Tagtools - Practical 3 - Tracks, pseudo-tracks, dead-reckoning and track reconstruction

(*Matlab & Octave version*)

7th October 2017

Introduction

This practical explores tracking data from GPS and dead-reckoning. Increasing numbers of tags now include a GPS to collect positional information. These consume a lot of power and only obtain a position when the animal is at the water surface and can be affected by over-covering structures such as trees. As a result, GPS positions are often sparse and irregularly sampled, leaving open the question of how animals move inbetween these positions. In this practical, you will explore this by merging GPS and dead-reckoned tracks. With the GPS data, you will convert latitude and longitude data to a local level plane and compute some track statistics. You will then try some methods to dead-reckon a track from accelerometer and movement data. You will use a simple track fitting method to reconcile the GPS and dead-reckoned tracks, and examine errors.

Section 1. Working with GPS data

1.1 Introduction

GPS data is almost always recorded as pairs of latitude and longitude values, usually in decimal degrees. Positive values mean north of the equator and east of the Greenwich meridian, respectively. Although latitude and longitude values are useful for plotting large scale movements or for integrating data from multiple sources, they are difficult to work with for measuring movements. For this reason, we usually convert to a local level frame.

1.2 Load the test data set

Load the test dataset testset4:

load_nc('testset4.nc')

You should see that variables A, M, P, POS and info have been made. Check out where the data came from. Variable POS contains the GPS positions. It is an irregularly sampled data variable. Type POS to see how this is represented in the structure elements. What are the columns in POS.data? How do you plot the track? It will be something like:

plot(POS.data(:,??),POS.data(:,??),'.-'),grid

where you need to replace ?? with the correct column numbers.

Extract the column of POS.data that contains the time of each capture:

T = POS.data(:,??);

T is in seconds since the start of the data sample. Use diff() to see what the shortest and longest time is between consecutive GPS samples.

Plot the dive profile and add the time of each grab as a dot.
plott(P)
hold on
*plot(T/3600,0*T,'g.')*

Zoom in to check where in the dive profile the GPS grabs occured.

1.3 Converting to a LLF

Use lalo2llf to convert the GPS latitude/longitude data to northing and easting:

NE = lalo2llf(POS.data(:,[?? ??]));

where you need to replace ?? by the column number of the latitude and longitude data in POS. NE is the resulting track in the local level frame. Use *help lalo2llf* to find out what the columns of NE are. Plot the track and add units to x and y axis to show you know what is being plotted, e.g.,

xlabel('Fried chicken')

As NE is now in a rectangular frame instead of a spherical frame, you can directly measure distances. The distance between positions 2 and 1 is:

dist = sqrt((NE(2,1)-NE(1,1))^2 + (NE(2,2)-NE(1,2))^2)

The time between these two points is T(2)-T(1). What was the effective speed of the animal? Is this its actual speed?

Measure the effective speed between all the pairs of positions and plot a speed histogram:

dist = sqrt(sum(diff(NE).^2,2));
speed = dist./diff(T);
hist(speed)

Add axis labels with units to show that you know what is being shown.

Note the Matlab/Octave shortcut for getting the RMS difference of a vector:

sqrt(sum(diff(NE).^2,2))

Make sure you are comfortable with what this is doing.

Section 2. Dead-reckoning

2.1 Introduction

Making a dead-reckoned track requires estimating the speed and direction of an animal. To do this, you need to first find out how it is moving, for example, is it moving in the direction that it is pointing? This will tell you which method (ptrack or htrack) to use for dead-reckoning.

2.2 Checking how an animal moves

For an animal constrained to a 2-d environment (e.g., a land or benthic animal) you would just use its heading as the movement direction for dead-reckoning - you would ignore the posture of the animal (e.g., an animal going northwards should move the same amount irrespective of whether it has its head up or down). For animals in a 3-d environment, the posture of the animal may help determine its direction of movement: many swimming animals move in the direction of their long body axis. In this case, their vertical movements will be closely related to their pitch angle: a negative pitch angle should be accompanied by an increasing depth and vice versa. For the test data set you have already loaded, compare the dive profile and the pitch angle:

```
pp = a2pr(A) ;
fs = A.sampling_rate;
plott(P,pp*180/pi,fs)
```

Zoom into individual dives to see whether there is a match between diving and pitch angle. Does this animal move in the direction that it is pointing? [Also check out the pitch angle during a few surfacings. Any thoughts on why this animal adopts several different pitch angles at the surface?]

