OpenPose Integration into iLex

Thomas Hanke

v 0	2021-01-03	First version of the text
v 1	2021-12-30	First release version: formatting and typo corrections

Introduction

OpenPose is very promising to enable accurate motion analysis on the DGS-Korpus data. For each frame of a video, OpenPose computes the number of persons visible, and for each person it determines pixel coordinates of a set of body joints. The 2018 edition of OpenPose adds coordinates on the face as well on the hands.
All data collected in the DGS-Korpus project, including studio reproductions, underwent OpenPose analysis (cf. AP04-2018-01), the data is available in the iLex database in a separate schema.

Inspection of OpenPose Data in iLex

In this report, we identify how OpenPose data are stored in iLex and how they can be inspected beyond simply querying the data tables.

Database Tables for OpenPose Data

in iLex, OpenPose data is stored in the table openpose.keypoints. openpose.keypoints has the following fields:

track as integer	identifies the movie_track that was the input to OpenPose
frame	the frame number of the record, i.e. a number between 0 and the duration of the track in frames
person	a number for each person recognized in the frame, starting with 0 (records with negative person numbers have been identified as ghosting and should be ignored)
pose_2d	text containing the array of coordinates for the pose recognizer, i.e. 25 triples of real numbers, being the x and y coordinate in screen coordinates plus a real between 0 and 1 as the confidence about the x and y coordinates reported. The numbers are enclosed in square brackets. Keypoints not found are reported as 0,0,0, or the array contains less numbers.
face_2d	text containing the array of coordinates for the pose recognizer, i.e. 70 triples of real numbers, being the x and y coordinate in screen coordinates plus a real between 0 and 1 as the confidence about the x and y coordinates reported. The numbers are

	enclosed in square brackets. Keypoints not found are reported as 0,0,0.
hand_right_2d	text containing the array of coordinates for the right hand recognizer, i.e. 21 triples of real numbers, being the x and y coordinate in screen coordinates plus a real between 0 and 1 as the confidence about the x and y coordinates reported. The numbers are enclosed in square brackets. Keypoints not found are reported as 0,0,0.
hand_left_2d	text containing the array of coordinates for the left hand recognizer, i.e. 21 triples of real numbers, being the x and y coordinate in screen coordinates plus a real between 0 and 1 as the confidence about the x and y coordinates reported. The numbers are enclosed in square brackets. Keypoints not found are reported as 0,0,0.
status	NULL in most cases, but contains the original person number for the records filtered out as ghosting.

The original json output from OpenPose contains additional fields, i.e. the 3d versions of the coordinates arrays. As 3d analysis is not yet used, these fields would always be empty, so they are simply left out.

