**Body Models 4**

Computer Vision ss 16

Gerard Pons-Moll, Javier Romero

**Due date:** Monday 27th of June

**Contact person:** [gerard.pons.moll@tue.mpg.de](mailto:gerard.pons.moll@tue.mpg.de), [jromero@tue.mpg.de](mailto:jromero@tue.mpg.de)

In these exercises you will learn two main things:

* Align a SMPL model to a 3D point
* Align a SMPL model to a 2D image

The exercises are prepared in the folder *06\_BodyModels\_04/* . There, you will find the scripts that you need to run to answer the questions below. The functions you need for running the scripts are in the folder *06\_BodyModels\_04/solution/* . The answers there contain lines with the word TODO to note that you should complete them in order to make the function work.

Use always empirical or theoretical evidence to answer the questions below. Include plots, graphs and figures where necessary. Always support your answers with arguments. Bonus questions are not required to achieve top marks, but can help in achieving them if the student fails to answer other questions successfully.

*Practical note*: In this exercise you will need to install download the SMPL model and install opendr.

For SMPL, go to SMPL for python users in <http://smpl.is.tue.mpg.de/downloads> and sign up. You should download the model as a pkl file and the code to load it. To use the code, update the path to the model in the files smpl\_align\_without\_correspondences\_gradient\_opendr.py

smpl\_align\_without\_correspondences\_gradient.py

and update the PYTHONPATH in the terminal by running (as explained in the README):

export PYTHONPATH=path\_to\_smpl\_code:$PYTHONPATH

To install OpenDR, run the following commands:

sudo apt-get install libgl1-mesa-dev libglu1-mesa-dev libosmesa6-dev

sudo pip install cython opendr

As in previous exercises, to better visualize the triangulated meshes, please install mayavi (http://docs.enthought.com/mayavi/mayavi/). For visualizing the derivative of your objective with chumpy (objective.show\_tree(), great debuggin tool!), you need to install python-pydot. You can install them in ubuntu from the terminal running:

sudo apt-get install mayavi2 python-pydot

**Exercise 1:** (6 points)

After filling the corresponding TODO lines in solution/icp\_smpl\_gradient.py, you can run the code for each of the subexercises as   
ipython --gui=wx smpl\_align\_without\_correspondences\_gradient.py 1a

changing 1a to 1b, 1c, 1d, 1e or 1f

1. Fill the objective in line 58 in solution/icp\_smpl\_gradient, using the point2point\_squared function. Sum\_on\_data refers to summing the distances between all points on the static mesh and their closest points in the deformable mesh (the smpl model).  
   Run the exercise with argument 1a. Does it converge satisfactorily?
2. Now run the subexercise 1b, which loads a different mesh. Does the objective used, which contains only a dataterm, converge? Why?
3. Define the simplest pose prior you can think of in line 60 and run 1c. Such prior would penalize poses deviating from zero without taking into account correlations between joints or their variances. Does it help compared to 1b? Does it converge satisfactorily?
4. Add the definition of the simplest shape prior you can think of and run 1d. Does it help compared to 1c? Does it converge satisfactorily?  
   *Bonus question*: *if I tell you that the shape directions (which get multiplied by the shape coefficients to generate the geometry) are the eigenvectors of PCA on vertices (see slides) scaled by the square root of the explained variance, what exactly does this prior express?*
5. Add the definition of the sum\_on\_model in line 59 (summing the distances between all points on the deformable mesh (the smpl model) and their closest points in the static mesh. Run 1e, which uses both sum\_on\_model and sum\_on\_data terms. What would be the reason of using both distances? Does it help in practice?
6. Implement a better prior in line 63, according to the definition in the slides and the provided pose mean (line 41) and pose covariance inverse (line 46). Does it help in practice? *Bonus question: could you imagine a use of the cholesky decomposition of the inverted covariance to make this prior more efficient?*

**Exercise 2:** (4 point)  
After filling the corresponding TODO lines in solution/icp\_smpl\_gradient\_opendr.py, you can run the code for each of the subexercises as   
python smpl\_align\_without\_correspondences\_gradient\_opendr.py 2a

changing 2a to 2b, 2c, 2d or 2e

1. In exercise 2a and 2b, the optimization should find the camera position that minimizes the differences in pixels between im and the model of the image, rdr. Fill the objective and variable of optimization in lines 92 and 93. Run 2a. Does it converge?
2. In exercise 2b the optimization is the same but the amount of noise added to the camera position is doubled. How does that affect to convergence?
3. To improve convergence, we switch the objective using differences between pixels to a Gaussian pyramid. Run 2c. Please search for information about what is a Gaussian pyramid (you can also check the code in opendr) and reason about why Gaussian pyramids help in the convergence of this optimization problem. Run 2d. Why does it fail to converge?

e) Add something to the objective in line 117 that helps the convergence. Note that this problem has lots of local minima so convergence might not be perfect, but it should be substantially better than c.

f) *Bonus question: Can you integrate exercise 1 and 2 and optimize an objective involving both geometric and image-based data terms?*