Automatic Rigging for Gaussian Splat Avatars

Leander Winters, *Student, UHasselt & KU Leuven*
Prof. dr. Nick Michiels (supervisor), dhr. Brent Zoomers (advisor)

***Abstract*—3D Gaussian Splatting (3DGS) has emerged as a high-fidelity alternative to mesh-based representations, but deforming 3DGS models remains challenging due to their unstructured, volumetric nature. The proposed pipeline introduces a skeleton-driven animation approach based on forward kinematics. A single-frame pose estimator is used to extract a 3D skeleton from the avatar, which is then aligned to the model through a similarity transform. By computing an orthogonal projection, each Gaussian is assigned to a skeleton bone, enabling joint rotations to drive the deformation of associated Gaussians. Interpolated pose transitions and motion transfer from external sources are supported via additional functionality. Results show that plausible deformations can be achieved without requiring temporal supervision, re-training, or mesh-based rigging.**

***Index Terms*—Forward Kinematics, Gaussian Splatting, Pose Estimation, Rigging, Skeleton-Based Animation**

I. Introduction

ANIMATED 3D content is widely used in modern computer graphics, whether for visual effects, games, medical imaging, or interactive VR/AR applications. However, many of the most common methods for modeling and animating 3D scenes rely heavily on polygon-based meshes, which require explicit connectivity and tend to exhibit lower visual fidelity due to limited geometric detail. Recently, 3D Gaussian Splatting (3DGS) has emerged as a relatively lightweight alternative [1] that represents scenes using volumetric ellipsoids rather than meshes. These ellipsoids efficiently capture both shape and view-dependent appearance of a scene without the need for explicit connectivity. Despite this potential, the problem of animating such Gaussian-based 3D representations in a skeleton-driven manner remains underexplored compared to traditional mesh-rigging pipelines.

The core question of this research is: *How can skeletal rigging be enabled for Gaussian Splat avatars so that their ellipsoids can be directly manipulated via forward kinematics?* To explore this, a Python-based pipeline was developed that performs the full rigging process, from skeleton extraction to deformation. The pipeline starts by using Google's MediaPipe framework [2] to estimate a 3D skeleton. Once the optimized skeleton and the Gaussian Splat avatar are correctly aligned, each Gaussian can be assigned to a particular bone. This Gaussian-to-bone assignment enables the use of a forward kinematics system, adding offsets to the joint positions from one pose to another. By applying the same transformation to each Gaussian as to its associated bone, including position, scale, orientation, and spherical harmonic (SH) features, the Gaussian Splat Avatar is deformed in a manner comparable to a traditionally rigged character model.

The remainder of this paper is organized as follows. Section II reviews related work on alternative 3D representations and skeleton-based animation. Section III describes the proposed pipeline for pose estimation, skeleton alignment, Gaussian assignment, the application of forward kinematics, and various applications for this system. The results are presented in Section IV, followed by a discussion and future directions as well as the final conclusion in Sections V and VI, respectively.

II. Related Work

The original 3D Gaussian Splatting method introduced by Kerbl et al. [1] provides a fast and flexible way to render radiance fields by modeling scenes as sets of anisotropic Gaussians. Through alpha blending and spherical harmonics, the method supports real-time rendering while preserving fine visual details. Several recent works have explored how to extend this method to enable animation. For example, Qian et al. [3] introduced GaussianAvatars. They propose parametric face rigs that are fitted to splat-based representations of human heads. Their method relies on existing blendshape models and uses optimization to align the Gaussians with facial controllers. Gaussians-to-Life, presented by Wimmer et al. [4], enables the animation of entire scenes from textual descriptions. This is achieved using neural controllers trained via diffusion models that learn to apply realistic motion to splatted scenes. While both approaches produce high-quality results, they typically rely on pre-existing rigging models or neural motion priors. This dependence can limit their usability in cases where only a generic, static Gaussian Splat reconstruction without related pose data is available.

