## Running the analysis

## All analysis (expect for R and Python code, which are run from their corresponding folders in ‘analysis/analysis\_code/’) is run from the analysis.m file. The file is divided to sections, use Ctrl+Enter to run each section separately or F5 to run the entire script. You can use Matlab shortcuts (Ctrl+down/up arrow) to jump between sections. Most sections save their results to the processed data folder (which is not included in github to save storage). This means that you don’t have to rerun the entire analysis every time you want to make a small change or make a new plot.

### MATLAB scripts

Sections and their purpose (some sections are further explained in the “additional information” section below):

#### Preprocess the data:

1. Define following parameters: if any of these are changed, rerun the entire analysis.
   1. SUBS
   2. DAY - only experiment 3 has ‘Day1’.
   3. picked\_traj - Wasn’t fully implemented, since we were only interested in the trajectory to the target.
   4. SIMULATE - Reduces the number of analyzed trials. Since adding a keyboard session in exp 4 prolonged the exp, we checked how many trials can we reduce from exp 2 / 3 and still receive an effect.
   5. NORMALIZE\_WITHIN\_SUB
   6. NORM\_TRAJ
   7. MIN\_SAMP\_LEN
2. Simulate (optional, see ‘SIMULATE’ in a.iv above)
3. Create Proc Data File
4. Add Fields - some fields were added after running some of the subjects. This section adds them to their data.
5. Add trials
6. Preprocess and Normalize
7. Trial screening
8. Subject screening
9. All of the above should be recompiled if you made changes to the analysis parameters at the top of the code (‘parameters’ section).

*Perform feature extraction*

producing meaningful variables from the data.

Maximum absolute deviation, Heading angle, implied endpoint, changes of mind, total distance traveled, AUC, Velocity, Max velocity, Acceleration.

*Average features within participant*

Also sorts them to according to trial type (e.g. congruent/incongruent). Bad trials are marked and only good ones are included in the averages.

*Extract feature that are computed over the average trajectories*

1. Reach area
2. D’ computation - Uses machine learning to classify trials to each of the conditions. Then measures the accuracy of classification to estimate the main tasks’ sensitivity to the conditions (see Meyen et al. (2022). Advancing research on unconscious priming: When can scientists claim an indirect task advantage?).
3. Velocity profile

*Average features between subjects.*

*Conduct FDA and then count trials within each condition.*

For explanations about FDA see Craig Chapman’s documentation within

‘/imported\_code/craig\_code/Using Functional Data Analysis\_v1\_april2011.pdf’).

we decided against using this analysis since it wasn’t fruitful.

*Plotting params section*

sets all the params for plotting (Duh….), but also aggregates all the subjects data to a single variable.

*Single sub plots*

makes a separate plot for each subject.

*Multiple subs average plots*

makes plots with the average data of all subjects.

*Plots for paper*

Neatly Organizes all the plots I needed for the paper. After running this run the next section ‘Add labels to subplot’.

*GUI*

This section creates a GUI that allows you to visually examine each trial and the processed trajectory that was fit to it.

*Movement Time Percentiles*

Helps determine what duration to trim the trajectories to (when NORM\_TRAJ = 0). Shorter trajectories are excluded from analysis. E.g. we are looking for a movement time that is longer than 10% of the trials (i.e. exclude only 10%). Then we look for the RT of the 10th percentile.

*Format to R*

Converts analysis output to R suitable format. Run this before running R scripts.

*Tree-BH Correction*

Run this after running the R scripts to get the final significance values of the MATLAB+R analysis.

### R Scripts

Use this code to either:

1. Create mixed regression model with ‘side of correct answer’ (left/right) as a random effect.

To do so, uncomment the functions that contain ‘trials’ and ‘coefs’ in their name, and comment those that have

‘avgs’.

1. Test assumptions of the statistical tests used in MATLAB, Run permutation tests where they fail, Compute effect sizes for these tests (and the regular t-tests).

This code updates the p-values for variables that violated assumptions. After running it, run ‘Tree-BH Correction’ section in MATLAB to correct p-values for multiple comparisons.

To run the analysis:

Use exe\_analysis.R, but first define the following parameters within main.R:

1. DAY
2. SUBS
3. PICKED\_TRAJS
4. NORM\_FRAMES
5. STNDRD - relevant when creating mixed regression models.
6. RAND\_EFF - relevant when creating mixed regression models.
7. R/K\_VAR\_NAMES

### Python Scripts

Their purpose is identical to the ‘D’ computation’ section in the MATLAB code. However, it Uses more complex machine learning methods.

To run the analysis:

Execute fit\_n\_pred.m but first define the following:

1. ‘SUBS’
2. ‘DAY’

Comment/uncomment algorithms according to the ones you wish to use (Naive base / Ensemble / Gradient boosting).

## Additional information

### Trajectory Preprocessing

All preprocessing was done according to Gallivan, J. P., & Chapman, C. S. (2014). Three-dimensional reach trajectories. A user manual is found at ‘/analysis/analysis\_code/imported\_code/craig\_code/Using Functional Data Analysis\_v1\_april2011.pdf’

#### Fill Missing Data

Applies only to trajectories that have more than 1 sample.

Uses inpaint\_nans to replace NaNs with interpolated values .

**Inpaint\_nans**:

Written by John D'Errico.

John D'Errico (2022). inpaint\_nans (https://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint\_nans), MATLAB Central File Exchange. Retrieved February 15, 2022.

Is an advanced method for interpolating trajectory data while keeping the dynamic information of the reach.

Was used by Gallivan, J. P., & Chapman, C. S. (2014). Three-dimensional reach trajectories

#### Filter Trajectory

Applies only to trajectories that have more than 1 sample.

