Data in Brief 48 (2023) 109087

Contents lists available at ScienceDirect

Data in Brief

journal homepage: www.elsevier.com/locate/dib

Data Article

C3I-SynFace: A synthetic head pose and facial depth dataset using seed virtual human models.

Shubhajit Basak[a,*], Faisal Khan[b], Hossein Javidnia[c], Peter Corcoran[b], Rachel McDonnell[d], Michael Schukat[a]

[a] School of Computer Science, University of Galway, Ireland
[b] School of Electrical and Electronics Engineering, University of Galway, Ireland
[c] School of Computing, Dublin City University, Ireland
[d] School of Computer Science and Statistics, Trinity College Dublin, Ireland

ARTICLE INFO

Article history:
Received 13 February 2023
Revised 9 March 2023
Accepted 17 March 2023
Available online 23 March 2023

Dataset link: SyntheticFaceDataset_Male_Part2 (Original data); SyntheticFaceDataset_Male_Part3 (Original data); SyntheticHeadPoseDataset_Female_Part1 (Original data); SyntheticHeadPoseDataset_Female_Part2 (Original data); SyntheticHeadPoseDataset_Female_Part3 (Original data); SyntheticHeadPoseDataset_Male_Part1 (Original data); SyntheticHeadPoseDataset_Male_Part2 (Original data); SyntheticHeadPoseDataset_Female_Part1 (Original data); SyntheticHeadPoseDataset_Female_Part2 (Original data); SyntheticFaceDataset_Male_Part1 (Original data)

ABSTRACT

This article presents C3I-SynFace: a large-scale synthetic human face dataset with corresponding ground truth annotations of head pose and face depth generated using the iClone 7 Character Creator "Realistic Human 100" toolkit with variations in ethnicity, gender, race, age, and clothing. The data is generated from 15 female and 15 male synthetic 3D human models extracted from iClone software in FBX format. Five facial expressions - neutral, angry, sad, happy, and scared are added to the face models to add further variations. With the help of these models, an open-source data generation pipeline in Python is proposed to import these models into the 3D computer graphics tool Blender and render the facial images along with the ground truth annotations of head pose and face depth in raw format. The datasets contain more than 100k ground truth samples with their annotations. With the help of virtual human models, the proposed framework can generate extensive synthetic facial datasets (e.g., head pose or face depths datasets) with a high degree of control over facial and environmental variations such as pose, illumination, and background. Such large datasets can be

* Corresponding author.
E-mail address: s.basak1@nuigalway.ie (S. Basak).
Social media: @shubhaBas (S. Basak), @pcor (P. Corcoran)

https://doi.org/10.1016/j.dib.2023.109087

Keywords:
Synthetic data
Computer vision
Virtual human
Facial depth
Head pose
Ground truth data

used for the improved and targeted training of deep neural networks.

Specifications Table

Subject	Computer Science
Specific subject area	Computer Vision, Head Pose Estimation, Monocular Depth Estimation
Type of data	Images Annotations
How the data were acquired	The initial 3D virtual human models are collected from the low-cost commercially available software iClone [15] and Character Creator [16]. The data has been produced with a 3D graphics rendering pipeline using the open-source Computer Graphics (CG) software Blender [17]. The source of the 3D virtual models and the generating Python scripts are included in this paper.
Data format	Raw
Description of data collection	Rendering - The face images and the corresponding raw depths have been rendered with the following camera parameters - camera near and far the clip is set to 0.001 and 5.0 meters. Camera sensor size and field of view (FOV) are set to 36 millimeters and 60°, respectively. The yaw, pitch, and roll of a head are constrained to ±30°. To render the face images and raw depths, the RGB and the Z-Pass compositor nodes of the Blender [17] are used with the cycle rendering engine. Background - For complex backgrounds, two scenes (Classroom and Barbershop) from the Blender [17] website have been used.
Data source location	Laboratory: C3Imaging, University of Galway Institution: School of Computer Science, University of Galway City/Town/Region: Galway Country: Ireland
Data accessibility	Direct URL to data: Synthetic Head Pose Datasets: https://data.mendeley.com/datasets/jd4jm3jpp2 [7] https://data.mendeley.com/datasets/mc9fzhkvwp [8] https://data.mendeley.com/datasets/vfrfb56sh4 [9] https://data.mendeley.com/datasets/pttvxjcmpd [10] Synthetic Face Depth Datasets: https://data.mendeley.com/datasets/z4454fyd8b [4] https://data.mendeley.com/datasets/yzjdjj5w39 [5] https://data.mendeley.com/datasets/tbt46rs4y6 [6] https://data.mendeley.com/datasets/33kjk7mj7y [1] https://data.mendeley.com/datasets/5wpj8nh2cv [2] https://data.mendeley.com/datasets/2c2r7998vs [3] Virtual Human Models: Due to licensing issues, we cannot release the virtual human models. But the models can be purchased from the Reallusion website from the following link. These models need to be extracted in fbx format and put in a folder structure as described in the 'Experimental Design' section. https://www.reallusion.com/contentstore/iClone/pack/Realistic_Human_100/default.html Code: https://github.com/shubhajitbasak/blenderDataGeneration
Related research article	S. Basak, P. Corcoran, F. Khan, R. Mcdonnell and M. Schukat, Learning 3D Head Pose From Synthetic Data: A Semi-Supervised Approach in IEEE Access, vol. 9, pp. 37557-37573, 2021, https://doi.org/10.1109/ACCESS.2021.3063884 [11]

