Dead Reckoning

Related terms:

Ultrasound, Extended Kalman Filter, Global Positioning System, Kalman Filter

View all Topics

Vehicle Navigation

Stefan Edelkamp, Stefan Schrödl, in Heuristic Search, 2012

Stand-alone Positioning Systems

Dead reckoning *(DR)* is the typical standalone technique that sailors commonly used in earlier times before the development of satellite navigation. To determine the current position, DR incrementally integrates the distance traveled and the direction of travel relative to a known start location. A ship's direction used to be determined by magnetic compass, and the distance traveled was computed by the time of travel and the speed of the vehicle (posing the technical challenge of building mechanical clocks that work with high accuracy in an environment of rough motions and changing climates). In modern land-based navigation, however, various sensor devices can be used, such as wheel rotation counters, gyroscopes, and inertial measurement units (IMUs). A common drawback of dead reckoning is that the estimation errors increase with the distance to the known initial position, so that frequent updates with a fixed position are necessary.

> Read full chapter

SLAM for Pedestrians and Ultrasonic Landmarks in Emergency Response Scenarios

Carl Fischer, ... Mike Hazas, in Advances in Computers, 2011

2.1.1 Inertial PDR with Zero-Velocity Updates

Dead-reckoning is a self-contained navigation technique in which measurements—typically from inertial sensors in the case of PDR—are used to update the position and orientation of an object, given an initial position, orientation, and velocity. No infrastructure is required but the position error can accumulate over time.

XSens's MTx3 is an IMU comprising a tri-axis accelerometer, gyroscope, and magnetometer. The on-board processor computes 3D orientation. Our PDR algorithm is similar to other work [47,48] which uses shoe-mounted IMUs and applies periodic zero velocity updates (ZUPTs). In order to convert the MTx measurements into meaningful positions, the raw accelerations are rotated from the sensor coordinate system into the world coordinate system using the rotation matrix computed by the MTx as shown in Fig. 1. The transformed accelerations are then double integrated to yield position estimates in the world coordinate system.

Fig. 1. Transformation from sensor to world coordinates via the direction cosine matrix: $\mathbf{pos}_{world} = R_{sensor \rightarrow world}\mathbf{pos}_{sensor}$.

In order to reduce the position error which increases quadratically with time, we reset the integrated velocities to zero at each footstep [47]. This has the effect of reducing the error, making it linear with distance covered. Two phases in walking are identified: the stance phase, when the foot is in contact with the ground, and the swing phase. During the stance phase, the velocity is reset and kept at zero; during the swing phase, the acceleration is double integrated. Our algorithm detects the stance phase of each step by comparing a threshold to the product of the norm of the acceleration and the norm of the rate of turn [48]. If this product is below an empirically determined threshold for more than 200 ms, then a stance phase is detected. When the product rises above the threshold again, a swing phase is

detected. This is illustrated in Fig. 2. If steps are taken at a significantly faster pace, then the stance phase may not always be detected and some opportunities for ZUPTs may be missed.

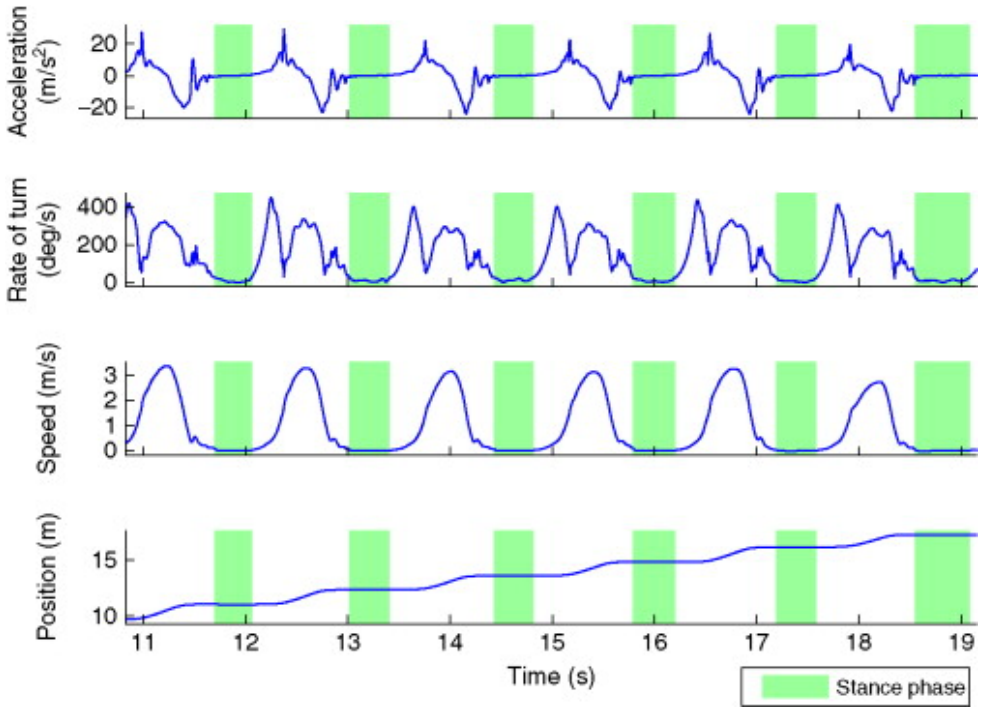

Fig. 2. PDR algorithm: each step has a stance phase (shaded) and a swing phase. Velocity is reset to zero during the stance phase, acceleration is double integrated during the swing phase.

