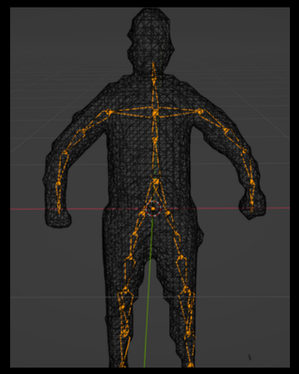
COMPUTER VISION 22/23

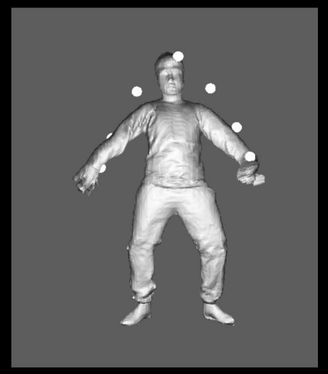
Ro.Bo.

Rigging the human body

Reale Lucchesi Pica Gioia
(giua.glb)

Goals of our project





Create a mesh from a picture of the human body

We create a 3D mesh of the human body starting from a single picture. To achieve this result, we use PIFUhd after taking a picture with our laptops.

Automatically rig the mesh

In order to animate the model, we need to create its joints. We use RigNet to make this process automatic.

Real time animation through motion tracking

We animate the resulting mesh with the help of MediaPipe and the Panda3D game engine. We do this in real time using just the webcam of our laptops.

Why though?

Simulate 3D style-transfer

Can be used to create a version of yourself which mimicks a character (e.g. ogre, demon, etc...) for role-playing games (real-time or offline).

VTubing

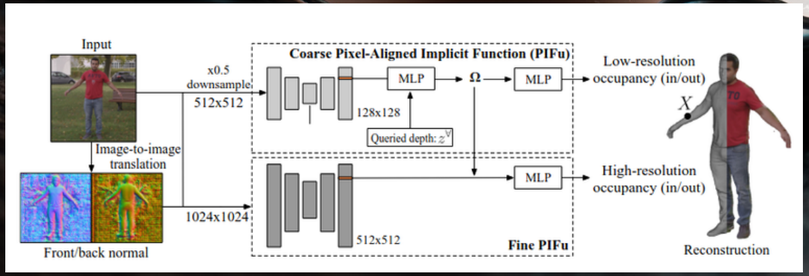
Cheap real-time 3D Virtual Youtubing for everyone!
The 3D model can be modified prior to rigging to make it more appealing.

3D printing a funny version of yourself

Why not?

PIFUhd

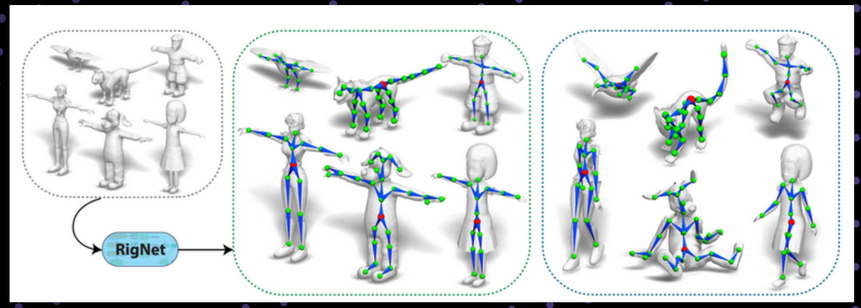
PIFUhd is an extension of the already existing PIFU model. It combines a coarse image encoder and a fine one in order to retain both robustness and details. This allows to reconstruct 3D meshes of the subject in different poses with just a single image.

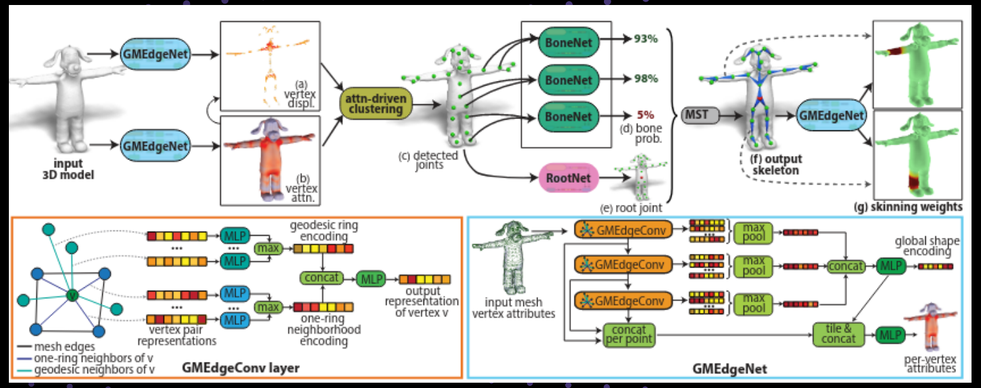


Taken from https://shunsukesaito.github.io/PIFuHD/

RigNet

RigNet is an end-to-end automated method for producing animation rigs from input character models. Given an input 3D model representing an articulated character, RigNet predicts a skeleton that matches the animator expectations in joint placement and topology.