Value of the Data

- The data can be used to train and evaluate computer vision models for head pose estimation, face depth estimation, and face reconstruction. The different lighting conditions and camera positions make the data set robust and capable of generalizing the learning model. Finally, the large scale of the dataset makes it an ideal candidate for deep learning training. For head pose estimation, only two datasets, 300WLP(real) [13] and Nvidia Synhead (synthetic) [14], are available for training. There is no real large-scale dataset available for face depth estimation tasks.
- As the dataset is generated synthetically, both the head pose distribution and the depth data cover a wide range of angles and a wide variation of background which can be challenging to acquire in a constrained laboratory condition.
- Both real head-pose and depth data are acquired by inertial measurement unit (IMU) sensors or depth sensors, which are both prone to sensor noise. For example, often, real depth data has missing depth values or holes in it. The most common head pose dataset, Biwi [12], has an average error of 1 degree [14]. These errors in ground truth data eventually pass to the trained model and affect the model performance. On the contrary, the synthetic head pose and depth data generated by our pipeline are pixel-accurate and do not have any of these issues.
- As both the acquisition of real head pose and face depth data required human subjects, they fall under different data protection and privacy regulations like GDPR, which makes them difficult to collect and use for research purposes. Synthetic data can does not fall under any of these rules, so they are easy to use and generate without any restrictions.
- Apart from the raw data, we have also provided the source for synthetic human models and open-sourced the data generation scripts. Using this code and the open-source CG software Blender [17], one can generate an unlimited amount of pixel-perfect data by changing the camera parameters, scene illumination, and background scenes.

1. Objective

Recent advancement of deep learning makes it the de-facto choice for facial analysis tasks. But the human face has a complex structure and requires high-quality ground truth data to learn the features. Collecting real ground truth 3D face data either requires expensive 3D scanners or depth sensors, which are prone to noise. Also, as this data acquisition involves human subjects, they often fall under data privacy restrictions and other ethical regulations. So, creating synthetic face datasets can be an alternative that can provide the freedom to generate high-quality, large, and diverse face datasets without any such restrictions. A key element of face analysis is the face alignment information as well as the face depth to get the 3D cues. So, in this work, with the help of low-cost 3D human assets and an open-source CG tool, we have created a large face dataset with their corresponding head pose and raw depth annotations. Further, we used this dataset to train a model for head-pose estimation and face-depth estimation to validate the generated synthetic data.

2. Data Description

The data set contains two parts: The synthetic faces with their ground truth depth and another set of synthetic faces with their ground truth head pose annotations. The following section will describe those two parts in detail:

- Face Depth dataset:
  Directory Structure **-** This part of the dataset contains all the rendered face images, their corresponding raw depth in *.exr format, and the head poses data in *.txt files. The root

folder contains two subfolders, 'male' and 'female', with all the identity folders with labels from '0001' to '0045'. Each identity folder has a subfolder called 'Complex', which signifies the complex background of the rendered images. There are two different background scenes in two subfolders; the first folder contains the Barbershop scene, and the second folder has the Classroom scene. Each of these folders contains the ground truth files with five expressions - angry, happy, neutral, sad, and scared. For each expression, there are three different datasets based on the rendered settings. We have collected the ground truth for three different camera and head movements. The 'CameraTran' folder contains the ground truths when the virtual human models are at the origin of the scene, and a random translation motion is added to the camera. The 'HeadCameraRotTran' folder has all the ground truths where the models are in the scene origin, and a head rotation {-30°, +30°} is applied to the head bone of the model, and a simultaneous rotation and translation are added to the camera. The last folder, 'HeadRot,' contains the ground truth, where the camera is placed in front of the face in a fixed location, and a rotation {-30°, +30°} is applied to the head of the model. Fig. 1a shows the directory structure, and 1b shows some examples of rendered ground truth face images and their corresponding raw normalized depth visualized in grayscale and plasma color maps.