Although some distance drift is inevitable due to the integration of noise and small offsets in the raw sensor data, we also believe that most of the distance error is due to the MTx incorrectly estimating its orientation as explained by Foxlin [49]. Thus, we might interpret some of the forward motion as vertical motion, or vice versa. The traces appearing later in this chapter (Fig. 10) show the effects of drift on position estimates. The MTx internal algorithm which supplies the rotation matrix has not been disclosed by XSens, and is effectively a "black box." Thus, we have very little information about how the sensors (gyroscope, magnetometers, and accelerometers) are used in computing the MTx orientation, and little control over the unit's internal parameters. We assume that most of the heading errors are due to metallic objects or magnetic fields interfering with the MTx magnetometers, as we have observed that these errors occur systematically in the same locations. We also note that when using the system outdoors or in an open space, the results are much better and the orientation drift is often negligible. So it seems that magnetometers help in outdoor situations where they accurately determine magnetic North but that indoors they cause heading errors due to interference.

Our PDR implementation is a naive version—(1) rotate the accelerations from the sensor coordinate system into the world coordinate system, (2) integrate the accelerations to determine the speed, (3) integrate the speed to determine the position, (4) if

the accelerations and the gyroscope readings are small then reset the speed to zero. More advanced algorithms (such as the one described by Foxlin) use the ZUPTs to correct not only the speed but also the position, and to estimate accelerometer and gyroscope biases which are responsible for much of the error accumulation. Such methods are able to achieve better accuracy over longer distances through tighter integration of the orientation computation, the dead-reckoning, and the ZUPTs. This improvement in performance comes at the expense of greater complexity in the design of the algorithm (typically an EKF). Beauregard gives more details on the challenges in his thesis [50]. Irrespective of the method used for PDR, the output will inevitably suffer from error accumulation. A more advanced method than ours could simply be used as a drop-in replacement in the algorithm presented above.[4]

> Read full chapter

Latency and consistency

Anthony Steed, Manuel Fradinho Oliveira, in Networked Graphics, 2010

11.7.1 Basic dead-reckoning

Dead-reckoning works with entities that have a simple dynamics model. The idea is that a local process will simulate an object and periodically send a message with state to a remote process which then extrapolate this state using the dynamics model. For example, the local process will simulate a vehicle with a position and velocity, and will send a message with the position and velocity of the vehicle to the remote process. When the remote process receives the message, it will know exactly where the vehicle is and will also know its velocity; therefore it can extrapolate its position. In papers about dead-reckoning, authors commonly use the terminology of Blau, et al. (1992) where the local object is called the *player object* and the remote object whose motion is extrapolated is called the *ghost object*.

Key to a dead-reckoning scheme is the dynamics model that the ghost object model uses. There are two parts: the *extrapolation scheme* and the *convergence scheme*. Two extrapolation schemes are commonly used: *first-order models* where each entity has a position and velocity, and *second-order models* where each entity has position, velocity and acceleration.

In a first-order model, if at T_0 the position is P_0 and the velocity V_0 the position can be extrapolated as follows:

In a second-order model, if at T_0 the position is P_0, the velocity V_0 and the acceleration A_0 then the position and velocity can be extrapolated as follows:

In both first and second orders, the rotation may be treated in the same way (rotation, angular velocity, angular acceleration). First-order models are good models of player movement in FPS games whereas second-order models are good models of vehicle dynamics.

Of course, motion extrapolation is imprecise because the behavior of the object changes. Eventually a message arrives from the player object model giving a position and velocity different than that computed with the ghost object model. Figure 11.14 shows some examples of first- and second-order models. When the path is predictable and smooth, extrapolation can be accurate. However, it can only be completely accurate if motion parameters are sent at the rate at which they change. In the top graph, we see that the first-order model is constantly diverging from the smoothly varying curve. On the top-left we see the when the motion parameters are sent, they include a position and a velocity (the arrow). Thus the first-order model effectively implements a linear approximation to the curve. On the top-right we see that the ghost model path has a series of alternating linear extrapolations followed by short jumps back on the true path of the object. With the second-order model, we see that effectively we get a lower error and that the extrapolated path is closer to the real path. However there are still gross errors. Roughly in the middle of the path, where the object doubles back on itself, the last update message has zero acceleration as the object is almost traveling in a straight line; however, shortly afterwards there is a large acceleration. Thus the error here still results in a large jump to the next update. At this point, because the acceleration is momentarily zero, the extrapolation is linear, and just as prone to error as the first-order model. Elsewhere, the extrapolation is almost exact, because the curve has constant acceleration. Thus second-order and first-order models are very sensitive to the sampling of their behavior.