Previous skeleton-based deformation techniques have mostly relied on mesh-based techniques such as SMPL [5], which parameterizes a human body via blendshapes and pose-dependent deformations. Guo et al. [6] presented Make-It-Animatable, a framework that converts raw 3D scans into mesh models ready for animation using automatic rigging and skinning prediction. Before that, Lin et al. [7] addressed the problem of skeleton extraction from point clouds by employing neural networks to infer medial structures. However, their method generates a skeletal mesh, an approach deliberately avoided in this study. While such techniques are effective for surface-based representations, their underlying assumptions and processing pipelines are not readily applicable to sparse volumetric formats such as 3D Gaussian Splatting.

Yao et al. [8] take a different approach with RigGS, where a 4D representation is constructed from Gaussians over time. They fit a skeleton by tracking motion trajectories directly through this temporal data, which enables joint-based animation. Their pipeline produces an animatable Gaussian Splat model, but it requires consistent input data over time and a custom training procedure.

This study aims to bridge this gap by enabling basic rigging functionality on standard Gaussian Splat avatars using a procedural alignment and bone assignment strategy. In contrast to previous works, the proposed approach aims to operate directly on the raw ellipsoidal primitives and provides a geometric interface for user-defined motion without requiring temporal information, precomputed correspondences, or re-training.

III. Method

A. Preliminary

To obtain a 3D Gaussian Splatting model, the original implementation proposed by Kerbl et al. [1] is utilized. In this representation, scenes are encoded using a set of anisotropic Gaussians positioned in 3D space. Each Gaussian is defined by a mean position $\boldsymbol{\mu} \in \mathbb{R}^3$, a 3D covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ that describes its spatial extent and orientation, and an opacity value $\alpha \in [0, 1]$ controlling its contribution to the final image.

The density of a single Gaussian is given by:

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{1}$$

To ensure valid covariance matrices during optimization, something not guaranteed by gradient descent, Kerbl et al. [1] define a parametric ellipse with a rotation matrix $\mathbf{R}$ and a scaling matrix $\mathbf{S}$, then construct de covariance matrix by:

$$\boldsymbol{\Sigma} = \mathbf{RSS}^\top\mathbf{R}^\top. \tag{2}$$

The appearance of each Gaussian is defined using third-order spherical harmonics to model view-dependent color, rendering is performed through alpha blending. For each pixel, the final color $\mathbf{C}$ is computed as:

$$\mathbf{C} = \sum_{i=1}^{n} \mathbf{c}_i \alpha'_i \prod_{j=1}^{i-1}(1-\alpha'_j), \tag{3}$$

where $\mathbf{c}_i$ is the color of the i-th Gaussian, and α'_i its opacity after accounting for depth-based sorting and projection.

To train the model, camera poses and sparse geometry are constructed from a video using COLMAP [9], [10], a structure-from-motion pipeline that estimates intrinsic and extrinsic parameters via feature matching. These calibrated poses serve as input for the Gaussian Splatting optimization process, which progressively refines Gaussian parameters to minimize the photometric error across all training views. The result is a ".ply" file containing hundreds of thousands of Gaussians, which form the geometric basis for the proposed rigging and animation pipeline.

B. Pose Estimation

In order to estimate an articulated skeleton for the purpose of rigging, the pose estimation functionality from Google's MediaPipe framework [2] is employed. This tool takes a single RGB frame as input and estimates 33 body landmarks with 3D coordinates, along with a predefined kinematic hierarchy. Figure 1 shows the joint definitions provided by the MediaPipe framework. For the purpose of automating the pipeline, a frontal render of the user-provided 3DGS file is used as the input for this estimation process.

This approach offers a practical trade-off between complexity and usability: it requires no multi-camera setup or depth sensors, and can be applied to any standard image or video frame. However, since the inferred depth is not metrically accurate, especially for self-occluded joints, the resulting skeleton can contain structural distortions along the depth-axis (z-axis).

