Human Pose Estimation for Biomechanics
-
Workflow and instructions

Azilis Even
Original video instructions: Qiantailang Yuan

Contents

1 Set-up

1.1 Get access to Alvis

- Create a SUPR/NAISS account
- Join required compute and storage projects
- Send a ”account request” for the cluster you need to use (see Getting Access instructions)
- After at least 1 working day, set your password

1.2 Connecting to Alvis with KTH VPN

If you are not on the KTH network or another SUNET network, you need to use a VPN to be able to connect to Alvis. For KTH, you can follow the following instructions.

1.3 Different ways to connect to Alvis

1.3.1 Alvis OnDemand

The OnDemand portal allows you to browse your files, check your disk and file quotas, check on your active jobs and also launch interactive apps on compute nodes like Jupyter notebooks or VSCode.

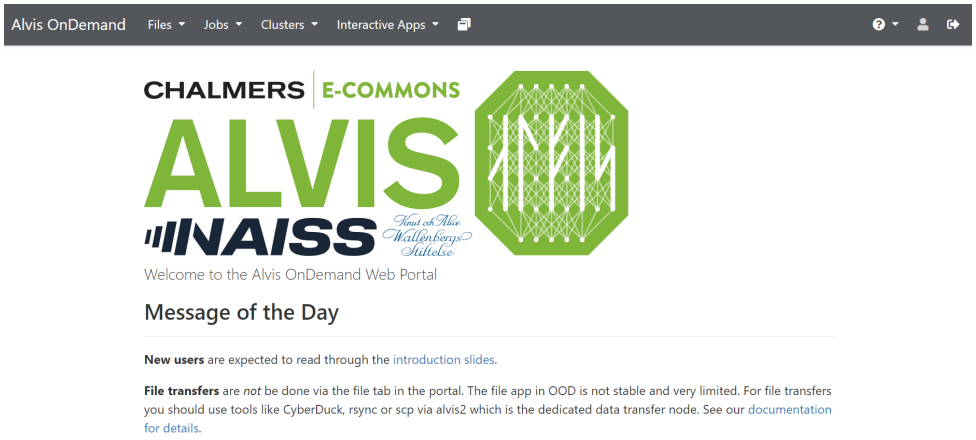

Figure 1: Alvis OnDemand portal

1.3.2 ThinLinc

For uses where a graphical interface is needed, Alvis can be accessed via the ThinLinc client. To connect to Alvis, you can choose to simply use your Alvis CID and password, or you can use connect with SSH (see section 1.4).

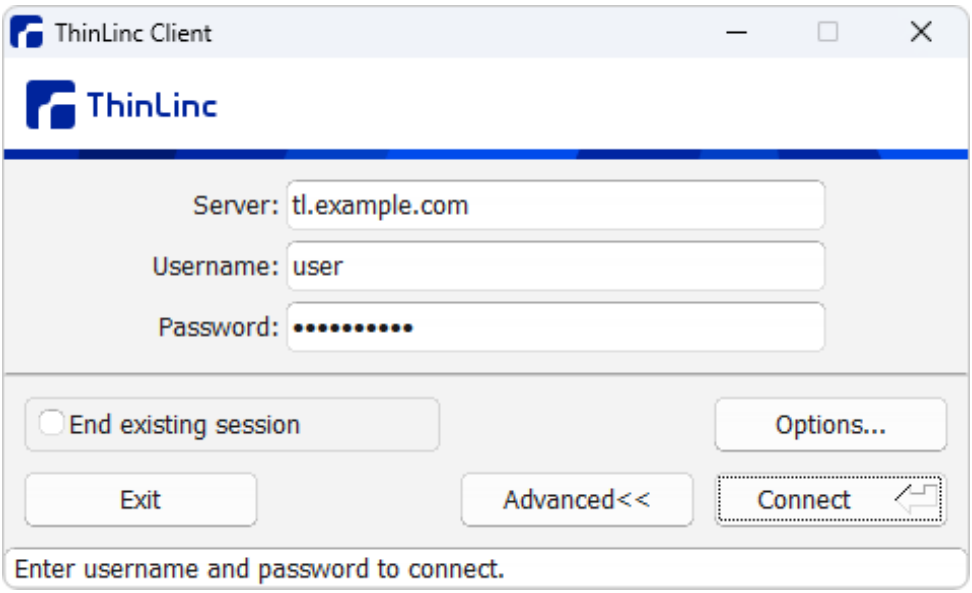

Figure 2: ThinLinc
(image from https://www.cendio.com/resources/docs/tag/client_usage.html)

1.3.3 VSCode

Another option is to use VSCode and connect to the host ”your-cid@alvis1.c3se.chalmers.se” (or alvis2, see Alvis documentation).

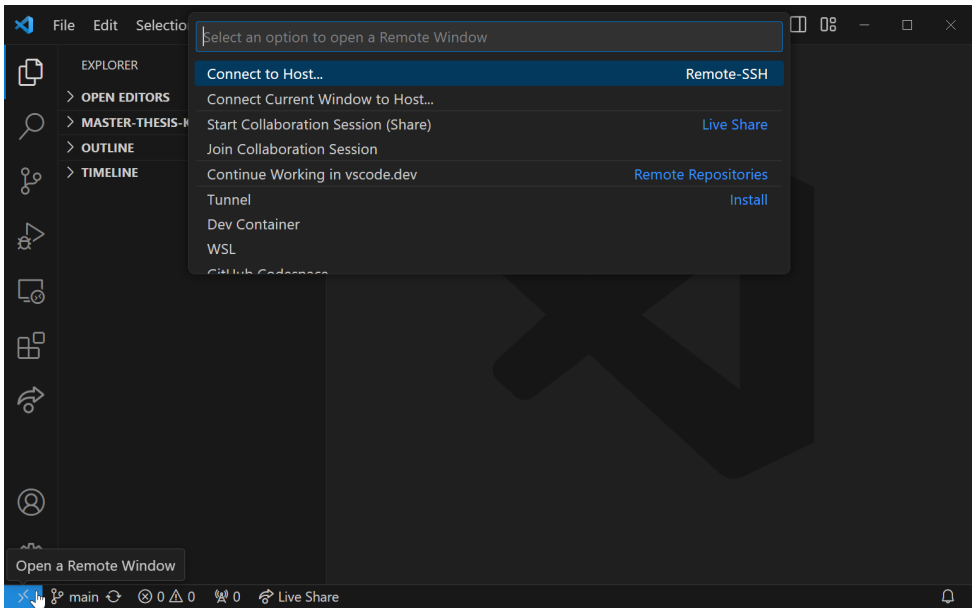

Figure 3: Connecting to Alvis with VSCode

1.4 Connect with SSH

Using PuTTY, generate a new SSH key-pair. From the command line, *cd* to your PuTTY folder (something along the lines of C:\Program Files\PuTTY) and use:

```
ssh-keygen -t rsa
```

Then connect to the OnDemand portal and copy the public key (.pub) into your home folder.

