Defining effort for photo-based analysis of 2012-2013 BOSS data

June 1, 2016

This document describes how effort is defined for photo-based analysis of 2012-2013 BOSS (Bering-Okhotsk Seal Survey) data. Previous analysis (e.g. Conn et al. 2014, MEE) used a hot spot based approach where total effort was defined by the width of the thermal swath. However, this approach results in a large number of observations where species is unobserved (hot spots outside of the photographed footprint) and can lead to extremely slow software execution times. As such, a decision was made to try to base effort on the photographed footprint and only include hot spots in the analysis if they have an accompanying photograph. Although calculation of such a footprint seems straightforward in principle, there are several issues with the data that made this effort complicated. First, the temporal resolution of photograph time stamps is limited to one second, but the duration between successive photographs was at a finer resolution. Also, photographs made during turns require use of a roll value to calculate effort (and extremely oblique angles may have poor detection). Unfortunately, plotting roll as a function of time suggests that the AHRS unit we were using to gather roll data often produced exaggerated or suppressed values. This was confirmed in a recent flight in the Chukchi Sea where banking turns of 25 degrees were being recorded as e.g., 6 degrees by the AHRS unit. Roll values also appeared to be biased. As such, we need some other way to deal with roll, and our solution is to filter out turns based on change in bearing (see below). This document does not contain complete code for effort calculations. Formatting data was actually accomplished in two R scripts, first with `Calculate_area_surveyed_crawl.R` to produce the dataset `Area_photographed.rda`. Further filtering and formatting into a form suitable for spatio-temporal analysis by the BOSSst package was done in `format_Bering20xx_all.R`.

Formulation of effort in `Calculate_area_surveyed_crawl.R` begins by loading in several data sets provided by JML. These include `boss_geo_sp.Rda`, which is a SpatialPointsDataFrame with a record for every on-effort photograph, and `grid_spdf.rda` which is a SpatialPolygonsDataFrame gives the BOSS survey grid (see the `bass` package for construction of this grid. The following steps were taken in formatting effort.

1. Remove photographs that don't interesect the survey grid or are over land

2. Remove photographs from the flights `13_AeroFl14` (starboard camera) and `13_AeroFl05` (port camera) - these had exposure or lens issues and aren't reliable.

3. Produce unique identifiers for each 'leg' of a trip by flight, grid cell, and camera. Legs were further split within a grid cell if the time between successive photographs exceeded 5 seonds.

4. Determine the position of the aircraft when each photograph was taken by distributing photographs systematically at equal intervals along the flight track of a given leg. If there were 10 or more photographs in the leg, this was accomplished using the `crawl` package, so that turns could be properly modeled. If fewer than 10 photographs, plane position was base on linear interpolation or extrapolation of positions, depending on whether GPS positional data were duplicated among sucessive photographs.

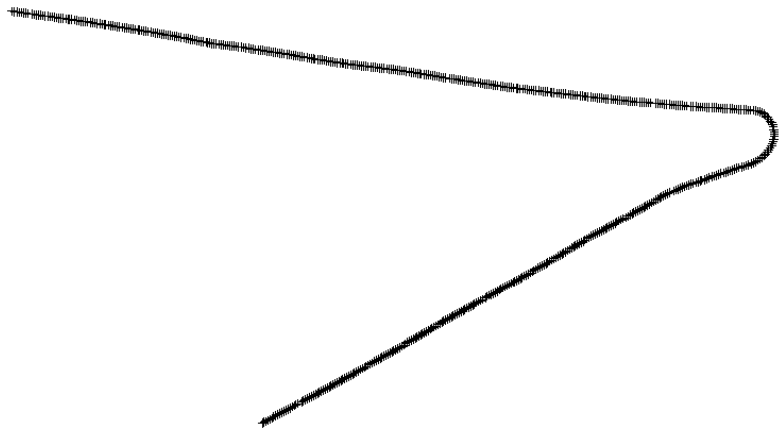
5. Acknowledging that roll data were poor, we eliminated photographs during turns by employing a filter based on bearing. Bearing (in radiaans) was calculated using the `atan2` function in R. Upon examining bearing data on several turns, it was apparent that it was necessary to separate geopositional error (which sometimes resulted in large changes in bearing at very short intervals) from actual turns. As aircraft tend to take a substantial amount of time to conduct banks, actual turns were better diagnosed using longer seqeuences of bearing changes than absolute numbers. To accomplish this we calculated $\Delta_i = Bearing_i - Bearing_{i-1}$. We then filtered out turns by deleting any photographs occurring during intervals for which $Delta_i > 0.01$ for 10 successive photographs.

