Analysis Overview

Daniel Parthier

13.11.2019

Analysis Brainstorming

The data generated by DeepLabCut will be always in this csv format, however, it can change in terms of labels or how many columns will be generated (Label number). We for now will work with a 1/3 of 25 Hz which makes it a frame rate of 8.33. I implemented a function which takes the number of objects and later seperates the coordinates without knowing where they were. This is implemented with k-means clustering. My hope is that it will be flexible for different (immobile) objects independently of the count.

```
# Information about recording
FileName <- "RawData/20180903_Schmitz_PaS_B1_C1_NL_Day1_NOVEL_DSC001663DLC_resnet50_NOVEL_VIDEONov7shuf
FrameRate <- 25/3
ObjectNumber <- 2
```

To analyse the data we will all have to start with the CSV. This is fairly easy to load but the naming convention requires some adjustments.

```
RawDT <- fread(file = FileName, skip = 0)
RawDT[1:3,1:3]
```

```
##       scorer DLC_resnet50_NOVEL_VIDEONov7shuffle1_100000
## 1: bodyparts                                     leftear
## 2:    coords                                           x
## 3:         0                           138.72007751464844
##    DLC_resnet50_NOVEL_VIDEONov7shuffle1_100000
## 1:                                    leftear
## 2:                                          y
## 3:                          422.8099060058594
```

As we can see the first two rows have labels. Therefore we have to clean and seperate them and later stitch them back together as unique column names.

```
# Load data
DataSet <- fread(file = FileName, skip = 2)
LabelNames <- fread(file = FileName, nrows = 1)

# Construct unique columns
ColumnNames <- paste(LabelNames, colnames(DataSet), sep="_")
colnames(DataSet) <- ColumnNames
# Print table
DataSet[1:3,1:3]
```

```
##    bodyparts_coords leftear_x leftear_y
## 1:                0  138.7201  422.8099
## 2:                1  141.6581  390.2830
## 3:                2  141.6607  390.1670
```

Having a "clean" data set allows us now to calculate properties of the movement or the position. Here we will quickly look at the Distance (length of the vector at any given point) and the speed (vector length as function of time).

```
# Calculate Speed/Distance Travelled
DataSet[
  ,InstDistance := sqrt((shift(nose_x, type = "lead") - nose_x)^2+
                          (shift(nose_y, type = "lead") - nose_y)^2)][
    ,NoseSpeed := InstDistance/(1/FrameRate)][,CumDist := cumsum(InstDistance)]
```

The syntax I use is with the `data.table` approach which is rather **SQL** like than **R** or **Python**. It is just super fast and keeps a clean table. Absolutely personal preference and I am willing to learn! The object coordinates can now be used to seperate **n** numbers of objects using k-means. This will assign cluster numbers to based on location. What this means in the end is that jumping labels will be corrected. Here I will use the median XY for any given object in order to estimate the exact location without jitter. Then we will create a new table which just holds the coordinates for the objects.

```
# Find Object Location
ObjectSet <- data.table(x=c(DataSet$object1_x, DataSet$object2_x),
                        y = c(DataSet$object1_y, DataSet$object2_y))
ObjectSet$ObjectLoc <- kmeans(x = ObjectSet[,.(x,y)], centers = ObjectNumber)$cluster
ObjectCoord <- ObjectSet[,.(x=median(x),y=median(y)), by=ObjectLoc]
# Print table
ObjectCoord
```

```
##    ObjectLoc        x        y
## 1:         1 461.5869 425.1844
## 2:         2 226.5102 131.8683
```

The location is calculated by vector length of mouse (nose) to the object. This should be further optimised using the centroid of the mouse. I guess it will be more stable too.

