The Fight Against Error: Dead Reckoning Effectiveness with Implementation of Correction Methods

1. Introduction

1.1 What is Dead Reckoning?

Dead reckoning is the process of estimating one's current position based upon a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time, and course. [1] A fix is a term used in navigation that entails a position that is derived from the known or obtained measure of two or more external references. One of these references can be the abstract current position. This idea is often used in robotic navigation (see 2.2 Robotic Navigation). Dead reckoning relies on fixes as a way of determining a position through the factors of an initial fix with direction taken, elapsed time, and estimation of speed. Traditional dead reckoning refers to any movement through the use of predetermined speed, time, and direction based on external references. Robotics limits this definition to movement based on the use of predetermined speed, time, distance, and direction with no sensory information not involved in these factors. In this paper, speed, distance, time, and direction will be referenced as the four fix factors of dead reckoning [1].

1.2 When is it Used?

Dead reckoning is used in many real world applications such as naval, aeronautic, automotive, and animal movement. In animals, dead reckoning, also loosely referred to as path integration, is used to determine current position based on what steps and directions were taken from the latest previously known location. In robotics, path integration is usually used by reference to the starting position from which the robot is run. Path integration in animals allows for things like being able to return to a nest or remember the layout of a territory due to many trips in all directions from the nest. This process gets even more effective when factoring in three dimensions, due to more overlapping trips; such as in flight [1].

In mechanical dead reckoning, the accuracy of navigation is dependent on consistent retrieval of fixes, or fixtures for one of the four fix factors of dead reckoning. Although sensory input is not in the essence of dead reckoning, sensors are often used in automotive technology to implement dead reckoning such as sensors that track wheel rotation and steering direction. This use of sensors to monitor dead reckoning is used in electronic stability control. Note that these sensors are not guiding navigation, but are making sure that the machine is simply keeping consistency of the current path.

1.3 Misconceptions

It is common misconception that dead reckoning is not usable in the field of robotics. It is believed that the error that is created when using dead reckoning is too great and not manageable or effective to solve navigation problems. The example that will be used to combat this belief is the Botball challenge of 2010 and the ways that correction methods used by team 10-0160 served a CBC that runs completely off of dead reckoning and is effective in collecting clean ducks.

2. Downfalls of Dead Reckoning and How They Can be Avoided

2.1 Robot Navigation

Turn error is one of the biggest problems in robotic dead reckoning. Because dead reckoning direction settings are not based on sensory input other than that of initial positions and directions, all turns are dependant on physical parts functioning and not on abstract calculations. In the real world environment, many things contribute to either the miscalculation or implementation of turn accuracy. These could include inability of the hardware to exact a measurement leading to rounded turns. Also included in these causes are environmental factors such as the texture of the ground traveled or object disorientation (see Figure 1). Texture can create slippage which can be neither accounted for nor accurately predicted as it happens based on instantaneous wheel to ground friction pressure.

The effects of turn error can be very substantial. When programming and testing a dead reckoning implementing robot, it can be observed that the discrepancy between robot position and predicted position varies. For example, in a path with one right turn of 90 degrees a turn error 5 degrees to the right can result in a distance discrepancy of 0.87 inches from the predicted end point to the actual endpoint.

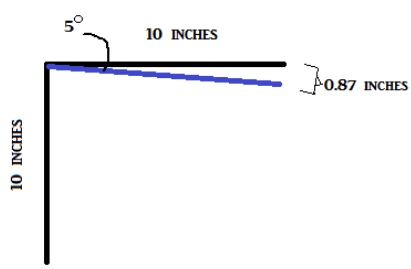

Figure 1 Effect of Turn Error

While this discrepancy may not seem like much it can however, when observing complicated dead reckoning paths, magnify the turn error over time. As progressions of error-inflicted turns occur, the error in a previous turn increases the total error from the next turn. This phenomenon is known as Abbe error. In laymen's terms, at the worst case scenario, the turn error may double at every turn. For example examine the following segment of the path of our robot's path.