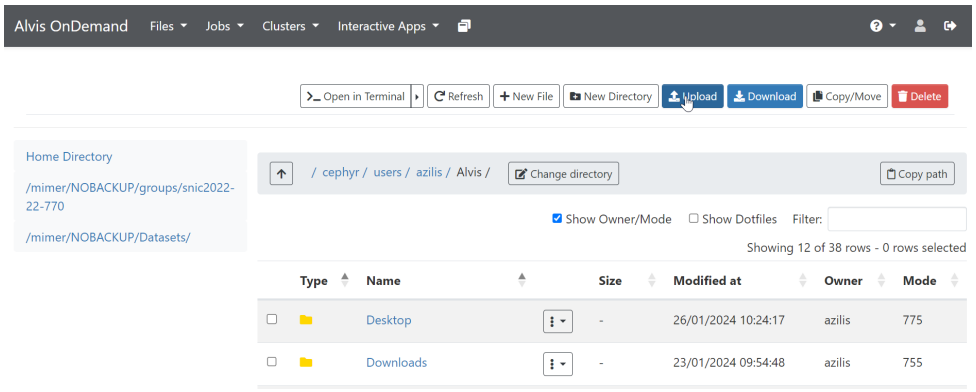

Figure 4: OnDemand Portal - Upload Public key

Open a shell and move your public key to the list of authorised keys for Alvis using:

```
mv your-public-key.pub .ssh/authorized_keys
```

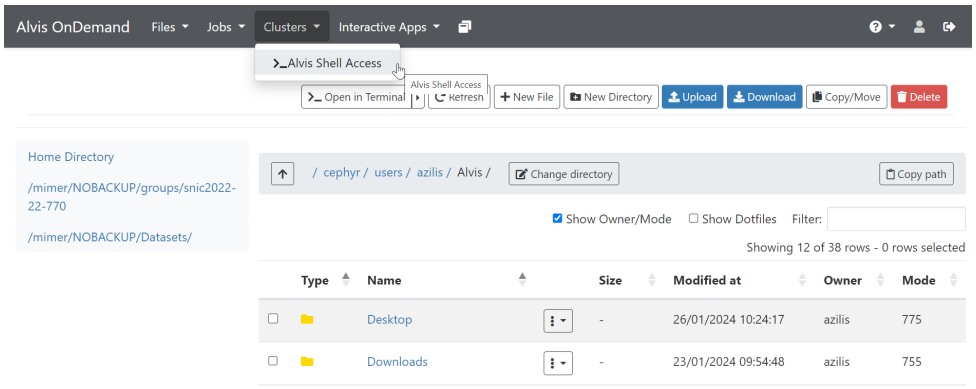

Figure 5: OnDemand Portal - Alvis Shell Access

Now when you open ThinLinc you can choose to use ssh identification instead of using your password: select Options > Security > Public Key, then choose your private key file in the login panel.

2 OpenPose

2.1 Description

The goal of this step is to get the position of the 2D joints from the chosen frames / monocular images. The output is a JSON file of the position of the joints relative to the initial image, as well as a rendered image with the skeleton over the original image.
For more information on OpenPose, refer to its project page.

2.2 Prerequisites

To run OpenPose, you need to have the image files containing the people of interest (whose pose you want to estimate).

You also need access to OpenPose and all its dependencies. The easiest way to achieve this is through the easymocap.img container image that is available on MIMER for the "Human pose estimation using AI for biomechanics application (SNIC 2022/22-770)" project. On Alvis it is accessible with the following path:

```
/mimer/NOBACKUP/groups/snic2022-22-770/singularity_image/easymocap/easymocap.img
```

You can either directly access it from this path or copy it in your home directory or a personal workspace folder (however, note that it takes a non-negligible amount of storage).

2.3 Instructions to run on Alvis

Open a terminal and start a bash job using the command below. You can change the duration of the job as needed and update the project ID to the one you are a part of (the information is on your NAISS SUPR project page), and you can choose the type of GPUs you want to use and the number per node (see Alvis documentation for more information).

```
srun -A NAISS2023-22-708 -p alvis --gpus-per-node=A40:1 -t 00:10:00 --pty bash
```

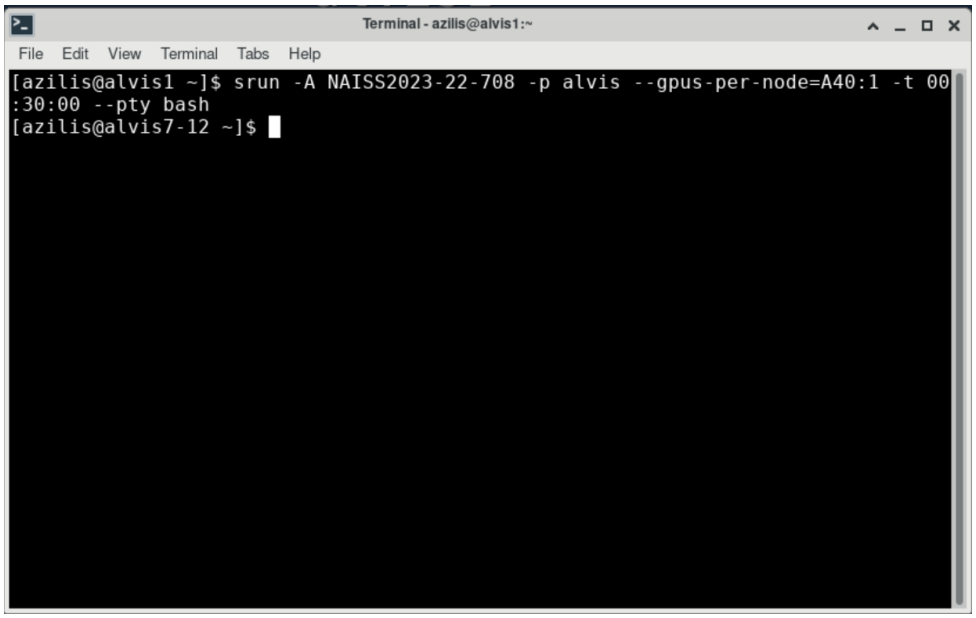

Figure 6: Alvis - Starting a bash job in the terminal

Once you have been attributed resources for this job, open a new terminal and ssh to the node you have booked (alvis0-00 is only an example) with the following command. The -Y flag indicates that displays and graphical interfaces will be forwarded to your local system (alvis1 on ThinLinc), even if they are run on a different booked node.

```
ssh -Y alvis0-00
```

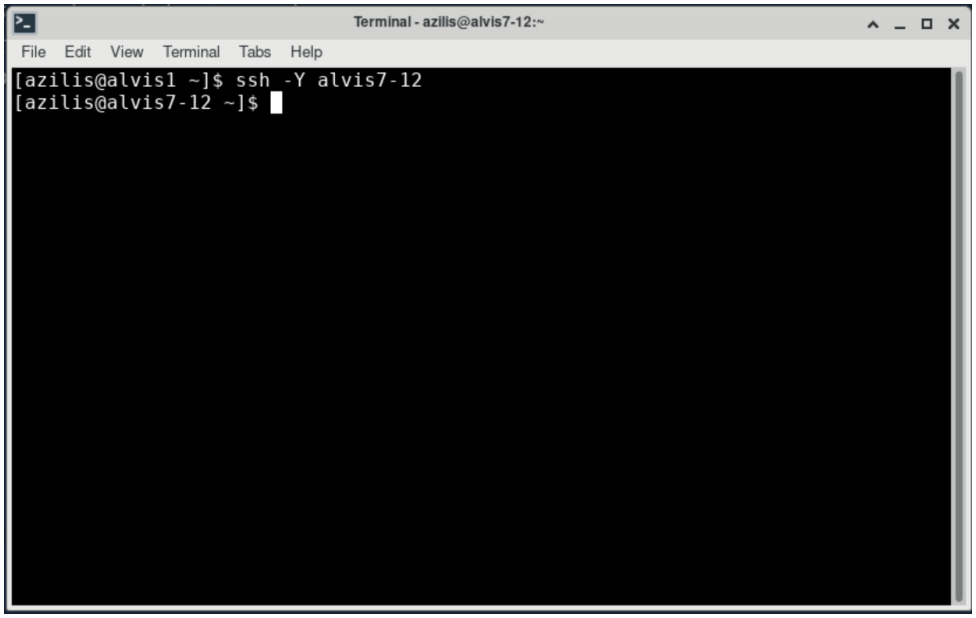

Figure 7: ALvis - ssh to booked node with display forwarding

Then, to run a shell within the easymocap container, use the following command. The "–writable-tmpfs" flag allows you to make changes to the container image, however, they will not remain once you exit it. If your easymocap image isn't in your home directory, update the path as needed.

```
apptainer shell --fakeroot --writable-tmpfs easymocap.img
```