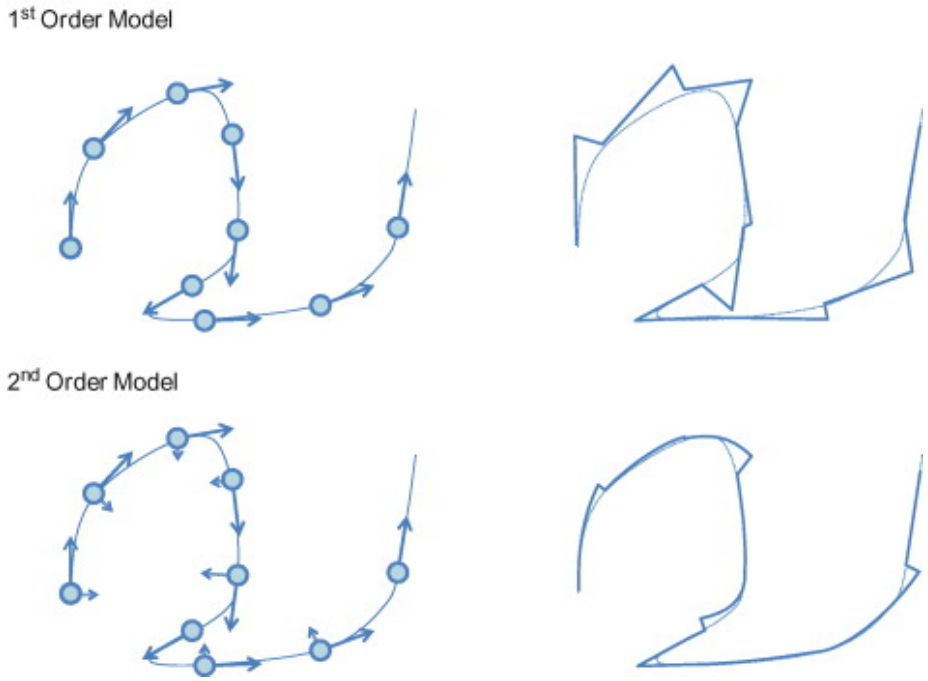

Figure 11.14. Dead-reckoning using first- and second-order models. In the left column we see the original player object's behavior. The circles indicate the location of the object at times when updates were sent. The single arrow indicates the velocity at that time. The arrows in the bottom-left figure shows the acceleration. On the right we see a solid line which is the ghost object model path, and a dotted line which is the original player object path for comparison. The ghost object model path includes alternating motion extrapolations and short jumps back to the true position of the player object when a new update is received. These jumps are generally larger in the first-order model, but there are also large errors in the second-order model

The ghost simulation thus needs a convergence scheme to reconcile the old, incorrect prediction and the new information. The situation is similar to that with the client predict ahead in the previous section: to preserve local plausibility the object shouldn't jump, but we should somehow plan to reduce the error over time. There are two main options: *path-interpolation strategies* or *path-planning strategies*.

Convergence techniques that use a path-interpolation strategy choose to reduce the error in position and velocity over a period of time. The prediction scheme prior to the new information (old ghost model) and the new prediction scheme (new ghost model) are both run forward in time. Then over a given period the actual displayed position and velocity are given as a weighted blend of the old ghost model and new ghost model, where the blend weight changes slowly over time. This is illustrated in Figure 11.15.

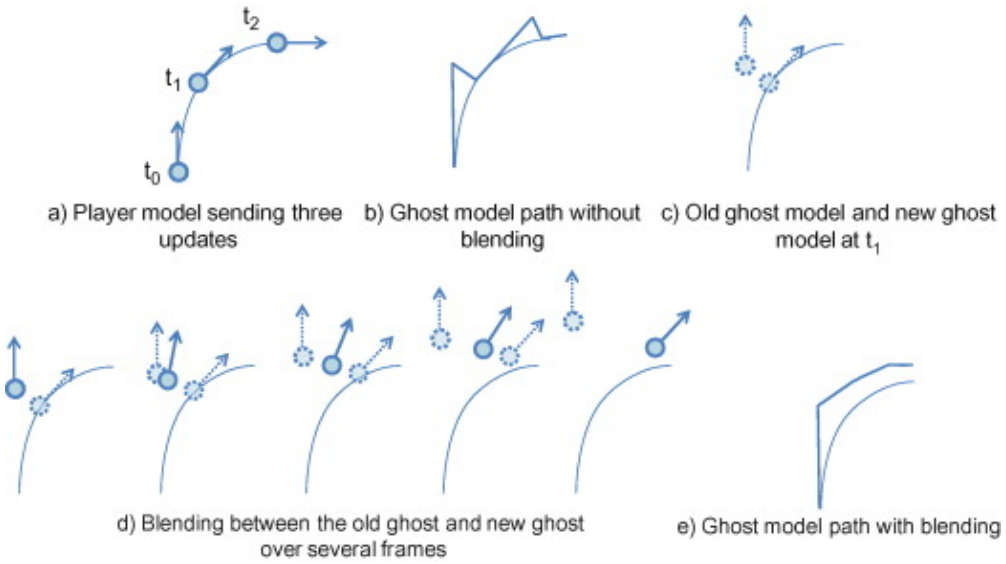

Figure 11.15. Blending between the old ghost model and the new ghost model over time. Note that the blending brings it back onto the path linearly extrapolated from the update at t_1, not back onto the original path. However, the path now contains no instantaneous jumps