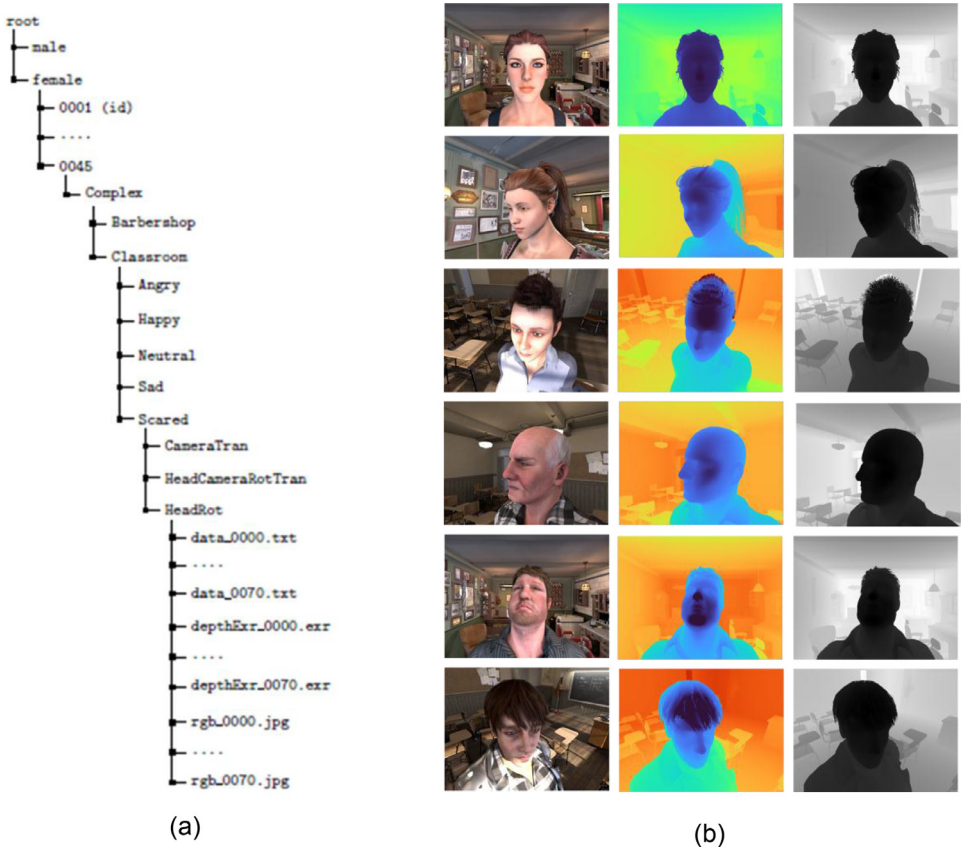

Fig. 1. (a) Folder structure for the facial depth data (b) Sample data rendered in Blender – 1st column is the RGB image, 2nd and 3rd are the normalized depth data (*.exr) visualized in colormap and grayscale.

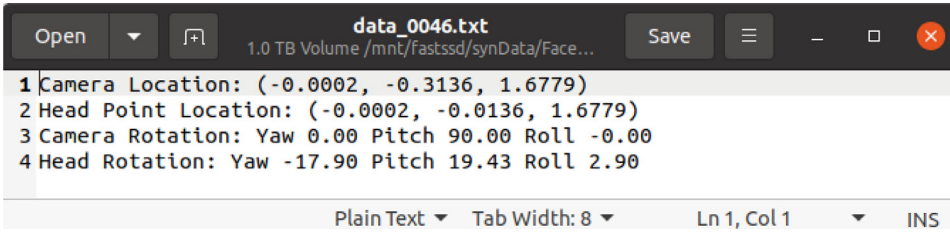

Fig. 2. Sample annotation data consists of the camera position, head point location, camera rotation, and head rotation.

Ground Truth Annotations - Each of these final folders contains three type of ground truth data –

I. **rgb_<id>.jpg** - The rendered ground truth face image in RGB (*.jpg) format.
II. **depthExr_<id>.exr** - The raw depth (distance of the face point from the camera center) values in *.exr format.
III. **data_<id>.txt** - The ground truth annotations in *.txt files. Each text file contains four ground truth annotations - camera location (x,y,z coordinates), head point location (x,y,z coordinates of the head bone), camera rotation (yaw, pitch, and roll of the camera) and head rotation (yaw, pitch, and roll of the head bone). Fig. 2 shows a sample annotation text file.

The depth dataset contains a total of 37670 sets of ground truth images and their corresponding annotations (.exr and .txt) with a total size of around 45 GB.

- Head Pose Dataset:
  Directory Structure – This part of the dataset contains the ground truth face images with varying head pose and their corresponding head pose annotations. The root folder contains two subfolders, 'male' and 'female', with all the identity folders with labels from '0001' to '0045'. Each of these folders contain ground truth RGB images and their corresponding head pose data annotations in a text file. Fig. 3a shows the directory structure and 3b shows some samples of ground truth RGB images with varying head poses.
  Ground Truth Annotations - Each of these folders contains the ground truth data –
  I. **rgb_<id>.jpg** - The rendered ground truth face image in RGB (*.jpg) format.
  II. **data_<id>.txt** - The ground truth annotations in *.txt files. Similar to the depth data each of these text file contains four ground truth annotations - camera location (x,y,z coordinates), head point location (x,y,z coordinates of the head bone), camera rotation (yaw, pitch, and roll of the camera) and head rotation (yaw, pitch, and roll of the head bone). Fig. 2 shows a sample annotation text file.

This part of the dataset contains a total of 72060 pairs of ground truth images and their corresponding annotations (.txt) with a total size of around 32 GB.