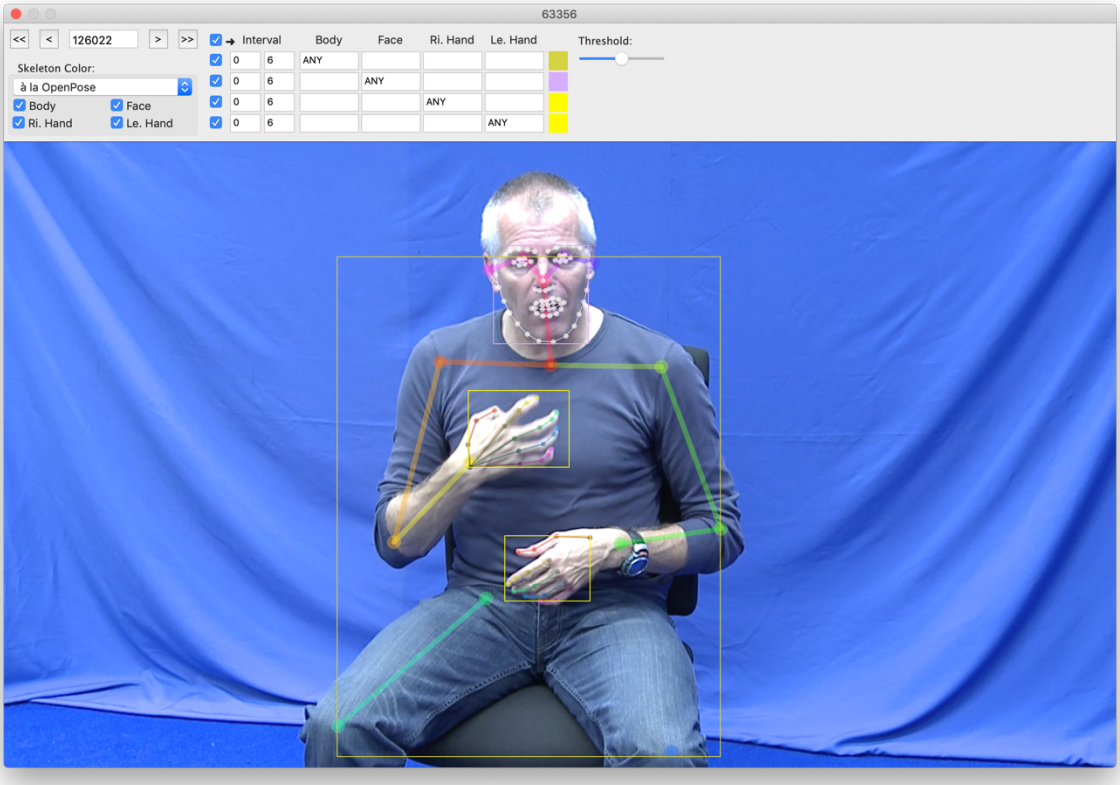
For performance reasons, a second table exists, openpose.tracks. It contains one record per movie_track for which there are keypoints available:

id	the id of the movie_track
width	the width of the movie_track input to OpenPose, relevant for interpreting the x and y coordinates of the keypoints as those are between 0 and width/height respectively
height	the height of the movie_track input to OpenPose, relevant for interpreting the x and y coordinates of the keypoints as those are between 0 and width/height respectively
target_count	the number of persons expected to be visible in the movie_track. In our case 1 for A and B perspectives, and 3 for the C perspective.
total_frames	the duration of the movie_track in frames, i.e. a positive integer
frames_with_data	the number of frames for this track where OpenPose has actually found something, i.e. a number between 0 and total_frames

Please be aware that the openpose.keypoints table is huge, with currently more than 720 million records. It is therefore important to work with the denormalized data in openpose.tracks whereever possible.