This method is simple to implement but can violate certain properties of the dynamics, such as the object's direction of movement not being the same as the direction of its velocity. Other schemes might include quadratic or cubic interpolation between the two paths (Singhal & Zyda, 1999); these have the effect of smoothing out any apparent abrupt changes in velocity, so the object doesn't appear to accelerate jerkily; however, they don't deal with the issue of consistency of position changes and apparent dynamics. For certain vehicles, the vehicle orientation can usually be determined by its velocity and acceleration, so the orientation can be ignored (Katz & Graham, 1995; Singhal & Zyda, 1999). The reader may recall that we did something similar in our boid demonstration in previous chapters. An example is normal aircraft motion where as long as the plane is not slipping or stalling, its direction can be inferred from its acceleration. However, this often isn't true for vehicles, and it is certainly not true for player movement in games: someone who travels in the direction they are facing in an FPS is easy to shoot and thus players try to strafe (move sideways) in an unpredictable manner.

The second strategy is to have the local vehicle plan to move back onto the extrapolated path over time using some in-built reasoning about the vehicle dynamics. For example, when using a first-order model, the local object could slowly and within known vehicle constraints rotate from its current heading to turn towards a target point on the new ghost model. This is appropriate for vehicles which have to be seen to turn and have momentum. An example is shown in Figure 11.16. Note that the path is continuous in velocity with the old ghost position and the extrapolated new ghost position. Note also though that the vehicle will be seen to steer quite violently,

turning more than double the angle between the old and the new ghost directions because it needs to compensate.

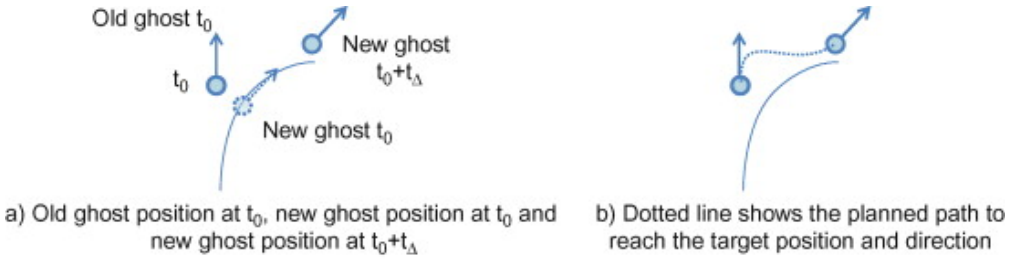

Figure 11.16. Instead of blending, the ghost model can plan to steer the object back onto the extrapolated path

This last point about the path-planning convergence technique hints at a larger problem with dead-reckoning: it can be unstable if the frequency of updates isn't sufficient to give a good sampling of the behavior. An example is shown in Figure 11.17 which shows a ghost model that is effectively "out of phase" with the player model.

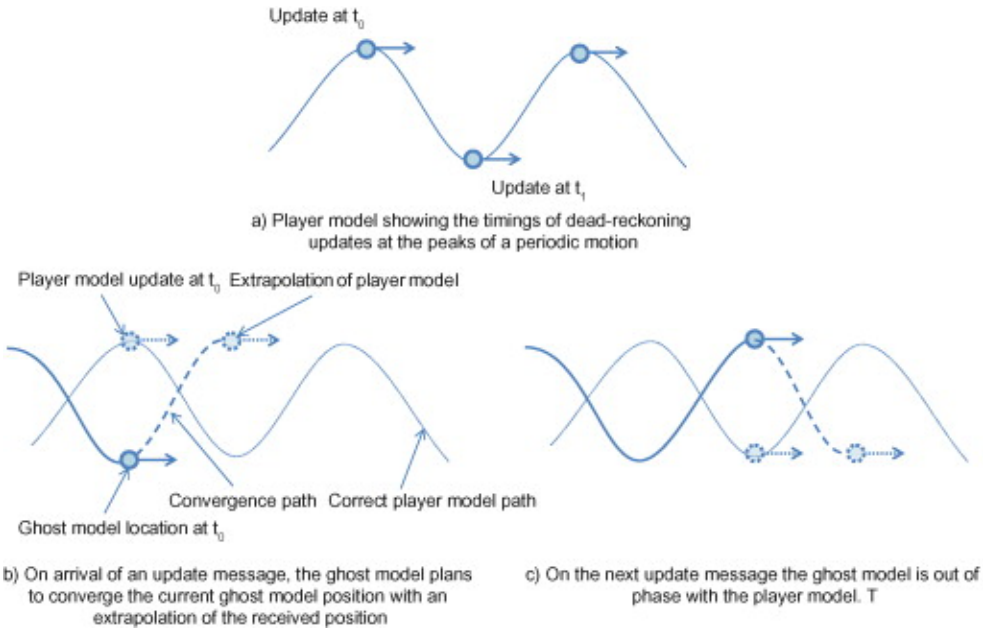

Figure 11.17. An example of dead-reckoning where the ghost model poorly represents the player model

