GaussianAvatar: Towards Realistic Human Avatar Modeling from a Single Video via Animatable 3D Gaussians

Reviewer: Victor Ferrari

Archeologist: Leonardo Nanci

Hacker: Horácio Macedo

PhD Student: Alberto Arkader Kopiler

GaussianAvatar: Towards Realistic Human Avatar Modeling from a Single Video via Animatable 3D Gaussians

Reviewer: Victor Ferrari

Method

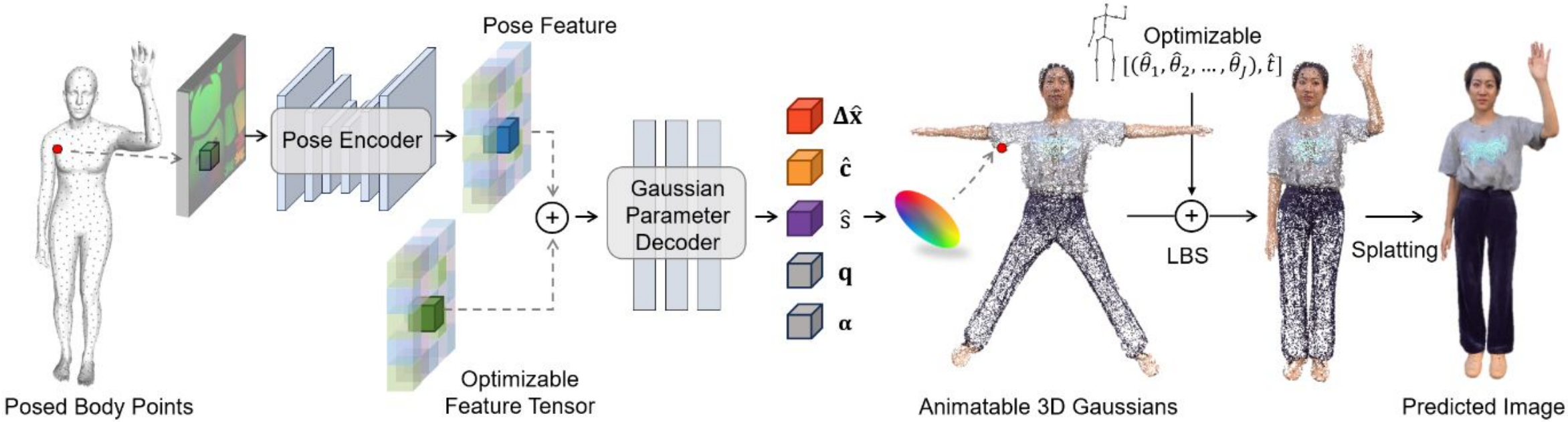


Illustration of the proposed method. Our method learns motion-to-appearance mapping by a dynamic appearance network and an optimizable feature tensor. The predicted point offsets, colors, and scales with fixed rotations and opacity constitute the animatable 3D Gaussians in canonical space. Following this, the 3D Gaussians undergo deformation into motion space via Linear Blend Skinning (LBS) and are subsequently rendered as images.

Objetivo do paper

- Criação de avatares humanos realistas com aparências 3D dinâmicas a partir de um único vídeo.
  - Introdução de gausianas 3D animáveis para criar avatares humanos realistas a partir de um único vídeo, **representando explicitamente as superfícies** humanas e fundindo as aparências 3D de forma eficiente e consistente a partir de observações 2D.
  - Augmentação das Gausianas 3D com **propriedades dinâmicas**, suportando a **modelagem da aparência dependente da pose**, utilizando uma rede de aparência dinâmica e um tensor de características otimizável para aprender o mapeamento de movimento para aparência.
  - Proposição de otimização conjunta do movimento e da aparência durante a modelagem do avatar, permitindo **corrigir desalinhamentos** iniciais de movimento e **melhorar a qualidade final da aparência**.

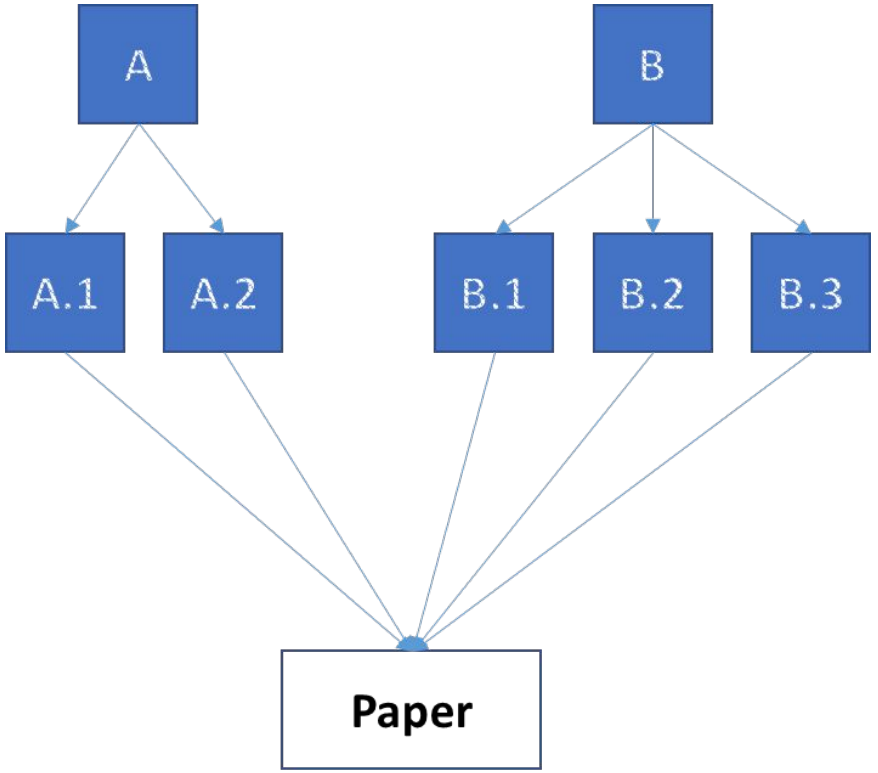
Trabalhos Relacionados

A

Neural Rendering for Human Reconstruction

B

Avatar Modeling from Monocular Videos



Trabalhos Relacionados

A.1

Neural Rendering for Human Reconstruction (Implicit)

20, 21, 24, 32, 33, 47, 67, 68

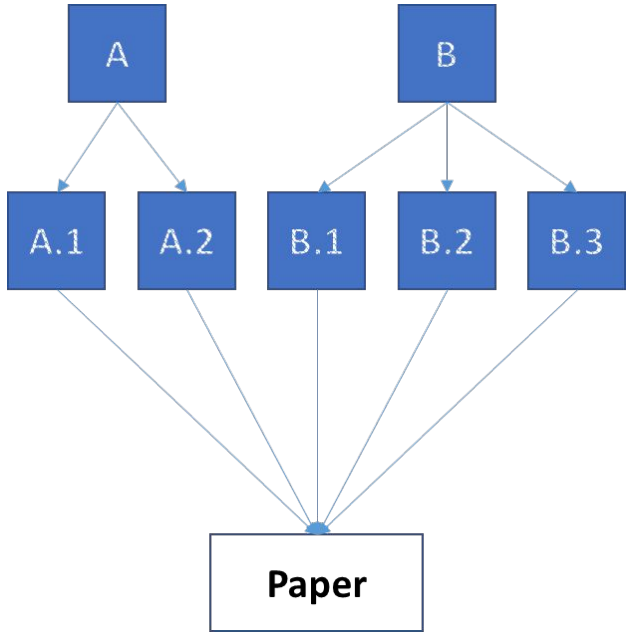
- Métodos baseados em NeRF têm demonstrado resultados atraentes de renderização na reconstrução de avatares humanos
- Dificuldade para **representar superfícies humanas** com o volume 3D implícito.

A.2

Neural Rendering for Human Reconstruction (Explicit)

20, 21, 24, 32, 33, 47, 67, 68

- Potencial significativo das representações explícitas e esse ponto é **pouco explorado**.



Trabalhos Relacionados

B.1

Regression-based methods

10, 11, 13, 38, 39, 51, 52, 66

- Produz resultados atrativos
- Não consegue recuperar a **aparência dinâmica** na reconstrução ao longo de toda a sequência de imagens.

B.2

Pre-scanned rigged templates

7, 8, 54, 1, 5, 30

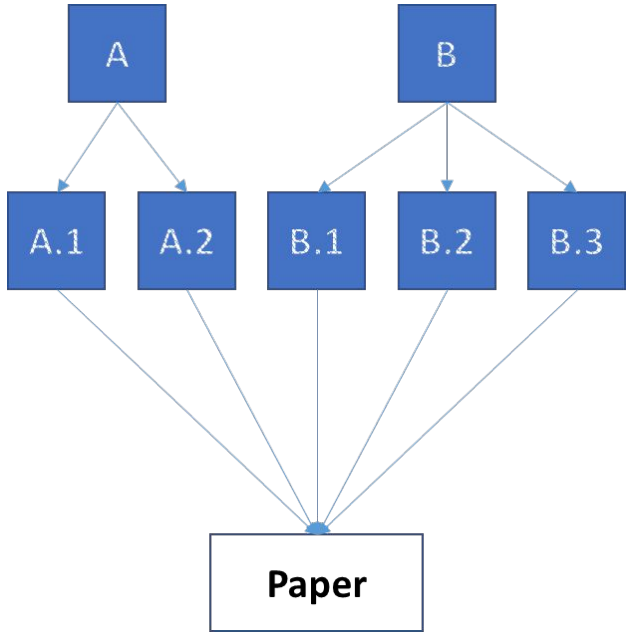
- Impedem suas aplicações no mundo real.
- Mesmo alguns tentando contornar a necessidade de modelos humanos predefinidos, ainda enfrentam desafios em preservar detalhes finos devido à **resolução fixa** da malha.

B.3

Neural rendering for reconstruction from video

3, 14, 16, 41, 43, 49

- Tentar resolver o problema da estimativa de pose imprecisa com **modelos implícitos** de NeRF para humanos tem se mostrado ineficaz e impreciso.