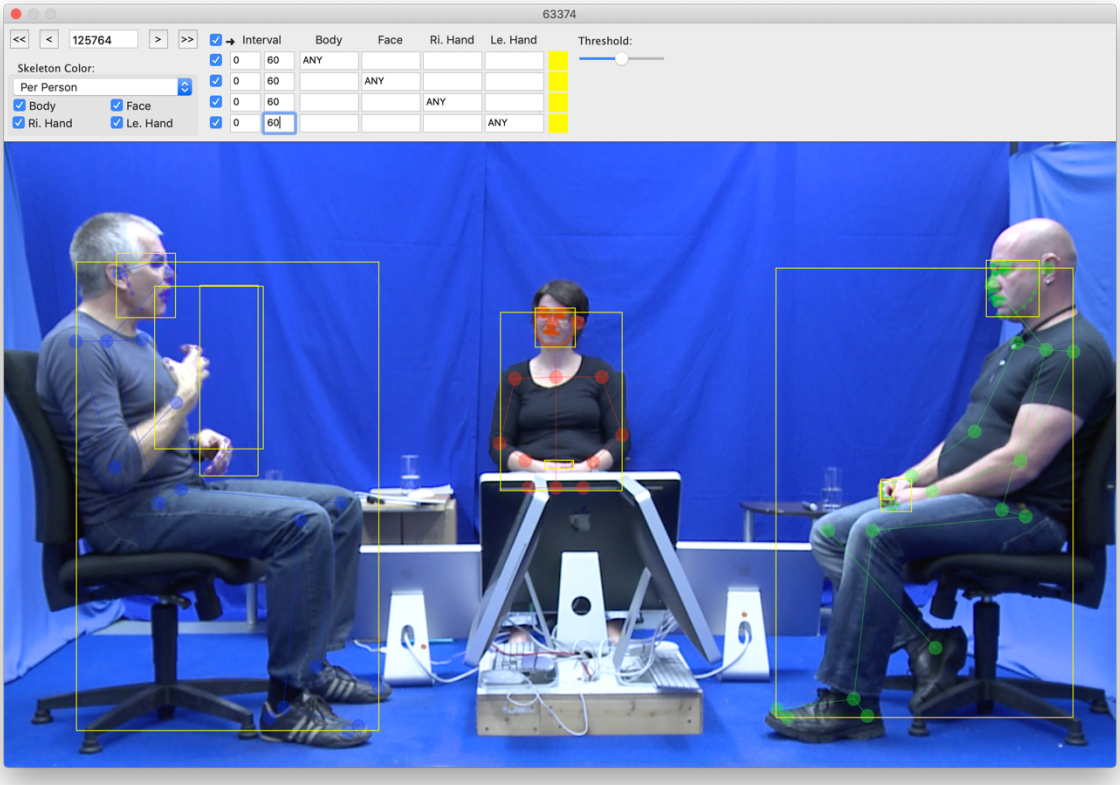
OpenPose Viewer

Apparently, it is very difficult to evaluate the quality of the OpenPose data by only querying the data tables. So iLex (from version 5.1b37 on) offers a special viewer:

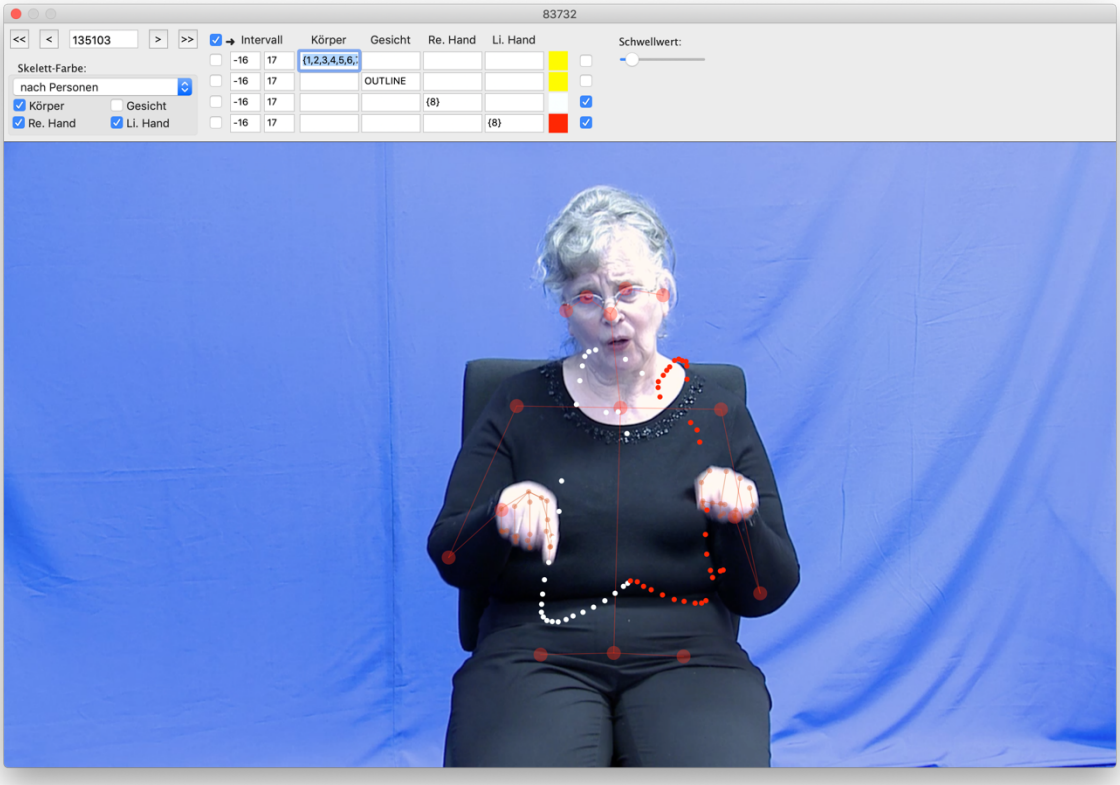


This viewer shows a specific frame of a movie_track with the OpenPose data overlaid to the video frame.
In the number field in the upper left you can type the number of the frame to be visualized, or use the step buttons (> steps forward by 1 frame, >> by 10 frames, option->> by 100 frames, < steps backwards by 1 frame, << by 10 frames, option-<< by 100 frames).

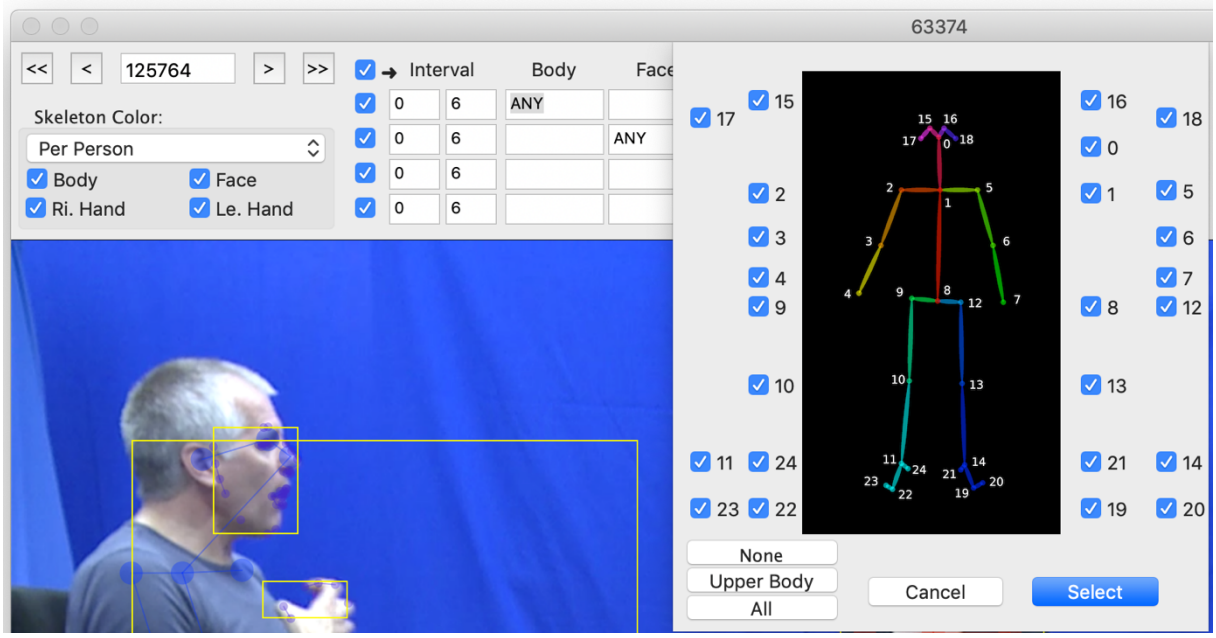
In the Skeleton Color section, you choose which coloring scheme to apply when showing the keypoints: You can choose to have one color per person or to use the OpenPose style, using different colors for the keypoints and the "bones" (the connecting lines). For the one-color-per-person scheme you can also include the ghosting data while the OpenPose style always suppresses the ghosts.
You can select which of the four recognizers is overlaid. Keypoints are only drawn if their confidence value exceeds a threshold that can be adjusted on the right end of the user interface elements.