Then, move into the OpenPose folder.

```
cd /workspace/openpose/
```

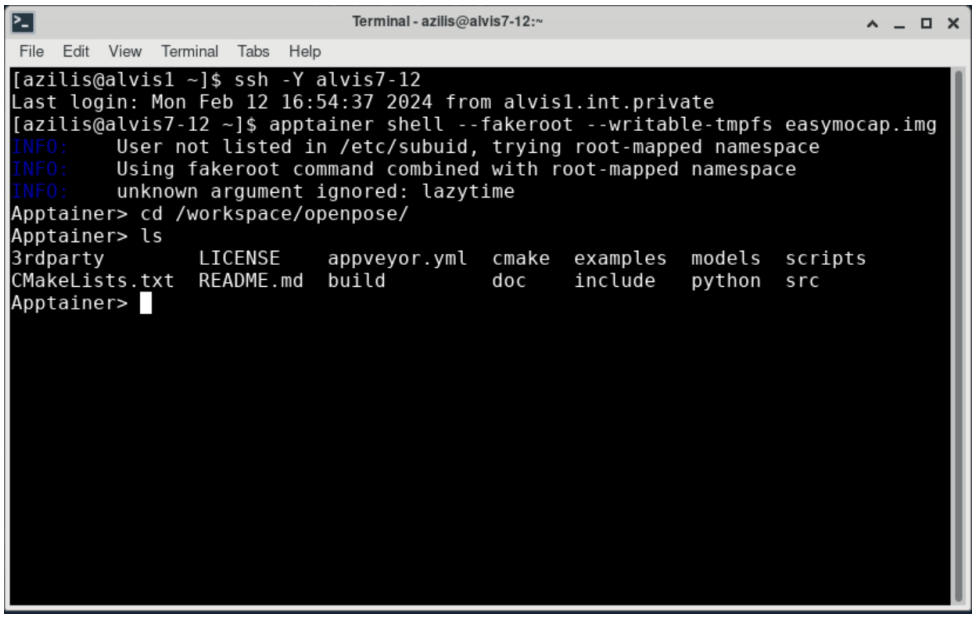

Figure 8: Using the OpenPose container

To run OpenPose, use the following command updated with the required folders. If they don't already exit, the output folders will be created. The flags "–hand" and "–face" indicate that the keypoints for the face and hands should also be found if possible.

```
./build/examples/openpose/openpose.bin \
    --hand --face \
    --image_dir PATH_INPUT_IMAGES  \
    --write_json PATH_OUTPUT_KEYPOINTS \
    --write_images PATH_OUTPUT_RENDERED_IMAGES \
    --display 0
```

When you are done, you can exit the container and the Alvis node you had booked with the "exit" bash command.

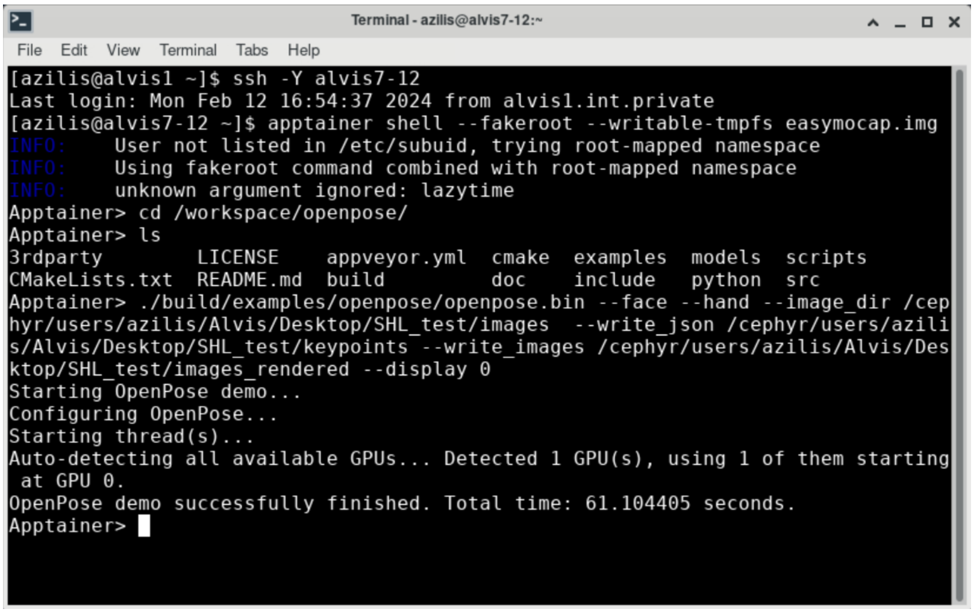

Figure 9: Running OpenPose

2.4 Outputs

The resulting folders and files should look as follow.

```
parent-directory
├── images
│   └── image.png … Source image
├── images_rendered
│   └── image_rendered.png … Rendered image with 2D OpenPose
│                            skeleton overlaid on top of it
└── keypoints
    └── image_keypoints.json … JSON keypoints results
```

An example of a rendered image with the joints is available in Figure 11 and the corresponding keypoints are displayed in Figure 10. For more information on keypoints and the structure of the JSON files, consult the OpenPose documentation.

{"version":1.3,"people":[{"person_id":[-1],"pose_keypoints_2d":[886.576,277.624,0.901608,945.731,386.748,0.772936,773.1,377.789,0.671817,604.821,495.838,0.717111,704.709,709.293,0.674922,1127.42,391.349,0.644726,1141.17,672.956,0.669166,1059.34,854.859,0.659328,1073,754.803,0.530356,959.239,777.631,0.524153,1104.78,1104.7,0.658024,1204.68,1363.91,0.746542,1195.7,741.152,0.518681,1404.78,1032.14,0.682203,1609.11,1300.21,0.726744,872.979,241.324,0.85006,922.986,259.464,0.883757,0,0,0,1000.49,268.588,0.875226,1709.11,1368.27,0.468114,1727.21,1341.01,0.551671,1595.53,1327.57,0.456889,1313.79,1409.32,0.69861,1263.81,1432.05,0.690248,1186.5,1400.01,0.493102],"face_keypoints_2d":[],"hand_left_keypoints_2d":[],"hand_right_keypoints_2d":[],"pose_keypoints_3d":[],"face_keypoints_3d":[],"hand_left_keypoints_3d":[],"hand_right_keypoints_3d":[]},{"person_id":[-1],"pose_keypoints_2d":[2372.52,127.738,0.534348,2386.2,236.788,0.629416,2290.77,232.371,0.515031,2236.46,377.594,0.240101,0,0,0,2459.11,241.473,0.294896,2459.1,459.549,0.0734553,0,0,0,2381.68,495.814,0.139698,2308.95,491.273,0.13882,0,0,0,0,0,0,2459.02,500.341,0.151035,0,0,0,0,0,0,2349.83,100.567,0.511413,2390.77,100.471,0.510353,2331.62,118.529,0.214753,2458.99,118.54,0.423932,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"face_keypoints_2d":[],"hand_left_keypoints_2d":[],"hand_right_keypoints_2d":[],"pose_keypoints_3d":[],"face_keypoints_3d":[],"hand_left_keypoints_3d":[],"hand_right_keypoints_3d":[]},{"person_id":[-1],"pose_keypoints_2d":[0,0,0,2227.26,11.7828,0.136634,2127.46,11.7889,0.0964227,2113.58,91.379,0.274912,2163.56,155.071,0.234484,2304.26,11.7623,0.180973,2304.5,86.8183,0.200648,2249.89,150.367,0.160908,2204.59,159.494,0.218546,2158.99,155.014,0.227643,0,0,0,0,0,0,2267.9,159.601,0.217614,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"face_keypoints_2d":[],"hand_left_keypoints_2d":[],"hand_right_keypoints_2d":[],"pose_keypoints_3d":[],"face_keypoints_3d":[],"hand_left_keypoints_3d":[],"hand_right_keypoints_3d":[]}]}

Figure 10: Keypoints OpenPose output example, without the advanced detection of the face and the hands

0 Nose	13 Left knee
1 Mid shoulder	14 Left ankle
2 Right shoulder	15 Right eye
3 Right elbow	16 Left eye
4 Right wrist	17 Right ear
5 Left shoulder	18 Left ear
6 Left elbow	19 Left big toe
7 Left wrist	20 Left small toe
8 Mid hips	21 Left heel
9 Right hip	22 Right big toe
10 Right knee	23 Right small toe
11 Right ankle	24 Right heel
12 Left hip	

Figure 11: Rendered image, without the advanced detection of the face and the hands

2.5 Possible problems

If you can't access the easymocap.img container image, it's probably a permission issue. Check with the project owner to get access to the desired folders and files.