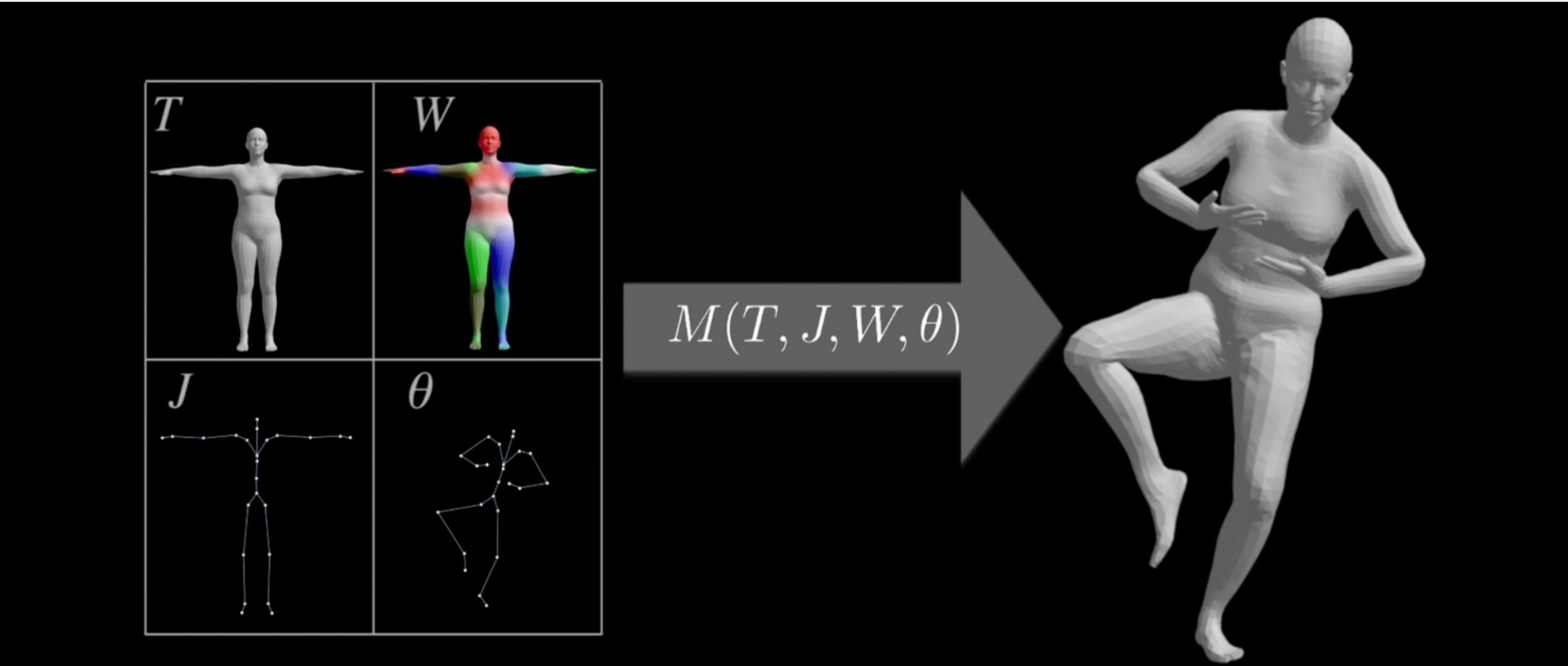


GaussianAvatar: Towards Realistic Human Avatar Modeling from a Single Video via Animatable 3D Gaussians

Archeologist : Leonardo Nanci

Trabalhos anteriores

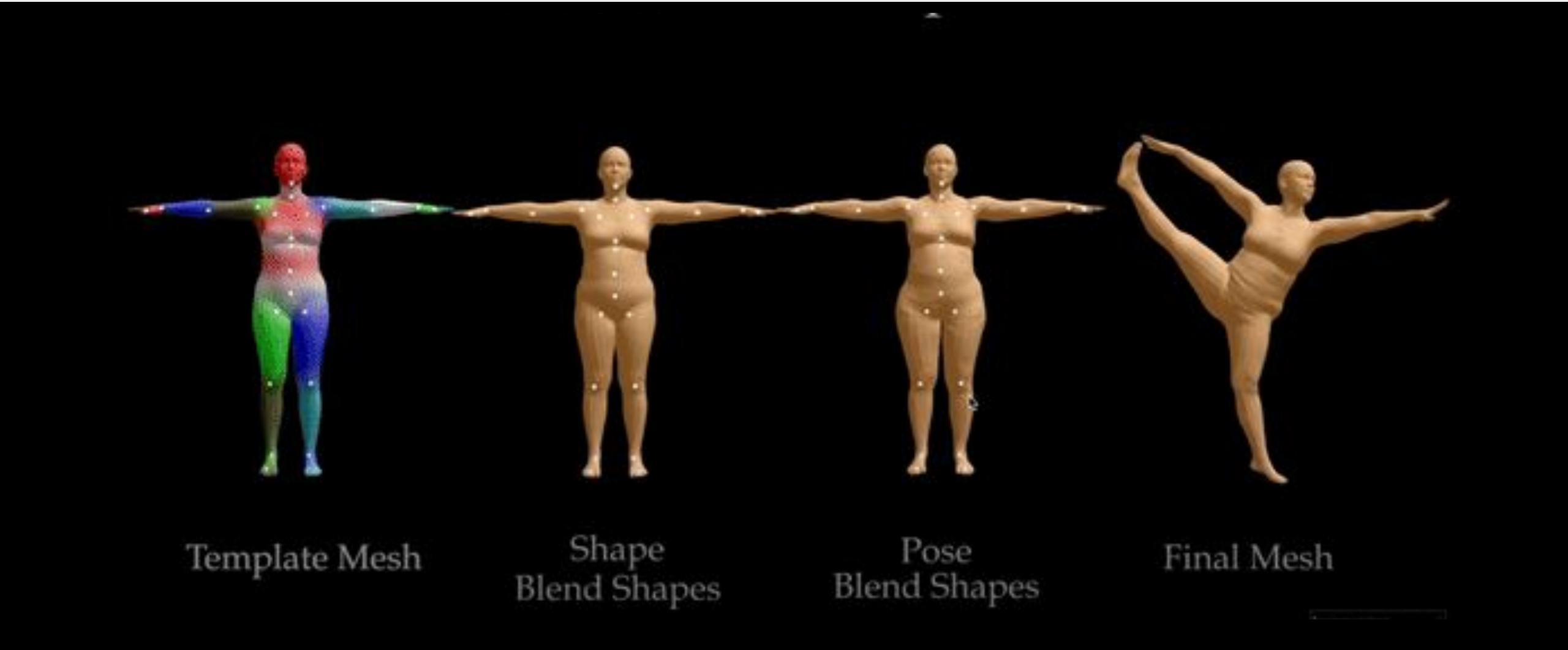
Standard Skinning



SMPL: A Skinned Multi-Person Linear Model

Matthew Loper [*12] Naureen Mahmood[†1] Javier Romero[†1] Gerard Pons-Moll[†1] Michael J. Black[†1]

[1]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[2]Industrial Light and Magic, San Francisco, CA



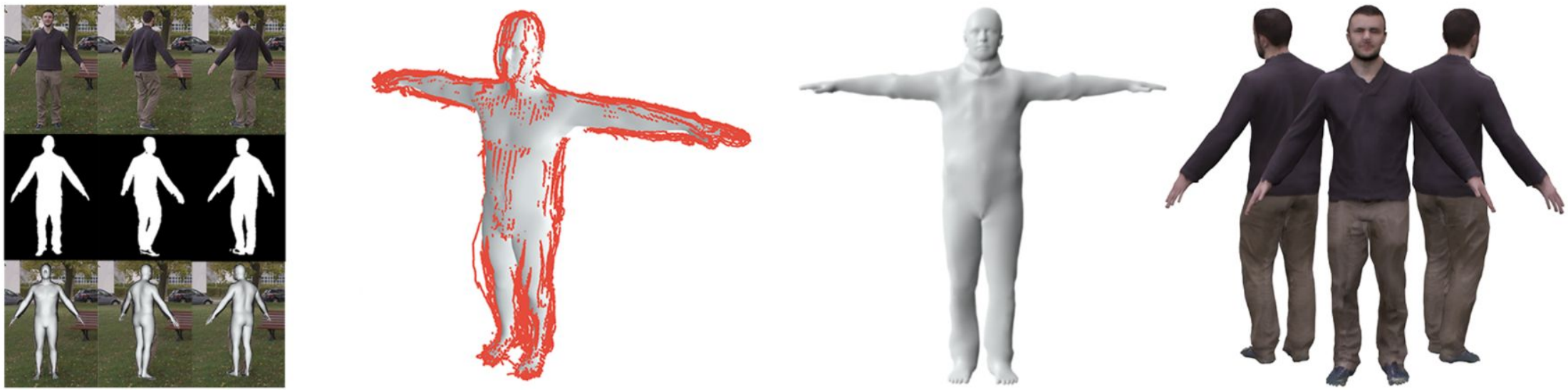
Expressive Body Capture: 3D Hands, Face, and Body from a Single Image

Georgios Pavlakos[*1,2], Vasileios Choutas[*1], Nima Ghorbani[1], Timo Bolkart[1], Ahmed A. A. Osman[1], Dimitrios Tzionas[1], and Michael J. Black[1]

Figure 2: We learn a new 3D model of the human body called *SMPL-X* that jointly models the human body, face and hands. We fit the female *SMPL-X* model with *SMPLify-X* to single RGB images and show that it captures a rich variety of *natural* and *expressive* 3D human poses, gestures and facial expressions.

Video Based Reconstruction of 3D People Models

Thiemo Alldieck[1] Marcus Magnor[1] Weipeng Xu[2] Christian Theobalt[2] Gerard Pons-Moll[2]



a) b) c) d)

Figure 2. Overview of our method. The input to our method is an image sequence with corresponding segmentations. We first calculate poses using the SMPL model (a). Then we unpose silhouette camera rays (unposed silhouettes depicted in red) (b) and optimize for the subjects shape in the canonical T-pose (c). Finally, we are able to calculate a texture and generate a personalized blend shape model (d).

Trabalhos Contemporâneos

Animatable Gaussians: Learning Pose-dependent Gaussian Maps for High-fidelity Human Avatar Modeling

Zhe Li[1], Zerong Zheng[2], Lizhen Wang[1], Yebin Liu[1]