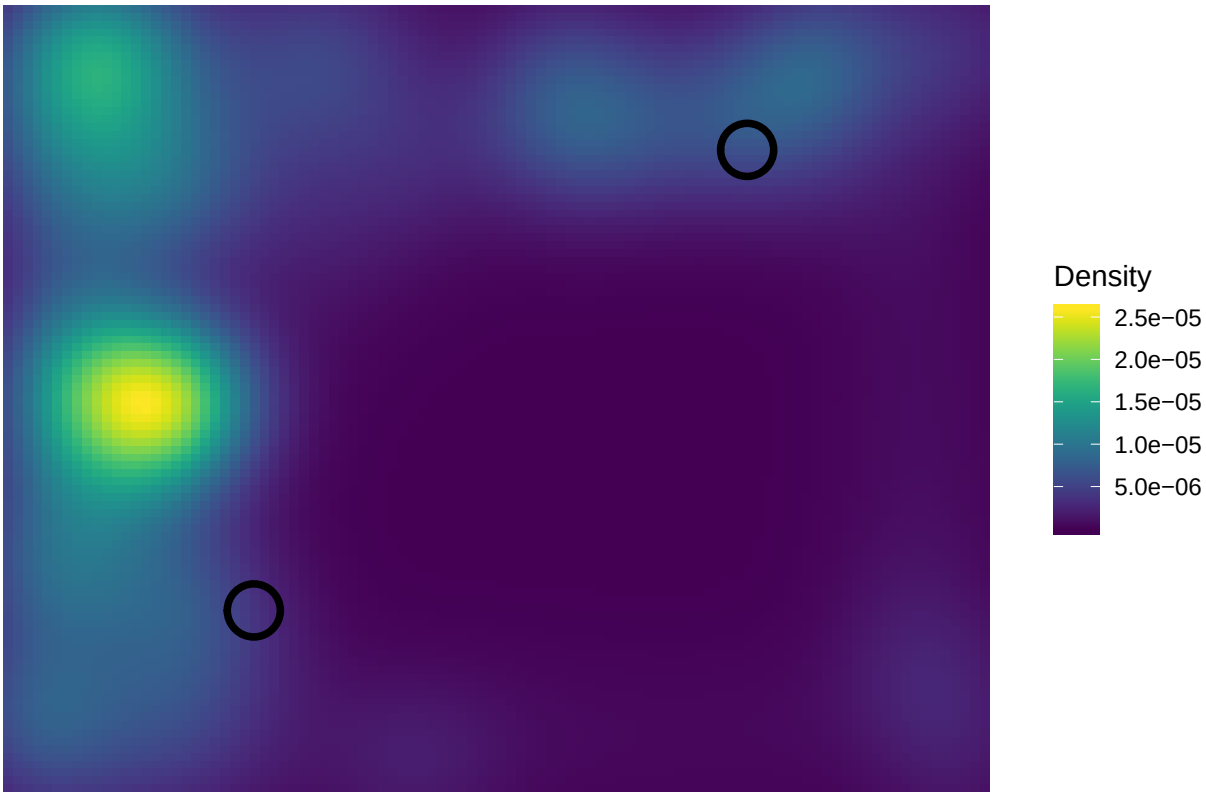
```
# Calculate object distance for two objects (should be made more flexible)
DataSet[
  ,DistToObject1:=sqrt((nose_x-ObjectCoord[ObjectLoc==1, x])^2+
                         (nose_y-ObjectCoord[ObjectLoc==1, y])^2)][
  ,DistToObject2:=sqrt((nose_x-ObjectCoord[ObjectLoc==2, x])^2+
                         (nose_y-ObjectCoord[ObjectLoc==2, y])^2)]
```

To tidy up the table and to make it easier to plot with `ggplot2` I will restructure the table to "long format". I could also use for this pupose the `reshape` function. However, here I will do it quickly manually to assign directly the right labels.