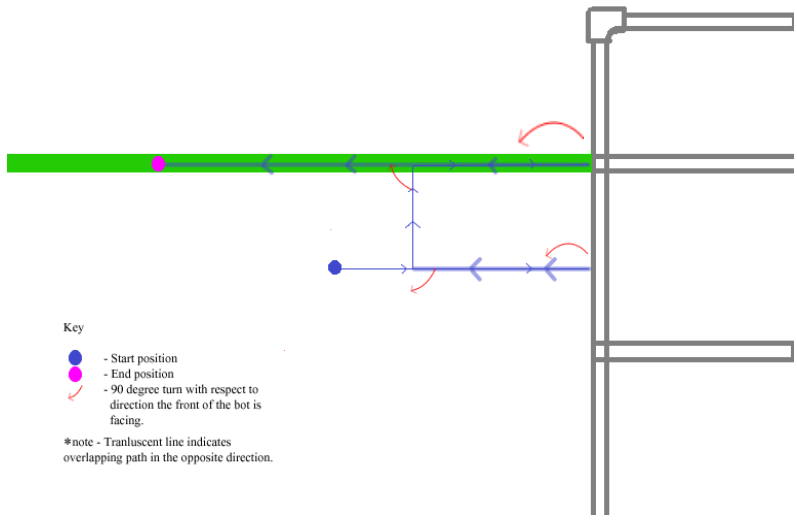

Figure 2. Dead Reckoning Path Segment without Turn Error

Now here is a representation of the same path with a 5 percent turn error in either direction.

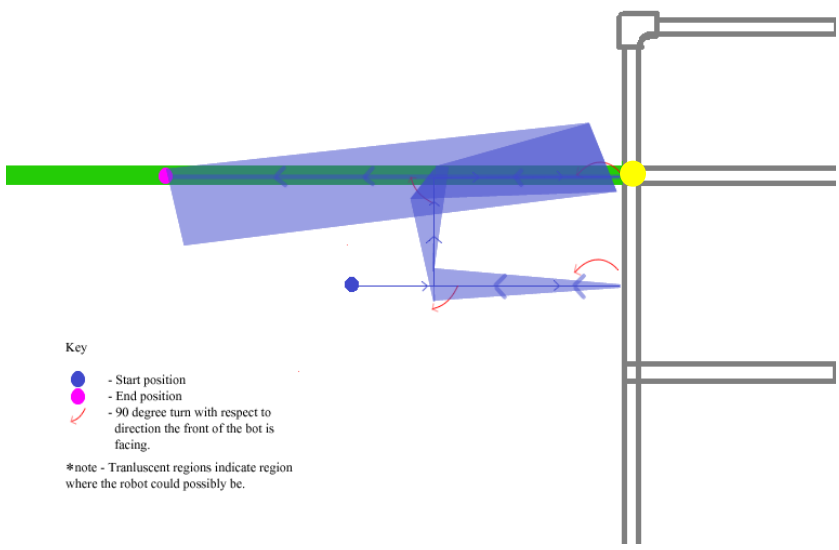

Figure 3. Dead Reckoning Path Segment with Turning Error

The region of space that the robot could end up in increases and so does the distance from the predicted endpoint. However, the effects of turn error do not end here. Turn error results in yet another problem with navigation. This problem is known as fix expansion. As said above a fix is a position from which a dead reckoning robot can obtain its position and direction, and from which it determines the rest of its path. Because of Abbe error the region from which the robot obtains its fixes (in this case since no correction methods are in place, the fixes are the turns and any points that have been decided by predetermined distances) increasingly varies. The increase of this space in which the region of space which the robot will next fix itself is called fix expansion. Fix expansion is the reason that the end position will be off, while Abbe error is the cause of fix expansion.

Another type of error present in dead reckoning is distance error. Distance error can lead to a very large instability in dead reckoning paths. Distance error is the inconsistency of the length of a robot's travel per 1 unit effort. The effort can be different depending on the mode of movement of a robot. For example, one full rotation of a wheel could be 1 unit effort. Now even though the circumference of a wheel cannot differ, the way in which it is turned and the way in which the turn itself moves the robot can. This means that in the process of turning the wheel, error can occur. A common wheel error is wobbling in the axle or wobbles by wheel shape. [3] Wobble can cause imbalance between sides of the robot and lead to tilt of movement in either direction. The effect of the wheel's turn on movement of the robot relates to distance error, which could be caused by a slippage in gears of the wheel or slippage from the surface traveled. Distance error in this way is not relatively consistent like turn error. It is also, relatively impossible to predict.

Now that the downfalls of robotic movement error have been outlined it may seem as if dead reckoning is not an effective approach to solve any navigation problem. However, these downfalls are correctable and reducible by some very easy methods.