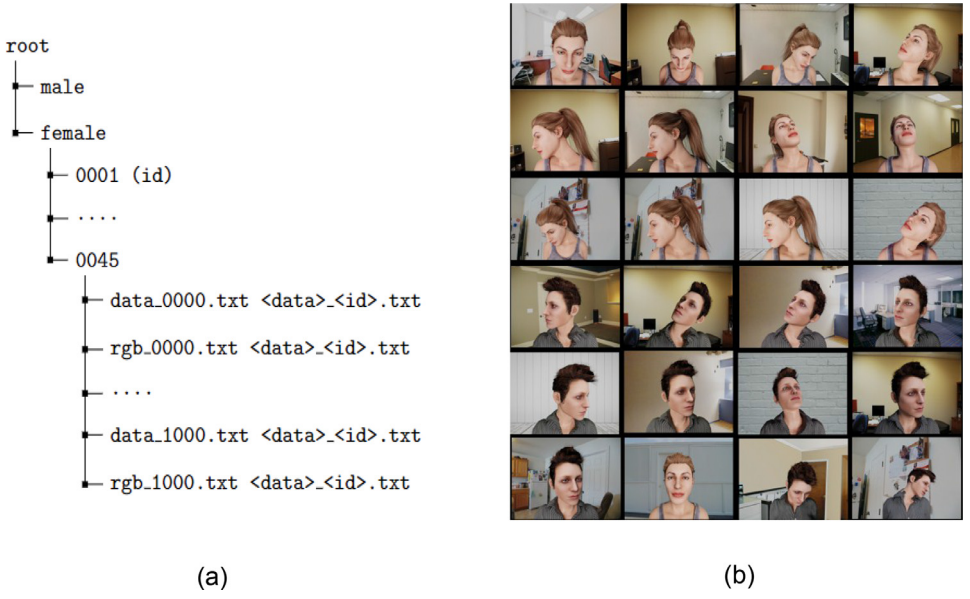

Fig. 3. (a) Folder structure for the head pose data (b) Ground truth synthetic face data rendered in Blender with varying head pose.

3. Experimental Design, Materials and Methods

This section describes the methodology for generating the data set, including the details of the FBX models and the rendering pipeline. It also provides the details of the Python code that has been used to generate the ground truth data with the help of Blender [17].

- 3D Model Generation
  Though we are not able to release the virtual model publicly, here we provide the detailed methodology to create this part of the data.

  I. The models can be generated from the 'Realistic Human 100' package in iClone [15] software. It provides the functionality to add expressions to the face morphs. The models can be exported in FBX formats from the iClone Character Creator [16].
  II. The iClone [15] tool provides a feature to add different facial expressions to a morph to enhance the facial mesh's diversity. We have added random changes in the face morph and added different clothing to the model. Then we added four different expressions angry, happy, sad, and scared. The default model is the neutral one.
  III. Then we export the models in fbx format through the iClone Character Creator [16] export pipeline[1].
  IV. The FBX files need to be structured in the following manner to run the python scripts provided to generate the ground truth data. The root folder contains the two subfolders, 'male' and 'female,' which have the male and female model files within them. In each of these folders, there are identity folders for female and male models, starting from '0001' up to '0100'. Each identity folder has a subfolder called 'Simple' with five subfolders containing the FBX model files with five different expressions - angry, happy, neutral, sad, and scared. Each of these folders has the

[1] https://www.reallusion.com/character-creator/blender.html.

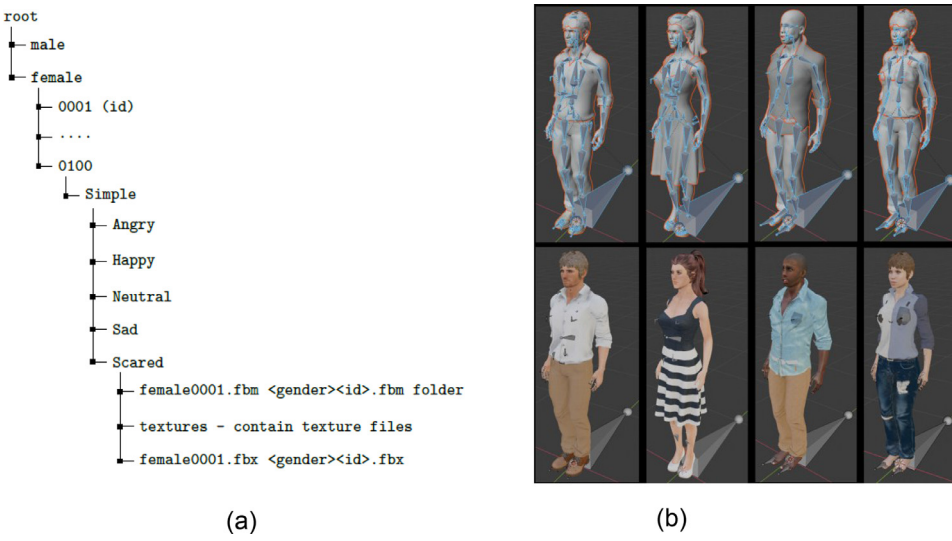

Fig. 4. (a) Folder structure for the virtual human model data (b) Samples of the synthetic human models in fbx format after exporting in Blender.

texture (in textures and fbm folders) and other image files associated with the FBX models. Fig. 4a shows the directory structure of this data set.