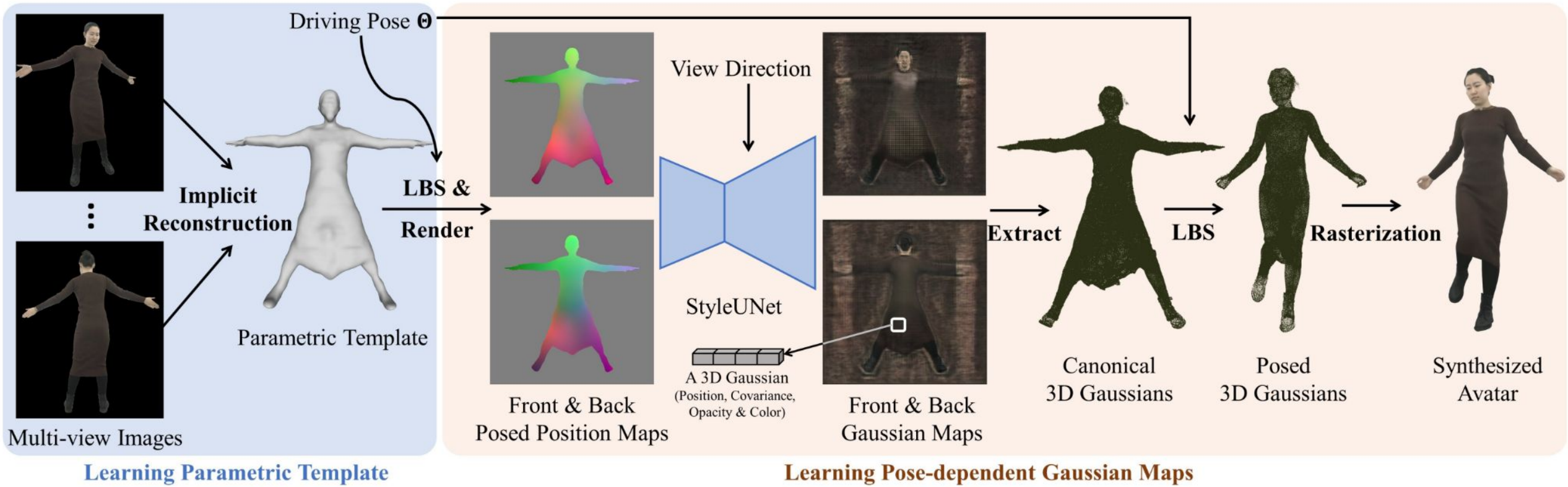
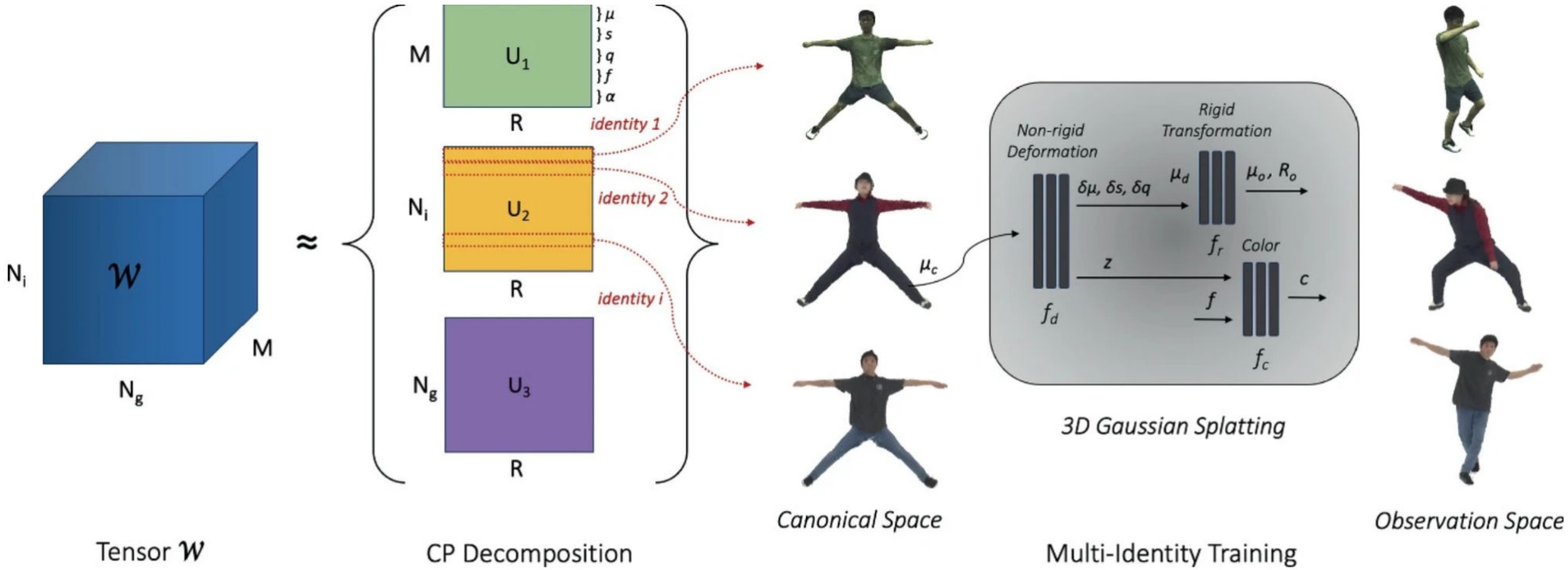


Figure 2. **Illustration of the pipeline.** It contains two main steps: 1) Reconstruct a character-specific template from multi-view images. 2) Predict pose-dependent Gaussian maps through the StyleUNet, and render the synthesized avatar by LBS and differentiable rasterization.

MIGS: Multi-Identity Gaussian Splatting via Tensor Decomposition

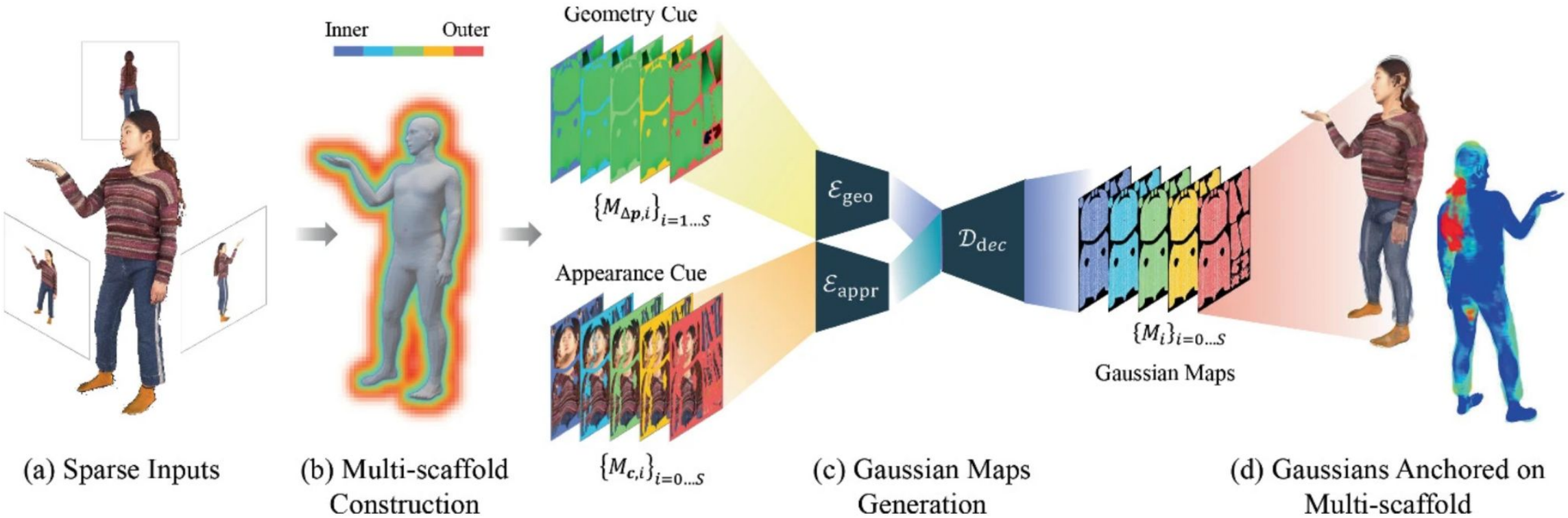
Aggelina Chatziagapi[1], Grigorios G. Chrysos[2], and Dimitris Samaras[1]



Overview of MIGS. Given monocular videos of multiple identities, we learn a unified 3DGS representation for human avatars based on CP tensor decomposition. We construct a tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{N_i \times N_g \times M}$, where N_i is the number of identities, N_g the number of 3D Gaussians and M the number of parameters per Gaussian. In practice, we assume a low-rank structure of the tensor $\boldsymbol{\mathcal{W}}$ and thus, we only learn the matrices $\boldsymbol{U}_1 \in \mathbb{R}^{M \times R}$, $\boldsymbol{U}_2 \in \mathbb{R}^{N_i \times R}$, $\boldsymbol{U}_3 \in \mathbb{R}^{N_g \times R}$ that approximate $\boldsymbol{\mathcal{W}}$ through the CP decomposition with $R << N_g$. By leveraging information from the diverse deformations of different subjects, MIGS enables robust animation under novel challenging poses.

Generalizable Human Gaussians for Sparse View Synthesis

Youngjoong Kwon[1], Baole Fang[1]*, Yixing Lu[1]*, Haoye Dong[1], Cheng Zhang[1],
Francisco Vicente Carrasco[1], Albert Mosella-Montoro[1], Jianjin Xu[1],
Shingo Takagi[2], Daeil Kim[2], Aayush Prakash[2], and Fernando De la Torre[1]



Overview of GHG. (a) We focus on generalizable human rendering under very sparse view setting. (b) We first construct the multi-scaffolds by dilating the human template surface. The 2D UV space of each scaffold serves to collect the geometry and appearance information from the corresponding 3D locations. (c) The aggregated multi-scaffold input is fed into the network, which generates multi-Gaussian parameter maps. (d) Finally, Gaussians are anchored on the corresponding surface of each scaffold, and rasterized into novel views.

GPS-Gaussian: Generalizable Pixel-wise 3D Gaussian Splatting for Real-time Human Novel View Synthesis

Shunyuan Zheng[†,1], Boyao Zhou[2], Ruizhi Shao[2], Boning Liu[2], Shengping Zhang[*,1,3], Liqiang Nie[1], Yebin Liu[2]

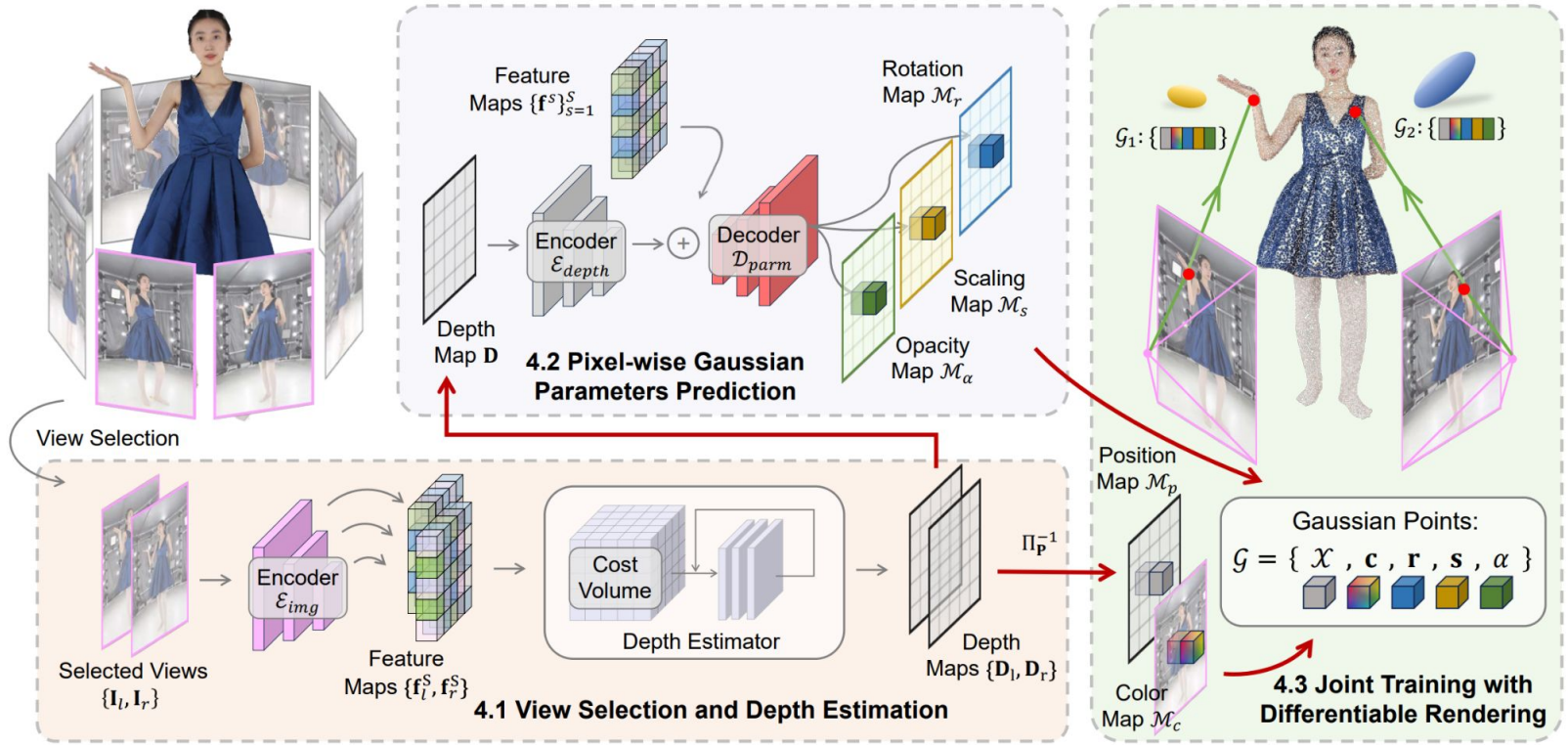


Figure 2. **Overview of GPS-Gaussian.** Given RGB images of a human-centered scene with sparse camera views and a target novel viewpoint, we select the adjacent two views on which to formulate our Gaussian representation. We extract the image features followed by conducting an iterative depth estimation. For each source view, the depth map and the RGB image serve as a 3D position map and a color map, respectively, to formulate the Gaussian representation while the other parameters of 3D Gaussians are predicted in a pixel-wise manner. The Gaussian parameter maps defined on 2D image planes of both views are further unprojected to 3D space and aggregated for novel view rendering. The fully differentiable framework enables a joint training mechanism for all networks.

