3D Computer Vision: HW3

Human Pose Estimation

1 Submission

- Assignment due: June 6th (Fri) 11:59pm
- For submission, submit your code and LaTex-exported PDF report.
- You are allowed to use customized helper functions.
- You may use your own function instead of the given one.
- Use the following script to install the dependencies:

```
conda create -n smplify python=3.7
conda activate smplify
pip install -r requirements.txt
```

- Please zip all your files into a single file. Keep submission format as `lastname_firstname_HW3.zip` (all capital letters).
- **Do not forget to submit your PDF report including the results, images.**

3D Computer Vision: HW3

Human Pose Estimation

- You will complete the following functions
  - `fit_3d.py`
    - `Main`
  - `hw3.py`
    - `find_bbox`
    - `drawKeypoints`
    - `drawSkeleton`
    - `exportMesh`
    - `drawMesh`
  - `eval.py`
    - `ReconstructionError`

3D Computer Vision: HW3

Human Pose Estimation

2 Overview

In this assignment, you will see how 3D human pose estimation is achieved by a optimization-based algorithm called SMPLify. More information on SMPLify can be found on the original paper: Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. It is recommended that you read the paper before looking at the given source code for better comprehension.

For this task you are expected to:

`Code`

- Use Deepcut detected joints and SMPLify optimization on the given 10 images chosen from the LSP(Leeds Sports Pose) Dataset.
- Draw bounding box around the detected human.
- Draw 2D Keypoints.
- Draw the skeleton(connected joints) using the predicted joint locations.
- Export predicted SMPL mesh as .obj file.
- Draw the projected SMPL mesh on the image.
- Find the reconstruction error.

You will implement the visualization code in `hw3.py` and reconstruction error evaluation code in `eval.py`. Then, complete the `fit_3d.py` to save your visualization results in `hw3/results` folder and print reconstruction error in the terminal. We will check your code by running `python code/fit_3d.py`.

`PDF Report`

- Draw a brief pipeline of the SMPLify algorithm.
- Attach the output images of the keypoints, skeleton, and projected mesh.
- Describe the structure of .obj file for saving triangular mesh, and attach the captured image of MeshLab with your exported mesh.
- Write a brief summary of SMPLify priors (why they are needed, how does it work) and compare the reconstruction error with/without these priors. (Please refer to sec. 8)

3D Computer Vision: HW3

Human Pose Estimation

3 Drawing Bounding Box, Keypoints

Figure 1: You will draw a bounding box and keypoints on the input image as above. Green points are the ground truth keypoints and the red points are the predicted keypoints after runnning SMPLify.

```
def find_bbox(keypoints):
  ...
  return bbox
```

Input: `keypoints` are keypoints
Output: `bbox` $\in \mathbb{R}^{4\times 1}$ is the coordinates of the bounding box lying on the input image.
Description: Find the bounding box coordinates from the estimated keypoints from SMPLify.

```
def drawKeypoints(img, keypoints, color):
  ...
```

Input: Input Image, Detected Keypoints, Keypoint Color
Description: Use OpenCV's circle function to draw keypoints on the input image.

`To Dos:`

- Implement above functions and draw bounding box, groundtruth keypoints, and predicted keypoints from SMPLify. Please save the resulting images as `bbox_keypoints_{i}.jpg` in `hw3/results` folder. You can load groundtruth from `hw3/results/est_joints.npz`.

3D Computer Vision: HW3

Human Pose Estimation

4 Draw Skeleton

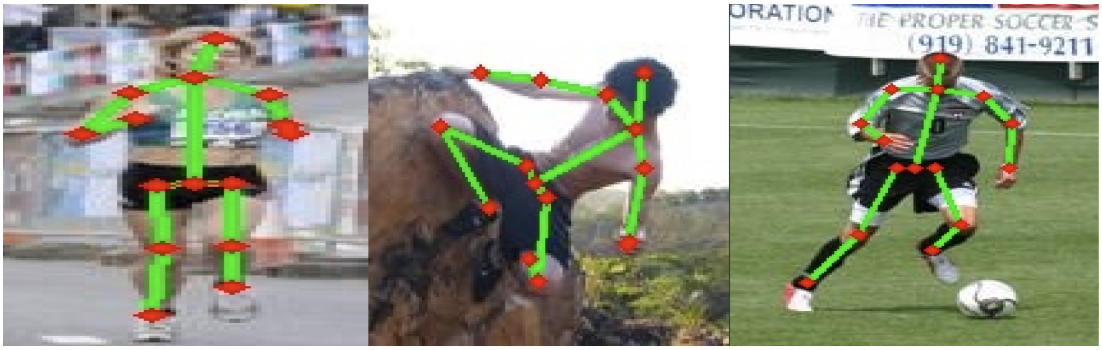

Figure 2: You will draw the 2D skeleton using the detected keypoints. Make sure they are distinguishable from one another.

