OpenPoseR demo (version 1.0)

Project web site: https://github.com/trettenbrein/OpenPoseR

Contents

What is this?

This is a quick intro to OpenPoseR. OpenPoseR is an R package that provides some functions for analyzing motion-tracking data derived from video files using OpenPose. The original motivation for creating this package was to control video stimuli in sign language and gesture research, but the provided functionality may also be useful for other purposes.

In this demo, you will learn how to (i) organize your video data, (ii) perform motion-tracking using OpenPose, (iii) process the raw data with OpenPoseR, and (iv) prepare these data for plotting and/or further statistical analysis.

Prerequisites

To run this tutorial, you will need a current version of R, RStudio, as well as a version of OpenPose installed on your machine. See the web sites of these projects for help with installation.

Windows users: Please note that in order to be able to run some of the sections in this document, you will need to install additional software like Cygwin or the Linux Subsystem to be able to run shell scripts (i.e. code chunks in this document related to the file system or running OpenPose).

Install OpenPoseR

OpenPoseR is distributed as an R package that can be loaded into your R session like any other package, so you may alrady be familiar with the process. Install the OpenPoseR package using the following commands (you will need to have the `devtools` package installed):

```
# Install devtools from CRAN (if not already installed)
install.packages("devtools")

# Install OpenPoseR package from Github
devtools::install_github("trettenbrein/OpenPoseR")
```

Load package

Once installed, the OpenPoseR package can be loaded with the following command:

```
library(OpenPoseR)
```

Look at data before analysis

Video data should be located in a single folder on your file system. In this demo, this is the folder named `data`:

```
ls
```

```
## data
## data_openposer
## demo.Rmd
## demo.pdf
## openposer-demo.Rproj
```

The folder `data` contains the following files included in this demo:

```
cd data
ls
```

```
## du_wohn_wo
## du_wohn_wo.mp4
## gebaerden
## gebaerden.mp4
## linguistik
## linguistik.mp4
## psychologie
## psychologie.mp4
```

In this demo, for every video file (e.g. `du_wohn_wo.mp4`), the folder also contains a sub-folder of the same name (`du_wohn_wo`). This folder contains all the motion-tracking data generated by OpenPose for this video. If you don't want to or cannot run OpenPose as part of this demo then this data can be used for all further processing in OpenPoseR.

The video clips show the German Sign Language (DGS) signs for

- "psychology" (`psychologie.mp4`),
- "linguistics" (`linguistik.mp4`),
- "to sign" (`gebaerden.mp4`), and
- the short DGS sentence "Where do you live?" (`du_wohn_wo.mp4`).

These videos are courtesy of Henrike Maria Falke, gebaerdenlernen.de (Creative Commons by-nc-sa/3.0/de).

Track videos with OpenPose

The actual motion-tracking for our video files is done using OpenPose, which you will need to have installed on your system in order to be able to run the commands below. If you don't want to or cannot run OpenPose as part of this demo then please skip to the section "Analyze motion-tracking data with OpenPoseR".

OpenPose uses machine-learning to identify people in video clips and fits a model of their body pose and, optionally, face, and hands to the video data. In other words, OpenPose is currently the most sophisticated means for tracking people in video clips. The results of this motion-tracking of people in video clips with OpenPose are then used for further analysis with OpenPoseR.

Fit body-pose model

As mentioned above in the discussion of directory structure, this demo already inludes tracking data for all demo video clips. To illustrate how this data was generated, we'll perform the tracking for one video clip (`psychologie.mp4`) below (provided you have OpenPose installed).

To fit the body-pose model (BODY25) using OpenPose for the example video file, run the following code (make sure to adapt the file path for the OpenPose binary to the path of your system and to uncomment the final lines of the code before running):

```
# Adjust to your system path to OpenPose binary
OPPATH="/path/to/openpose"
# Name of file used in demo
FILE="psychologie.mp4"

# Path to demo files
FILEPATH="$(pwd)/data/"


# Get file name without extension
FILENAME=$(basename "${FILE%.*}")

# Check if directory already exists
if ! [ -d "$FILEPATH$FILENAME" ]; then
  # If not, create a directory for output JSON files
  mkdir "$FILEPATH$FILENAME"
fi


# # Run OpenPose
# cd $OPPATH
# # Uncomment next line if you actually want to run OpenPose (this may take a while, depending
# # on the specifications of your system!)
# .$OPPATH/build/examples/openpose/openpose.bin --video "$FILEPATH$FILE" --number_people_max 1 --write_
```

This bash script first creates a directory for the output files with the name of the given video file (without file ending) and then runs OpenPose for this video. The argument `--write_json` writes the output to the new directory. This is the data that will be further processed using OpenPoseR.