To mitigate this, the subject is instructed to adopt a front-facing pose with arms extended away from the body during capture to ensure maximal joint visibility and minimizing ambiguity in limb placement. The estimated 2D joint positions on a correctly positioned subject are visualized in Figure 2.

Following this estimation, the MediaPipe framework leverages a trained regression model to infer an approximate depth, resulting in pseudo-3D keypoints. The resulting skeleton is then partially flattened along the z-axis to improve spatial correspondence with the 3DGS model, which is treated as a geometrically consistent 3D reconstruction.

Finally, the optimized joint positions and bone connections are exported in a structured JSON format. Each joint entry includes a unique index, a descriptive name, and an absolute position represented as a list of $[x, y, z]$ values. The connection list specifies which pairs of joint indices are connected to form the skeletal structure. This representation allows for easy processing in later stages of the rigging pipeline.

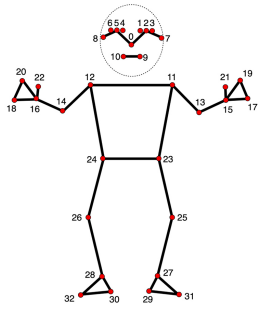

Fig. 1: MediaPipe joint definitions and hierarchy.

Fig. 2: Mediapipe pose estimation result on a frontal render of the 3DGS avatar.

C. Skeleton-to-Model Alignment

To apply joint-based deformations to the Gaussian Splat Avatar, its positioning must first be aligned with the optimized skeleton. While both are roughly normalized in the same

coordinate space, a rigid similarity transformation is still required to ensure that the Gaussians correspond spatially to the relevant bones of the skeleton.

This is achieved by identifying four reliable anatomical landmarks in both the Gaussian Splat avatar and the skeleton: the left and right hands, and the left and right feet. These correspond to joints 19, 20, 31, and 32 according to the MediaPipe definition. For the skeleton, the absolute 3D positions of these joints are retrieved directly from the previously stored JSON file. For the 3DGS model, the corresponding locations are identified heuristically by selecting extremal points along the x, y, and z axes, depending on the expected orientation of limbs.

Once these four pairs of corresponding 3D points are established, a similarity transform is computed. Specifically, the transformation is estimated using a variant of the orthogonal Procrustes analysis, which minimizes the mean squared error between two sets of corresponding 3D points [11]. Given the matched landmarks from the 3DGS model and the skeleton, this method computes a similarity transform consisting of a uniform scale factor, an orthogonal rotation matrix, and a translation vector.

The computed transformation is applied solely to the 3DGS model. Specifically, the mean position $\boldsymbol{\mu}$ of each Gaussian is transformed directly, while the scaling factor and associated covariance matrix are adjusted to maintain visual consistency. The covariance matrix of each Gaussian defines its shape and orientation in 3D space. In addition, the spherical harmonic (SH) features that encode view-dependent appearance are rotated with the same transformation to ensure consistency in lighting and reflectance behavior. This results in a transformed Gaussian Splat Avatar that is spatially aligned with the skeleton as illustrated in Figure 3.

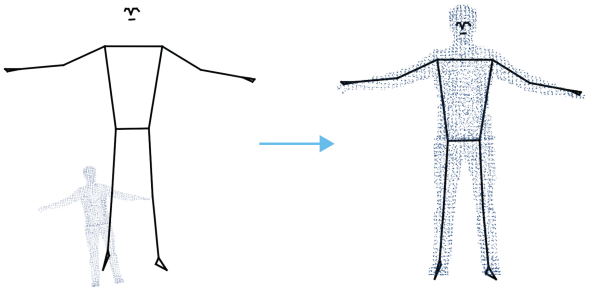

Fig. 3: Initial and transformed skeleton to 3DGS model alignment.

D. Gaussian-to-Bone Assignment

Once the Gaussian Splat avatar and the optimized skeleton are aligned, each Gaussian is assigned to a corresponding bone in the skeletal structure. Bones are defined as oriented line segments between joint pairs, as encoded in the connection list of the skeleton JSON file. This association enables later deformation of the Gaussians based on articulated joint motion.