Taken from https://zhan-xu.github.io/rig-net/

```python
# function that checks if the coordinates are aligned
median = (left_shoulder_y + right_shoulder_y) / 2
if alignment(median, left_elbow_y) and alignment(median, right_elbow_y) and alignment(median, left_wrist_y) and alignment(median, right_wrist_y):
  t_pose = True
else:
  print("Please stand in a t-pose")
  t_pose = False
# check if the legs are visible
if (left_ankle_visibility < 0.9) and (right_ankle_visibility < 0.9):
  print("Please make sure that the legs are visible")
  t_pose = False
# check if the arms are fully extended
if ((right_wrist_x - right_elbow_x) > 30) and ((left_wrist_x - left_elbow_x) > 30):
  print("Please make sure that your arms are not fully extended")
  t_pose = False
```

```python
img = cv2.imread("photo.png")
smoothed = cv2.GaussianBlur(img, (9, 9), 10)
unsharped = cv2.addWeighted(img, 1.5, smoothed, -0.5, 0)
cv2.imwrite("unsharped.png",unsharped)
```

```python
SEGMENTATION_NETWORK = "tracer_b7" #@param ["u2net", "deeplabv3", "basnet", "tracer_
POSTPROCESSING_METHOD = "fba" #@param ["fba", "none"]
SEGMENTATION_MASK_SIZE = 640 #@param ["640", "320"] {type:"raw", allow-input: true}
TRIMAP_DILATION = 30 #@param {type:"integer"}
TRIMAP_EROSION = 5 #@param {type:"integer"}
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'
```

```python
#Target number of vertex
TARGET=1000

#Estimate number of faces to have 100+10000 vertex using Euler
numFaces = 100 + 2*TARGET

#Simplify the mesh. Only first simplification will be agressive
while (ms.current_mesh().vertex_number() > TARGET):
    ms.apply_filter('meshing_decimation_quadric_edge_collapse', targetfacenum=numFaces
    print("Decimated to", numFaces, "faces mesh has", ms.current_mesh().vertex_number(
    #Refine our estimation to slowly converge to TARGET vertex number
    numFaces = numFaces - (ms.current_mesh().vertex_number() - TARGET)
```

Pipeline and our improvements

We engineered a pipeline in order to make the entire process automatic.

1. Automatic acquisition of the picture when in the right pose (**MediaPipe**)
2. Background removal (CarveKit) and sharpening (**OpenCV**)
3. PIFUhd on the pre-processed image
4. Remesh (reducing nr. of faces) of the result (PyMeshLab)
5. RigNet on the remeshed result

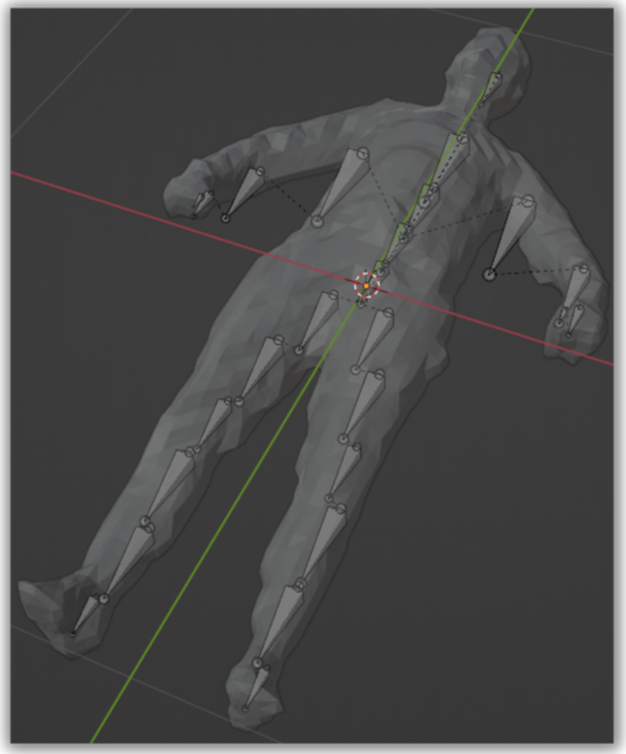
This pre-processing ensures better results than just taking a simple picture and applying PifuHD + RigNet. It also runs on **Colab**.