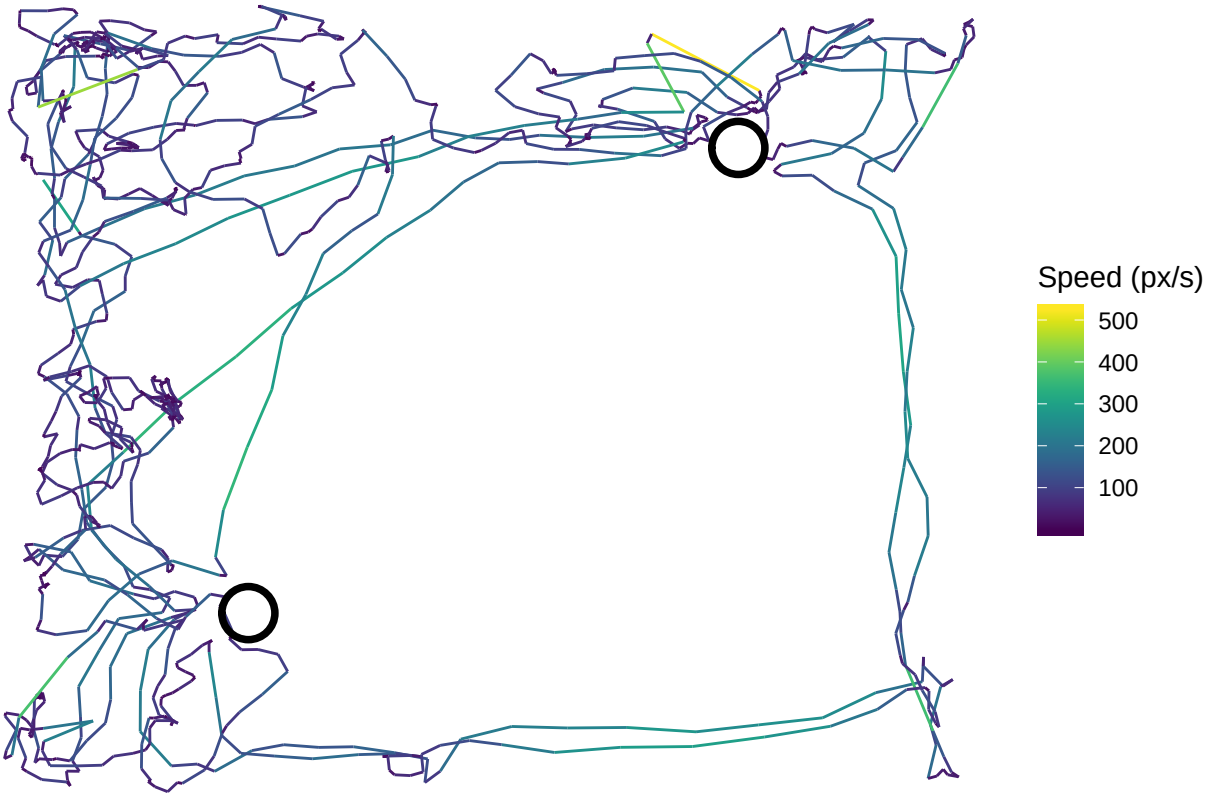
```
# Generate distance table
ObjectDistance <- data.table(Time = rep(DataSet$bodyparts_coords/FrameRate, times = 2),
                             Distance = c(DataSet$DistToObject1, DataSet$DistToObject2),
                             ObjectLoc = rep(c("1","2"),
                                             each = length(DataSet$bodyparts_coords)))
```

We can then generate plots showing our computed parameters. Here we should check though which parameters are of importance and how to present them. Furthermore, we will have to check for the object size and validate the pixel size to get later proper measurements.

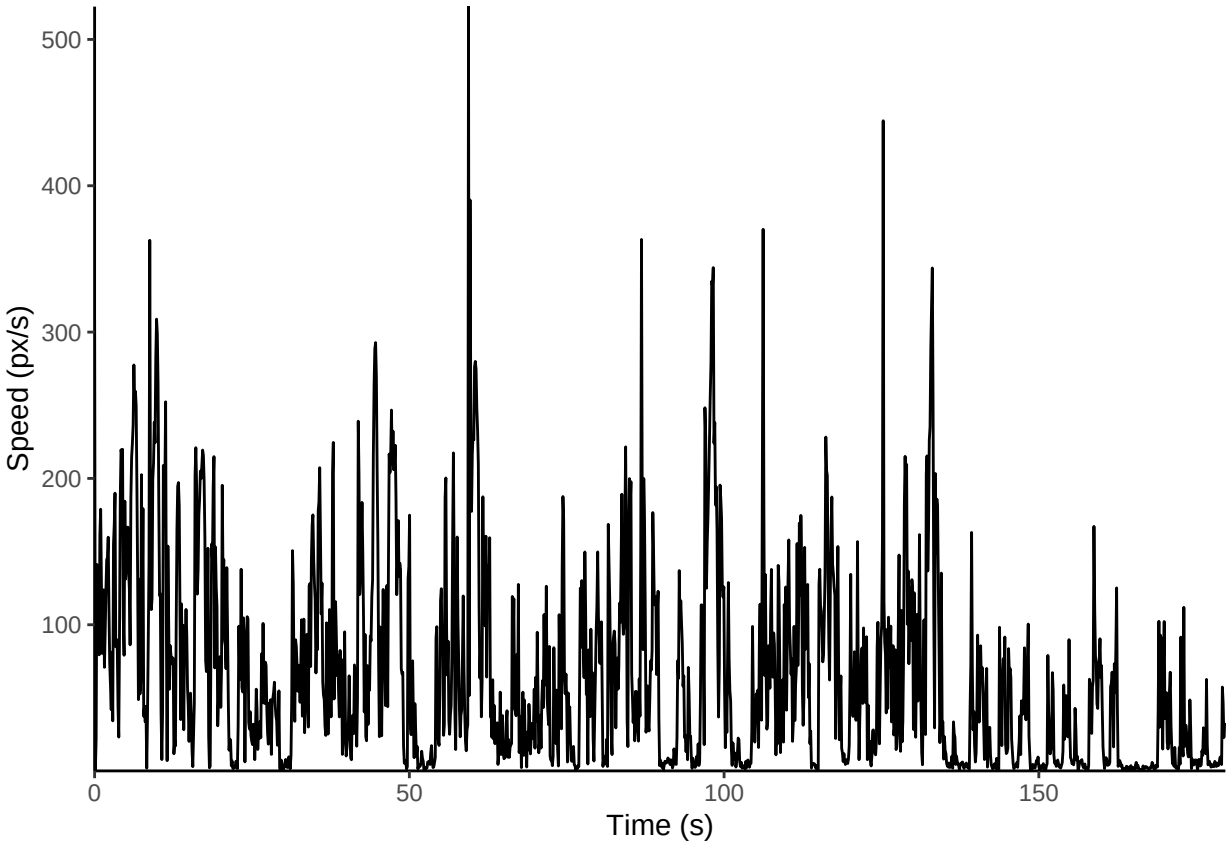
We will take a look at the density of the location. This shows the likelihood of the animal being at any given point on the map. Yellow will indicate a high probability of the animal visiting the spot. Blue is very low or not even visited once. I also added the objects as circles. The size of course is not correct and would need to be changed according to the real object (I guess but it is not the highest priority).