The back align is a correction method that is very commonly used in Botball and is effective. A back align allows us to face forward in the right direction before undergoing the next element of navigation; but we often under estimate how much error it actually accounts for. Let's take into consideration our previous turn error illustration. (Figure 3)

Let us consider doing a back align at the point where the yellow dot is located. A back align here would account for all of the horizontal (in this diagram) positioning error that was created by previous turn error. The new error is greatly reduced and when taking into consideration that the thickness of the robot is greater than the green line, the final position of the robot would have a much greater probability of being on the green line (bear in mind that the illustration is not exact).

Another turn error result, fix expansion, may be a problem; however, it is expected by navigators in the real world and so can be accounted for. By making fix expansion calculations, navigators can know how much error there can be in their position. They can thus account and look for a fix location that would be suitable to lower the error no matter where they are in the possible region. For example, given the Figure 4 (with an implemented back align), because the position could be anywhere in the gray region, it can be concluded that there is a greater chance that the final position of the robot will have a greater probability of being too far to the right of the line than to the left. Coupled with these visual calculations, in the real world, navigators use motion sensors (accelerometers) and rotation sensors (gyroscopes) to continuously calculate via dead reckoning the position, orientation, and velocity of a moving object without the need for external references. This is termed inertial navigation and is a sophisticated correction method. It can greatly reduce the effects of fix expansion. In this way aligns and digital sensors can be a great way to fix dead reckoning implementing robots for distance error as well [4].

Another very simple correction method is known as a twitch. A twitch is used when a navigational error exists that can't be fixed (like a robot having a tendency to go to the left in a particular area). In dead reckoning it is a small heading correction, typically used to account for a known error that occurs in movement. As such twitches are normally used for planned routes and used to correct error that was not already accounted for. They are instantaneous corrects for at-a-

point corrections. However, relative to the whole path, they can improve the accuracy o the end position.

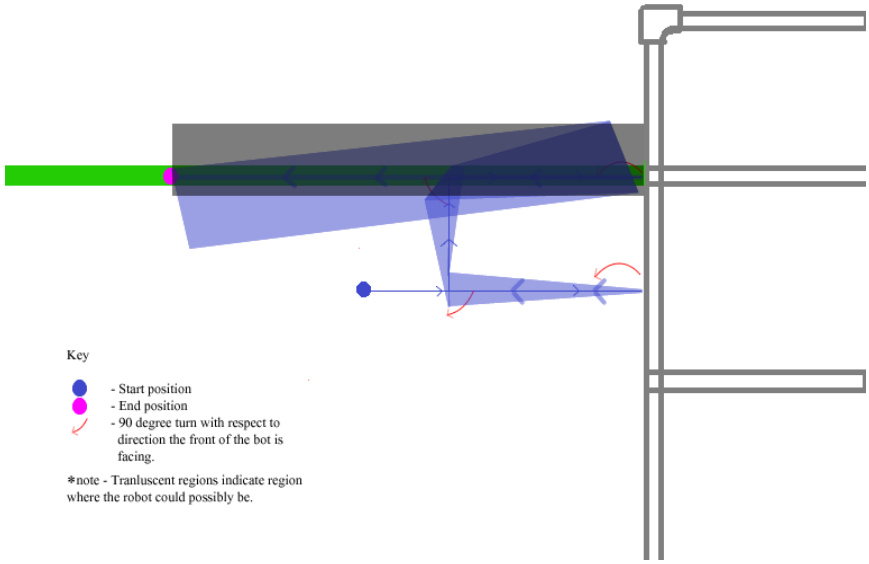

Figure 4. Dead Reckoning Path Segment with Turning Error After a Back Align

2.2 Interaction with Objects in the Environment

In Team 10-0160's efforts to obtain all 7 ducks and place them on the green line, we used complete dead reckoning to position the robot and grab ducks. The positioning of the robot in the path to obtain the first 3 ducks and place them on the line is shown in the illustrations above.

Another problem that can occur is that the positioning of the robot can give error. This can also be caused by the positions of the ducks. The robot is using dead reckoning to capture the ducks in its claw and so must be able to position itself into the position it thinks the ducks should be.

A jig is one way that our team made sure that the ducks' positions remained accurate and consistent. For each run this provided a great increase in the chances of the robot obtaining the ducks as they would always be in the right positions.