To determine the bone to which a Gaussian is mapped, an orthogonal projection is computed between the Gaussian's center and each bone segment. The bone with the smallest distance to the Gaussian is selected as its parent structure, as visualized in Figure 4. This procedure results in a one-to-one mapping between each Gaussian and a specific bone, represented as a pair of joint indices. For consistency, joint indices within each pair are sorted in ascending order to avoid ambiguity in later lookup operations.

The outcome of this process is a mapping structure, also stored in JSON format, which links each splat to its associated bone segment. This mapping file is also a requirement for the animation step, where bone transformations propagate to their assigned Gaussians via forward kinematics. Figure 5 illustrates the anatomical segmentation of the 3DGS model.

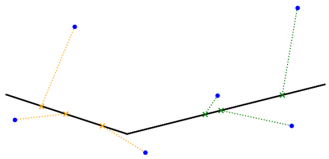

Fig. 4: Orthogonal projection for bone assignments.

Fig. 5: Anatomical segmentation of the 3DGS model.

E. Forward Kinematics

Following the initial alignment and assignment of each Gaussian, articulated motion can be simulated using a forward kinematics system. A forward kinematics implementation propagates transformations from parent joints to their children based on the defined skeletal hierarchy. This mechanism allows the deformation of the volumetric model to reflect human motion.

A subtree-based rotation is applied to the skeleton by traversing its joint hierarchy in a constrained breadth-first manner. Starting from a pivot joint, all descendant joints that are not explicitly blocked are rotated around the pivot using a user-defined rotation input. Such an input includes: a joint id, the rotation axis, and the angle in degrees. The adjacency structure is used directly from the bone connection list defined in the skeleton JSON file.

To deform the Gaussian model accordingly, the associated Gaussians are repositioned in space by rotating their positions relative to the pivot joint. In addition, each affected Gaussian's orientation, encoded as a quaternion in $[w, x, y, z]$ format,

and spherical harmonic features up to order 3 are rotated accordingly. This is accomplished by composing the global joint rotation with the original orientation using quaternion algebra. As a result, both geometric deformation and view-dependent appearance remain consistent with the new skeletal configuration.

Figure 6 illustrates an example of the forward kinematics implementation. Each frame shows an intermediate state resulting from a single subtree rotation step applied to the Gaussian Splat Avatar. Although the order in which subtrees are updated does not affect the final result, multiple updates are required to reach the desired pose. To construct the target pose shown, three consecutive subtree-based rotations are applied. First, a rotation of -75° around the x-axis is applied to joint 24 (right hip). Second, a +45° x-axis rotation is applied to joint 26 (right knee). Finally, a +45° y-axis rotation is applied to joint 11 (left shoulder).

Fig. 6: Stepwise Subtree Rotations for Pose Construction.

F. Motion Transfer

Another feature of the proposed pipeline is motion transfer, which enables the static Gaussian Splat avatar to reproduce motion captured from an external video source. The input video is first decomposed into individual frames. These frames serve as input for the estimation step, extracting 3D joint positions for each frame. These estimated skeletons are then compared to the reference pose, a rest or T-pose configuration as described previously.

This comparison is performed by a command extraction script, which identifies pairs of limb segments (e.g., upper and lower arm or leg) and computes the relative rotations between corresponding bones in the reference and target skeletons. Each frame appends to a JSON file that lists the necessary joint rotation instructions to replicate the observed pose. This list can then be used as input for the forward kinematics step to animate the 3DGS model over time.

Unlike mesh-based rigging pipelines, the proposed method does not rely on any mechanism to fit or deform the Gaussian Splat avatar to a target skeleton. Instead, thanks to the skeleton-driven forward kinematics implementation, the avatar can assume any pose as long as the relative difference in joint orientations between the input and reference skeleton can be computed. As a result, global scale differences between the source motion and the avatar do not affect the deformation process, making the approach inherently robust to size mismatch between the input subject and the reconstructed avatar.