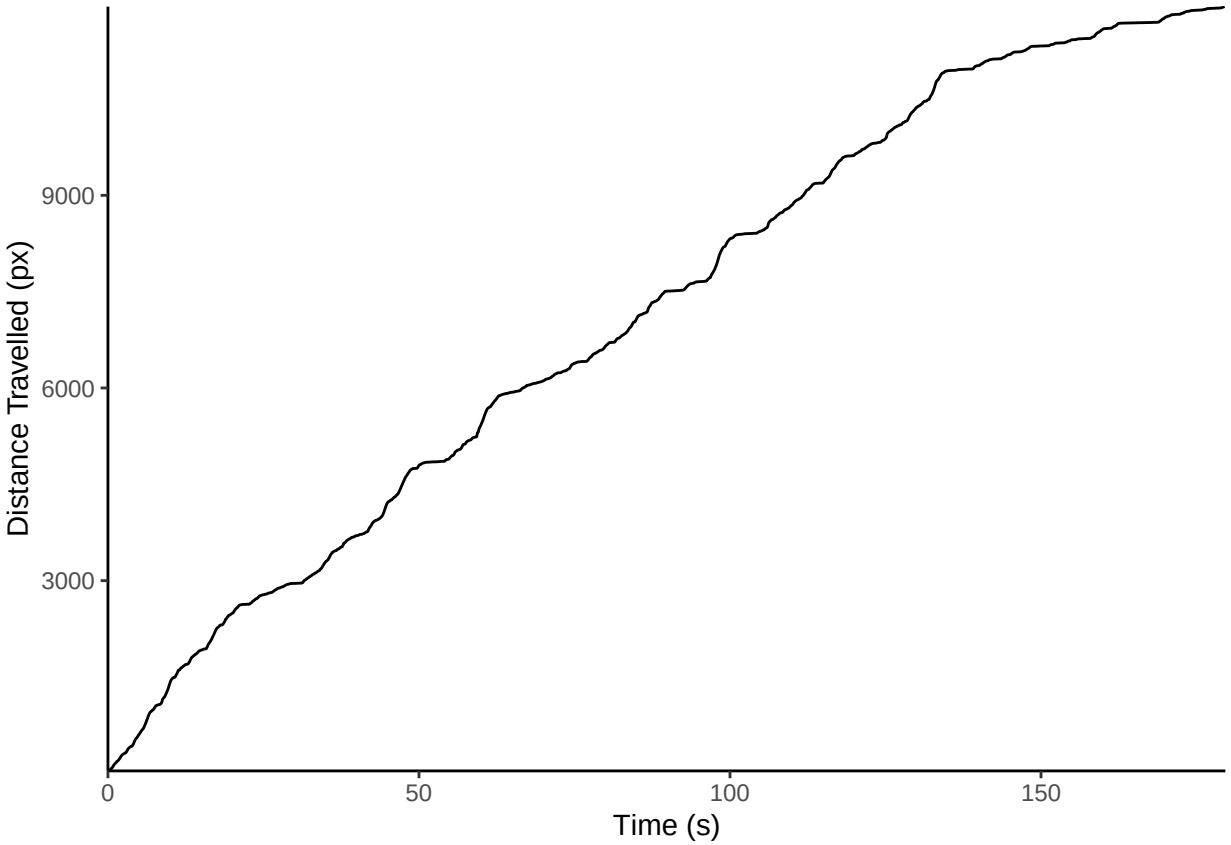
The second plot shows the trajectory of the animal and in colour code the speed at the given segment. This might This will indicate where the animal is standing still and where it is just running from A to B.