In addition, we speed up processing a little bit by specifiyng `--number_people_max 1` as we tell OpenPose that our video clip contains only a single person.

Analyze motion-tracking data with OpenPoseR

Now that the motion-tracking has been performed using OpenPose, we can carry on with our actual analysis using the different function provided by the OpenPoseR package.

Convert OpenPose data to CSV

OpenPose stores data in JSON format. A single file is saved for every frame. To make the data somewhat easier to handle, we convert this data to CSV files using OpenPoseR's `create_csv()` function:

```
create_csv("data/psychologie", "psychologie", "data_openposer/")
```

This function will create a CSV files with the name `psychologie_body25.csv` in the given directory (`data_openposer`).

Data structure of OpenPoseR files

The file created by OpenPoseR has the following structure which (we hope) is also easily readable by humans:

x0	y0	c0	x1	y1	...
362.312	119.557	0.913335	362.388	233.906	...
362.307	119.561	0.912487	362.390	233.929	...
...	...	...	...	...	...

Every point tracked in the body-pose model (BODY25) by OpenPose has an `x` and `y` coordinate (depending on the video format). Points are nnumbered from 0–25 (see this illustration. In addition to `x` and `y` coordinates for every point, the third column for a point labelled `c` provides the confidence in the tracking of this point by OpenPose (whereas `0` means no confidence at all `1` indicates absolute certainty).

Clean data

Upon inspection, we will see that out data is likely to contain either some `0` values for `x`, `y` and `c`, indicating that a point was not tracked. This may have different causes, but we will need a consistens way of handling this. Similarly, we may want to remove points with extremly low `c` values (indicating that OpenPose had low certaintiy in having correctly detected this point).

A simple way of doing this is to use OpenPoseR's `clean_data()` function. This function allows us to remove points with 0 values as well as points with extremly low confidence ratings. For this reason, we can specify the `cutoff` argument, indicating the lowest confidence rating we are willing to accept.

In addition, as we are handling files here, we can use `file_clean()` instead of directly calling `clean_data()`. This allows us to directly work with files instead of the underlying functions, as `file_clean()` essentially provides a wraapper for OpenPoseR's `clean_data()`:

```
file_clean("data_openposer/psychologie_body25.csv", cutoff = .3, overwrite = FALSE)
```

```
## [1] TRUE
```

This creates a file called `psychologie_body25_cleaned.csv` (because we specified `overwrite` as `FALSE` meaning that the original CSV file was not overwritten by the function) with the same data structure illustrated above. The only difference is that in this file all points with 0 values as well as points below the specified `cutoff` for `c` have been imputed based on the other tracked data.

Compute velocity

Next, we can now compute the velocity of the different points of the body-pose model using OpenPoseR's `file_velocity()` function.

As we have already seen above, functions starting with `file_` ususally provide wrappers for other functions which make it easier to directly work with files. In this spirit, `file_velocity()` provides a wrapper for the functions `velocity_x()` and `velocity_y()` which makes it possible to direclty pass a file to the function and create output files. These functions compute the velocity of a point on either the x-axis or y-axis according to the following formula:

$\frac{p_t - p_{t-1}}{t-(t-1)}$

Luckily, we need not be concerned about the details, all we have to do is pass the path to a CSV file in the OpenPoseR format specififed above (i.e. created in the previous step[s]) to `file_velocity()`:

```
file_velocity("data_openposer/psychologie_body25_cleaned.csv")
```

```
## [1] TRUE
```

Compute Euclidean norm of sums of velocity vectors

Our goal is to capture the total amount of bodily motion between frames in a single value. This can be done with our data by computing the Euclidean norm of sums of velocity vectors with another OpenPoseR function called `en_velocity()`.

The `en_velocity()` function computes the Euclidean norm of sums of velocity vectors (`x`,`y`) for a tracked video using the `x`-axis and `y`-axis data generated using `velocity_x()` and `velocity_y()`. The computaton is performed according to the following formula:

$MF_1 = \|\sum_{i=1}^{N} v_i\|$

Again, we need not worry much about the details–we can simply pass the two files created in the previous step to `file_en_velocity()` which is a wrapper for `en_velocity()`:

```
file_en_velocity("data_openposer/psychologie_body25_cleaned_velocity_x.csv",
                 "data_openposer/psychologie_body25_cleaned_velocity_y.csv")
```

```
## [1] TRUE
```