The tabular input in the middle allows you to specify which bounding rectangles to display. For up to four bounding rects, you specify the color of the rect and which keypoints are to be included. By setting a look-behind and look-ahead around the current frame, you cannot only visualize the bounding rect at the current frame but also the area covered across a range of frames. Instead or in addition to the bounding rects, you can also display traces of selected coordinates across time by selecting checkboxes right to the tabular data. Note that is best used with only very few keypoints selected:



When choosing the keypoints to be displayed, it is best to use the data picker available from the context menu:



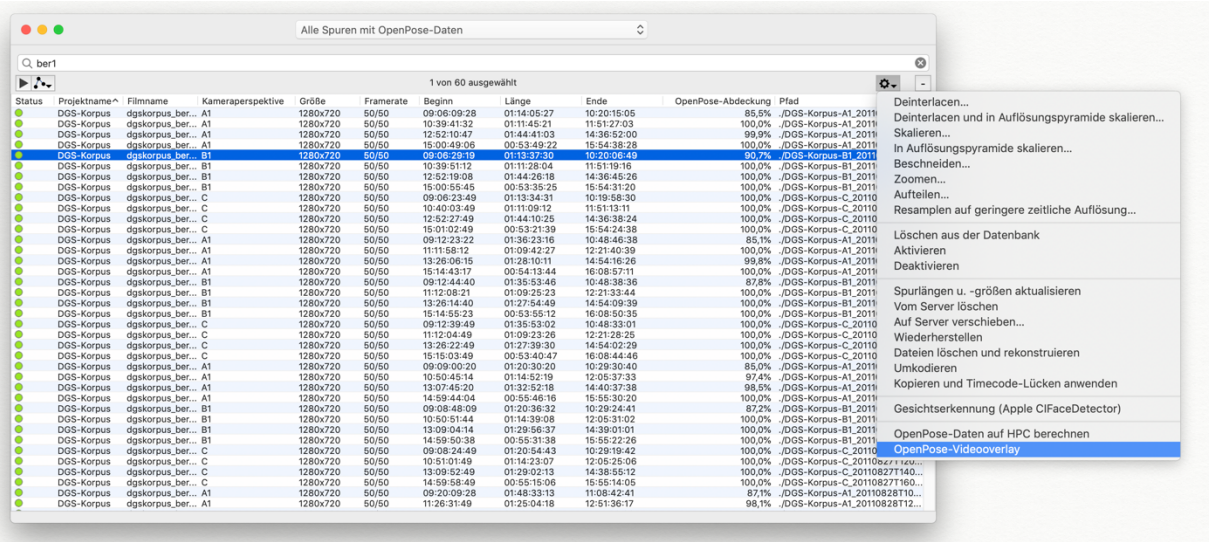
Accessing the OpenPose Viewer

In a Free Query window (“File” menu), you can enter a statement such as

select *'63374#125764.6' as openposevideoid*

Double-clicking the result will open the movie_track with id 63374 at frame 125764. The third element, here .6 , is optional and defines the length of the frame range of interest. If specified, that number will be used as look-ahead together with a look-behind of 0 the default. As long as the ➜ checkbox is selected, stepping back and forward will adapt the look-behind and look-ahead to identify the range of interest, but the values can of course be changed manually.