3 SMPLify-X

3.1 Description

This part aims to obtain the 3D pose estimate from the impact frames selected previously and the 2D joints from OpenPose. The output is a SMPLX surface mesh of the body.
For more information on SMPLify-X, refer to its project page.

3.2 Prerequisites

Your inputs should be organised in 2 subfolders sharing the same parent folder: an "images" folder containing the image and a "keypoints" folder with the results of the OpenPose step (2D joints JSON files). If there are several people in one image, you can separate keypoints in one JSON file per person, but then you also need to duplicate the source image to go with it.

```
parent-directory
├── images
│   ├── image.png
│   └── …
└── keypoints
    ├── image_keypoints.json
    └── …
```

You also need access to SMPLify-X and all its dependencies. The easiest way to achieve this is through the smplifyx.sif container image that is available on MIMER for the "Human pose estimation using AI for biomechanics application (SNIC 2022/22-770)" project. On Alvis it is accessible with the following path:

```
/mimer/NOBACKUP/groups/snic2022-22-770/singularity_image/smplifyx/smplify_cuda.sif
```

You can either directly access it from this path or copy it in your home directory or a personal workspace folder (however, note that it takes a non-negligible amount of storage).

3.3 Instructions to run on Alvis

Open a terminal and start a bash job (change the duration of the job as needed). Change the project ID to the one you are a part of (the information is on your NAISS SUPR project page), and you can choose the type of GPUs you want to use and the number per node (see Alvis documentation for more information).

```
srun -A NAISS2023-22-708 -p alvis --gpus-per-node=A40:1 -t 00:10:00 --pty bash
```

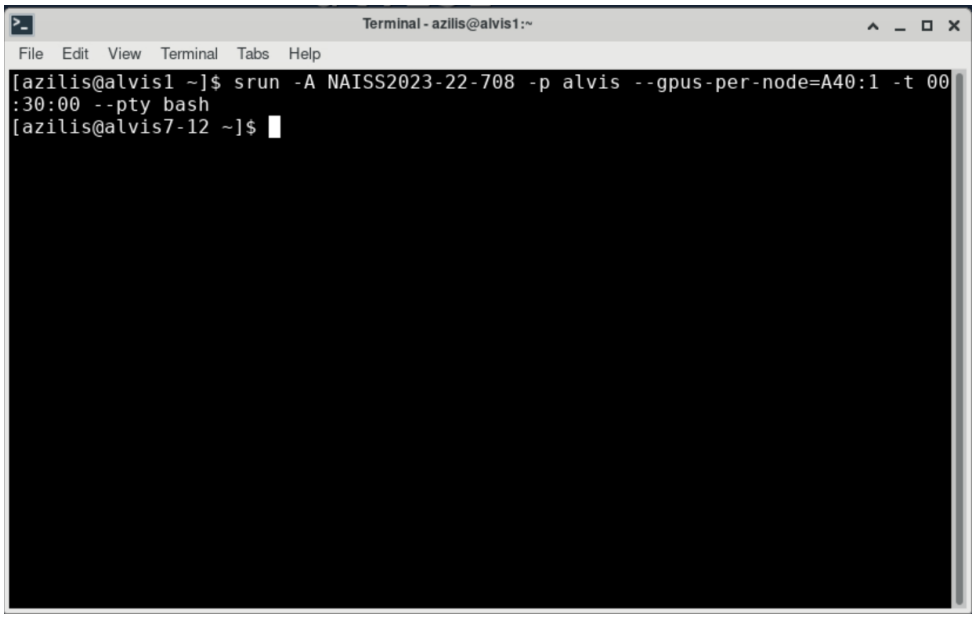

Figure 12: Alvis - Starting a bash job in the terminal

Once you have been attributed resources for this job, open a new terminal and ssh to the node you have booked (alvis0-00 is only an example) with the following command. The -Y flag indicates that displays and graphical interfaces will be forwarded to your local system (alvis1 on ThinLinc), even if they are run on a different booked node. This is essential for this step to work (unless you later disable the visualize option of SMPLify-X).

```
ssh -Y alvis0-00
```

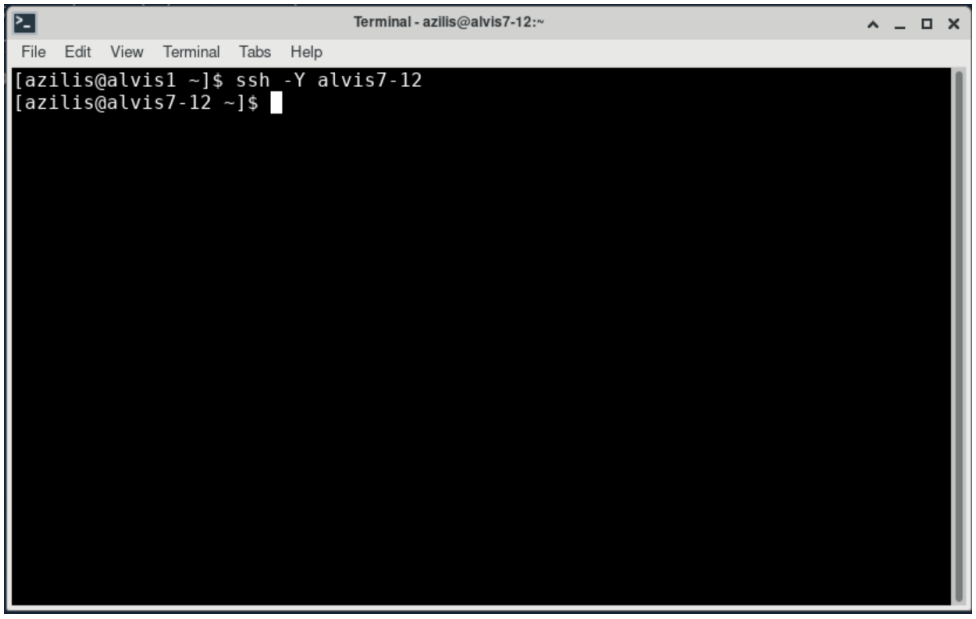

Figure 13: ALvis - ssh to booked node with display forwarding

Then, to run a shell within the smplifyx container, use the following command. The ”–writable-tmpfs” flag allows you to make changes to the container image, however they won’t remain once

you exit it. If your smplifyx image isn't in your home directory, update the path as needed.

```
apptainer shell --fakeroot --writable-tmpfs smplify_cuda.sif
```