Another difficult case is shown in Figure 11.18 where local constraints mean that the old ghost model and new ghost model can't converge easily. If all else fails, the object could jump to the new ghost model's position and carry on. At this point it is worth noting the relationship between convergence in dead-reckoning and rollback in time-warp algorithms. The assumption that is usually made in NVEs and NGs is that the movement of the player is continuous (i.e. doesn't jump) and possibly smooth (i.e. no rapid changes in velocity). Thus, as we noted when discussing time-warping, convergence is a general technique that could be used to cover up

any discrepancies that are found in the state, be they because of incorrect prediction or over optimism in lookahead simulation.

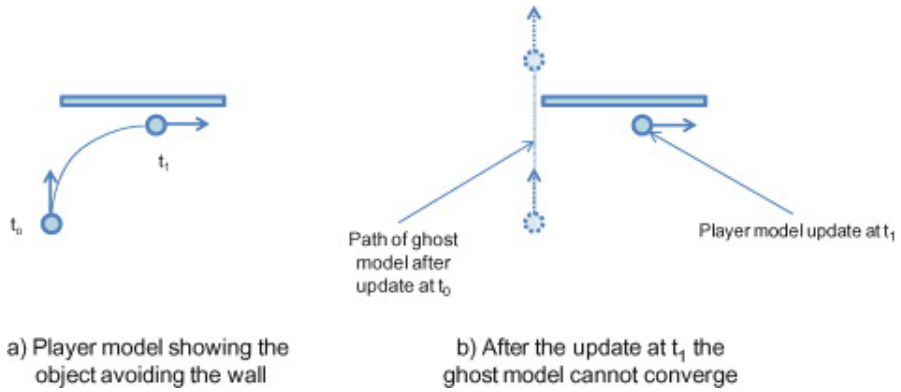

Figure 11.18. A situation where the ghost model has difficult converging with the new player model update

A final aspect of dead-reckoning that is commonly, but not always, used is instead of sending updates with dynamics periodically, only send update when necessary. The idea is simple: the sending process runs both the player model and the ghost model. It only sends an update when the two models diverge by more than a given error threshold. This is illustrated in Figure 11.19, which uses the same player model path as Figure 11.14 using the second-order model. As can be seen, we get the same number of updates as before but the timing of the updates is not regular. We get a much better fit of the ghost model path to the player model path. In particular the gross errors we saw in Figure 11.14 are gone. With a convergence model applied, the small jumps would of course be removed, and the two paths would now look very similar.

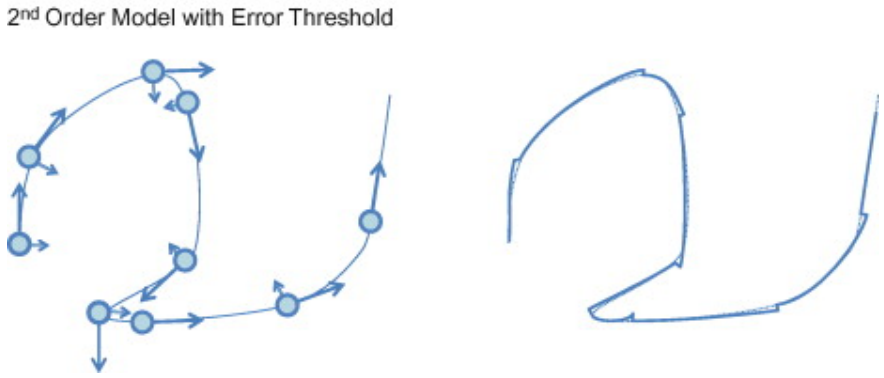

Figure 11.19. Dead-reckoning with an error threshold. The ghost model is the model closer to the player model

Dead-reckoning with an error threshold can work extremely well for vehicles that only accelerate infrequently. It is less useful for high frequency changes, such as player positions in a skirmish in an FPS or MMOG. Cado (2007) notes that dead-reckoning was examined for use in an MMOG but dropped because most of the objects being represented were humanoids for which smooth extrapolation didn't work.

Within the context of the next chapter, dead-reckoning is a technique that can be used to reduce the packet send rate. The error threshold might be dynamically changed to reduce the number of messages sent. Of course this can put more strain on the convergence technique used, so the saving in bandwidth from dropping a packet needs to be weighed against the larger inconsistency that will be produced.

