Surface Navigation for Robots

Tyler C. Folsom

From Raw GPS

GPS coordinates are spherical: Latitude, Longitude and distance from the center of the earth (or altitude above mean sea level). Angle measurements are given in either decimal degrees or degrees, minutes and seconds. This paper assumes use of decimal degrees.

Local maps are in two dimensions; the Earth is locally flat. We will assume that the robot is operating in a small enough area where the curvature of the Earth is not significant. We will pick an origin for a local map, and measure distances in meters. The positive X-axis measures distance east from the origin and the Y-axis gives distance north. Compass direction uses 0 degrees as north, 90 degrees as east, 180 as south, and 270 as west. Compass direction will be referred to as "bearing". It differs from the "angle" of a standard coordinate system in which 0 is east and 90 is north.

angle = 90 - bearing

There are sophisticated algorithms to find the great circle distance between two (latitude, longitude) readings. With the flat earth assumption, conversion between the two coordinate systems is simple.

x = EARTH_RADIUS * (Longitude – LONGITUDE_ORIGIN) * π / 180 * cos(LATITUDE_ORIGIN *π /180)

y = EARTH_RADIUS * (Latitude – LATITUDE_ORIGIN) * π / 180

z = EARTH_RADIUS

Latitude = (y * 180) / (π * EARTH_RADIUS) + LATITUDE_ORIGIN

Longitude = (x * cos(LATITUDE_ORIGIN *π /180) * 180) / (π * EARTH_RADIUS) + LONGITUDE_ORIGIN

You should set the origin to be near the center of the area in which you intend to operate.

Example: using Seattle Center House as the origin

LATITUDE_ORIGIN = 47.6213

LONGITUDE_ORIGIN = -122.3509

EARTH_RADIUS = 6371000

A reading of (47.620941, -122.351206) is 22.934 meters west of Center House and 39.918 meters south. A point 151.77 meters east and 66.16 meters north is at (47.621895, -122.348875).

Note that measuring distance to a centimeter requires about 7 places beyond the decimal point in latitude and longitude. Thus latitudes and longitudes should be stored as 64 bit doubles. If your processor (such as the Arduino) does not have this capability, an alternative is to record distances in mm as a 32 bit long integer, which will provide a range of 2147 km from the origin. If you travel that far, the flat earth assumption is no longer valid and you are on a new map with a new origin.

GPS is not sufficiently accurate for robot navigation. It typically has a standard deviation of 3-5 meters, which would mean that it can drift by 15 meters. Even if you achieve a 1 meter standard deviation, you

do not have enough accuracy to keep a vehicle in its lane. The GPGGA format includes latitude, longitude, time and Horizontal Dilution of Precision (HDOP). The standard deviation of the GPS reading can be estimated as

sigma = sqrt((3.04*HDOP)^2 + 3.57^2)

GPS can be further degraded by multipath reflections from buildings. It may become unavailable indoors, in tunnels, or if satellites are not properly positioned.

Latitude and longitude coordinates are available from Google Earth, Google Maps, and OpenStreetMaps. Open Street maps lets you download the positions of streets and buildings in latitude and longitude coordinates precise to seven decimal places. Thus you can construct a map of your robot's expected environment.

References: users.erols.com/dlwilson/gpshdop.htm

http://en.wikipedia.org/wiki/Error_analysis_for_the_Global_Positioning_System

Refining GPS with Kalman Filter

A Kalman filter (KF) can use all available information to provide the optimal prediction of position. We will start with a simplified version based only on the GPS position. Please refer to the references for the theory.

Start with a column vector of the state:

$\mathbf{X} = [x \;\; y \;\; dx/dt \;\; dy/dt]^T$

We could use $[x \;\; y \;\; z \;\; dx/dt \;\; dydt \;\; dz/dt \;\; d^2x/dt^2 \; d^2y/dt^2 \; d^2z/dt^2]^T$

The state might also include the yaw, pitch and roll of the vehicle. We will start with the simpler 4 by 1 matrix. All data are in meters and seconds. Suppose that we had determined our state at a previous time t_0 and we want to estimate the state at time $t_1 = t_0 + \Delta t$. We estimate the new state as

$\mathbf{X}_{et1} = \mathbf{F} * \mathbf{X}_{t0}$

where **F** is the state transition matrix

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Suppose that we also knew the uncertainties of the estimated positions and velocities at time t_0. For simplicity, assume that all the measurements have the same standard deviation σ and that they are uncorrelated. The covariance matrix is P

$$\mathbf{P} = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & \sigma^2 \end{bmatrix}$$

Since time has elapsed, uncertainty increases. We estimate the new uncertainty by

$\mathbf{P}_{et1} = \mathbf{F} * \mathbf{P}_{t0} * \mathbf{F}^T$

Thus the extrapolated covariance matrix is

$$\mathbf{P}_{et1} = \begin{bmatrix} \sigma^2 & 0 & \sigma^2\Delta t & 0 \\ 0 & \sigma^2 & 0 & \sigma^2\Delta t \\ \sigma^2\Delta t & 0 & \sigma^2(1+\Delta t^2) & 0 \\ 0 & \sigma^2\Delta t & 0 & \sigma^2(1+\Delta t^2) \end{bmatrix}$$