Computing a rigged 3D model

RigNet generates a .txt file with a list of joints, not a 3D model!

- ➔ Clean-up joints (remove duplicates, rename)
- ➔ Combine OBJ with joints
- ➔ Generate GLTF file

All done by automating Blender.



Panda3D 

Panda3D is an open-source engine for realtime 3D games, visualizations, simulations and experiments.

We used panda to render the final model with the animation.

- ➔ we loaded the rigged model
- ➔ then we included MediaPipe's landmarks
- ➔ finally we matched the landmarks to the bones

But how to animate the joints?

But how to animate the joints?

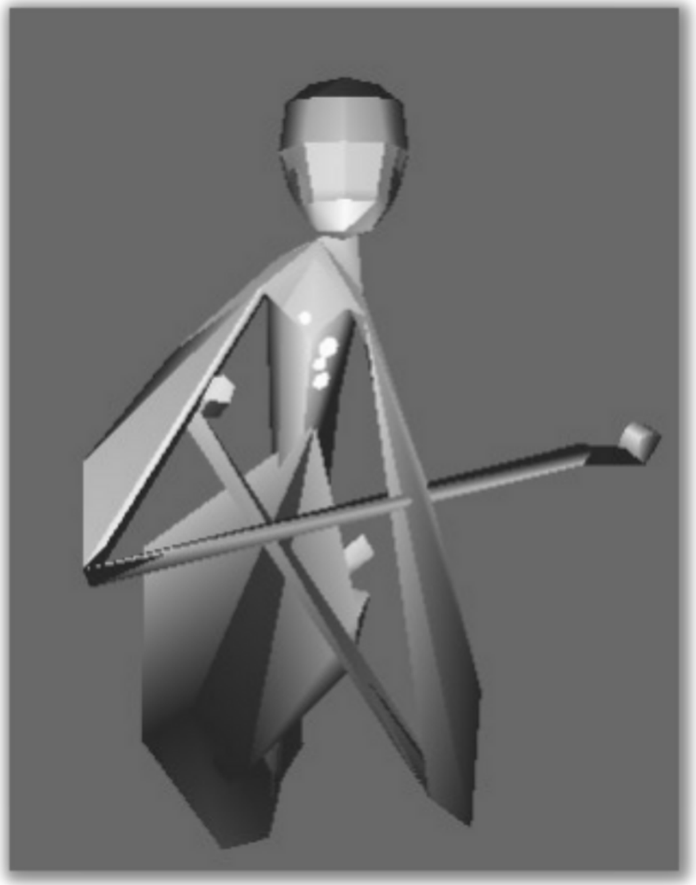
Move the joints into place?

But how to animate the joints?

Move the joints into place?

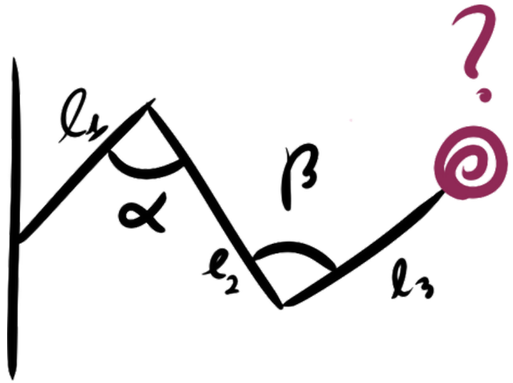
Creates Lovecraftian eldritch horrors beyond our human comprehension.

We clearly need a better strategy. Tipically joints are rotated, not moved.



So, how to **rotate** the joints?

[Forward] Kinematics studies how an object moves over time without forces acting on it.



Inverse Kinematics is the process of calculating joint parameters to position the end of a kinematic chain.

➔ With an **Inverse Kinematics solver** (like **CCD** or **FABRIK**) we can approximate the joint rotations and animate the model.

Future Work

Provide better joint constraints

Make the model move in a more natural way by better constraining joints.

Texturing

Figure out a way to extract a "best effort" texture from a photo and UV-match it to the model.

Style Transfer

Other than attach horns, tails and other fantasy objects, we could actually change the face shape altogether.

Recap

Generating 3D model

PIFUhd

Allows to reconstruct 3D meshes of the subject with just a single image.

Generating Joints

RigNet

Predicts a skeleton that matches the animator expectations in joint placement and topology

Rigging

bpy (Blender)

We used python scripting in Blender to apply the skeleton to the 3D model and to refine its topology

Animating

Panda 3D

We used panda to render the final model with the animation.

Result