6. Using several functions `get_footprint_corners_port`, `get_footprint_corners_center`, and `get_footprint_corners_starboard` together with the altitude, interpolated aircraft position, and camera angles associated with a given flights camera setup, calculation the position of photograph vertices. Note that these functions also work with roll values (up to say, 10 degrees or so), but these calculation were all made assuming roll=0 given that roll data were unreliable and turns were already filtered out.

7. Photographs were then georeferenced by their corner vertices within a SpatialPolygonsDataFrame in R, and the total area surveyed for each flight and grid cell could be determined by forming using the functions `gUnionCascaded` and `gArea` in the `rgeos` package. These functions properly account for overlap among photographs.

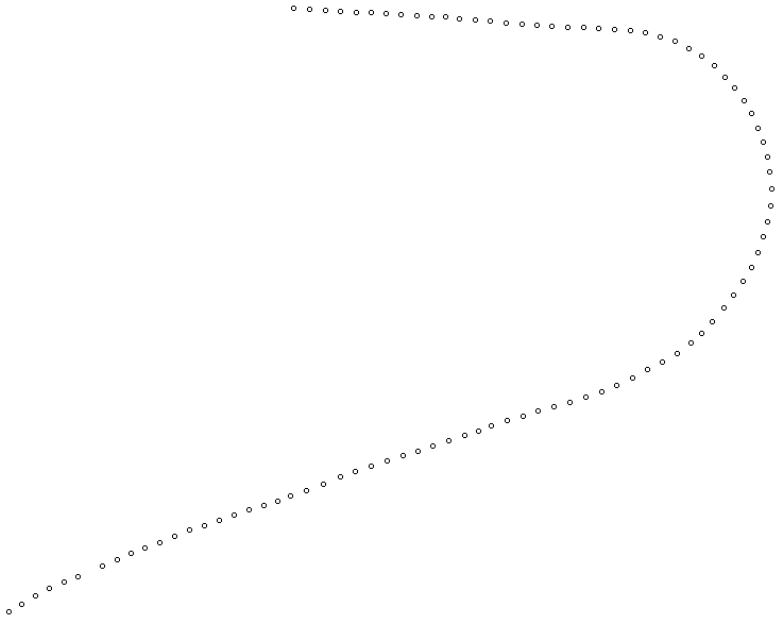
Let's take a quick look at some of these calculations in practice, using some previously saved data from `Calculate_area_surveyed_crawl.R`. Most 'legs' are straight lines, but we'll pick one out with a turn for illustration. In particular, we'll examine the port camera in flight `12_AeroFl01`'s travel through grid cell 701.

```
library(sp)
library(rgeos)

## rgeos version:  0.3-15, (SVN revision 515)
##  GEOS runtime version:  3.4.2-CAPI-1.8.2 r3921
##  Linking to sp version:  1.2-1
##  Polygon checking:  TRUE

library(crawl)

## Loading required package:  mvtnorm
## Loading required package:  raster
## crawl 1.5 (2015-8-19)
##  Type 'demo(package='crawl')' to see a list of demos for this package.
##  The raw code for the demos can be found by typing 'system.file('demo',
package='crawl')'

setwd('c:/users/paul.conn/git/BOSSst')
load('boss_geo_sp_tmp3b.Rda')
FlightCellSide_unique_IDs = unique(FlightCellSide_ID)
Interp_loc = coordinates(boss_geo)
Which_photos = which(FlightCellSide_ID==FlightCellSide_unique_IDs[11])
plot(boss_geo[Which_photos,])
```



There's a lot of photographs here, so let's zoom in on the turn:

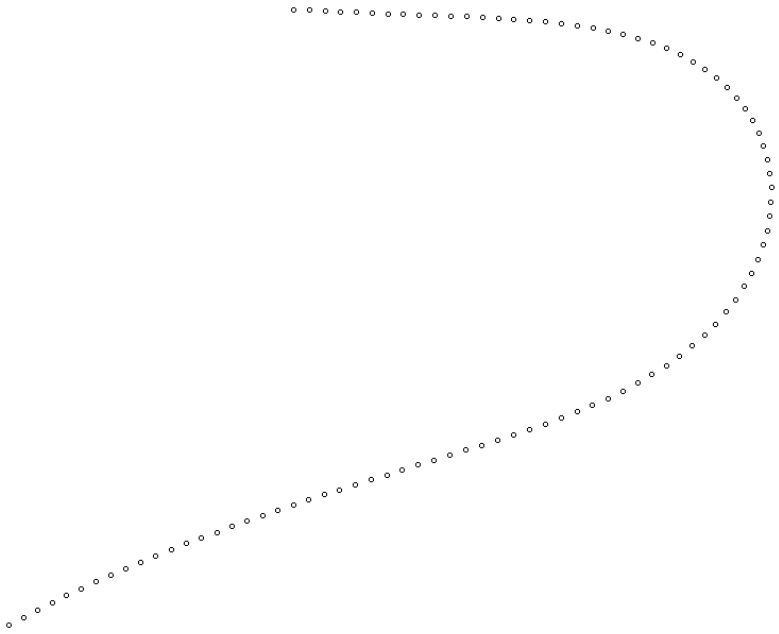
```
plot(boss_geo[Which_photos[311:405],],pch=1)
```



This plot brings to light a few issues. First, there is a small gap in the photographs, likely owing to time stamp precision error. We'd like to redistribute photographs so they are roughly equally spaced along this trajectory. We'd also like to filter out photographs that occur during the turn because roll is not known, and oblique angles may affect detection rates. Let's start by running a state space model on the positions to reallocate spatial positions so they are more even.

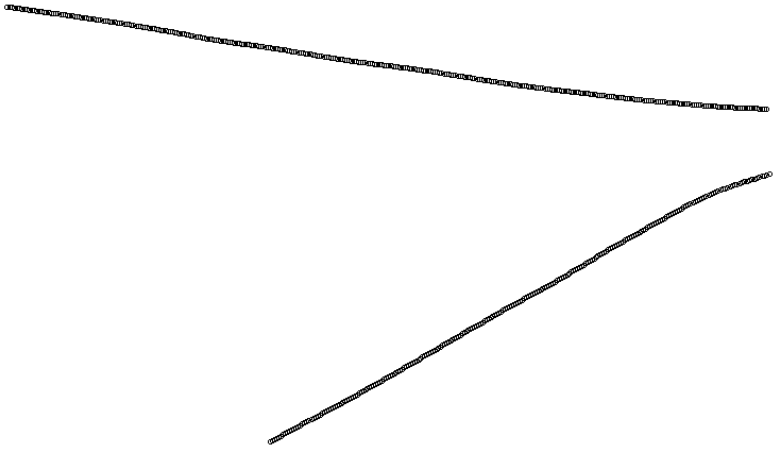
```
n_photos=length(Which_photos)
 Coords = coordinates(boss_geo[Which_photos,])
 I_dup = (Coords[2:n_photos,1]==Coords[1:(n_photos-1),1] & Coords[2:n_photos,2]==Coords[1
 Duplicated = as.numeric(which(I_dup==1))+1
 Cur_df=data.frame(img_dt=as.numeric(difftime(boss_geo[Which_photos,]$img_dt,boss_geo[Whi
 Cur_df$x = Coords[,"coords.x1"]
 Cur_df$y = Coords[,"coords.x2"]
 if(length(Duplicated)>0)Cur_df=Cur_df[-Duplicated,]
 Duplicated = which(duplicated(Cur_df$img_dt)==TRUE)
```

```
    if(length(Duplicated)>0)Cur_df=Cur_df[-Duplicated,]
    Inits = list(a1.x = c(Coords[1,1],0), a1.y =c(Coords[1,2],0),P1.x=diag(c(10000,10000)),P
    #now run through crawl
    Fix=c(log(100),log(100),NA,NA)
   my_fit <- crwMLE(mov.model=~1,err.model=list(x=~1,y=~1),data=Cur_df,coord=c("x","y"),Time

##
## Cannot calculate covariance matrix

    if(is.character(my_fit)==TRUE){
      Fix=c(log(150),log(150),NA,NA)
      my_fit <- crwMLE(mov.model=~1,err.model=list(x=~1,y=~1),data=Cur_df,coord=c("x","y"),T
    }
    elapsed = Cur_df[nrow(Cur_df),"img_dt"]-Cur_df[1,"img_dt"]
    t_diff=elapsed/(n_photos-1)
    predTime=as.numeric(Cur_df[1,"img_dt"]+c(0:(n_photos-1))*t_diff)
    Preds=crwPredict(my_fit,predTime=predTime)
    Which_pred = which(Preds$locType=="p")
    Interp_loc[Which_photos,]=cbind(Preds$mu.x[Which_pred],Preds$mu.y[Which_pred])
#formulate new sp points object with interpolated locations
  laea_180_proj <- paste("+proj=laea +lat_0=90 +lon_0=180 +x_0=0 +y_0=0",
                         "+datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0")
boss_geo=SpatialPointsDataFrame(coords=Interp_loc,data=boss_geo@data,proj4str=CRS(laea_180_p
plot(boss_geo[Which_photos[311:405],],pch=1)
```