GaussianAvatar: Towards Realistic Human Avatar Modeling from a Single Video via Animatable 3D Gaussians

Hacker: Horácio Macedo

Reprodutibilidade do GaussianAvatar

- Frágil
  - O código está cheio de erros
- Readme problemático
  - As instruções são bem simplórias, em algumas seções incompletas
- Mistura estranha de outros trabalhos
  - 3D Gaussian Splatting (https://github.com/graphdeco-inria/gaussian-splatting)
  - POP (https://github.com/qianlim/POP)
  - HumanNeRF (https://github.com/chungyiweng/humannerf)
  - InstantAvatar (https://github.com/tijiang13/InstantAvatar)

Reprodutibilidade do GaussianAvatar

- Datasets providenciados:
  - m4c_processed: é *straightforward* (já está pré-processado para treino)
  - neuman_bike: em tese está pronto para treino, smpl_parms.pth precisa de reparos? Ele não funciona!
  - dynvideo_female, dynvideo_male: precisa de pré-processamento dos datasets
  - Scripts para pré-processamento não funcionam (e são de outro trabalho)

Reprodutibilidade do GaussianAvatar

- O environment.yml está incompleto
  - Uso de bibliotecas defasadas que dão trabalho para encontrar a versão correta
  - Rotina de pré-processamento exige baixar aplicações que não são listadas no próprio repositório
  - PyOpenGl está *quebrado* no Windows há algumas versões
    - Versões antigas que em tese funcionam são problemáticas de instalar
    - O conserto para isso consistiu em achar um .dll do *freeglut* na internet para colocar no $PATH

Falando algo de bom sobre a reprodutibilidade do GaussianAvatar

- Qualidade boa do render final
- Scripts de render e avaliação autorais funcionam
  - É só não fazer nada que o readme não diga pra fazer
- Recomendaram 24GB de VRAM, dá pra rodar tendo 12GB de VRAM
  - Só precisei diminuir o batch size para o valor de 1

O código do GaussianAvatar

To extend this representation for human avatar modeling, we integrate it with either the SMPL [26] or SMPL-X [31] model as follows:

$$G(\beta, \theta, \mathbf{D}, \mathbf{P}) = Splatting(W(\mathbf{D}, J(\beta), \theta, \omega), \mathbf{P}), \quad (1)$$

where $G(\cdot)$ represents a rendered image, and $Splatting(\cdot)$ denotes the rendering process of 3D Gaussians from any viewpoint, $W(\cdot)$ is a standard linear blend skinning function employed for reposing 3D Gaussians, $\mathbf{D} = T(\beta) + dT$ represents the locations of 3D Gaussians in canonical space, formed by adding corrective point displacements dT on the template mesh surface $T(\beta)$, $\mathbf{P}$ denotes the remaining properties of 3D Gaussians, excluding the positions. β and θ are the shape and pose parameters, $J(\beta)$ outputs 3D joint locations. Note that we propagate the skinning weight ω from the vertices of the SMPL or SMPL-X model to the nearest 3D Gaussians. With the proposed representation, we can now repose these canonical 3D Gaussians to the motion space for free-view rendering.

```python
class AvatarModel:
    def __init__(self, model_parms, net_parms, opt_parms,
    load_iteration=None, train=True):

        self.model_parms = model_parms
        self.net_parms = net_parms
        self.opt_parms = opt_parms
        self.model_path = model_parms.model_path
        self.loaded_iter = None
        self.train = train
        self.train_mode = model_parms.train_mode
        self.gender = self.model_parms.smpl_gender

        if train:
            self.batch_size = self.model_parms.batch_size
        else:
            self.batch_size = 1

        if train:
            split = 'train'
        else:
            split = 'test'

        self.train_dataset  = MonoDataset_train(model_parms)
        self.smpl_data = self.train_dataset.smpl_data
```

Estratégia de Treino do GaussianAvatar

3.4. Training Strategy

In this section, we outline our approach to training the network with inaccurate human motions. We conduct a two-stage optimization process using different loss functions. In

Estágio Um

stage optimization process using different loss functions. In the first stage, we aim to fuse the sequential appearances to the optimizable feature tensor and conduct motion optimization to get accurate poses for the dynamic appearance network. In this stage, we optimize the framework without incorporating any pose-dependent information by excluding the training of the pose encoder. Specifically, we utilize the following loss functions to train our network:

$$\begin{aligned}\mathcal{L}_{stage_1} =& \lambda_{rbg}\mathcal{L}_{rbg} + \lambda_{ssim}\mathcal{L}_{ssim} + \lambda_{lpips}\mathcal{L}_{lpips} \\ &+ \lambda_f\mathcal{L}_f + \lambda_{offset}\mathcal{L}_{offset} + \lambda_{scale}\mathcal{L}_{scale},\end{aligned} \tag{4}$$

where $\mathcal{L}_{rbg}$, $\mathcal{L}_{ssim}$, and $\mathcal{L}_{lpips}$ are the L1 loss, SSIM loss [48], and LPIPS loss [61], respectively. $\mathcal{L}_f$, $\mathcal{L}_{offset}$, $\mathcal{L}_{scale}$ calculate the L2-norm of the feature map, predicted offsets and scales, respectively. We set $\lambda_{rbg} = 0.8$, $\lambda_{ssim} = 0.2$, $\lambda_{lpips} = 0.2$, $\lambda_f = 1$, $\lambda_{offset} = 10$, $\lambda_{scale} = 1$.

```
for _, batch_data in enumerate(train_loader):

    first_iter += 1
    batch_data = to_cuda(batch_data, device=torch.device('cuda:0'))
    gt_image = batch_data['original_image']

    if model.train_stage ==1:
        image, points, offset_loss, geo_loss, scale_loss = avatarmodel.
        train_stage1(batch_data, first_iter)
        scale_loss = opt.lambda_scale  * scale_loss
        offset_loss = wdecay_rgl * offset_loss

        Ll1 = (1.0 - opt.lambda_dssim) * l1_loss_w(image, gt_image)
        ssim_loss = opt.lambda_dssim * (1.0 - ssim(image, gt_image))

        loss = scale_loss + offset_loss + Ll1 + ssim_loss + geo_loss
    if epoch > opt.lpips_start_iter:
        vgg_loss = opt.lambda_lpips * loss_fn_vgg((image-0.5)*2, (gt_image- 0.5)
        *2).mean()
        loss = loss + vgg_loss

    avatarmodel.zero_grad(epoch)

    loss.backward(retain_graph=True)
    iter_end.record()
    avatarmodel.step(epoch)
```

Estágio Um

-

```
(gs-avatar) PS E:\trabalhos\GaussianAvatar> python train.py -s .\gs-data\m4c_processed -m .\output\m4c_processed_1 --bat
ch_size 1 --stage1_out_path .\output\m4c_processed_1\stage1 --train_stage 1
Optimizing .\output\m4c_processed_1
Output folder: .\output\m4c_processed_1 [27/10 23:16:06]
loading smpl data  E:\trabalhos\GaussianAvatar\gs-data\m4c_processed\train\smpl_parms.pth [27/10 23:16:06]
total pose length 110 [27/10 23:16:06]
inv_mat shape:  torch.Size([1, 24, 4, 4]) [27/10 23:16:09]
Setting up [LPIPS] perceptual loss: trunk [alex], v[0.1], spatial [off] [27/10 23:16:09]
C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torchvision\models\_utils.py:208: UserWarning: The parameter
'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torchvision\models\_utils.py:223: UserWarning: Arguments othe
r than a weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. The current beh
avior is equivalent to passing `weights=AlexNet_Weights.IMAGENET1K_V1`. You can also use `weights=AlexNet_Weights.DEFAUL
T` to get the most up-to-date weights.
  warnings.warn(msg)
Loading model from: C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\lpips\weights\v0.1\alex.pth [27/10 23:16:
09]
Training progress:  60%|                                  | 13200/22000 [52:30<28:33,  5.14it/s, Loss=0.0184575]
[Epoch 120] Saving Model [28/10 00:08:41]
Training progress:  75%|                                | 16500/22000 [1:05:26<17:50,  5.14it/s, Loss=0.0195978]
[Epoch 150] Saving Model [28/10 00:21:36]
Training progress:  90%|                             | 19800/22000 [1:18:24<07:15,  5.05it/s, Loss=0.0174267]
[Epoch 180] Saving Model [28/10 00:34:34]
Training progress: 100%|                           | 22000/22000 [1:27:01<00:00,  4.21it/s, Loss=0.0169756]

Training complete. [28/10 00:43:11]
(gs-avatar) PS E:\trabalhos\GaussianAvatar>
```

Avaliando o Estágio Um

```
(gs-avatar) PS E:\trabalhos\gaussianavatar> python .\eval.py -s .\gs-data\m4c_processed -m .\output\m4c_processed_1 --ep
och 180
Looking for config file in .\output\m4c_processed_1\cfg_args
Config file found: .\output\m4c_processed_1\cfg_args
Rendering .\output\m4c_processed_1
C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torchmetrics\utilities\prints.py:62: FutureWarning: Importing
 `PeakSignalNoiseRatio` from `torchmetrics` was deprecated and will be removed in 2.0. Import `PeakSignalNoiseRatio` fro
m `torchmetrics.image` instead.
  _future_warning(
C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torchmetrics\utilities\prints.py:62: FutureWarning: Importing
 `StructuralSimilarityIndexMeasure` from `torchmetrics` was deprecated and will be removed in 2.0. Import `StructuralSim
ilarityIndexMeasure` from `torchmetrics.image` instead.
  _future_warning(
loading smpl data  E:\trabalhos\gaussianavatar\gs-data\m4c_processed\train\smpl_parms.pth [12/11 22:31:58]
total pose length 110 [12/11 22:31:58]
E:\trabalhos\gaussianavatar\gs-data\m4c_processed\test\smpl_cano_joint_mat.pth [12/11 22:32:01]
inv_mat shape:  torch.Size([1, 24, 4, 4]) [12/11 22:32:01]
load pth:  .\output\m4c_processed_1\net/iteration_180\net.pth [12/11 22:32:01]
loading smpl data  E:\trabalhos\gaussianavatar\gs-data\m4c_processed\test\smpl_parms.pth [12/11 22:32:01]
total pose length 36 [12/11 22:32:01]
Rendering progress: 100%|                                                                | 36/36 [00:17<00:00,  2.03it/s]
PSNR: 28.51 [12/11 22:32:19]
SSIM: 0.9735 [12/11 22:32:19]
LPIPS: 0.0288 [12/11 22:32:19]
save video... [12/11 22:32:19]
(gs-avatar) PS E:\trabalhos\gaussianavatar> _
```