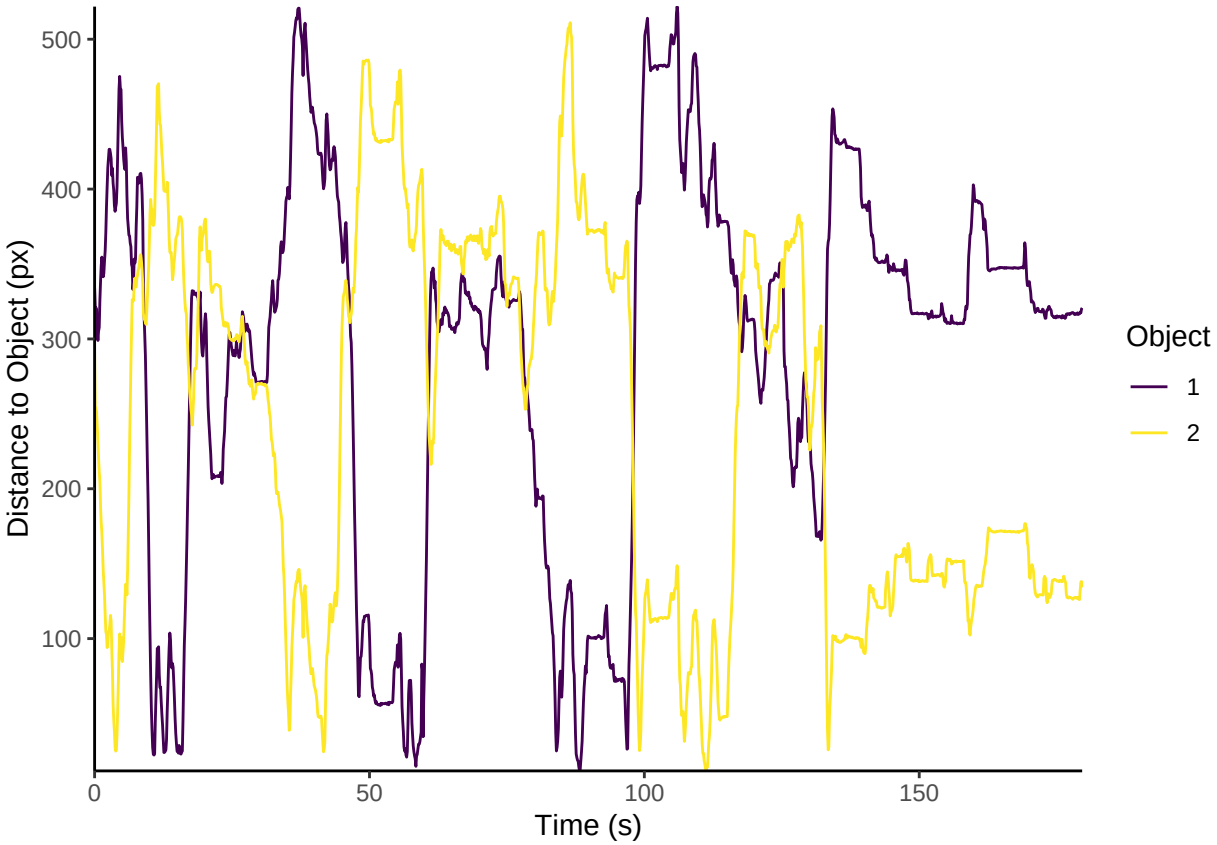
The speed plot is showing the change of speed over time. It looks like it might not change much but more analysis could indicate a trend that they are just moving less in the end due to boredom or exhaustion.



Similarly to the speed the distance over time as cumulative distance could also be a measure of exploration. A linear relationship would mean equal exploration whereas a flattend curve would mean less exploration in the end.



The object distance plot shows the distance to object 1 and 2 at any given point in time. This plot would show a continous preference or equally distributed exploration.


Distance to Object (px)
500
400
300
200
100
0
50
100
150
Time (s)
Object
1
2