Another correction method that was implemented was correcting the duck positions with the claw itself. When going to grab the ducks a loop of closing and opening the claw provided for realignment of ducks and for centering the ducks in proper position relative to the claw. This provided a fix (duck repositioning) if the robot was not in the exact position it should be. The long tube shape of the claw allowed for loss and improper capture of the ducks. Adding simple open and close movements before actually grabbing with the claw aligns the ducks and can even put them from an askew to an upright position. (A correction is modeled in Figure 5 below.)

3. Conclusion: The Impact of Simplistic Methods & Comparison to Sensors

These correction methods are simple but provide tremendous success when solving navigation tasks. They allow dead reckoning to be implemented by stabilizing object positioning, reducing fix expansion error in navigation, and allowing for more accurate predictability of robot position.

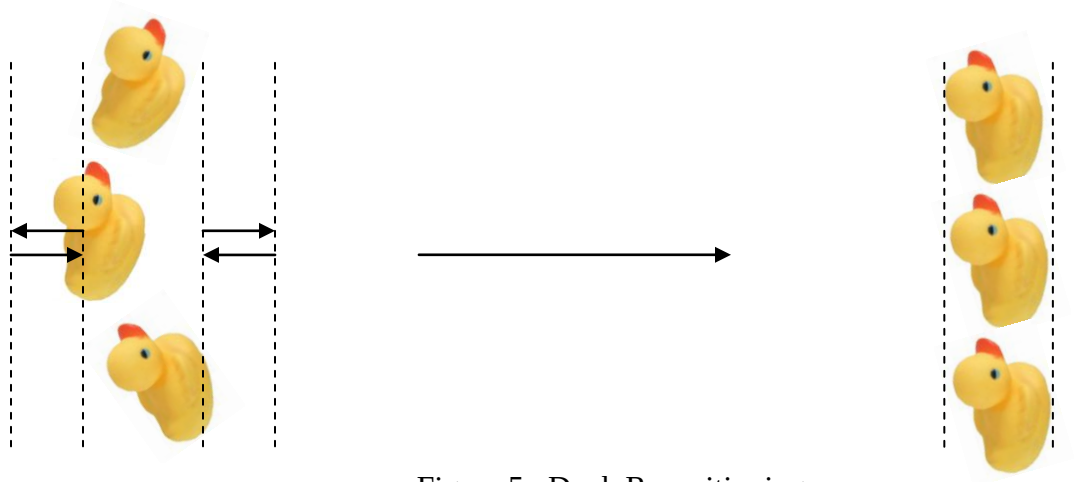

Figure 5. Duck Repositioning

Now some may say that sensor oriented movement is the epitome of dead reckoning. Complete sensor navigation can be looked upon as dead reckoning while getting consistency and repeatedly getting fixes. In this way sensor oriented movement is better suited for a dynamic environment. Dead reckoning can never be implemented where the surroundings are changing, and objects move and change properties. This provides for greater accuracy and less error. However, sensor oriented movement is less controllable. Because it is the sensors that are determining the path the end path may or may not come out to be the same every time. This is why dead reckoning and sensor oriented movement are commonly combined. In Botball because the objects are only moved in either pre determined time settings or by your or the other team's robots sensors should be used to regain positioning. Dead reckoning can be used for everything else, and as explained above, the error caused by dead reckoning is manageable and thus this method is effective [5].

References

[1] Bowditch, N. (n.d.). *Bowditch -The American Practical Navigator* (IRBS, Ed.). Retrieved from NATIONAL IMAGERY AND MAPPING AGENCY website: http://www.irbs.com/bowditch/

[2] Linear Stage Terminology. (2009). *Aerotech Engineer Reference Linear Stage Terminology*. Retrieved from http://www.aerotech.com/products/engref/strait.html

[3] Motion Control Coordinate System. (2010). *Newport Corporation Motion Control Coordinate System*. Retrieved from http://www.newport.com/servicesupport/Tutorials/default.aspx?id=143

[4] Stovall, S. H. (1997, September). *Basic Inertial Navigation*. Retrieved from http://www.fas.org/spp/military/program/nav/basicnav.pdf

[5] Straight Dope Science Advisory Board. (n.d.). Is "Dead Reckoning" short for "deduced reckoning"? In *The Straight Dope* (www.straightdope.com). Retrieved from http://www.straightdope.com/columns/read/2053/is-dead-reckoning-short-for-deduced-reckoning