Figure 6 shows four key instances during the demonstration. The developing trajectory was plotted over a detailed floor plan in Figure 6 to illustrate the accuracy of the trajectory plot. However, during the actual demonstration, while the trajectory developed on a large screen in front of the audience and the operator, only the contour of the building (but not the detailed floor plan) was shown as the fixed background.

After starting, the operator drove the robot to the right, into the first visible corridor, with the intention of performing a Counter-ClockWise (CCW) surveillance run on the first floor. Due to the one-second latency of the video feedback, the operator missed the left turn into the corridor leading CCW and drove instead into the room in the lower right corner (see Figure 6a). He noticed and turned around, but when he thought the robot was in the CCW-leading corridor, he was actually driving back toward the entry door.

The operator noticed his mistake only when he looked at the trajectory plot on the screen, as shown in Figure 6a. He then turned back and eventually followed the corridors. When the robot reached the upper left corner of the corridors (Figure 6b), the operator decided to investigate that area further and steered the robot into a room. However, seconds after driving into that room the wireless video feedback signal deteriorated, producing even larger latencies and repeated freezing, until it eventually blanked out entirely. In order to recover from that complete loss of video feedback (a condition from which conventional tele-operated robots cannot recover), the operator decided to backtrack using the TelOpTrak trajectory plot alone for guidance. Figure 6c shows the robot during the backtrack manoeuver, which eventually led the robot out of the room.

Once the robot was out of that room and back in the corridors, the video signal was re-acquired and allowed the operator to drive the robot back to the original starting point to complete the trip. Figure 6d shows the final state of the trajectory, reflecting an estimated return position error of about 1·5 metres.

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

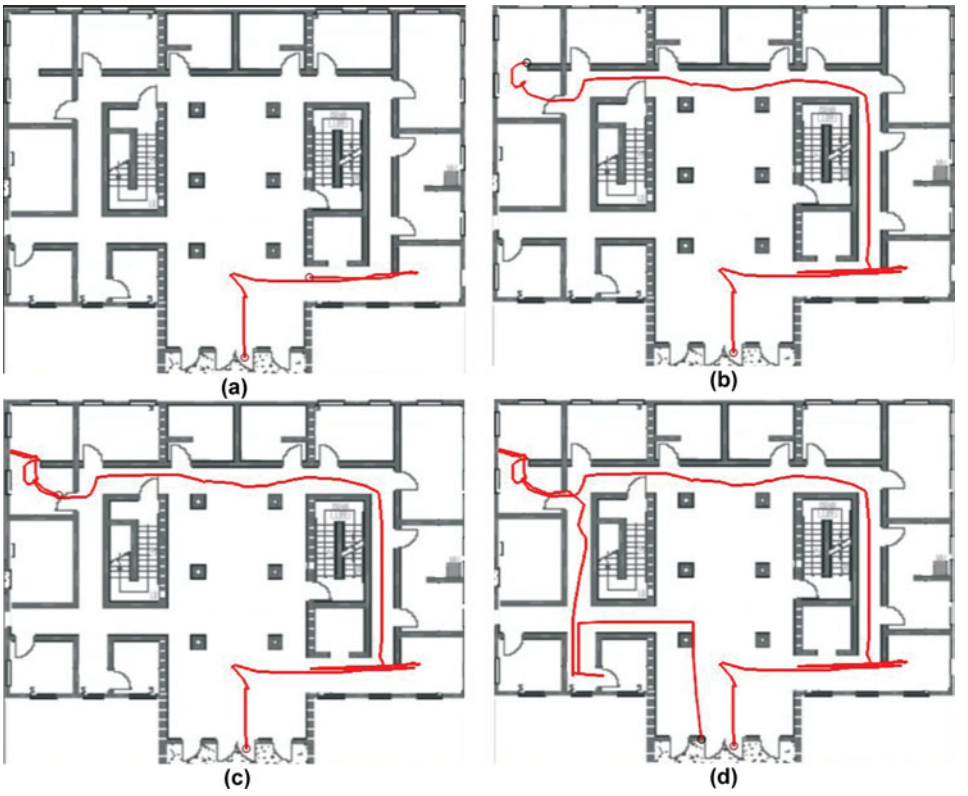

Figure 6. Trajectory plots during different stages of the RTOD2 demo. In order to highlight the accuracy of the plot, we overlaid the trajectory over the actual floor plan of the building in these plots. However, during the actual demonstration and to match realistic conditions, the operator's screen only showed the contour of the building, but no floor plan details.

A video clip, recorded during the demo by a videographer walking behind the robot, shows the robot during the demo. An animated and synchronized trajectory plot is shown as well, picture-in-picture style. This and other TelOpTrak video clips are available at http://mrl.engin.umich.edu/video/RTOD2_2xspeed.wmv.

5. CONCLUSIONS. This paper introduced the TelOpTrak algorithm with heuristics-enhanced dead-reckoning for precision indoor tracking of tele-operated robots. TelOpTrak 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.

TelOpTrak is applicable in structured indoor environments, in which much (but not necessarily 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.

https://doi.org/10.1017/S0373463311000725 Published online by Cambridge University Press

The TelOpTrak algorithm itself is relatively simple. It can be implemented in just a few lines of program code running either on a mobile robot's onboard computer or on a separate low-cost microcontroller.

In buildings or other environments that meet the dominant direction criteria, TelOpTrak eliminates heading errors caused by gyro drift and other slow-changing sources of errors, effectively maintaining zero heading errors in drives of unlimited duration at steady state. As a direct result, TelOpTrak significantly reduces position errors, as accumulated heading errors are almost always the primary source of position errors in a dead-reckoning system.

ACKNOWLEDGEMENTS

This research was supported by the Ground Robotics Reliability Centre (GRRC) at the University of Michigan, with funding from government contract DoD-DoA W56H2 V-04-2-0001 through the Joint Centre for Robotics.

REFERENCES

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

Borenstein, J. and Ojeda, L. (2009). Heuristic Reduction of Gyro Drift in Vehicle Tracking Applications. *Proceedings of the International Journal of Vehicle Information and Communication Systems*, **2**, 78–98.

Borenstein, J. and Ojeda, L. (2010). Heuristic Drift Elimination for Personnel Tracking Systems. *The Journal of Navigation*, **63**, 591–606.

Borenstein, J., Miller, R., Borrell, A. and Thomas, D. (2010a). Heuristics-Enhanced Dead-Reckoning (HEDR) for Accurate Position Tracking of Tele-operated UGVs. *Proceedings of the SPIE Defence, Security + Sensing; Unmanned Systems Technology XII*, Orlando, Florida.

Borenstein, J., Miller, R., Borrell, A. and Thomas, D. (2010b). Heuristics-Enhanced Dead-reckoning for Improved Situation Awareness with Tele-operated Robots. *Proceedings of the 2010 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*, Detroit, MI.

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

De Agostino, M., Manzino, A. M. and Piras, M. (2010). Performances comparison of different MEMS-based IMUs. *Proceedings of the Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, Indian Wells/Palm Springs, California USA.

Johnson, A. E., Goldberg, S. B., Cheng, Y. and Matthies, L. H. (2008). Robust and Efficient Stereo Feature Tracking for Visual Odometry. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Pasadena, CA.

Levinson, J. and Thrun, S. (2010). Robust vehicle localization in urban environments using probabilistic maps. *Proceedings of the IEEE International Conference on Robotics and Automation* (ICRA), Anchorage, Alaska.

Microfinity, Spec Sheet, http://www.cruizcore.com.

Nistér, D., Naroditsky, O. and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, **23**, 3–20.

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