Apply low pass butter worth filter of 2nd order with cutoff at 8Hz on each axis separately to clean any noise in the recording.

Since the noise is canceled, we can later over-fit the data without concern for capturing noise with our fit (Using Functional Data Analysis\_v1\_april2011).

#### Set Origin

Set the first sample in each trial as the axes origin by reducing it from each sample.

Set the time of the first sample in each trial as t=0.

#### Trim Onset Offset

Applies only to trajectories that have more than 1 sample.

Compute velocity and apply low pass butter worth filter of 2nd order to it with a 10Hz cutoff.

**GetOnset:**

Trim trajectory to the first sample in which the 3 consecutive samples had a velocity greater than 2cm/s and a total acceleration greater than 2cm/s^2.

**GetOffset:**

Trim trajectory ending as the point closest to the screen.

#### Normalize\_trajs

To compare X coordinates between trials, Z coordinatesmust be equal between trials.

Achieve that by fitting a function to the data and extracting 200 points equally spaced along the Z trajectory.

ToNormalize – Number of the Motive marker you used to track the movement.

**General information about B-spline:**

A function composed of sections (splines), each section is a separate polynomial function.

Degree – the highest order of the polynomials.

Knot – the point where two splines connect.

Knots = coefficients + degree.

B – basis functions. A list of functions that are used to create each of the spline functions.

Cubic spline – spline function where all the polynomials are of the 3rd degree.

C – Continuity, a property of a spline indicating how smooth it is.

Roughness penalty – **Used in a “smoothing spline”, and determines how smooth (as**

**opposed to over-fit) will the spline be.**

Error function (which we try to minimize when fitting the spline):

Text, whiteboard

Description automatically generated

– penalty factor, determines how much weight will be given to smoothness as opposed to

overfitting the data.

Chart, diagram

Description automatically generated

For more information about the roughness penalty:

* [Mathlab's cubic spline function](https://www.mathworks.com/help/curvefit/csaps.html" \l "d123e27998)
* [Article about functional data analysis](https://www.researchgate.net/publication/227578062_Functional_Data_Analysis)
* [R Function for fitting splines](https://link.springer.com/content/pdf/10.1007%2F978-0-387-98185-7_5.pdf)

**From “Using Functional Data Analysis\_v1\_april2011”:**

Fit a B-spline function to the each axis with a spline at each data point. The B-spline is of the 6th order which keeps the original data up to the 3rd derivative.

A roughness penalty is applied to the 4th derivative of the data to smooth the data.

Use the fitted function to produce a high-resolution representation of the trajectory (1000 samples).

Perform the normalization in space by extracting 200 points equally spaced along the Z axis (perpendicular to the screen) from the high-resolution representations.

Since the subject can move backwards the movement is divided to sections according to the movement direction (towards / away from the screen). The 200 points are equally divided along the TOTAL path traveled in the Z axis (if the subject moved 2 forward, 1 backward and 3 forward, the total is 6).

### Trial screening

Test the original data (before preprocessing) and excludes trials that fall within the following criteria:

* Too early – movement initiation started lees than 100ms after target was displayed indicating a predictive response.
* Late response – movement didn't start 320ms after target display.
* Extremely slow movement – movement time was 3SD above the subject's mean, or simply above a preset threshold.
* Short reach distance – distance along the Z axis between first and last sample was shorter than:

*Onset variation* is a 3cm error margin that compensates for small variations in the location of movement onset

* Too many missing samples – trajectory has more than 100ms of missing samples (usually due to recording issues or obstruction of the marker from the camera's view).
* Short sample – Recorded trajectory is shorter than 100ms.
* Missed targets – The last sample is further than 12cm from either of the targets (on the X,Y plane).
* Bad stimulus duration – the duration of one of the stimuli (fixation, masks, prime, target) deviated by more than 2ms from its designated duration.
* Incorrect – target classification was wrong.
* Subject didn't perform the trials (quit)
* Bad preprocessed length – preprocessed trajectory’s length is different than the one assigned to ‘trim\_len’ in ‘preprocessing’ section in ‘analysis.m’.

### Subject screening

Excludes subjects that fall withing the following criteria:

* Not enough trials – The number of valid trials which also had a PAS rating of 1 is smaller than 60.
* Not enough trials in each condition – The number of valid trials in each condition (congruent /

incongruent) which also had a PAS rating of 1 is smaller than 25.

* Bad performance – subject was correct less than 70% of the times in the target classification task (counts only trials with PAS=1).
* Aware of prime – prime recognition was above chance indicating the subject is aware of the prime.

### Parameters extraction

#### MAD

Maximal absolute deviation.

Locates for each trial the point that is furthest away from the line that connects the start and end points (known as the maximally deviating point, mad\_p).

MAD is the distance of that point from that line.

#### Functional data analysis (FDA)

getRMMeans – calculates the mean trajectory for each subject at each condition.

Fanovan – runs a repeated measures aNOVA analysis at every point along the trajectory.

For an explanation why this doesn't violate multiple comparisons limitations see Craig's document " Using Functional Data Analysis\_v1\_april2011.pdf" section: 5.2 – Statistical violation and functional interpretation.

Subject is used as a random factor.

#### Reach area calculation

Calculates the area circumscribed between the average trajectory to the left target and the average trajectory to the right target.

Does so for each condition separately.

Area calculation:

Finds minimal X value and draws a line at that value parallel to Z axis.

Computes the area between each trajectory and this line and subtracts the results, producing the area between the trajectories.

To avoid negative values after intersections, calculate the area seperatly between each pair of intersections..