V. These models are then imported into the existing Blender scene. We have used two different Blender scenes – Barbershop and Classroom collected from the Blender [17] free library. Fig. 4b shows the sample models (in FBX format) imported into Blender [17] with their armatures (bones) and textures. The imported FBX models are rigged with armatures, later used to add head movements.

- Rendering Ground Truth

  To render the ground truth, we followed the following steps. All these steps are performed in batch through python scripts that will be explained in the next section.

I. A camera is added to the scene in perspective mode, and adequate illumination (e.g., area, sun, point, and spotlight) is added on top of the existing light in the Blender [17] scenes.

II. To apply the head pose, the neck bone is selected. An empty object is added to the center of the two eyeballs, chosen as the center of the head. The camera's optical axis is set as normal to the plane of the two eyeballs to select the initial head pose. The neck bone's translation and rotation have been copied to the empty object, which adds constraints to the empty object to follow the neck bone.

III. Once the initial setup is completed, a uniform rotation is applied to the neck bone in the sequence of PRY (pitch, roll, and yaw), and the keyframes are saved.

IV. After the final design, the ground truth face RGB images with their corresponding raw depths and annotations are generated by a Python script with the help of Blender [17]. Blender's [17] in-build Python support is used to run these scripts. Details about the rendering parameters used while generating the data are shown in Table 1.

V. We have also provided a separate dataset emphasizing head pose annotations only. We have not used virtual environments because we have not collected the depth information for this part of the data. Instead, we have set real images as the background and put the virtual models in front of them to render the ground truths.

Table 1
Rendering parameters used to generate the ground truth in Blender.

Parameters	Values
Camera center and model head center distance	30 centimeters
Camera Near Clip	0.001 meter
Camera Far Clip	10.0 meters
Camera Sensor Size	36 millimeters
Camera Field of View (FOV)	60 degrees
Blender Rendering Engine	CYCLES
Cycles Progression	BRANCHED PATH
Cycles AA samples	256
Cycles Min transparent bounces	32
Cycles Light sampling threshold	0
Cycles Sample clamp indirect	0
Cycles Max bounces	32
Cycles Diffuse bounces	0
Cycles Glossy bounces	0
Cycles Transparent max bounces	16
Cycles Transmission bounces	16
Rendering Resolution X	640
Rendering Resolution Y	480

VI. We have added discrete head rotations to the models in an interval of 3°. The yaw, pitch, and roll ranges are ±80°, ±70°, and ±55°, respectively. Though these rotations cover an extensive range of angles, as these are discrete linear sequences, these do not cover some cross-rotation angles. So, to cover all the practical human head pose angles, we have also applied the rotations collected as the ground truth of a real dataset called Biwi [12].
VII. After applying the head rotation and saving the keyframes, we render each frame through the Blender [17] cycle rendering engine.

• Explanation of Generating Code -

The following section elaborates on the main components of the Python code to generate the ground truth data. The complete code for ground truth generation with complex background is attached to this paper as supplementary material. Also, additional codes for simple and textured background generation can be found on the GitHub page mentioned in the specification table.

importFbx.py: In the first step, the FBX models are imported to the Blender [17] scene (classroom.blend or barbershop.blend), and the imported model is scaled in proper scale to match the Blender [17] scene.

```
bpy.ops.import_scene.fbx(filepath=fbxFilePath)
bpy.context.object.scale = (0.01, 0.01, 0.01)
#save the blend file
bpy.ops.wm.save_mainfile(filepath=blendFilePath)
```

importMisFileBlender.py: First, we set the blender properties. Next, a script is run to add missing texture files (if any) to the *. blend file. sceneSetup.py: In the next step, the Blender [17] scene with the model is set up and head movements are added. Also, the other rendering parameters, like camera properties, are set. In the next step, the midpoint of the two eyeballs is computed by setting empty objects in the eyeball positions and calculating the midpoint of those object locations in global coordinates.

```
emptyList = []
# set the empty at the center of two eye balls
for _obj in _objects:
    if _obj.type == 'MESH':
        if 'Eye' in _obj.name:
            for name in _obj.vertex_groups.keys():
                mt = bpy.data.objects.new("empty", None)
                bpy.context.scene.collection.objects.link(mt)
                mt.name = f"{_obj.name}_{name}"
                emptyList.append(mt.name)
                cl = mt.constraints.new('COPY_LOCATION')
                cl.target = _obj

# get the mid-point of two eyeballs
l_eye = bpy.data.objects[emptyList[1]]
r_eye = bpy.data.objects[emptyList[0]]
l_eye_pos = Point(l_eye.matrix_world.translation)
r_eye_pos = Point(r_eye.matrix_world.translation)
global_location = map_tuple_gen(float,
l_eye_pos.midpoint(r_eye_pos))
```