Then, move into the smplifyx folder.

```
cd /workspace/smplify-x/
```

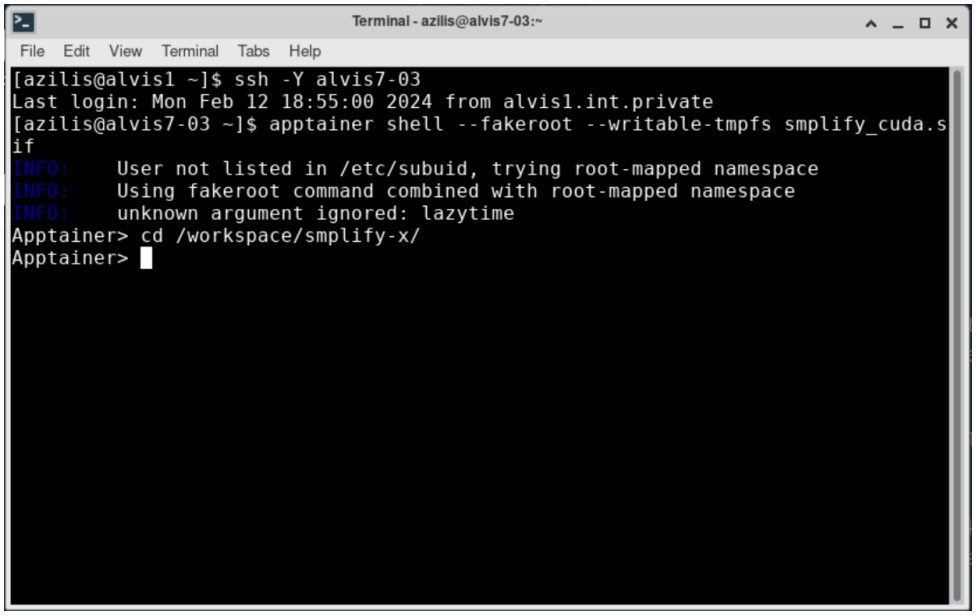

Figure 14: Using the SMPLify-x container

Then, to run SMPLify-X, use the following command. You can change the gender of the model as required between male, female and neutral.

```
python3.8 smplifyx/main.py \
    --config cfg_files/fit_smplx.yaml \
    --data_folder PATH_INPUT_DIR \
    --output_folder PATH_OUTPUT_DIR \
    --visualize="True" \
    --model_folder models/ \
    --vposer_ckpt ../vposer_v1_0/ \
    --part_segm_fn smplx_parts_segm.pkl \
    --use_vposer True \
    --gender male
```

NB: The runtime is higher each time you open a shell inside the container and run SMPLify-X for the first time. If you run it again on different files it will be faster.

A Scene Viewer should open where you can visualize what is going on with your model as it gets fitted to the image and the keypoints (see Figure 15).

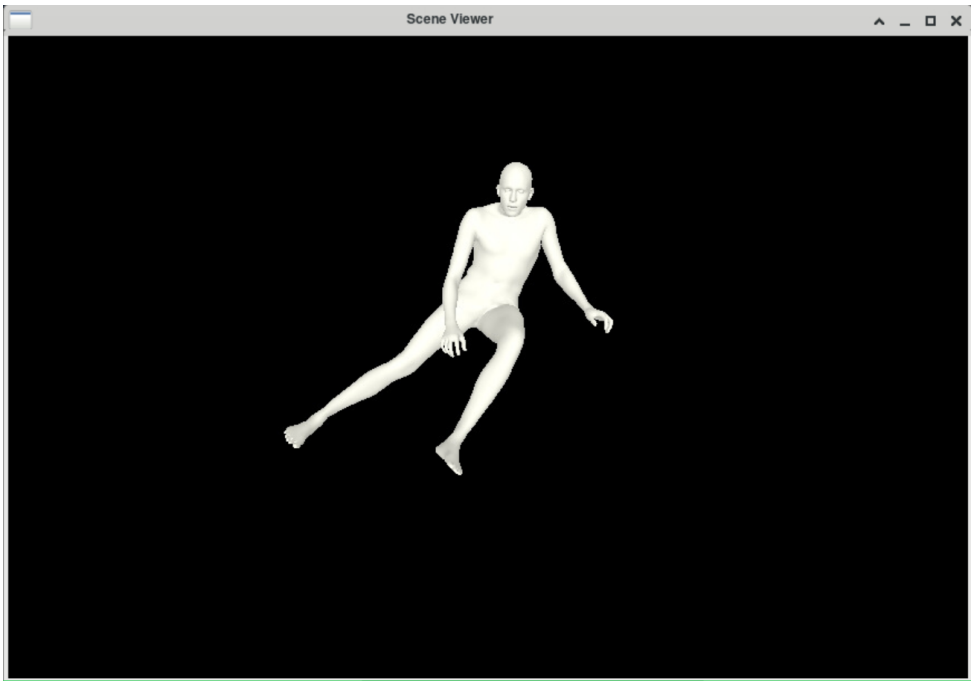

Figure 15: SceneViewer example while SMPLify-X is running

You can also follow the different steps directly in the terminal (see Figure 16).

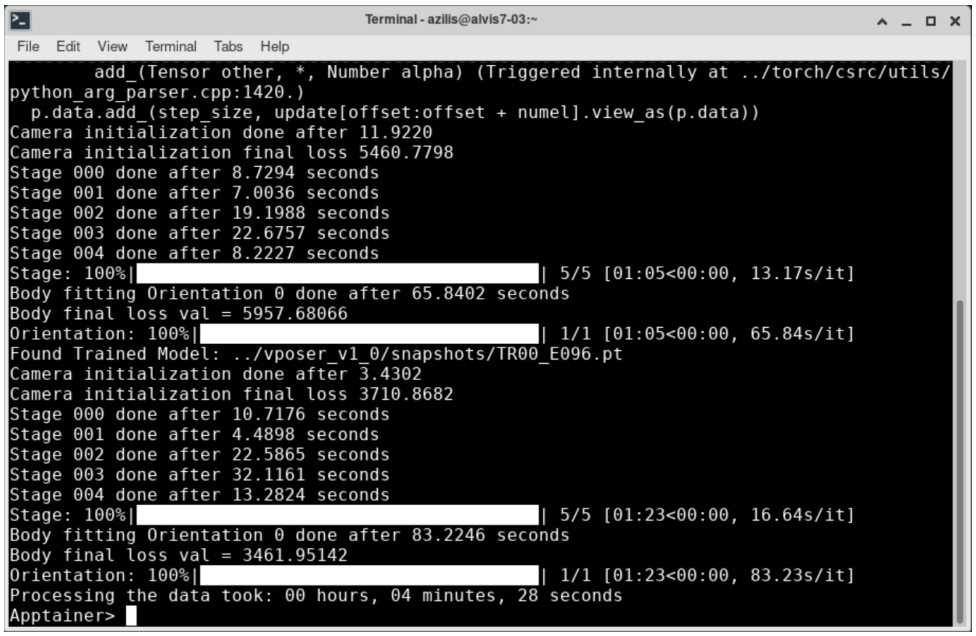

Figure 16: Example from running SMPLify-X

When you are done, you can exit the container and the Alvis node you had booked with the ”exit” bash command.

3.4 Outputs

The resulting folders and files should look as follow (if you didn’t separate the keypoints in the case of images with multiple people in them).

```
parent-directory
├── images
│   ├── 000
│   │   └── output.png … Rendered image with 3D pose estimation result
│   │                    overlaid over the original image
│   ├── 001
│   └── …
├── keypoints
├── meshes
│   └── 000.obj … Result mesh (can be opened with MeshLab)
└── results
    └── 000.pkl
```

An example of a rendered image with the 3D pose overlaid on top of the original image is available in Figure 17.

3.5 Possible problems

3.5.1 Container access issue

If you can’t access the smplifyx.sif container image, it’s probably a permission issue. Check with the project owner to get access to the desired folders and files.