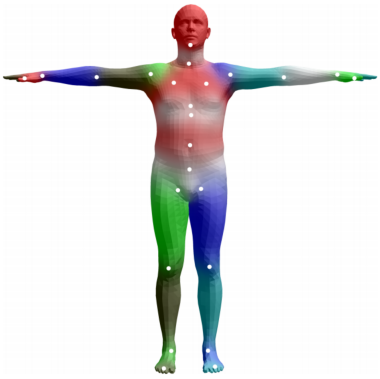

Figure 3: In the SMPL model, the human skeleton is described by a hierarchy of 24 joints as shown by the white dots in the below figure. This hierarchy is defined by a kinematic tree that keeps the parent relation for each joint.

```
def drawSkeleton(img, joints, keypointColor, lineColor):
  ...
```

Input: `joints` are the optimized joint locations.
Description: Draw keypoints(joints) and lines(links) based on the skeleton hierarchy with given keypointColor, lineColor.

`To Dos:`

- Implement above functions and draw skeleton(joints + links) with predicted keypoints from SMPLify. Please save the resulting images as `skeleton_{i}.jpg` in `hw3/results` folder.

5 Export Mesh as an .obj File

```
def exportMesh(savepath, vertices, faces):
  ...
```

Input: `savepath` is a path to save the .obj file using predicted vertices and faces. `vertices` are the vertices from the SMPL output. `faces` contains the vertex to face relation.

Description: Export the SMPL mesh as an .obj file. Do not use the off-the-shelf function from 3D libraries (e.g., trimesh, Open3D, etc.). If you understand the structure of .obj file, you may write with standard file I/O function to save the mesh as an .obj file.

`To Dos:`

- Implement above function and save resulting mesh as `mesh_{i}.obj` in `hw3/results` folder.

3D Computer Vision: HW3

Human Pose Estimation

6 Draw Mesh

Figure 4: You will draw the mesh of the estimated human pose using the vertices and faces from the SMPL output.

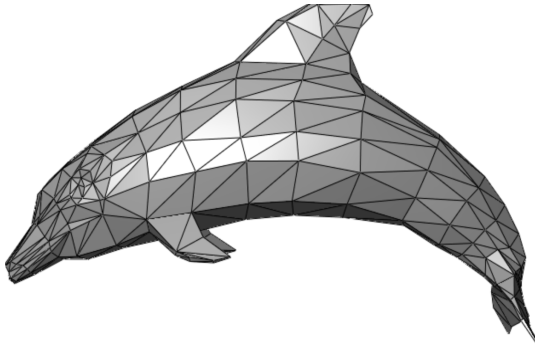

Figure 5: Triangle mesh is a type of polygon mesh. It comprises a set of triangles (typically in three dimensions) that are connected by their common edges or corners. Use vertices and faces from SMPL output to draw the triangular mesh like this dolphin. There are 6890 vertices(many on a small image) so results should look like Figure 4.

```
def drawMesh(img, vertices, faces, lineColor):
  ...
```

Input: `vertices` are the vertices from the SMPL output. `faces` contains the vertex to face relation.
Description: Draw the triangular mesh of the detected human.

`To Dos:`

- Implement above function and draw projected mesh on the given images. Please save resulting image as `mesh_{i}.jpg` in `hw3/results` folder.

7 Calculate Reconstruction Error

```
def reconstrutionError(gt_joints, predicted_joints):
  ...
  return error
```

Input: `gt_joints` are the ground truth joints. `predicted_joints` are the estimated joints after the SMPLify fitting process.

Description: Compute the distance between each ground truth and estimated joints, then provide the mean error.

`To Dos:`

- Implement above function and call this function in the loop to print the reconstruction error after optimization is done for each image.

3D Computer Vision: HW3

Human Pose Estimation

8 SMPLify without Priors

```
opt_weights = zip([4.04 * 1e2, 4.04 * 1e2, 57.4, 4.78],
                  [1e2, 5 * 1e1, 1e1, .5 * 1e1])
```

Figure 6: In `fit3d-optimizeOnJoints` function, you can find the weight configuration used in the paper, with joints + confidence values from the CNN (all the weights used in the code were obtained via grid search, see the paper for more details) the first list contains the weights for the pose priors,the second list contains the weights for the shape prior

`To Dos:`

- Adjust the weights for the pose, shape prior and summarize the quantitative results (using the reconstruction error function in section 7), and the qualitative results (output images).
- Compute the reconstruction error when run SMPLify without each of the pose, shape, and interpenetration prior (turn off with the `--nointerpenetration` argument) and attach each of the fitting result images.
- Compute the reconstruction error without any priors and attach the fitting result images.
- Which prior is SMPLify most dependant on and why?
- Which prior is SMPLify least dependant on and why?
- All reconstruction error must be organized into a table format.

Excluded Prior	Reconstuction Error	Importance
Shape		
Pose		
Interpenetration		
All		NULL

Figure 7: Required Table. On the `Importance` column, provide a rank on (1st,2nd,3rd) the prior with most importance.