Avaliando o Estágio Um

Ground truth

Render

Avaliando o Estágio Um

```
(gs-avatar) PS E:\trabalhos\gaussianavatar> python .\render_novel_pose.py -s .\gs-data\m4c_processed -m .\output\m4c_pro
cessed_1 --epoch 180
Looking for config file in .\output\m4c_processed_1\cfg_args
Config file found: .\output\m4c_processed_1\cfg_args
Rendering .\output\m4c_processed_1
loading smpl data  E:\trabalhos\gaussianavatar\gs-data\m4c_processed\train\smpl_parms.pth [12/11 22:58:03]
total pose length 110 [12/11 22:58:03]
E:\trabalhos\gaussianavatar\gs-data\m4c_processed\test\smpl_cano_joint_mat.pth [12/11 22:58:06]
inv_mat shape:  torch.Size([1, 24, 4, 4]) [12/11 22:58:06]
load pth:  .\output\m4c_processed_1\net/iteration_180\net.pth [12/11 22:58:06]
loading smpl data  E:\trabalhos\GaussianAvatar/assets/test_pose\smpl_parms.pth [12/11 22:58:06]
total pose length 480 [12/11 22:58:06]
novel pose shape torch.Size([480, 72]) [12/11 22:58:06]
novel pose shape torch.Size([480, 3]) [12/11 22:58:06]
Rendering progress: 100%|                                                                | 480/480 [01:19<00:00,  6.01it/s]
(gs-avatar) PS E:\trabalhos\gaussianavatar> _
```

Avaliando o Estágio Um

Render (180 epochs)

Estágio Dois

the training of the pose encoder. Specifically, we utilize the following loss functions to train our network:

$$\begin{aligned}\mathcal{L}_{stage_1} =& \lambda_{rbg}\mathcal{L}_{rbg} + \lambda_{ssim}\mathcal{L}_{ssim} + \lambda_{lpips}\mathcal{L}_{lpips} \\ &+ \lambda_f\mathcal{L}_f + \lambda_{offset}\mathcal{L}_{offset} + \lambda_{scale}\mathcal{L}_{scale},\end{aligned} \tag{4}$$

where $\mathcal{L}_{rbg}$, $\mathcal{L}_{ssim}$, and $\mathcal{L}_{lpips}$ are the L1 loss, SSIM loss [48], and LPIPS loss [61], respectively. $\mathcal{L}_f$, $\mathcal{L}_{offset}$, $\mathcal{L}_{scale}$ calculate the L2-norm of the feature map, predicted offsets and scales, respectively. We set $\lambda_{rbg} = 0.8$, $\lambda_{ssim} = 0.2$, $\lambda_{lpips} = 0.2$, $\lambda_f = 1$, $\lambda_{offset} = 10$, $\lambda_{scale} = 1$.

After the first stage of training, we obtain more accurate human motions and an optimized feature tensor F. The optimized feature tensor F captures a coarse appearance of human avatars. In the second stage, we incorporate the pose features encoded by the pose encoder with the trained feature tensor F. We replace $\mathcal{L}_f$ with the L2-norm loss $\mathcal{L}_p$, which plays the same role as $\mathcal{L}_f$ in regularizing the limited pose space. By penalizing the pose-dependent features, we can eliminate the strong bias of limited training poses and thus generalize to unseen viewpoints and poses.

```
else:
    image, points, pose_loss, offset_loss, = avatarmodel.train_stage2
    (batch_data, first_iter)

    offset_loss = wdecay_rgl * offset_loss

    Ll1 = (1.0 - opt.lambda_dssim) * l1_loss_w(image, gt_image)
    ssim_loss = opt.lambda_dssim * (1.0 - ssim(image, gt_image))

    loss =  offset_loss + Ll1 + ssim_loss + pose_loss * 10
```

```
if epoch > opt.lpips_start_iter:
    vgg_loss = opt.lambda_lpips * loss_fn_vgg((image-0.5)*2, (gt_image- 0.5)
    *2).mean()
    loss = loss + vgg_loss

avatarmodel.zero_grad(epoch)

loss.backward(retain_graph=True)
iter_end.record()
avatarmodel.step(epoch)
```

Pré-processamento do Estágio Dois

- Instruções para rodar o segundo estágio precisam que um script específico funcione (não funcionou)

```
(gs-avatar) PS E:\trabalhos\GaussianAvatar> cd .\scripts\; python .\gen_pose_map_our_smpl.py
saving obj...
frame_num 110
saving pose_map 128 ...
Traceback (most recent call last):
  File "E:\trabalhos\GaussianAvatar\scripts\gen_pose_map_our_smpl.py", line 124, in <module>
    save_npz(smpl_parm_path, 128)
  File "E:\trabalhos\GaussianAvatar\scripts\gen_pose_map_our_smpl.py", line 100, in save_npz
    posmap128, _, _ = render_posmap(body_mesh.vertices, body_mesh.faces, uvs, faces_uvs, img_size=128)
  File "E:\trabalhos\GaussianAvatar\scripts\gen_pose_map_our_smpl.py", line 26, in render_posmap
    rndr = PosRender(width=img_size, height=img_size)
  File "E:\trabalhos\GaussianAvatar\scripts\posmap_generator\lib\renderer\gl\pos_render.py", line 9, in __init__
    CamRender.__init__(self, width, height, name, program_files=['pos_uv.vs', 'pos_uv.fs'])
  File "E:\trabalhos\GaussianAvatar\scripts\posmap_generator\lib\renderer\gl\cam_render.py", line 9, in __init__
    Render.__init__(self, width, height, name, program_files, color_size)
  File "E:\trabalhos\GaussianAvatar\scripts\posmap_generator\lib\renderer\gl\render.py", line 18, in __init__
    glutInit()
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\OpenGL\GLUT\special.py", line 333, in glutInit
    _base_glutInit( ctypes.byref(count), holder )
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\OpenGL\platform\baseplatform.py", line 405, in __call
__
    raise error.NullFunctionError(
OpenGL.error.NullFunctionError: Attempt to call an undefined function glutInit, check for bool(glutInit) before calling
```

Pré-processamento do Estágio Dois

- Exige versão antiga do PyOpenGL (não instalou no Windows)
- A solução foi baixar o freeglut.dll de um repositório aleatório e colocar no $path

Estágio Dois

- Depois de resolver esse problema no Windows, ele continua… não funcionando.

```
raceback (most recent call last):
  File "E:\trabalhos\GaussianAvatar\train.py", line 179, in <module>
    train(lp.extract(args), np.extract(args), op.extract(args), args.save_epochs, args.checkpoint_epochs)
  File "E:\trabalhos\GaussianAvatar\train.py", line 119, in train
    tb_writer.add_scalar('train_loss_patches/scale_loss', scale_loss.item(), first_iter)
UnboundLocalError: local variable 'scale_loss' referenced before assignment
Training progress:   0%|                                                                    | 0/22000 [00:06<?, ?it/s]
(gs-avatar) PS E:\trabalhos\GaussianAvatar> _
```

- Neste momento eu tive a realização de que as funções de Loss estão codificadas erradas?????

Consertando o Estágio Dois? O que é esse *10 na loss?!

```
else:
    image, points, pose_loss, offset_loss, = avatarmodel.train_stage2
    (batch_data, first_iter)

    offset_loss = wdecay_rgl * offset_loss

    Ll1 = (1.0 - opt.lambda_dssim) * l1_loss_w(image, gt_image)
    ssim_loss = opt.lambda_dssim * (1.0 - ssim(image, gt_image))

    loss =  offset_loss + Ll1 + ssim_loss + pose_loss * 10
```

```
if epoch > opt.lpips_start_iter:
    vgg_loss = opt.lambda_lpips * loss_fn_vgg((image-0.5)*2, (gt_image- 0.5)
    *2).mean()
    loss = loss + vgg_loss

avatarmodel.zero_grad(epoch)

loss.backward(retain_graph=True)
iter_end.record()
avatarmodel.step(epoch)
```

```
else:
    image, points, pose_loss, offset_loss, = avatarmodel.train_stage2(batch_data,
    first_iter)
    scale_loss = 10 * opt.lambda_scale
    offset_loss = wdecay_rgl * offset_loss

    Ll1 = (1.0 - opt.lambda_dssim) * l1_loss_w(image, gt_image)
    ssim_loss = opt.lambda_dssim * (1.0 - ssim(image, gt_image))

    loss =  offset_loss + Ll1 + ssim_loss + pose_loss * scale_loss


if epoch > opt.lpips_start_iter:
    vgg_loss = opt.lambda_lpips * loss_fn_vgg((image-0.5)*2, (gt_image- 0.5)*2).mean()
    loss = loss + vgg_loss

avatarmodel.zero_grad(epoch)

loss.backward(retain_graph=True)
iter_end.record()
avatarmodel.step(epoch)
```