After setting the midpoint of the eyeballs, which will be the center of the head, another empty object is placed at that point. It will provide the head pose ground truth data. Also, the camera is positioned perpendicular to this point as its initial position.

```
# Create camera object
camera = Camera()
# set the camera in the perspective mode
camera.set_perspective(focal_length=render_focal_length,
sensor=render_sensor_size)

# Offset the camera by the default camera position along the z-axis
camera_location = [global_location[0], -(-global_location[1] +
default_camera_position), global_location[2]]
camera.set_location(camera_location)
camera.set_rotation((radians(90), 0, 0))

# Set the far clipping plane of the camera to a multiple of its
distance from the origin
# far clip around 10 meter
camera.set_far_clipping_plane(default_camera_position * 50)

# create an empty at the center of the two eyeballs
empty1 = bpy.data.objects.new("empty", None)
empty1.location = global_location
```

Finally, the neck bone is chosen on which the head rotation is applied. The rotation is applied while inserting a keyframe to save the animation. The following shows a sample example of adding the animations:

```
pbone = arma.pose.bones['NeckTwist01']   # G6Beta_Neck  NeckTwist01
# Set rotation mode to Euler XYZ, easier to understand
# than default quaternions
pbone.rotation_mode = 'XYZ'

# add head rotations and save the frames
for i in range(0, 10):
    pbone.keyframe_insert(data_path="rotation_euler", frame=i + 1)
    pbone.rotation_euler.rotate_axis('Y', radians(-3))
```

compositorSetup.py: Using this script, the compositor nodes in Blender [17] are declared to set the output ground truth paths. It also sets the output data format (e.g., JPEG for RGB and EXR for raw depth data). Finally, the rendering parameters are established, as stated in Table 1.

Following is a code snippet where we set the Blender [17] scene properties.

```
basePath = '../data/Data'
filepath = basePath + '/'.join(bpy.data.filepath.split('/')[-5:-1])
filepath = filepath.replace('Simple', 'Complex/Barbershop')

# create Render Layer node
renderLayer_node = tree.nodes.new('CompositorNodeRLayers')
renderLayer_node.location = 0, 0

# create Normalize node
normalize_node = tree.nodes.new('CompositorNodeNormalize')
normalize_node.location = 200, 0

# create output node
comp_node = tree.nodes.new('CompositorNodeComposite')
comp_node.location = 400, 0

# link nodes
links = tree.links
links.new(renderLayer_node.outputs[2], normalize_node.inputs[0])
links.new(normalize_node.outputs[0], comp_node.inputs[0])

# create file output node
fileOutput_node = tree.nodes.new('CompositorNodeOutputFile')
fileOutput_node.location = 400, -100
fileOutput_node.base_path = filepath
fileOutput_node.format.color_mode = 'RGB'
fileOutput_node.file_slots[0].path = f'rgb'
fileOutput_node.file_slots[0].format.file_format = 'JPEG'
fileOutput_node.file_slots[0].use_node_format = False

fileOutput_node.file_slots.new("depthExr")
fileOutput_node.file_slots['depthExr'].format.file_format =
'OPEN_EXR'
fileOutput_node.file_slots['depthExr'].format.color_mode = 'RGB'
fileOutput_node.file_slots['depthExr'].format.use_zbuffer = True
fileOutput_node.file_slots['depthExr'].use_node_format = False

# link nodes
links.new(renderLayer_node.outputs[0], fileOutput_node.inputs[0])
links.new(renderLayer_node.outputs[2], fileOutput_node.inputs[1])
```

captureWithGaze.py: Finally, the ground truth is rendered by running this script. As stated in 1.2, the ground truth is generated with three different camera and head rotation settings:

- **Head Rotation:** In this scenario, the camera is placed in its initial location and applied to the frames saved during the above scene setup with the head rotation.

```
#iterate to get the saved key frames
for i in range(0, 70):
# set the key frame
    bpy.context.scene.frame_set(i)

    # set the generation file path
    bpy.data.scenes["Scene"].render.filepath = dataPathHeadRot +
f'/depth_{i:04d}.png'
    textFilePath = dataPathHeadRot + f'/data_{i:04d}.txt'

    # render
    bpy.ops.render.render(write_still=True)

    # get the camera and the empty (center of head) data
    matrix_empty = bpy.data.objects['empty'].matrix_world
    matrix_camera = bpy.data.objects['Camera'].matrix_world

    # covert to tuple
    r_empty = tuple(map(round,
np.degrees(np.array(matrix_empty.to_euler('XYZ')[0:3])), repeat(4)))
    r_camera = tuple(map(round,
np.degrees(np.array(matrix_camera.to_euler('XYZ')[0:3])),
repeat(4)))

    # write in a text file
    with open(textFilePath, "w") as f:
        f.write('Camera Location: ' + str(tuple(map(round,
matrix_camera.translation, repeat(4)))) + '\n')
        f.write('Head Point Location: ' + str(empty_init_loc) +
'\n')
        f.write('Camera Rotation: ' + 'Yaw %.2f Pitch %.2f Roll
%.2f' % (r_camera[2], r_camera[0], r_camera[1]) + '\n')
        f.write('Head Rotation: ' + 'Yaw %.2f Pitch %.2f Roll %.2f'
% (r_empty[2], r_empty[0], r_empty[1]) + '\n')
```