Now at time t_1 we get a new GPS fix and have a new measured position
$\mathbf{Z}_{t1} = [e \;\; n]^T$
We define a matrix **H** which reduces our estimated state to the parts that we can measure.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The difference between our extrapolated position and measured position is $\mathbf{Y}_{t1} = \mathbf{Z}_{t1} - \mathbf{H} * \mathbf{X}_{et1}$.

$$\mathbf{Y} = \begin{bmatrix} e \\ n \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x+\Delta t*dx/dt \\ y+\Delta t*dy/dt \\ dx/dt \\ dy/dt \end{bmatrix} = \begin{bmatrix} e - x - \Delta t*dx/dt \\ n - y - \Delta t*dy/dt \end{bmatrix}$$

The matrix **S** reduces the uncertainties to the items that we can measure. $\mathbf{S}_{t1} = \mathbf{H}\,\mathbf{P}_{et1}\,\mathbf{H}^T + \mathbf{R}$
The diagonal matrix **R** is the measurement noise, which is formed from the HDOP of the GPS. The GPS variance is r. The covariance matrix P describes how much confidence we have in our estimate. The initial estimate of σ is that of the GPS.

$$\mathbf{R} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} \sigma^2+r & 0 \\ 0 & \sigma^2+r \end{bmatrix}$$

$$\mathbf{S}^{-1} = \begin{bmatrix} 1/(\sigma^2+r) & 0 \\ 0 & 1/(\sigma^2+r) \end{bmatrix}$$

$\mathbf{K}_{t1} = \mathbf{P}_{t1}\,\mathbf{H}^T\,\mathbf{S}_{t1}^{-1}$

$$\mathbf{K} = \sigma^2/(\sigma^2+r) \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}$$

The Kalman matrix K lets us establish our best estimate of state and uncertainty at time t_1.
$\mathbf{X}_{t1} = \mathbf{X}_{et1} + \mathbf{K}_{t1}\,\mathbf{Y}_{t1}$
$\mathbf{P}_{t1} = (\mathbf{I} - \mathbf{K}_{t1}\,\mathbf{H})\,\mathbf{P}_{et1}$
If we stay in one place, the filter is averaging the GPS signals, and reducing the uncertainty of position. If a GPS reading is noisy compared to the position averaged from several previous readings, it is lightly weighted. If our previous readings were based on signals with high HDOP and the new reading is much better, the new reading will be heavily weighted.

$$\mathbf{X} = \begin{bmatrix} e * \sigma^2/(\sigma^2+r) + (x + \Delta t*dx/dt) * r/(\sigma^2+r) \\ n * \sigma^2/(\sigma^2+r) + (y + \Delta t*dy/dt) * r/(\sigma^2+r) \\ dx/dt * \sigma^2/(\sigma^2+r) + \Delta t*dx/dt * r/(\sigma^2+r) \\ dy/dt * \sigma^2/(\sigma^2+r) + \Delta t*dy/dt * r/(\sigma^2+r) \end{bmatrix}$$

$$\mathbf{P} = \sigma^2/(\sigma^2+r) \begin{bmatrix} r & 0 & r\Delta t & 0 \\ 0 & r & 0 & r\Delta t \\ 0 & 0 & \sigma^2+r(1+\Delta t^2) & 0 \\ 0 & 0 & 0 & \sigma^2+r(1+\Delta t^2) \end{bmatrix}$$

Expanding the filter

The basic KF will average the readings, but it is not as smooth as we want. This is because we take the difference between previous fixes as the velocity, and use the velocity to predict the next position. If we are standing still, we get a psuedo-velocity due to GPS drift which can make the drift worse. Thus it is common to augment GPS with an Inertial Measurement Unit (IMU), also known as an Inertial Navigation Unit (INU). Set

$\mathbf{X} = [x \;\; y \;\; dx/dt \;\; dy/dt \;\; d^2x/dt^2 \;\; d^2y/dt^2]^T$

The IMU will measure acceleration and bearing. Add a Hall effect sensor (cyclometer pick-up) to click on each wheel revolution. Let each wheel click trigger an interrupt to measure the time since the last click and compute speed = wheel circumference / time of revolution. We then have instruments that can measure speed, bearing and acceleration.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

H is a 6 by 6 identity matrix; Z and Y are 6 by 1; R, S, P, K and I are 6 by 6. A more sophisticated form of state estimation is

$\mathbf{X}_{et1} = \mathbf{F} * \mathbf{X}_{t0} + \mathbf{B} * \mathbf{U}_{t1}$

where $\mathbf{U} = [\,0\;0\;0\;0\;\Delta a\;\Delta b]^T$ is the commanded change in acceleration, and **B** scales the controls into a predicted acceleration in m/sec^2 in the x and y direction. The acceleration **U** will include any change of speed or direction that is about to happen.

The predicted acceleration and speed must be restricted by the vehicle's physical limits. The minimum turn radius is a function of speed.

We can also define a 4 by 6 form of **H** to use when GPS is not available.

References:

S Thrun, W. Burgard and D. Fox, Probabilistic Robotics, MIT Press, 2006.

A. Gelb (ed) Applied Optimal Estimation, MIT Press, 1974.

P. S. Maybeck, Stochastic Models, Estimation, and Control, Vol 1, Navtech, 1994.