Avaliando o Estágio Dois

```
Rendering .\output\m4c_processed_1_stage2
C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torchmetrics\utilities\prints.py:62: FutureWarning: Importing
 `PeakSignalNoiseRatio` from `torchmetrics` was deprecated and will be removed in 2.0. Import `PeakSignalNoiseRatio` fro
m `torchmetrics.image` instead.
  _future_warning(
C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torchmetrics\utilities\prints.py:62: FutureWarning: Importing
 `StructuralSimilarityIndexMeasure` from `torchmetrics` was deprecated and will be removed in 2.0. Import `StructuralSim
ilarityIndexMeasure` from `torchmetrics.image` instead.
  _future_warning(
loading smpl data  E:\trabalhos\gaussianavatar\gs-data\m4c_processed\train\smpl_parms_pred.pth [12/11 23:14:38]
total pose length 110 [12/11 23:14:38]
E:\trabalhos\gaussianavatar\gs-data\m4c_processed\test\smpl_cano_joint_mat.pth [12/11 23:14:40]
inv_mat shape:  torch.Size([1, 24, 4, 4]) [12/11 23:14:41]
load pth:  .\output\m4c_processed_1_stage2\net/iteration_180\net.pth [12/11 23:14:42]
loading smpl data  E:\trabalhos\gaussianavatar\gs-data\m4c_processed\test\smpl_parms_pred.pth [12/11 23:14:42]
Traceback (most recent call last):
  File "E:\trabalhos\gaussianavatar\eval.py", line 103, in <module>
    render_sets(model.extract(args), network.extract(args), op.extract(args), args.epoch,)
  File "E:\trabalhos\gaussianavatar\eval.py", line 47, in render_sets
    test_dataset = avatarmodel.getTestDataset()
  File "E:\trabalhos\gaussianavatar\model\avatar_model.py", line 250, in getTestDataset
    self.test_dataset = MonoDataset_test(self.model_parms)
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\utils\_contextlib.py", line 115, in decorate_co
ntext
    return func(*args, **kwargs)
  File "E:\trabalhos\gaussianavatar\scene\dataset_mono.py", line 283, in __init__
    self.smpl_data = torch.load(join(self.data_folder, 'smpl_parms_pred.pth'))
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\serializ
    with _open_file_like(f, 'rb') as opened_file:
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\serializ
    return _open_file(name_or_buffer, mode)
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\serializ
    super().__init__(open(name, mode))
FileNotFoundError: [Errno 2] No such file or directory: 'E:\\trabalhos\\gaussiana
pl_parms_pred.pth'
(gs-avatar) PS E:\trabalhos\gaussianavatar>
```

```
Traceback (most recent call last):
  File "E:\trabalhos\gaussianavatar\render_novel_pose.py", line 48, in <module>
    render_sets(model.extract(args), network.extract(args), op.extract(args), args.epoch,)
  File "E:\trabalhos\gaussianavatar\render_novel_pose.py", line 26, in render_sets
    for idx, batch_data in enumerate(tqdm(novel_pose_loader, desc="Rendering progress")):
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\tqdm\std.py", line 1181, in __iter__
    for obj in iterable:
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\utils\data\dataloader.py", line 633, in __next__
    data = self._next_data()
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\utils\data\dataloader.py", line 1345, in _next_data
    return self._process_data(data)
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\utils\data\dataloader.py", line 1371, in _process_data
    data.reraise()
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\_utils.py", line 644, in reraise
    raise exception
FileNotFoundError: Caught FileNotFoundError in DataLoader worker process 2.
Original Traceback (most recent call last):
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\utils\data\_utils\worker.py", line 308, in _worker_loop
    data = fetcher.fetch(index)
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\utils\data\_utils\fetch.py", line 51, in fetch
    data = [self.dataset[idx] for idx in possibly_batched_index]
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\utils\data\_utils\fetch.py", line 51, in <listcomp>
    data = [self.dataset[idx] for idx in possibly_batched_index]
  File "E:\trabalhos\gaussianavatar\scene\dataset_mono.py", line 471, in __getitem__
    return self.getitem(index, ignore_list)
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\torch\utils\_contextlib.py", line 115, in decorate_context
    return func(*args, **kwargs)
  File "E:\trabalhos\gaussianavatar\scene\dataset_mono.py", line 479, in getitem
    inp_posmap = np.load(inp_posmap_path)['posmap' + str(self.dataset_parms.inp_posmap_size)]
  File "C:\Users\horac\miniconda3\envs\gs-avatar\lib\site-packages\numpy\lib\npyio.py", line 390, in load
    fid = stack.enter_context(open(os_fspath(file), "rb"))
FileNotFoundError: [Errno 2] No such file or directory: 'E:\\trabalhos\\GaussianAvatar/assets/test_pose/inp_map/inp_posemap_128_00000110.npz'
```

“Ah, vou pré-processar outro dataset, o que pode dar errado?”

-

Run on Your Own Video

Preprocessing

- masks and poses: use the bash script `scripts/custom/process-sequence.sh` in InstantAvatar. The data folder should have the followings:

```
smpl_files
 ├── images
 ├── masks
 ├── cameras.npz
 └── poses_optimized.npz
```

- data format: we provide a script to convert the pose format of romp to ours (remember to change the `path` in L50 and L51):

```
cd scripts & python sample_romp2gsavatar.py
```

- position map of the canonical pose: (remember to change the corresponding `path`)

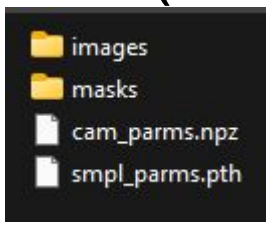
```
python gen_pose_map_cano_smpl.py
```

“Ah, vou pré-processar outro dataset, o que pode dar errado?”

- O que eu deveria ter (de acordo com o readme):

```
smpl_files
  ├── images
  ├── masks
  ├── cameras.npz
  └── poses_optimized.npz
```

- O que eu tenho:



- O que me instruíram a fazer:

Preprocessing

- masks and poses: use the bash script `scripts/custom/process-sequence.sh` in InstantAvatar. The data folder should have the followings:

```
smpl_files
  ├── images
  ├── masks
  ├── cameras.npz
  └── poses_optimized.npz
```

- data format: we provide a script to convert the pose format of romp to ours (remember to change the `path` in L50 and L51):

```
cd scripts & python sample_romp2gsavatar.py
```

- position map of the canonical pose: (remember to change the corresponding `path`)

```
python gen_pose_map_cano_smpl.py
```

(No fim do dia era só gerar o mapa de posições da pose canônica do SMPL)

“Ah, vou pré-processar outro dataset, o que pode dar errado?”

- Deu certo. Não graças ao readme do projeto.
- Consegui treinar o primeiro estágio do treino do dynvideo_female
  - `Training progress: 100%|| 155600/155600 [9:03:53<00:00,`

```
PSNR: 22.68 [13/11 23:22:03]
SSIM: 0.9333 [13/11 23:22:03]
LPIPS: 0.0441 [13/11 23:22:03]
```

“Ah, vou pré-processar outro dataset, o que pode dar errado?”

Render (120 epochs)

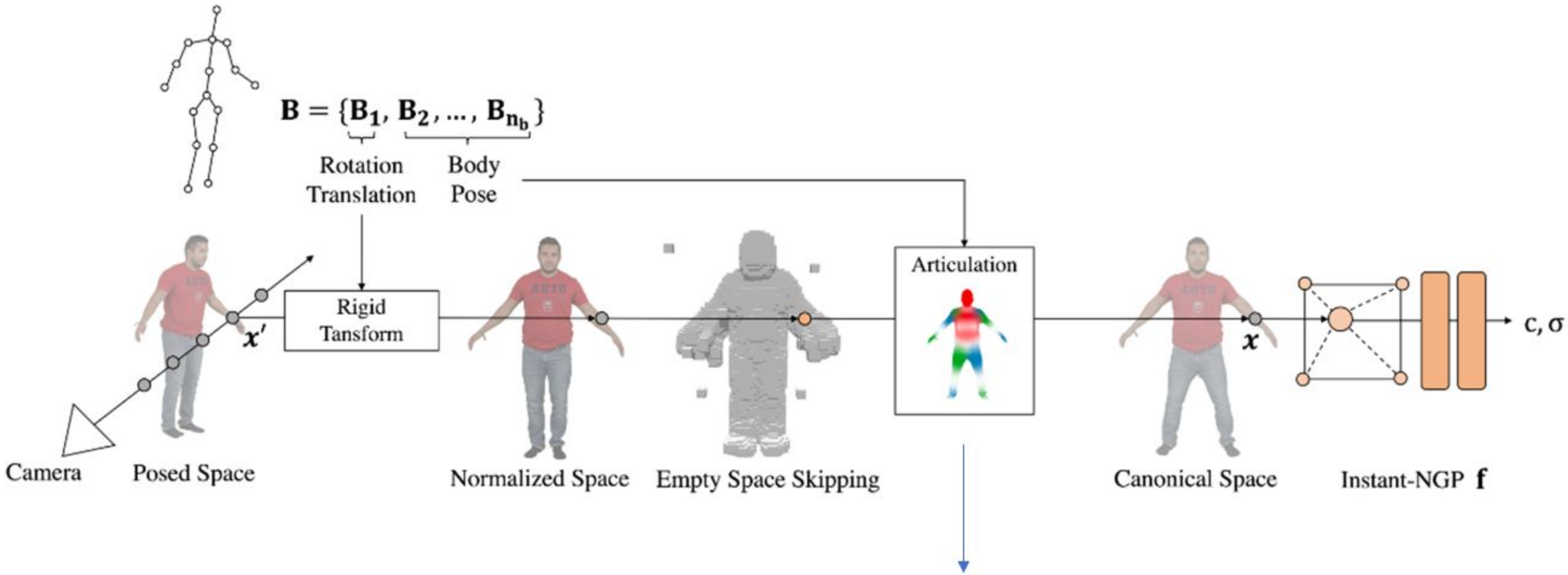
Render (150 epochs)

Render (180 epochs)

GaussianAvatar: Towards Realistic Human Avatar Modeling from a Single Video via Animatable 3D Gaussians