It is important to note that this process is made possible by the consistent joint definitions used by the MediaPipe pose estimation framework. Since the joint indices and anatomical correspondences remain fixed across all frames, the pivot joints for each rotation can be reliably identified.

Figure 7 shows selected frames from an input video and how the Gaussian Splat avatar was deformed to mimic its poses. The motion data was extracted from the ”Jumping Jacks” video by FitnessBlender [12].

Fig. 7: Motion transfer from input video.

G. Interpolation and Rendering

To visualize the animated results, we interpolate joint rotations over time and render a sequence of frames that capture the deformation of the Gaussian Splat avatar. The set of joints to be rotated, their corresponding axes, target angles, and the number of interpolation steps are all specified by the user. This provides flexibility in defining pose transitions for specific use cases or gestures.

For each rotation, interpolation is performed in angular space over the desired number of steps. At each step, the corresponding transformation is applied to the affected subtrees of the skeleton. The positions and orientations of all affected Gaussians are updated directly within the model’s data structure, using the same forward kinematics system as described earlier.

Each intermediate model state is rendered to a high-resolution PNG image using a lightweight renderer based on the differentiable splatting framework by Kerbl et al. [1]. We employ an open-source implementation developed by Meng [13], which supports GPU-accelerated rendering of 3DGS models. To ensure visual continuity, we fix the virtual camera in space and render the subject from a frontal viewpoint across all frames. The resulting sequence of images is compiled into an animated GIF with a uniform frame duration. This output can serve both as a qualitative validation of the deformation pipeline and as a practical end result for users who wish to visualize animated Gaussian Splat avatars in motion.

Figure 8 illustrates an example of such an interpolation sequence. In this case, we interpolate from the rest pose to a target pose extracted via the motion transfer functionality. Each frame in the figure shows an intermediate state resulting

from incremental joint updates. While the final pose matches that of the transferred motion, the interpolation process enables the gradual deformation of the Gaussian Splat avatar across multiple steps.

Fig. 8: Interpolation from rest pose to target pose.

IV. Results

The proposed pipeline was used to generate a sequence of animated poses from a static Gaussian Splat model. The input for these generations consisted of the original ".ply" file and a series of joint update commands. Each update specifies a pivot joint (by ID), a rotation axis, and an angle in degrees. These updates were applied to the optimized skeleton using forward kinematics.

Simultaneously, the Gaussian Splat avatar was deformed by utilizing the binary mask generated by the bone assignment and forward kinematics step. Each Gaussian was either marked as associated (1) or not associated (0) with a given bone. All Gaussians associated with affected bones were transformed accordingly. This included updating their position, orientation, scale, and SH's. Figure 9 illustrates several realistic poses such as raised arms, a walking position, and a running position.

Although the generated skeletons represent plausible human poses, the orientation of the hands and feet are inaccurate in some iterations. Additionally, artifacts can be seen around the armpits and gaps appear in between the limbs and the torso.

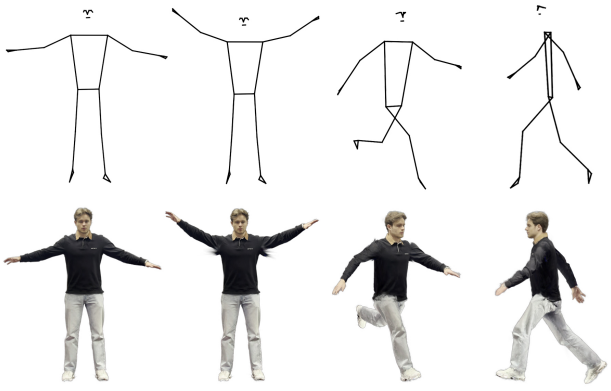

Fig. 9: Skeleton and 3DGS generations.