Figure 17: Rendered image with the 3D pose estimation from SMPLify-X on top of the original, with expressive gands and face

3.5.2 File not found error

```
FileNotFoundError: [Errno 2] No such file or directory
```

If you get an error message mentioning a non-existent file, check that each image has a corresponding keypoint file in the right format: ”image.jpg” should correspond to ”image_keypoints.json” for example. This should be fine if you directly use the output files from OpenPose, but errors can happen if you modify the files afterwards.

3.5.3 Display error

```
NoSuchDisplayException
```

If you get this exception and stay stuck at the final Orientation stage with no SceneViewer, check if you forwarded the graphical interfaces from the booked node to your local system (alvis1 or alvis2 with ThinLinc). See the beginning of the instructions.

3.5.4 RunTime dimensions errors

```
RuntimeError: The size of tensor a (118) must match the size of tensor b
(25) at non-singleton dimension 1
```

Check that you have activated the ”–hand” and ”–face” options in the OpenPose step, or change the configuration file to indicate that the hands and face should not be taken into account.

4 Conversion of SMPLX meshes to SMPL meshes

4.1 Description

This part describes how to convert SMPLX meshes to SMPL meshes.

4.2 Prerequisites

You need access to SMPLX and all its dependencies. The easiest way to achieve this is through the smplx.sif container image that is available on MIMER for the ”Human pose estimation using AI for biomechanics application (SNIC 2022/22-770)” project. On Alvis it is accessible with the following path:

```
/mimer/NOBACKUP/groups/snic2022-22-770/singularity_image/smplx/smplx.sif
```

You can either directly access it from this path or copy it in your home directory or a personal workspace folder (however, note that it takes a non-negligible amount of storage).

4.3 Instructions to run on Alvis

Open a terminal and start a bash job (change the duration of the job as needed). Change the project ID to the one you are a part of (the information is on your NAISS SUPR project page), and you can choose the type of GPUs you want to use and the number per node (see Alvis documentation for more information).

```
srun -A NAISS2023-22-708 -p alvis --gpus-per-node=A40:1 -t 00:30:00 --pty bash
```

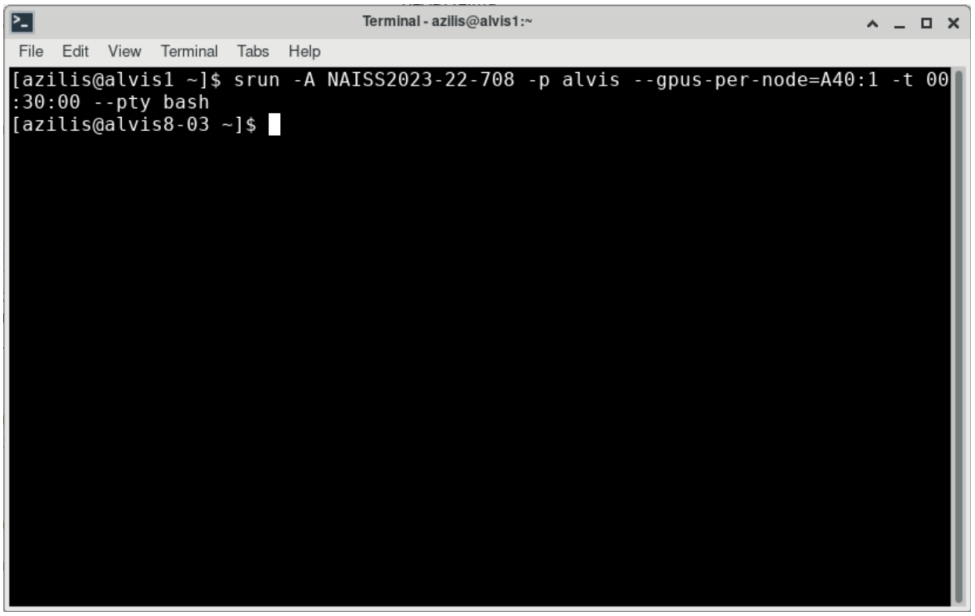

Figure 18: Alvis - Starting a bash job in the terminal

Once you have been attributed resources for this job, open a new terminal and ssh to the node you have booked (alvis0-00 is only an example). Newt, to run a shell within the smplx container, use the following command. The ”–writable-tmpfs” flag allows you to make changes to the container image, however they won't remain once you exit it. If your smplx image isn't in your home directory, update the path as needed. Then, move into the smplx folder.

```
ssh -Y alvis0-00
apptainer shell --fakeroot --writable-tmpfs smplx.sif
cd /workspace/smplx/
```

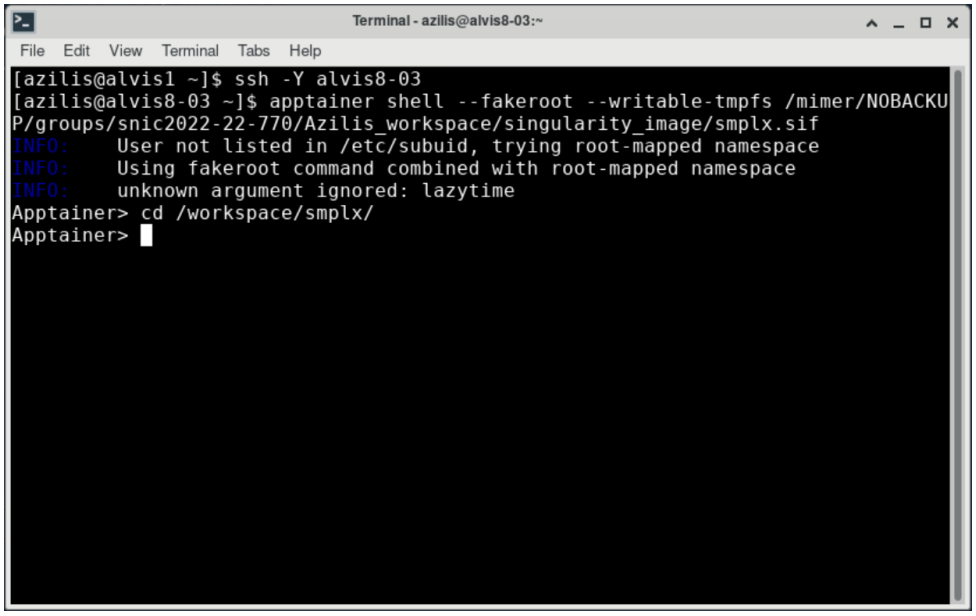

Figure 19

Open the configuration file for the conversion tool and use *vim* to comment out or remove line 5 according to Figure 20 (see Appendix 5.5.2 for more information on vim).
NB: This step needs to be done each time since the modifications to the container are discarded upon exiting the shell.

```
vim config_files/smplx2smpl.yaml
```

Another non-interactive option is to use *sed* as follows:

```
sed -i '5d' config_files/smplx2smpl.yaml
```