- **Camera Translation:** In this scenario, the camera is placed in its initial location, and translation is applied while keeping the head model stationary at its initial location. The sample code to apply this translation is given below:

```
for i in range(0, 1):
    # set camera initial location and rotation
    camera.location = camera_init_loc
    camera.rotation_euler = camera_init_rotation

    # apply uniform translation to the camear in the Y plane
    camera.location.x = camera.location.x + random.uniform(0.001,
0.140)
    camera.location.z = camera.location.z + random.uniform(0.001,
0.04)
```

- **Head Rotation and Camera Rotation & Translation:** In this scenario, firstly, a virtual half sphere centering the empty object (center of the eyeballs) is constructed before randomly generating distributed points on that half sphere to where the camera is moved. In contrast, the camera axis is pointed to the empty object. At the same time, the previously saved frames are applied to the head model to rotate the head linearly. The sample code to generate the points in a half sphere and apply them to the camera position is shown below:

```
from sympy.geometry import Point
def point_at(obj, target, roll=0):
    """
        Rotate obj to look at target

        :arg obj: the object to be rotated. Usually the camera
        :arg target: the location (3-tuple or Vector) to be looked
at
        :arg roll: The angle of rotation about the axis from obj to
target in radians.

        Based on: https://blender.stackexchange.com/a/5220/12947
(ideasman42)
    """
    if not isinstance(target, mathutils.Vector):
        target = mathutils.Vector(target)
    loc = obj.location
    # direction points from the object to the target
    direction = target - loc

    quat = direction.to_track_quat('-Z', 'Y')
    quat = quat.to_matrix().to_4x4()
    rollMatrix = mathutils.Matrix.Rotation(roll, 4, 'Z')

    # remember the current location, since assigning to
obj.matrix_world changes it
    loc = loc.to_tuple()
    obj.matrix_world = quat @ rollMatrix
    obj.location = loc
```

```
def generatePoints(r, center, num):
    """
        Generate the cartesian co-ordinates of random points on a
surface of sphere
        Constraints : phi - with (60,120) degree
                      theta - with (-45,45) degree

        :arg r: radius of the sphere
        :arg center: center of the sphere
        :arg num: number of random locations to be generated

        Based on:
https://stackoverflow.com/questions/33976911/generate-a-random-
sample-of-points-distributed-on-the-surface-of-a-unit-sphere
        """


    pts = []
    for i in range(0, num):
        phi = radians(random.uniform(60, 120))
        theta = radians(random.uniform(-45, 45))
        x, y, z = (center.x - sin(theta) * sin(phi) * r), (center.y
- (cos(theta) * sin(phi) * r)), (
                center.z + cos(phi) * r)
        pts.append((x, y, z))
    return pts

center = Point(tuple(empty_init_loc))
r = 0.3
num = 70
pts = generatePoints(r, center, num)
start = 0
for i, pt in enumerate(pts, start):
    frame = i
    # set heead rotation frame
    bpy.context.scene.frame_set(frame)
    bpy.context.view_layer.update()

    # apply camera location
    camera.location = tuple(map(float, tuple(pt)))
    # point the camera to the empty
    point_at(camera, empty_init_loc, roll=radians(0))
```

renderFromCMD.py: To run correctly, we must execute these individual scripts within the Blender [8] python console. So, to generate the ground truth in batch, we pass these individual scripts as a command line argument to the Blender [17] executable while iterating through all the fbx files from the model root folder. We execute the scripts in the same order as discussed above - importFbx.py → importMisFileBlender.py → sceneSetup.py → compositorSetup.py → captureWithGaze.py. A sample execution (for importFbx.py) is shown in the following code snippet.

```
import os
# set all the directories
target_folder = r'blenderDataGenerationFinal/data/FullBodyModels' #
fbx model path
blenderScenePath =
r'blenderDataGenerationFinal/data/Environments/Barbershop/barbershop_i
nterior_gpu.blend'  # blender scene path
sceneName = '_barbershop'
blenderPath = '/blender_path/blender-2.81-linux-glibc217-
x86_64/blender ' # blender executable path
importfbxScript = r'\importFbx.py'
# iterate through all the fbx files
for root, dirs, files in os.walk(target_folder):
    for dir in dirs:
        for file in os.listdir(str(root) + '/' + str(dir)):
            if file.endswith(".fbx"):
                fbxFilePath = os.path.join(root, dir, file)
                blenderFile =
os.path.basename(fbxFilePath).split('.')[0] + sceneName + '.blend'
                blenderFilePath = os.path.join(root, dir, blenderFile)
                if os.path.exists(blenderFilePath):
                    os.remove(blenderFilePath)
                pwd = 'XXXXXXXX'
                cmd = blenderPath + blenderScenePath + " --background
-P " \
                      + importfbxScript + " " + fbxFilePath + " " +
blenderFilePath
                # call blender executable with the required arguments
                p = subprocess.call('echo {} | sudo -S {}'.format(pwd,
cmd), shell=True)
```