> Read full chapter

Aircraft Avionics

Robert G. Loewy, in Encyclopedia of Physical Science and Technology (Third Edition), 2003

III.E Doppler Radar and Dead Reckoning Systems

"Dead reckoning" is an old maritime term used to describe navigating (itself a maritime term) by using known initial position, the vehicle's velocity vector (speed and direction), and how long that velocity has been maintained, to determine the vehicle's new position. If velocity is measured, say, relative to the surface of a body of water, it is clear that positions determined by dead reckoning will be in error by the existence of currents in that water. For ships whose speed is not great relative to currents, this is important; for fast flying aircraft it is much less so. Use of Doppler radars to measure relative speed in modern dead reckoning systems, however, has some significant advantages; for example, like INS, they are self contained, needing no terrestrial or satellite cooperative station; their transmitter power requirements are small; and they work very well for low vehicle velocities.

As to the principle on which Doppler radars are based, consider that wave motion emanating from a source moving with respect to the receiver is sensed as having a changed frequency; the magnitude of the change depending on the relative velocity, higher if the source and receiver are moving closer, lower if they are moving farther apart. This so-called "Doppler effect" is experienced almost every day acoustically, for example, if a fast-moving auto or train passes with its horn or whistle blowing. In a directly analogous way, the frequency of a radar signal return shifts if the transmitter and reflecting surface have relative velocity along the line of EM transmission. This provides a means, using reflection returns, to determine the speed of an aircraft relative to the ground or water over which it is flying. Doppler radars, mounted on an aircraft, use microwave frequencies in an internationally authorized band, between 13.25 and 13.4 GHz. This provides narrow beams of EM radiation, which can be pointed at the ground at relatively steep angles. The last has the additional benefit of reducing the probability of detection in military applications.

For Doppler navigation, at least three radar beams are needed to determine three components of velocity relative to the earth's surface, and three aircraft attitude measurements in three perpendicular planes are needed to resolve the Doppler radar measurements into components in an earth-related, geodetic coordinate system, as needed for dead reckoning navigation (Fig. 6). If the three Doppler radar beams are arranged as shown in Fig. 7, and a difference taken of the returns from signals A and B, the Doppler shifts of the lateral components will cancel, whereas the longitudinal components, being of opposite sign, will be added. This arrangement, known as a "Janus" system (after the Roman God who could see both backward and forward), increases system accuracy. For the usual beam angles to the horizontal of about 70°, a Janus system will have an error in horizontal velocity of only about 0.015% per degree of error in the aircraft's pitch attitude. Without the Janus arrangement the same metric would be about 5%.

FIGURE 6. Resolution of aircraft velocity into navigable components.

FIGURE 7. Lamda arrangement of three Doppler radar beams.

> Read full chapter

Context-aware computing

Manish J. Gajjar, in Mobile Sensors and Context-Aware Computing, 2017

Sensors

Sensors can be used to enhance the accuracy of determining the location of a device. For example, in dead reckoning, sensors can be used to determine relative motion from a reference point, such as to detect whether the system moves outside of a 3-m radius, or to determine relative positioning of devices, such as the case of bumping two devices up against each other to establish common reference and then they can track their relative positions. Sensors can also be used standalone when other methods are not available. First let us understand what dead reckoning is. Dead reckoning (deduced reckoning) is the process of calculating current position by using previously determined reference position and advancing that position based upon known or estimated speeds over elapsed time and course. Although it provides good information on position, this method is prone to errors due to factors like inaccuracy in speed or direction estimations. Errors will also be cumulative since new estimated value would have its own errors and it will also be based on previous position which had errors, thus resulting in cumulative errors. Some of the sensors used are accelerometers and gyroscopes for acceleration/velocity integration for dead reckoning, accelerometers for bump events, pressure for elevation, and so on.

> Read full chapter

Wayfinding and Navigation Behavior

S.M. Freundschuh, in International Encyclopedia of the Social & Behavioral Sciences, 2001

2 Wayfinding Strategies

Though there are only three classes of wayfinding tasks, there are various strategies used to carry out these tasks. These strategies include piloting, following marked trails, habitual locomotion, dead reckoning, path integration, and lastly the use of internal representations. These strategies, though presented here as separate individual strategies, can be used together to carry out a wayfinding task. In addition, a particular strategy can be used to carry out more than one wayfinding task.

Piloting relies on the use of landmarks for orientation. The immediate goal in piloting is to traverse from the present landmark to the next landmark. The present landmark serves as a basis for orientation by providing a link back to the familiar environment, and information on where to go next. The spatio-temporal sequence of landmarks provides information on wayfinding progress, indicating how much of

the trip has been completed and how much remains. Piloting is a strategy typically used in explores by tourists, or a person recently locating to a new city.

Following marked trails entails wayfinding that relies on a system of trail markers to highlight a route. Marked trails can include a road network with road signs and mile markers, multiple walking paths through a wooded area, or color-coded lines along the floor of a corridor in an office complex. Marked trails are becoming critical in the design of large build environments, such as hospitals, to aid people in finding a particular unit, such as X-ray or pediatric cardiology, while under stressful situations (Carpman and Grant 1997).