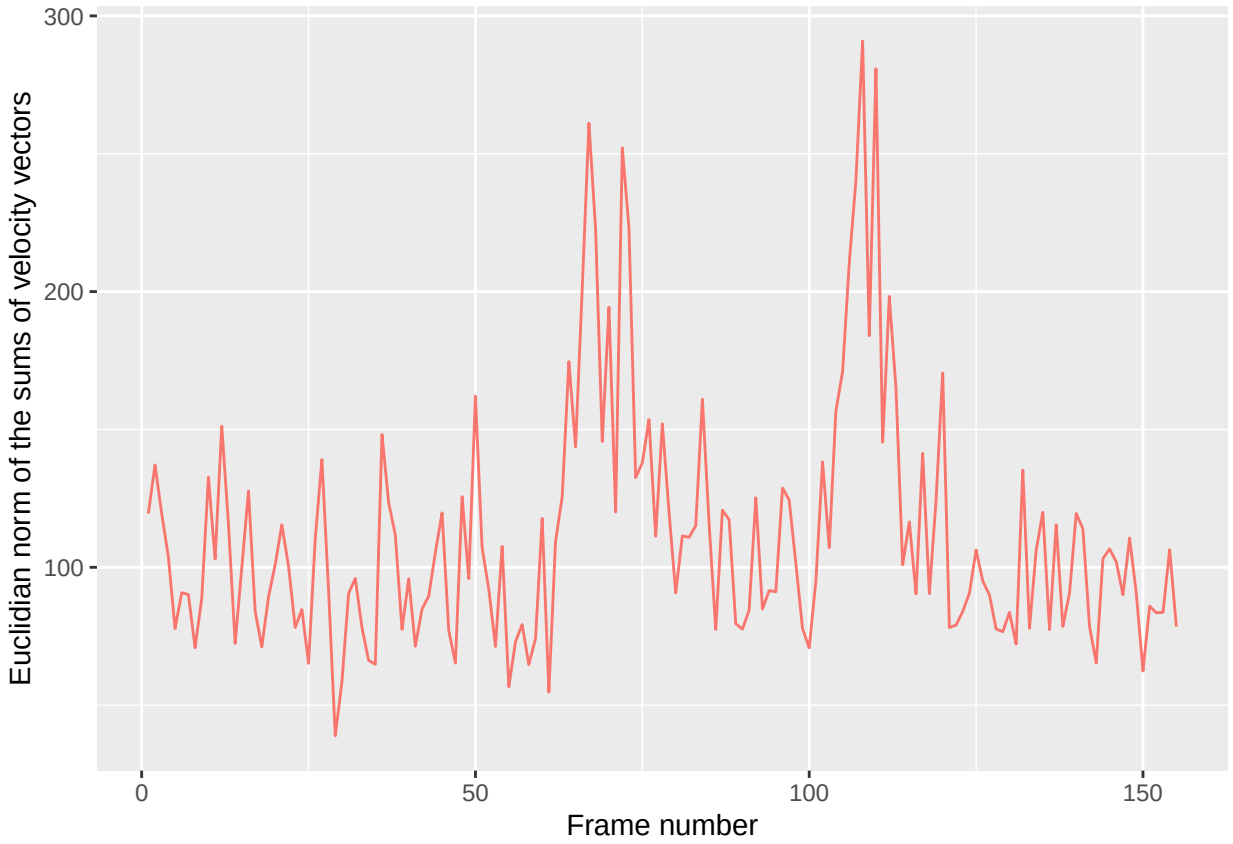
This generates a file named `psychologie_body25_cleaned_en_velocity.csv` in the folder `data_openposer` which contains the Euclidean norm of sums of velocity vectors for our input video.

Illustrating the result

Finally, to see the result of our efforts, let's have a look at a graphical illustration of the results. We can use the function `plot_timeseries()` which uses `ggplot2` to create a simple plot of the time series we have created as part of our analysis.

```
# First we have to load the data from our file:
motionPsychology <- read.csv("data_openposer/psychologie_body25_cleaned_en_velocity.csv",
                             sep="")

plot_timeseries(motionPsychology)
```



This simple plot provides an illustration of how we have quantified the total amount of bodily motion occurring in the input video clip. Now rewatch the video clip `psychologie.mp4` and you will see that the DGS sign for "psychology" is articulated with an initial large movement to the chest (i.e. sign onset), hand movement on the chest, followed by another large movement (i.e. sign offset). In the plot above, both sign onset and offset are clearly visible as peaks in the timeseries.