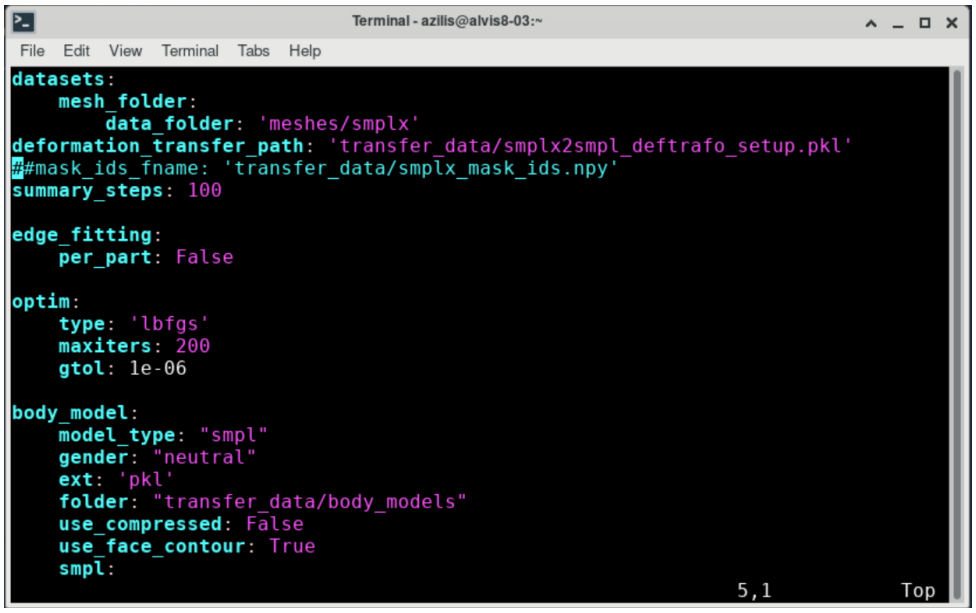

Figure 20: Line to comment out in the conversion configuration file

All the meshes that you wish to convert should be copied to a specific folder, using the following command. You can also remove all the example meshes from the folder to keep only the ones you want to use (*.ply refers to all files with the .ply extension).

```
cp PATH_TO_YOUR_MESH meshes/smplx/
rm meshes/smplx/*.ply
```

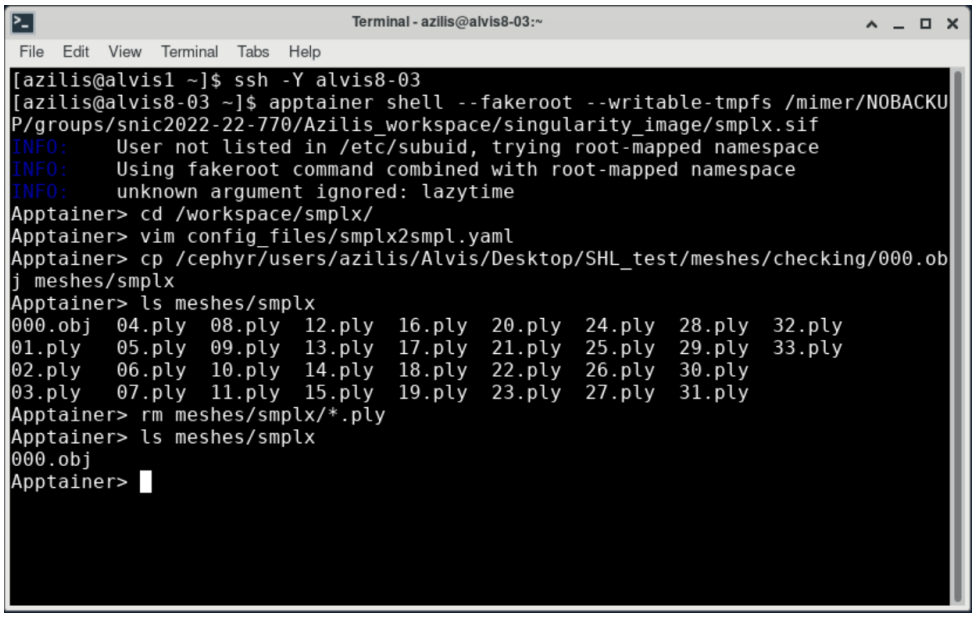

Figure 21: Preparation of the input files

Then, to run the conversion from SMPLX model to SMPL model, use the following command.

```
python3 -m transfer_model --exp-cfg config_files/smplx2smpl.yaml
```

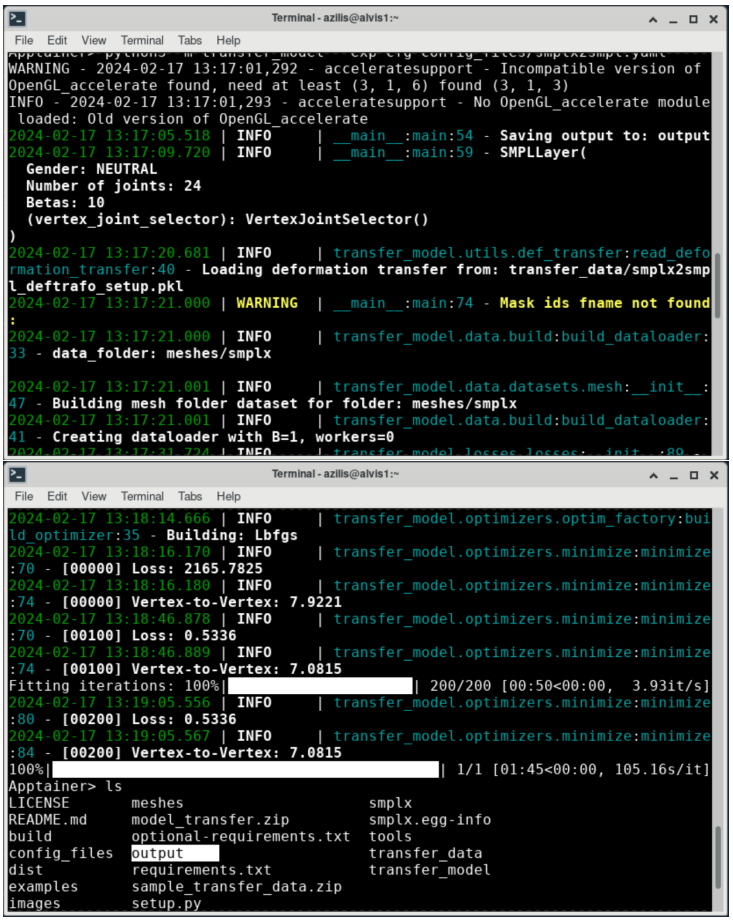

Figure 22: Conversion tool running example

4.4 Outputs

The output is SMPL versions of the input SMPLX meshes that can be used with OSSO.
Don't forget to copy the content of the output folder to your local directory so it isn't lost when the job finishes.

```
cp -r output/ PATH_TO_YOUR_DIRECTORY
```

4.5 Possible problems

5 OSSO

5.1 Description

The goal of this part is to infer the position and the size of the skeleton from the surface of the body (equivalent to the position of the skin).
For more information on OSSO, refer to its project page.

5.2 Prerequisites

You need access to OSSO and all its dependencies. The easiest way to achieve this is through the OSSO_post.sif container image that is available on MIMER for the "Human pose estimation using AI for biomechanics application (SNIC 2022/22-770)" project. On Alvis it is accessible with the following path:

```
/mimer/NOBACKUP/groups/snic2022-22-770/singularity_image/OSSO/OSSO_post.sif
```

You can either directly access it from this path or copy it in your home directory or a personal workspace folder (however, note that it takes a non-negligible amount of storage).

5.3 Instructions to run on Alvis

Open a terminal and start a bash job (change the duration of the job as needed). Change the project ID to the one you are a part of (the information is on your NAISS SUPR project page), and you can choose the type of GPUs you want to use and the number per node (see Alvis documentation for more information).

```
srun -A NAISS2023-22-708 -p alvis --gpus-per-node=A40:1 -t 00:30:00 --pty bash
```

Once you have been attributed resources for this job, open a new terminal and ssh to the node you have booked (alvis0-00 is only an example). Newt, to run a shell within the smplx container, use the following command. The "–writable-tmpfs" flag allows you to make changes to the container image, however they won't remain once you exit it. If your OSSO image isn't in your home directory, update the path as needed. Then, load the environment and move into the OSSO folder.

```
ssh -Y alvis0-00
apptainer shell --fakeroot --writable-tmpfs OSSO_post.sif
. /opt/conda/etc/profile.d/conda.sh
conda activate OSSO
cd /workspace/OSSO/
source osso_venv/bin/activate
```