Habitual locomotion, on the other hand, is a strategy that develops after repeated navigation through an environment, resulting in what has been referred to as route knowledge. Generally, habitual locomotion develops from piloting strategies, and after repeated exposure, the sequence of landmarks and the actions to be taken at each landmark are remembered. With increasing exposure to a route, habitual locomotion results in the development of commutes.

Path integration strategies, unlike habitual locomotion or piloting, are not based on the current landmark, or on the sequence of landmarks, but instead are based on orientation to the starting point of a route. Path integration enables travel directly back to the starting point along a novel route, after having traveled a complex route directed away from that point. This strategy necessitates constant knowledge of the direction or orientation back to the starting point. This wayfinding strategy is difficult, and is not common for humans in everyday wayfinding tasks, but instead is a common behavior in nonhuman species, such as honey bees and rats, studied extensively by behavioral psychologists (Gould 1986, Tolman 1948). It is also a strategy used in the sport of orienteering, where participants use a map and compass to navigate an unknown wilderness course.

Dead reckoning is a wayfinding strategy that entails first determining the direction to a destination, and then systematically searching until the destination is reached. Methods for orientation for dead reckoning generally include reliance on solar and lunar information, auditory information, and vestibular and proprioceptive information. Some cultures studied by anthropologists are able to use wind and water currents for orientation. Reliance on dead reckoning requires that the navigator perform periodic updates on position, making small corrections in distance and direction estimates while moving through the environment.

Wayfinding based on use of an internal representation requires reliance on a 'cognitive map' in which relative positions of landmarks, routes between landmarks, and the orientation and distances between landmarks are represented mentally. This spatial knowledge is referred to as survey or configurational knowledge. This is the most complex and sophisticated wayfinding strategy. Geographers and psy-

chologists have long studied internal representations and the implication these representations have on navigation and wayfinding behavior; (for overviews see Downs and Stea 1973, Golledge 1999, Kitchin and Freundschuh 2000, Moore and Golledge 1976).

> Read full chapter

Navigation

B. Finney, in International Encyclopedia of the Social & Behavioral Sciences, 2001

5 Navigating Electronically

However, the weak point in navigating by magnetic compass, chronometer, and stellar observations remained the changing weather. When clouds, rain, or fog prevented navigators from seeing the stars they had to fall back upon dead reckoning, which, even with instruments for measuring direction of travel and distance run, suffered from cumulative error. Radio direction finding pioneered in the 1910s was an early but only partial solution. However thankful fogbound navigators were to gain radio bearings, these were not very accurate, and available only along the coasts of industrial nations. The first truly all-weather position finding system, LORAN, was developed during World War II for wartime navigation. The reception of synchronized pulses from two pairs of transmitters located at carefully surveyed positions resulted in accurate navigational fixes. The subsequent development of inertial navigational systems was stimulated by further military requirements: guiding missiles and providing bombers and submarines with position information needed for targeting. Employing accelerometers and gyroscopes to determine changes in velocity and direction, these systems provide a continuous indication of position, subject, however, to cumulative error. The most accurate and revolutionary of the electronic navigational technologies is the Global Positioning System (GPS), a US military system made accessible to civilian users, though initially degraded in accuracy. (Russia has its own *Glonass* system.) The GPS system depends upon a constellation of 24 satellites in low earth orbit, which transmit signals that cheap hand-held receivers can process to determine latitude, longitude, and altitude with great accuracy. It was fully operational by the early 1990s, and is now widely used around the world, though not without some misgivings over military control.

> Read full chapter

Dead Reckoning of Multi-legged Robot —Error Analysis—

Toru Masuda, ... Kenji Inoue, in Human Friendly Mechatronics, 2001

1 INTRODUCTION

Recently, we hope locomotive robots play an active part for rescue operation and construction and civil engineering in outdoor unknown environment (Fig.1). It is important to measure the posture of the robots for doing various tasks in such environments. A robot with external sensor such as camera and ultrasonic sensor can measure the self posture by observing landmarks in the known environment. In outdoor environment, however, it is difficult to detect landmarks stably. Besides the positions of the landmarks are not known previously. Thus, dead reckoning of locomotive robots in unknown outdoor environment is required.

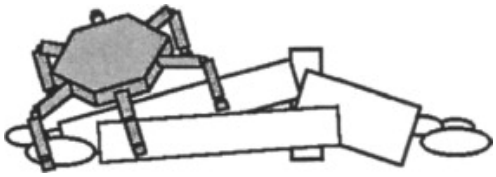

Fig.1. Image of Multi-legged robot.