V. Discussion and future work

While the results demonstrate that the proposed method can produce pose deformations on Gaussian Splat avatars, several imperfections remain that limit its robustness and realism.

As mentioned, certain Gaussians are incorrectly assigned to bones and result in artifacts when rotating limbs. These effects are noticeable around areas like the armpits because the one-to-one bone assignment proves insufficient to capture the anatomical complexity of the human body. Gaussians that were assigned to the bones of the torso, rather than bones of the arm can be seen in Figure 9. Since the current bone assignment implementation relies on a simple orthogonal projection and one-to-one mapping, it fails to model the anatomical overlap between adjacent regions. A more robust approach could incorporate soft weighting or volumetric distance metrics to assign Gaussians to multiple bones proportionally.

Besides artifacts, gaps between neighboring regions of the model emerge when larger rotations are applied. This occurs because the deformation technique that was used does not guarantee blending between deformed and stationary regions. On the contrary, it is also possible for Gaussians of different regions to overlap if bones rotate into each other's vicinity. Introducing skinning weights could reduce both under- and over-deformation in future implementations.

Another problem with this method lies in the initial alignment between the MediaPipe skeleton and the Gaussian model. While a global similarity transform offers a good approximation, it assumes that the skeleton is accurate. This assumption does not hold in practice since MediaPipe's pose estimation does not produce a reliable depth axis, only an approximation. As a result, small errors in the initial alignment step can amplify through successive forward kinematics steps, leading to structural drift. Improving the initialization with multi-frame depth refinement could reduce these errors.

Animating fine anatomical structures such as hands, fingers, and feet remains challenging. This is expected because these regions are underrepresented in the skeleton estimation but it is also a consequence of the Gaussian representation, which lacks detail in sparsely covered areas.

Finally, the current bone assignment technique is computationally expensive due to the naive nested loop over all Gaussians and all bones. This step can be significantly optimized by implementing spatial indexing or other similar techniques.

VI. Conclusion

This work demonstrates that a forward kinematics system can be used to animate static Gaussian Splat avatars without requiring temporal supervision, re-training, or mesh-based rigging. The Gaussian Splat avatars are successfully deformed in a skeleton-driven manner and are able mimic motion from external sources. Further improvements could focus on enhancing skeleton estimation accuracy, developing a more robust bone assignment strategy, and introducing smooth blending at anatomical boundaries. With these refinements, this approach could be extended to support basic animation tasks.

REFERENCES

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[2] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee *et al.*, "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[3] S. Qian, T. Kirschstein, L. Schoneveld, D. Davoli, S. Giebenhain, and M. Nießner, "Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 299–20 309.

[4] T. Wimmer, M. Oechsle, M. Niemeyer, and F. Tombari, "Gaussians-to-life: Text-driven animation of 3d gaussian splatting scenes," *arXiv preprint arXiv:2411.19233*, 2024.

[5] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 851–866.

[6] Z. Guo, J. Xiang, K. Ma, W. Zhou, H. Li, and R. Zhang, "Make-it-animatable: An efficient framework for authoring animation-ready 3d characters," *arXiv preprint arXiv:2411.18197*, 2024.

[7] C. Lin, C. Li, Y. Liu, N. Chen, Y.-K. Choi, and W. Wang, "Point2skeleton: Learning skeletal representations from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4277–4286.

[8] Y. Yao, Z. Deng, and J. Hou, "Riggs: Rigging of 3d gaussians for modeling articulated objects in videos," *arXiv preprint arXiv:2503.16822*, 2025.

[9] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[10] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 501–518.

[11] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, pp. 1–10, 1966.

[12] FitnessBlender, "Jumping jacks," https://www.youtube.com/watch?v=c4DAnQ6DtF8, 2010, accessed: 2025-05-10.

[13] W. Meng, "3dgs_poserender: A lightweight renderer for gaussian splatting models," https://github.com/guaMass/3DGS_PoseRender, 2024.