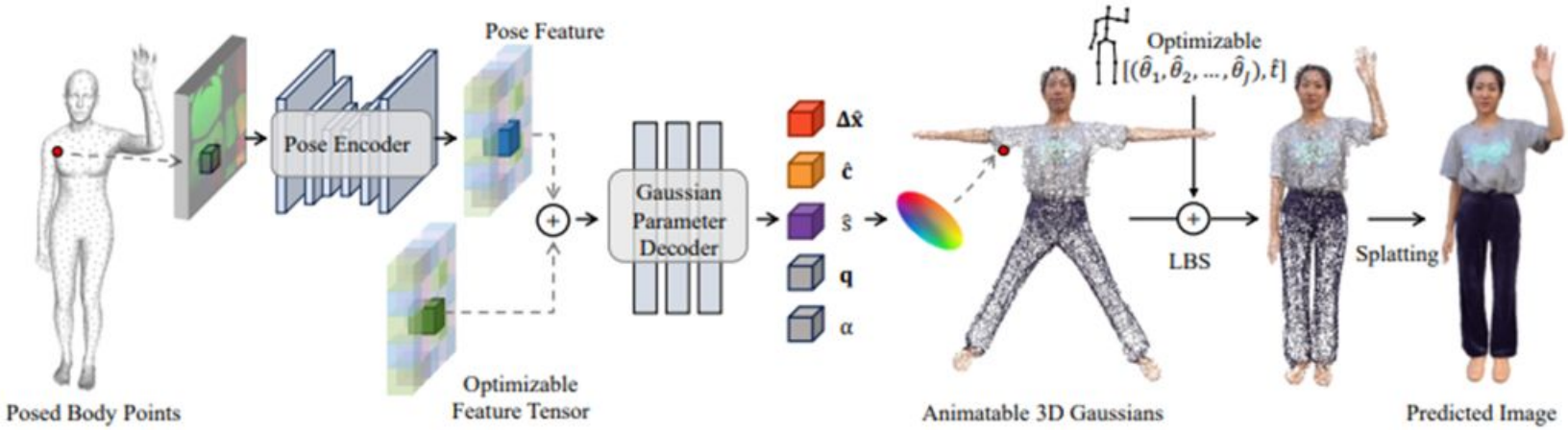
PhD Student: Alberto Arkader Kopiler

Instant Avatar



Fast-SNARF: A Fast Deformer for Articulated Neural Fields

Gaussian Avatar



INPUTS

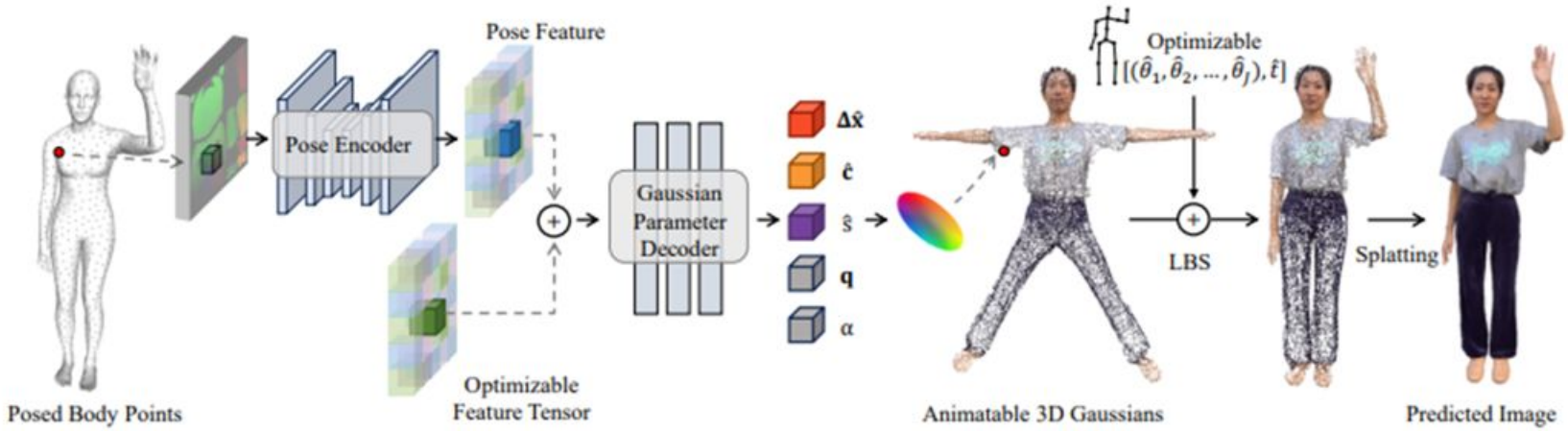
Single Monocular 2D Video

Articulation/ Animation

3D Reconstruction

Novel View Synthesis

Gaussian Avatar



Human Body Model

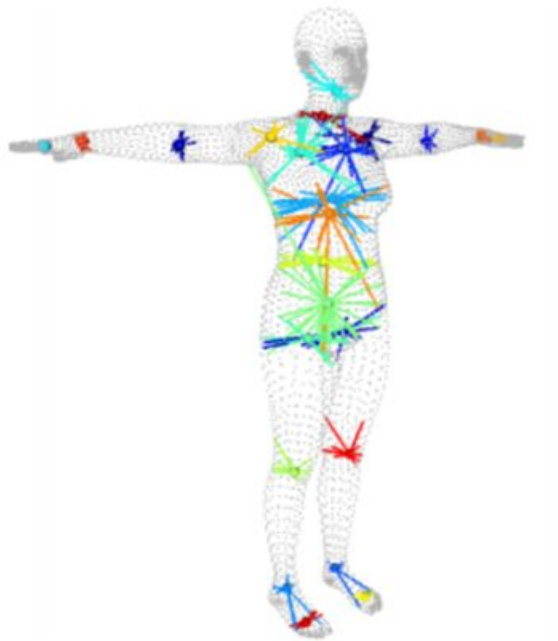
Skinned Multi-Person Linear Model (SMPL)

Linear Blend Skinning (LBS) → problems such as volume collapsing and candy wrapper → unrealistic body deformation

Pose corrective blendshapes → A blendshape is a vector of vertex displacements to the original template mesh

Gaussian Avatar

Training of pose-related parameters



Identity-dependent (Shape) blendshapes

DMPL [1] to model body and dynamic soft tissue movements

SMPL-H [2] to model body and hand/finger movement

SMPL-X [3] to model body, hand, and facial expression.

https://blog.ai.aioz.io/guides/computer-graphics/IntroductionToHumanBodyModel_14

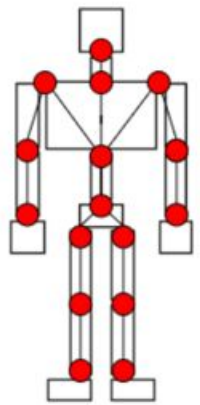
[1] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., & Black, M. J. SMPL: A skinned multi-person linear model. ACM transactions on graphics (TOG).

[2] Romero, J., Tzionas, D., & Black, M. J. Embodied hands: Modeling and capturing hands and bodies together. ACM transactions on graphics (TOG).

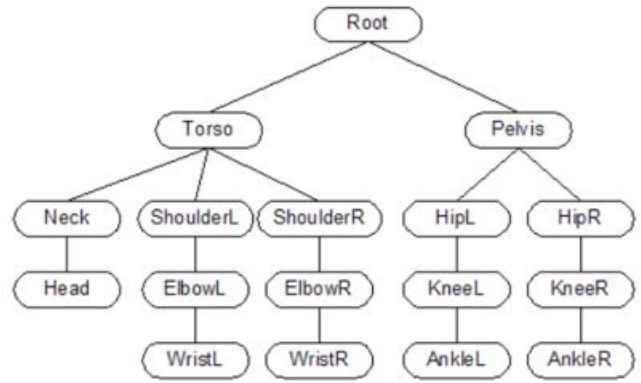
[3] Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A. A., Tzionas, D., & Black, M. J. Expressive body capture: 3d hands, face, and body from a single image. In CVPR 2019.

Kinematics structure of the character

Joints



(a) Character Skeleton



(b) Hierarchy of skeleton joints

https://ai.aioz.io/guides/computer-graphics/CharacterAnimationandSkinning_part1/

Types of animation:

Keyframe-based animation
Procedural animation
Physically-based animation

https://ai.aioz.io/guides/computer-graphics/CharacterAnimationandSkinning_part2/

Animating and Skinning the character:

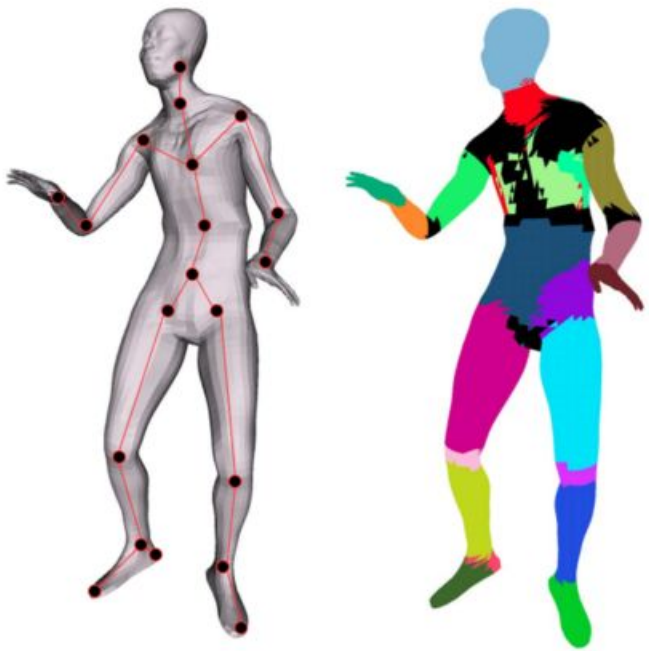


Figure 2. Example skinning weights of a human mesh

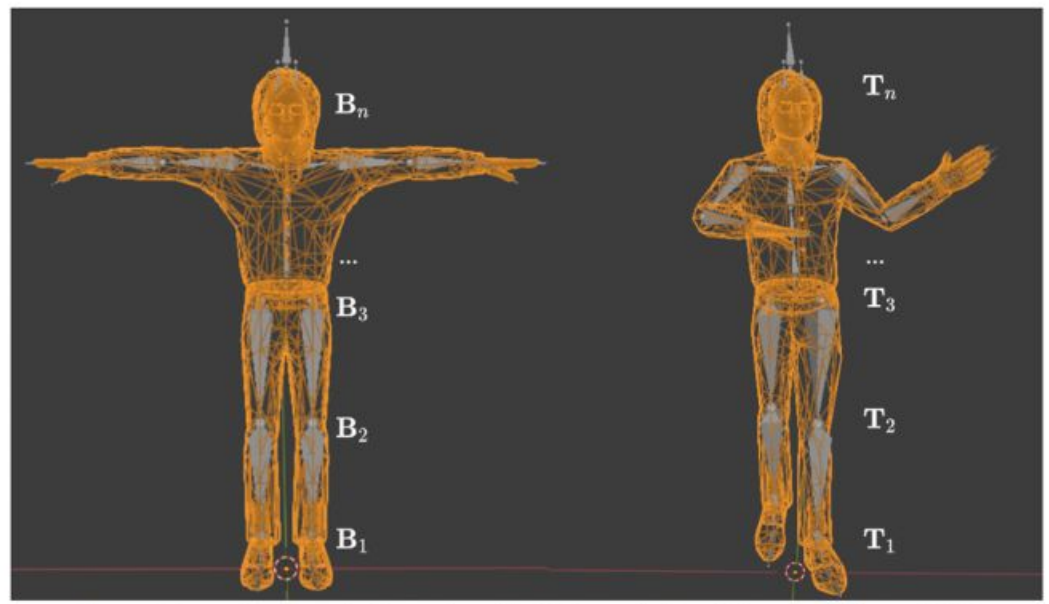


Figure 3. Examples of mesh deformation with bind pose (left) and the animated pose (right).

Gaussian Avatar - Limitations

Limitation. Similar to [15, 49, 57], our method may generate artifacts due to inaccurate foreground segmentations in videos and encounter challenges in modeling loose outfits such as dresses.

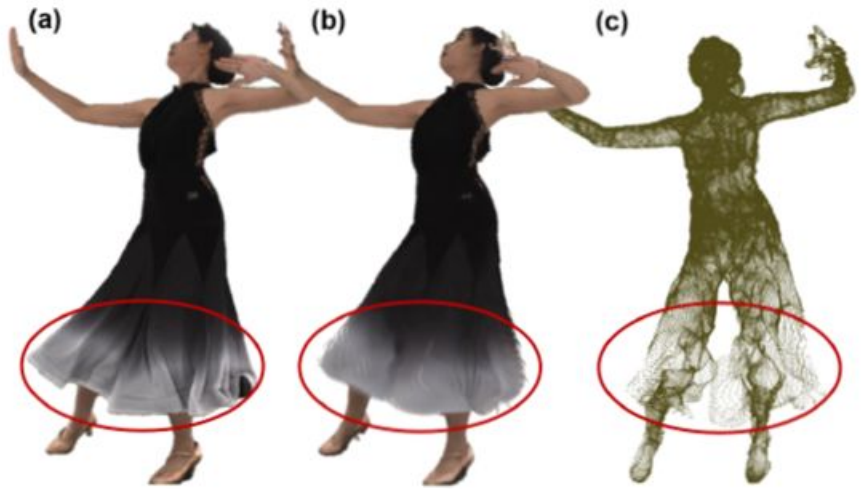


Figure 9. **Results of loose clothing.** (a) is the ground truth, (b) and (c) are the rendered image and Gaussian points.