As locomotion mechanisms of robots, wheeled and legged mechanisms have been studied Since most conventional robots use wheel mechanisms in planar environments dead reckoning of wheeled robots have been studied[1][2]. If rough terrain is anticipated as a work environment, however, wheeled robots have disadvantages in locomotion. Wheeled robots are not suitable for moving in a rough terrain which is not a planar environment, because the wheels must continuously contact the environment. Besides, in a rough terrain, not only position but also orientation of a robot must be measured. On the other hand, legged robots, especially multi-legged robots, are suitable for locomotion than wheeled robot, since multi-legged robots are superior to wheeled robots in moving and traverse avoidance ability.

In this paper, we propose dead reckoning of multi-legged robots and analyze the effect of the error of the joint displacement of legs on the body posture. The body posture of a legged robot can be estimated by inverse kinematics if the joint displacement of legs is known. We assume the robot is equipped with touch sensors at every tip of legs. Thus, the current robot posture is estimated only from the joint displacement and the contact points of the legtips. This is called dead reckoning of multi-legged robot[3]. In a walking operation, the error of dead reckoning is accumulated, which includes the error of joint displacement and the error of touch sensors. The error will be analyzed for the purpose of raising the reliability of dead reckoning in rough terrain.

Furthermore, it is possible to model the unknown rough terrain environment, because the contact points of legs can be estimated. Therefore, the proposed method can be used as one of the environmental modeling techniques in rough terrain.

> Read full chapter

Animal Cognition

W.A. Roberts, in International Encyclopedia of the Social & Behavioral Sciences, 2001

3.1 Dead Reckoning

Both insects and mammals can travel long and twisting paths from their home base in total darkness and still return home along a straight-line route. A person wearing a blindfold can be led some distance from his/her starting position through several turns; when asked to return to the start, the person will walk directly to an area near the start. In all of these cases, the return route is computed by dead reckoning, also known as path integration. The return route has two components, direction and distance. As an organism moves through space, it has several sources of internal cues that are recorded and indicate position in space. In mammals, vestibular organs in the semicircular canals measure angular acceleration created by inertial forces when turns are made. In addition, distance information is provided by proprioceptive feedback from the muscles involved in locomotion and by efference copies from motor commands. These cues are integrated to plot a return vector to home that contains information about both direction and distance. Although dead reckoning is highly effective, errors arise as the number of turns in a trip increases, and the angle of the return vector may deviate progressively from the correct one. Thus, animals often use dead reckoning to return to an area near their home and then use local landmarks to find their nest or hole.

> Read full chapter

Multimodal Localization for Embedded Systems: A Survey

Imane Salhi, ... Maroun Ojail, in Multimodal Scene Understanding, 2019

8.5.2.1 Indoor localization in large-scale buildings

Localization systems have various application areas, including critical and constraining ones that require the use of rather embedded systems. For example, in [235], an indoor localization system is proposed to help pedestrians to locate themselves in large-scale buildings like hospitals. Implemented on an *ASUS Z00Ld* smart phone, this system is based on inertial sensors (gyroscope and accelerometer) as a dead reckoning system and a light sensor embedded in the smart phone. As most of the systems are based on the combination of different sensors and data fusion, the proposed system adopts the same general system architecture as shown in Fig. 8.21. It proceeds through the data acquisition step where the data is resampled and filtered in order to be exploited for the end of processing. Then comes the modeling step. In this case, this operation is illustrated by two models: the motion model and the light model. The motion model mainly concerns three functional components: the step detection, the heading direction estimation and the positioning. The step detection is managed by the accelerometer, in particular the accelerometer's z-axis, which is easily affected by the smart phone vibration caused by the pedestrian movement. For the second functional component it is handled by the gyroscope, the heading direction estimation is done thanks to the z-axis gyroscope data. Finally, for the positioning the authors use the PDR system (see Sect. 8.3.1.1) and the footpath approach by fusing locally the different pieces of information having been produced in the components before. The final position resulting from this step is used after making a decision regarding the location of the pedestrian. On the other hand, the light model aims to modify the pedestrian's position and correct the step length information; it is based on the light sensor and the step detection data fusion. This is a second local data fusion. It is followed by the decision making phase which represents the global data fusion process. The final correct and accurate position is calculated and produced in this step.

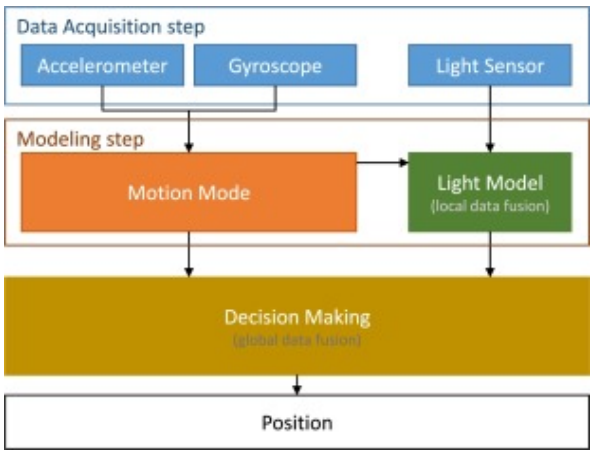

Figure 8.21. A global system architecture of the indoor localization system discussed in [235].

> Read full chapter