More evidence for which type of dead-reckoning to do can be gained by plotting pitch angle against vertical speed:

```
v = depth_rate(P);
k=P.data>0.5;
plot(pp(k)*180/pi,v(k),'.')
```

Does the scatter plot show a simple negative regression that passes through (0,0) ? If so, we can use the pointing angle of the animal to dead-reckon (tagtool function: *ptrack*). If not, we need to ignore the posture of the animal and just use its heading when dead-reckoning (tagtool: *htrack*).

2.3 Estimating speed

We will choose *htrack* for this animal. To use this function, we first have to estimate the speed of the animal. This estimate does not need to be very precise because we will later use the GPS positions to correct the track. But the speed estimate should take into account when in each dive the animal moves more or less horizontally. For example, the animal is probably not moving very fast horizontally when it is moving rapidly vertically, i.e., when descending or ascending in a dive. We can't use pitch angle to test when the animal is descending or ascending as we have decided that body posture is an unreliable indicator of movement for this animal. However the depth_rate that you calculated above is a good indicator of vertical movement. Let's say that the animal moves at sf = 1.2 m/s on average. Thus, whenever it has a vertical speed of 1.2 m/s or more, it must be making no horizontal speed. This leads to a speed estimate like this:

```
s = sqrt(max(sf^2 - v.^2,0));
```

where sf is the forward speed value that we are assuming. Plot P and s using plott and zoom in to see when in each dive cycle this speed estimates predicts that the animal is moving horizontally.

2.4 Dead-reckoning

We now have everything we need to compute the dead-reckoned track using *htrack*. You call it as follows:

DR = htrack(A,M,s);
Look at *help htrack* to find out what is in DR and then make a plot of the track. Add axis labels with units to the plot to show that you know what is being shown. In another figure re-plot the GPS track in the local level frame (Section 1.3) and compare this to the dead-reckoned track. Are the tracks qualitatively similar? How about quantitatively (e.g., compare the overall movement in northing and easting predicted by dead-reckoning to the actual movement in the GPS track). What happens if you use a different forward speed value (*sf*)? Try to figure out a better speed value and re-compute the dead-reckoned track. Does the track more closely match the GPS track now (try plotting them on the same figure if you like to make the differences more clear)?

Section 3. Fitting dead-reckoned and GPS tracks

3.1 Introduction

Dead-reckoned tracks are inaccurate and have errors that increase with time but have very high time resolution. In comparison, GPS tracks are very accurate but have poor and inconsistent time resolution. To get the best out of the two types of movement information, we need to merge the tracks.

3.2 Fitting the tracks

There is no standard way to merge DR and GPS tracks. A Kalman filter would be a good idea provided that you know something about the dynamics of the animal and the errors in the position measurements. Here we will use a simpler method that simply forces the dead-reckoned track to coincide with the GPS points. It does this by estimating a 'current' vector for segments of the DR track between two GPS points. The current gets added to each DR point in the segment to correct the positional error accumulated over the interval. The function to do this is:

[DD,CC] = fit_tracks(NE,POS.data(:,1),DR,fs);

See *help fit_tracks* to find out what the outputs are and what the function is doing. Plot the resulting merged track (DD) on top of your GPS track (the LLF one - we always work in a LLF for track estimation):

```
plot(NE(:,2),NE(:,1),'b.-')
hold on
plot(DD(:,2),DD(:,1),'g')
```

Zoom in to see the difference in time resolution of the GPS and DR track. Compare the DR track prediction during relatively straight line travel to times when the animal is staying longer in an area (i.e., a knot in the track such as at [4700,-1500]). What can you infer about the animal's movements below the surface in these two types of travel?