Ethics Statements

The work did not involve human or animal subjects or data from social media platforms.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

SyntheticFaceDataset_Male_Part2 (Original data) (Mendeley Data).
SyntheticFaceDataset_Male_Part3 (Original data) (Mendeley Data).
SyntheticHeadPoseDataset_Female_Part1 (Original data) (Mendeley Data).
SyntheticHeadPoseDataset_Female_Part2 (Original data) (Mendeley Data).
SyntheticHeadPoseDataset_Female_Part3 (Original data) (Mendeley Data).
SyntheticHeadPoseDataset_Male_Part1 (Original data) (Mendeley Data).
SyntheticHeadPoseDataset_Male_Part2 (Original data) (Mendeley Data).
SyntheticHeadPoseDataset_Female_Part1 (Original data) (Mendeley Data).
SyntheticHeadPoseDataset_Female_Part2 (Original data) (Mendeley Data).
SyntheticFaceDataset_Male_Part1 (Original data) (Mendeley Data).

CRediT Author Statement

Shubhajit Basak: Conceptualization, Methodology, Software, Writing – original draft; **Faisal Khan:** Data curation, Resources, Investigation; **Hossein Javidnia:** Conceptualization, Formal analysis, Investigation; **Peter Corcoran:** Formal analysis, Validation, Supervision; **Rachel McDonnell:** Supervision, Funding acquisition; **Michael Schukat:** Writing – review & editing, Supervision, Project administration.

Acknowledgments

Funding: This work was conducted with the financial support of the Science Foundation Ireland Centre for Research Training in Digitally-Enhanced Reality (D-REAL) under Grant No. 18/CRT/6224.

Supplementary Materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.dib.2023.109087.

References

[1] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticfacedataset_male_Part1", Mendeley Data V1 (2022), doi:10.17632/33kjk7mj7y.1.
[2] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticfacedataset_male_Part2", Mendeley Data V1 (2022), doi:10.17632/5wpj8nh2cv.1.
[3] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticfacedataset_male_Part3", Mendeley Data V1 (2022), doi:10.17632/2c2r7998vs.1.
[4] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticfacedataset_female_Part1", Mendeley Data V1 (2022), doi:10.17632/z4454fyd8b.1.
[5] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticfacedataset_female_Part2", Mendeley Data V1 (2022), doi:10.17632/yzjdjj5w39.1.
[6] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticfacedataset_female_Part3", Mendeley Data V1 (2022), doi:10.17632/tbt46rs4y6.1.
[7] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticheadposedataset_male_Part1", Mendeley Data V2 (2022), doi:10.17632/jd4jm3jpp2.2.
[8] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticheadposedataset_male_Part2", Mendeley Data V2 (2022), doi:10.17632/mc9fzhkvwp.2.
[9] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticheadposedataset_female_Part1", Mendeley Data V2 (2022), doi:10.17632/vfrfb56sh4.2.
[10] S. Basak, H. Javidnia, F. Khan, R. Mc Donnell, P. Corcoran, M. Schukat, "Syntheticheadposedataset_female_Part2", Mendeley Data V2 (2022), doi:10.17632/pttvxjcmpd.2.
[11] S. Basak, P. Corcoran, F. Khan, R. Mcdonnell, M. Schukat, Learning 3d head pose from synthetic data: a semi-supervised approach, IEEE Access 9 (2021) 37557–37573, doi:10.1109/ACCESS.2021.3063884.
[12] G. Fanelli, M. Dantone, J. Gall, A. Fossati, L. Van Gool, Random forests for real time 3d face analysis, Int. J. Comput. Vis. 101 (3) (2013) 437–458.
[13] X. Zhu, Z. Lei, X. Liu, H. Shi, S.Z. Li, Face alignment across large poses: a 3d solution, in: Proceedings of the IEEE Conference On Computer Vision And Pattern Recognition, 2016, pp. 146–155.
[14] J. Gu, X. Yang, S. De Mello, J. Kautz, Dynamic facial analysis: from bayesian filtering to recurrent neural network, in: Proceedings of the IEEE Conference On Computer Vision And Pattern Recognition, 2017, pp. 1548–1557.
[15] Real-Time 3D Animation Software | iClone | Reallusion. Accessed: Feb. 24, 2023. [Online]. Available: https://www.reallusion.com/iclone/.
[16] Character Creator—Fast Create Realistic and Stylized Characters. Accessed: Feb. 24, 2023. [Online]. Available: https://www.reallusion.com/character-creator/.
[17] Blender—Home of the Blender Project—Free and Open 3D Creation Software. Accessed: Feb. 24, 2023. [Online]. Available: https://www.blender.org/.