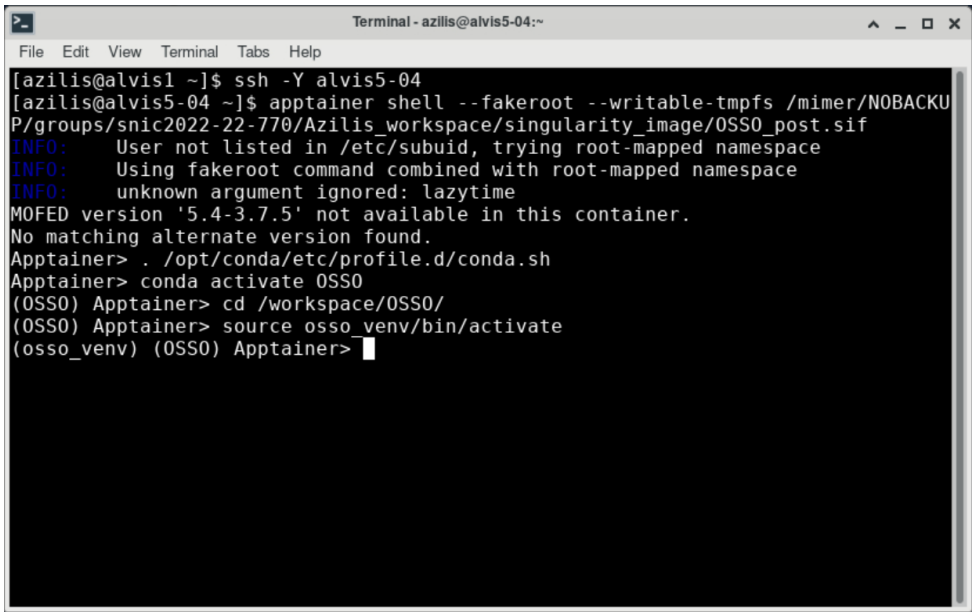

Figure 23: Preparing to run osso

Then, run OSSO. You can choose the gender of the model (female, male, or neutral).

```
python3 main.py  \
    --mesh_input PATH_TO_YOUR_INPUT_MESH \
    --gender male -D -v
```

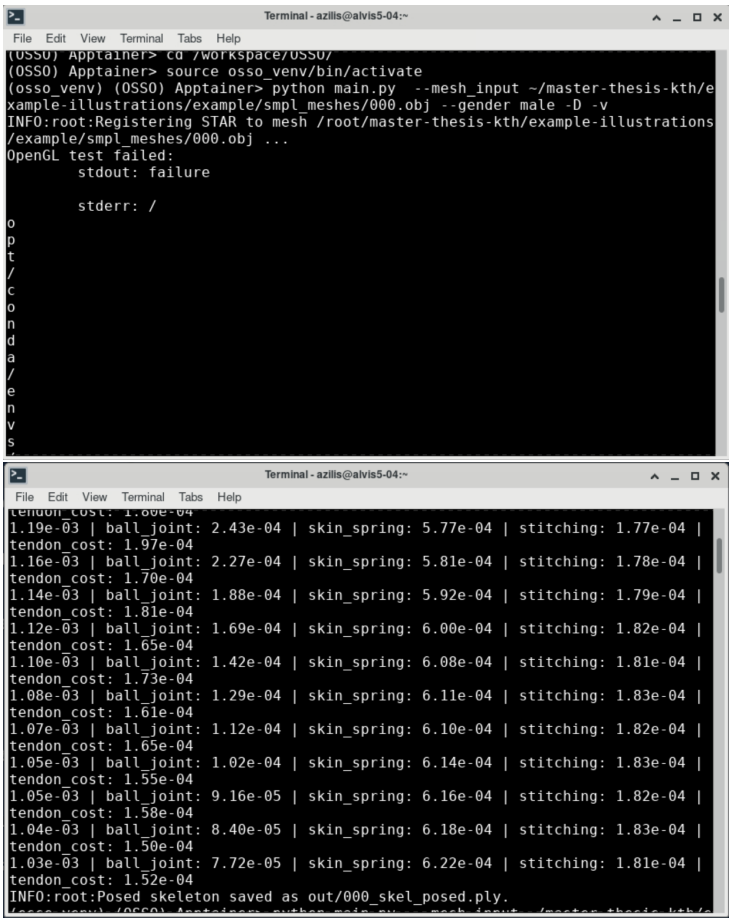

Figure 24: OSSO running example

5.4 Outputs

The output is a PLY file. If the input mesh was named ”mesh.obj” then the output file will be named ”mesh_skel_posed.ply”. It is saved to the ”out/” folder. Thus you need to copy the file to your local directory before the end of the job. For example you can use:

```
cp out/mesh_skel_posed.ply ~/Desktop/
```

You can see the output meshes using software such as MeshLab. Figure 25 shows an example of such an OSSO skeleton.

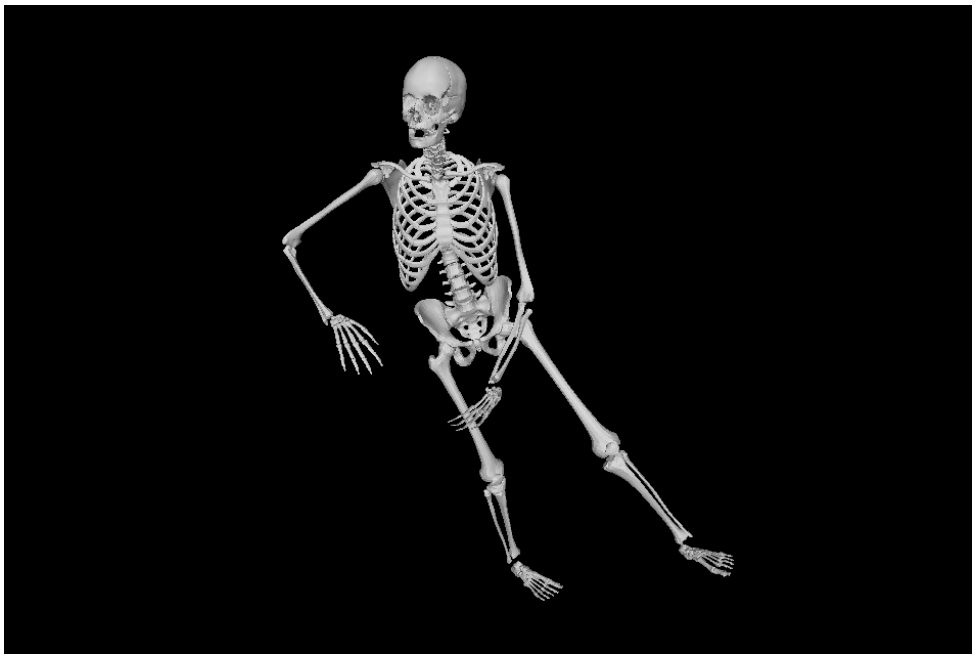

Figure 25: Posed skeleton obtained with OSSO

5.5 Possible problems

5.5.1 ValueError

```
    ValueError: Could not load file
```

Check the path to your file.

5.5.2 AttributeError - Mesh type problem

```
    AttributeError: 'add' object has no attribute 'v'
```

Check that your input mesh is either an SMPL mesh or a STAR mesh. If it's an SMPLX mesh, you need to convert it first then use the SMPL mesh to run OSSO.

Appendices

Command line

Useful commands

Some basic bash commands used in this document:

```
cd
pwd
mv
rm
ls
```

For more Linux bash commands, consult bash cheat sheets such as this one or that one (and there are plenty more available on the Internet).

vim basics

Open a file with vim:

```
vim PATH_TO_FILE
```

Move in document: use direction arrows
Insert something: press the ”I” key on the keyboard and use the Escape key to quit the INSERT mode
Save modification: type ”:w” + enter
Close vim: type ”:q” + enter

For more information on using vim, consult bash cheat sheets such as this one.

SMPLX Blender add-on