The second output variable from fit_tracks contains the 'currents' computed by the algorithm to force the two tracks to match. These may reflect real currents that are advecting the animal but could also be due to an incorrect speed estimate or errors in the sensors. Nonetheless, the predicted

currents may provide an indication of the movement of the water. You can plot the currents as a function of time to visualize this:

plott(CC,fs)

This plots two lines: the northing and easting components of the currents. The step pattern is due to the current being re-estimated for each gap in the GPS, i.e., for each dive. The overall variation in the predicted current could indicate a tidal factor but the data set here is too short to evaluate that.

3.3 Estimating errors

Just because we force the DR track to coincide with some GPS points, it doesn't mean that the resulting track is accurate between GPS points. To get an idea of what sort of errors there could be, we can try removing some GPS points and refitting the track. For example, let's remove 3 of every 4 GPS points but keep the start and end points of the whole track:

k = [1:4:size(NE,1),size(NE,1)] ;
DD4 = fit_tracks(NE(k,:),POS.data(k,1),DR,fs);

Add this track to your previous track plot - use a different colour so you can see which track is which. Zoom in to parts of the track to see how much the two different track estimates differ. We can use the same trick as before to measure the difference between DD (dead-reckoned track merged with all GPS points) and DD4 (dead-reckoned track merged with every 4th GPS point):

err=sqrt(sum((DD-DD4).^2,2));
plott(err,fs)

These are the distances between the two track estimates in metres. To put these differences in context, compute the average time between GPS points used to fit each of the tracks:

mean(diff(POS.data(:,1)))/60 % time in minutes between points in DD
mean(diff(POS.data(k,1)))/60 % time in minutes between points in DD4

This animal surfaces often giving an opportunity to collect GPS positions frequently. What magnitude of errors might you expect for an animal such as a sperm whale that dives for an hour?

3.4 Finding features in the track

Now we have a track estimate with fine temporal resolution, we can run quantify measures such as residence index to see when and where the animal spends more time:

[RI,tri] = residence_index(DD,fs,1000,50) ;
plot(tri/3600,RI)

This plots time in hours and residence index. The residence index is computed with a 50 m circle and with a maximum memory time of 1000 seconds. See *help residence_index* for more details about what this function is doing. Intervals with a high residence index represent times when the animal is staying in a small area. Conversely, low residence indices are associated with largely straight-line swimming. It is often assumed that intervals of high residence index indicate times of foraging. An independent indication of foraging that will be introduced in another lecture is the norm jerk. This works best with a high accelerometer sampling rate, higher than the 5 Hz being used here. But just for fun, let's plot the dive profile, the norm jerk and the residence index to see if

this data set supports the notion that foraging occurs primarily in times of 'area restricted search' (ARS) i.e., high residence index:

plott(P,RI,1/mean(diff(tri)),njerk(A),fs)

Zoom in to see if there are more spikes in the norm jerk during dives when RI is high.

In case all of that is not enough: If you want to practice your Matlab skills, how about writing a function called llf2lalo() that converts positions in a local level frame into latitude and longitude. It will need as inputs both the LLF and the anchor point of the LLF, i.e., the latitude and longitude of (0,0) in the LLF.

Section 4. Dead-reckoning with ptrack

4.1 Introduction

Up until now, we used htrack to dead-reckon because the animal in testsert4 (a harbour seal) spent a lot of time in dives moving horizontally but at a high pitch angle, i.e., its pointing vector (body longitudinal axis) did not coincide with its direction of motion. Here we will dead-reckon an animal that does seem to move in its pointing direction meaning that ptrack can be used.

4.2 Decimating

Load data set testset3.nc. Check the metadata to find out what animal it is from.

The sampling rate of these data is unnecessarily high - 25 Hz - for dead-reckoning so we need to decimate it before computing the track. A, M and P all need to be decimated by 5 to get a 5 Hz sampling rate, e.g.,

Ad = decdc(A,5) ;

repeat for M and P.

Plot the decimated dive data to see what the animal is doing.

4.3 Dead-reckoning

.