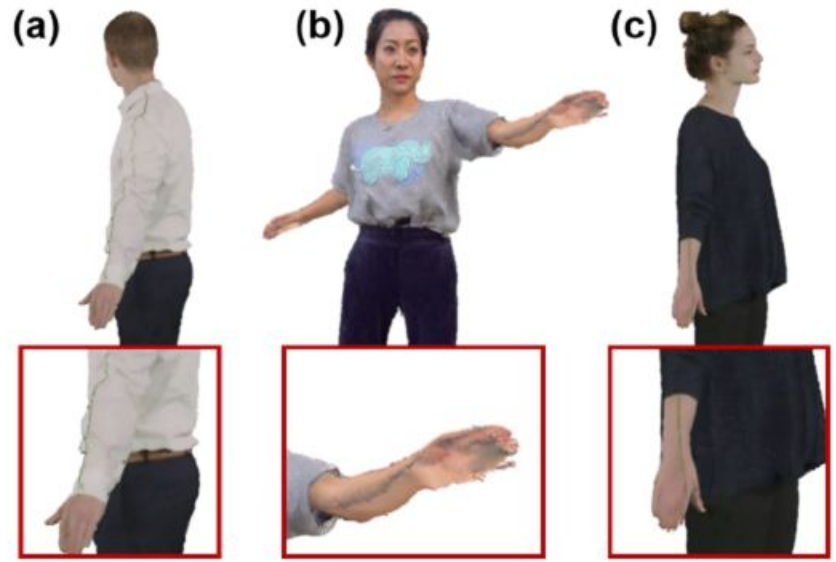
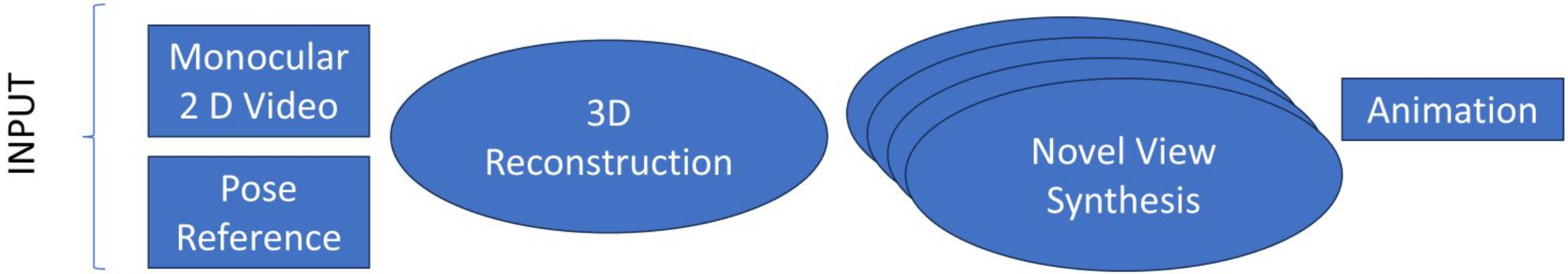
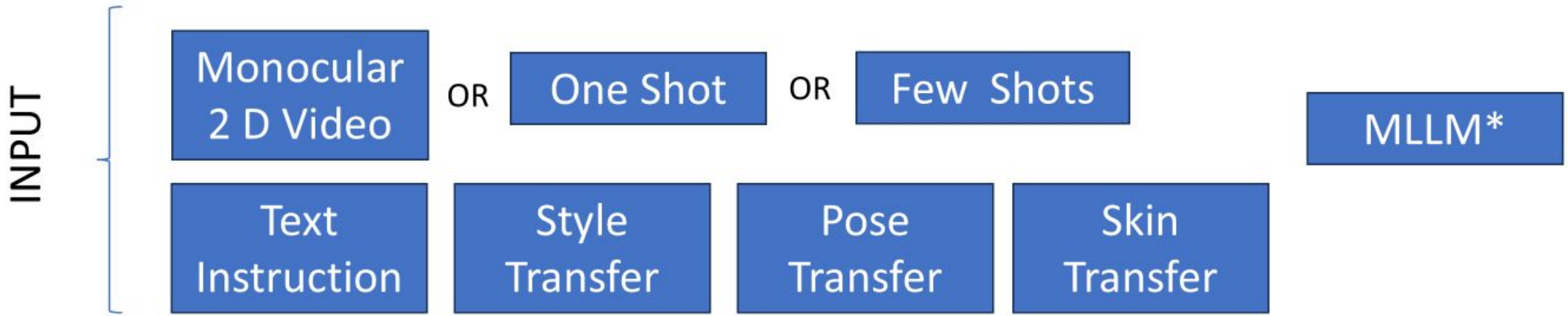


Figure 8. **Results of inaccurate segmentation.** We showcase the artifacts resulting from the inaccurate segmentation boundary.

Gaussian Avatar

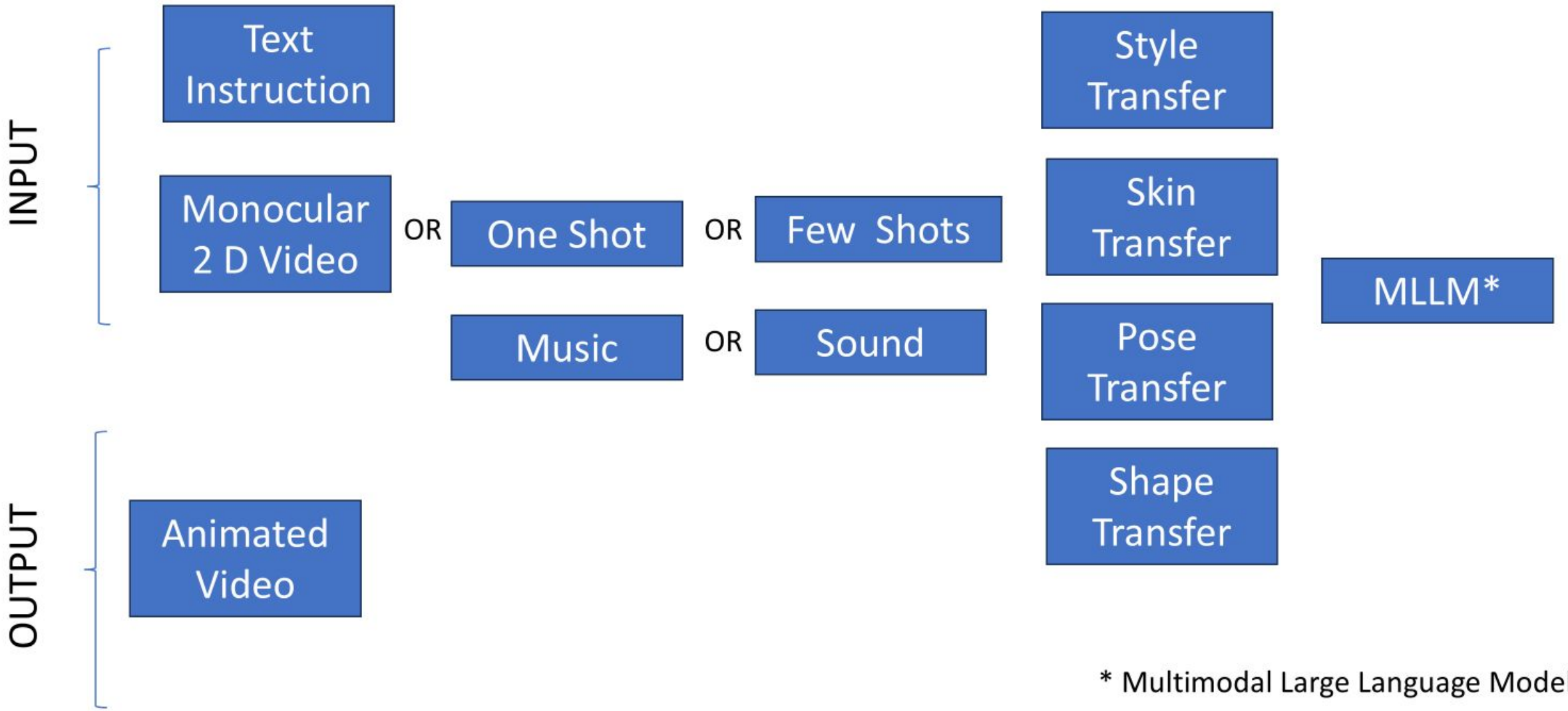


Proposed Project: Generative 3D Animated Agent



* Multimodal Large Language Models

Proposed Project: Generative 3D Animated Agent



* Multimodal Large Language Models

Related Work

https://www.deepmotion.com/post/saymotion-v2-3-video-text-to-3d-animation-in-one-platform-animate-3d-integration

https://www.deepmotion.com/animate-3d

https://adsknews.autodesk.com/pt-br/news/autodesk-launches-wonder-animation-video-to-3d-scene-technology/

https://www.tracking4all.com/

mediapipe tracking puppet

https://app.justsketch.me/

https://openai.com/index/sora/

https://runwayml.com/research/introducing-gen-3-alpha

Just Sketch Me

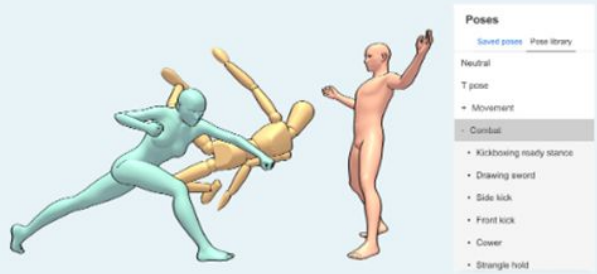
app.justsketch.me

Atualização

Atualização

Leve sua arte para o próximo nível!

Desbloqueie todos os 30+ modelos, 200+ poses, objetos úteis, armazenamento em nuvem e mais!



Obtenha o JustSketchMe Pro!

Já tem uma chave de licença?

Chave de licença...

Ativar

4.3

Introducing Wonder Animation: Powered by Video to 3D Scene Technology

Assistir mais tarde Compartilhar

MAIS VÍDEOS

0:14 / 1:05

YouTube

Features

High Quality Tracking with Only Webcam

Widely Accessible for End-Users

- Real-time body, hand, and finger tracking smoothed to 60fps.
- Runs on consumer hardware including PC and phones.
- Functions fully offline.

deepmotion.com/post/saymotion-v2-3-video-text-to-3d-animation-in-one-platform-animate-3d-integration

DEEPMOTION

SayMotion Animate 3D Resources Sign up Sign in

EXXO34 +1,52%

Pesquisar

10:39 13/11/2024

deepmotion.com/animate-3d

DEEPMOTION

SayMotion Animate 3D Resources Sign up Sign in

Create 3D Animations From Video Using AI

Unleash your creativity with Animate 3D - the leading AI motion capture solution for digital creators.

Free to Use - Sign Up Now

EXXO34 +1,52%

Pesquisar

10:40 13/11/2024

Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She...

All of the videos on this page were generated with Gen-3 Alpha with no modifications.

Prompt: Subtle reflections of a woman on the window of a train moving at hyper-speed in a Japanese city.

Many possibilities...

- Video to 3D Animations
- Text to 3D Motion
- Text to Video
  - “Director mode”: control of the camera position, direction, and movement, lighting effects, ...
- Text/Image to Video
- Text/Image/Video to Video
- Text/Image/Video/Sound to Video