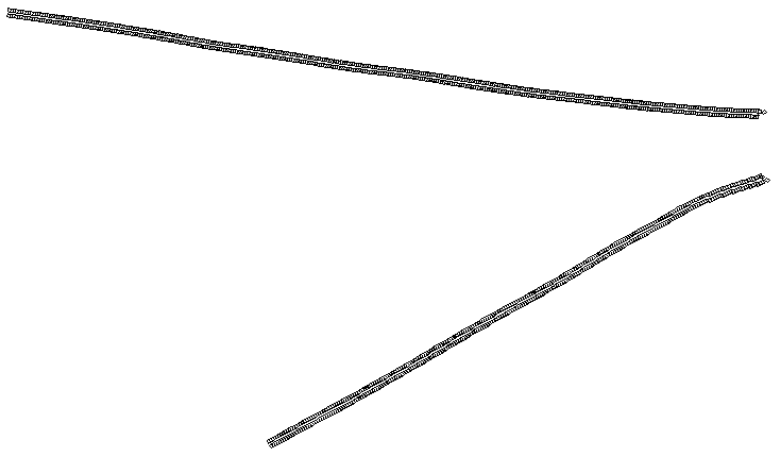
See that the gap in the lower left corner has gone away; presumably there is now less overlap in photographs now too. Rather than run the bearing filter directly, we'll just load a file with it already done to see that the turn is properly deleted.

```
setwd('c:/users/paul.conn/git/BOSSst')
load('boss_geo_sp_tmp5.Rda')
Which_photos=which(boss_geo$flightid=="12_AeroFl01" & boss_geo$side=="Port" & boss_geo$Gridl
plot(boss_geo[Which_photos,],pch=1)
```



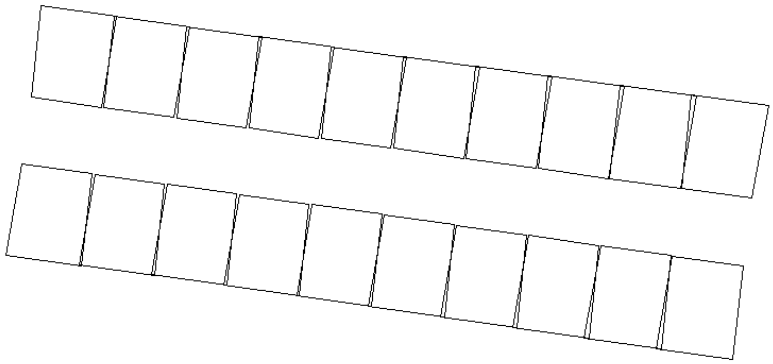
Now, it's time to georeference all of the photographs and calculate area surveyed. This takes a fairly long loop, so we'll just load a SpatailPolygons object where this where this is already done.

```
setwd('c:/users/paul.conn/git/BOSSst')
load("12_AeroFl01_701_spdf.Rda")
plot(SPDF)
```



This is kind of hard to see, so let's take a look at the first 10 photos from each camera

```
plot(SPDF[c(1:10,562:571)])
```



So looks like we do a good job of addressing a small amount of overlap in Aero Commander photos from this flight.