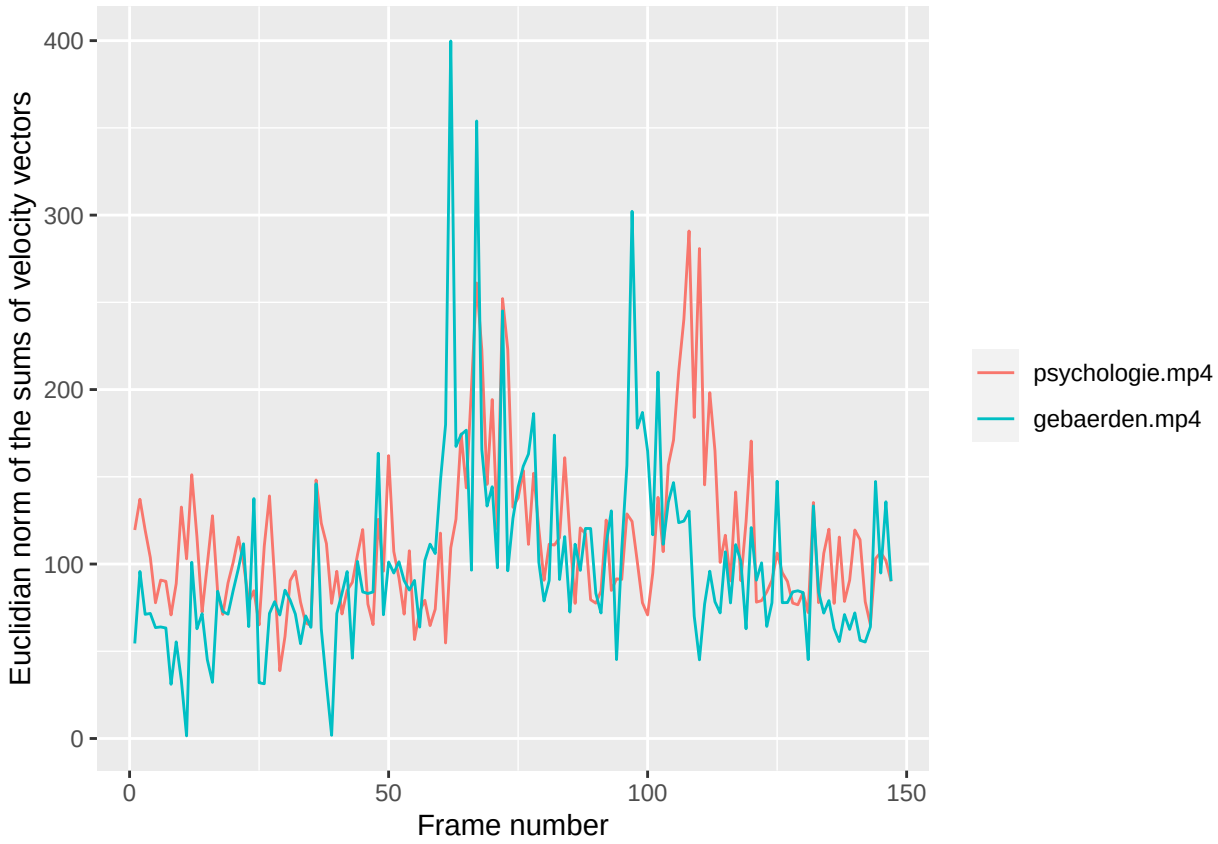
Comparision to another video clip

Because this demo already contains the motion-tracking data for all included video clips, we can use this data to compare our quantification of the motion in the DGS sign for "psychology" to that of the sign of "linguistics".

```
# Again, we load the data first
motionToSign <- read.csv("data_openposer/gebaerden_body25_cleaned_en_velocity.csv",
                         sep="")

# For the sake of simplicty we'll simply cut a few frames from the "psychology" clip
plotData <- cbind(motionPsychology[1:nrow(motionToSign),], motionToSign)
colnames(plotData) <- c("psychologie.mp4", "gebaerden.mp4")

plot_timeseries(plotData)
```



Of course, what we can do with this quantification of motion information is not limited to visual comparison of two videos. Instead, we can apply the method outlined in this demo to a larger number of files used, for example, in an experiment showing sign language video clips to participants and use this quantification of motion as a way of controlling our stimuli (i.e. making sure that the videos shown in different conditions of an experiment are actually comparable).

Creators & development

The OpenPoseR package was created at the Max Planck Institute for Human Cognitive & Brain Sciences by Patrick C. Trettenbrein in collaboration with Emiliano Zaccarella under the supervision of Angela D. Friederici.

If you have found a bug, please report it here. In case you have any questions, criticism, or suggestions that do not belong into the bugtracker, please e-mail Patrick at trettenbrein@cbs.mpg.de.

License