Similar queries can be stored as scripts, so the Hamburg database contains a script making the OpenPose viewer available from windows showing movie_tracks:

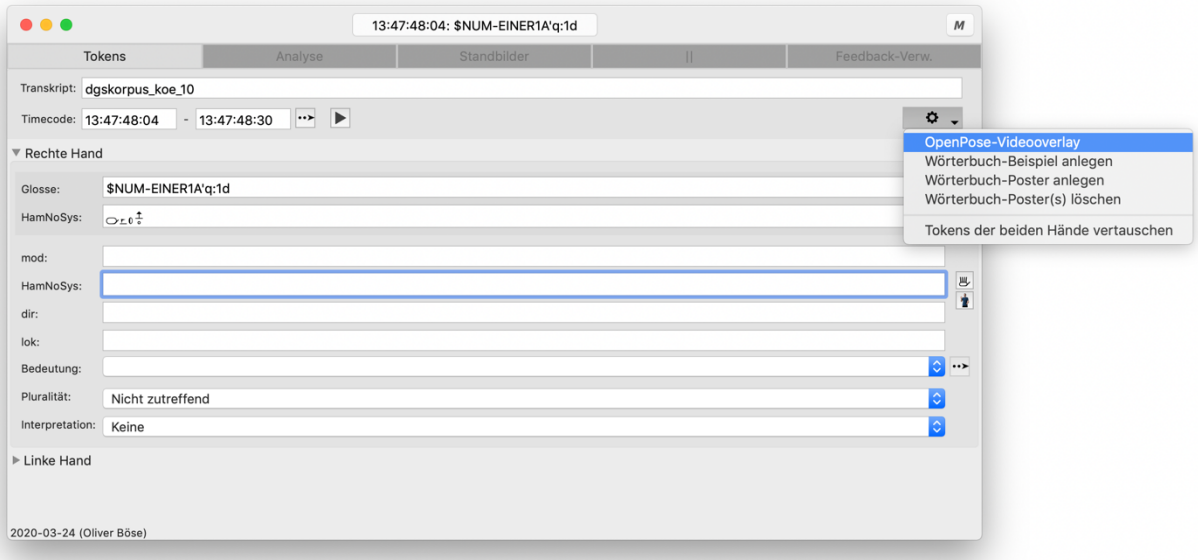


Here, the SQL code in the script definition looks as follows:

select *id as openposevideoid* ***from*** *openpose.tracks* ***where*** *id in* ***($1)***

The reference to openpose.tracks avoids error messages when movie_tracks without associated OpenPose data have been selected.

Finally, the function is available in the Gear (⚙) menus of tag detail windows:



Here, the SQL definition makes use of a helper function to determine which tracks contain the video data belonging to the tag and have OpenPose data associated, plus the frame offset for these tracks. As a result, the function typically opens two windows: One for the frontal showing the signer, a second for the total showing the signer (plus the other two people in the setting).

```
SELECT track||'#'||start_frame||'.'||(end_frame-start_frame) as
openposevideoid FROM tags JOIN LATERAL
openpose.tag2openposetrackandtimeframe(id) on tags.id  in ($1)
```

Ghosting

There are several reasons why there can be more openpose.keypoints records for one specific frame of one specific track than expected, meaning we have more person ids than openpose.tracks.target_count:

- There is actually another person visible, e.g. a technician crossing or the moderator leaving or returning to his/her seat.
- The other informant stands up or stretches high up getting into the camera view.
- OpenPose for some reason does not consider all parts of the signer to belong to the same individual.

The first case will mainly occur in breaks, so outside subtasks, and therefore is less of a concern. In the first two cases we want to throw out the additional person, but have to make sure that we do not throw out the target informant. In the third case, we need to check whether we can merge the data into one record as we know the bits belong together.

Person Identification

After removal of ghosting, there should be no need to do any further identification processing for our camera perspectives A and B as there should be exactly one person left. For the C perspective, however, there are three persons, and we have to make sure that we know which person id in the OpenPose data belongs to which informant.
OpenPose works on each frame without information on the neighbourhood and optionally does some post-processing that the person with a certain id in one frame gets the same id in the next frame. As OpenPose has to cope with a rather unrestricted matching problem, it

seems a better idea to use the information we have available for our setting. That is, for the C camera perspective we have three persons from left to right: Informant B (right side view), moderator (frontal view), informant A (left side view). It seems easier to work from the left-to-right arrangement than to determine which direction each person identified by OpenPose is facing.