

# Appendix

## MotionDreamer: Exploring Semantic Video Diffusion Features for Zero-Shot 3D Mesh Animation

Lukas Uzolas   Elmar Eisemann   Petr Kellnhofer  
Delft University of Technology  
The Netherlands

### A. Additional Implementation Details

We optimize the pose fitting for 1 000 iterations. We initially optimize only  $\mathbf{p}^0$  and linearly increase the number of optimized frames from 1 to  $L$  between iterations 0 and 500. We use a constant learning rate of 0.0005 and Adam optimizer [33] in Pytorch [58]. Our MLP  $m$  consists of 6 layers, each with a hidden dimension size of 256, and we scale the final output by a constant  $\alpha = 0.01$ . We apply a frequency encoding [51] for the input  $l$ :  $\gamma(\cdot) = (l, \sin(2^0\pi l), \cos(2^0\pi l), \dots, \sin(2^{k-1}\pi l), \cos(2^{k-1}\pi l))$  with  $k = 6$ .

For each input shape  $\mathcal{M}$ , we define the canonical camera  $\mathcal{C}$  manually.

#### A.1. Video Diffusion models

We use the official implementations for VideoComposer [91] (VC), DynamiCrafter [93] (DC), as well as for Stable Video Diffusion [5] (SVD), where the latter two are accessed through the Diffusers library [87]. We adopt the same hyperparameters for all wherever possible. We set the classifier-free guidance [22] to 6 and we generate 16 frames with an assumed framerate of 16 fps. We use the recommended schedulers with  $T = 50$  inference steps. We discuss the choice of conditioning images for each model and the omission of SVD from our experiments in Appendix D.1. Finally, we observe that VC provides faster inference than DC, which is why we adopt it for our quantitative experiments that require large number of optimizations.

**Matching rendered image to VDM features for DC** Unlike VC and SVD, Dynamicrafter does not enforce the input image to be the first frame of the output video, which is an assumption of our method. This is because all VDM frames are initialized with the same embedded input image:  $\mathcal{E}(\mathbf{x}) = \mathbf{z}^0 = \mathbf{z}^1 = \dots = \mathbf{z}^L$  and then they drift during the inference. We observe that this drift is minimized for the output frame matching the input image and hence we explicitly detect the frame  $l^*$  where features change the least

between the inference steps  $t$ :

$$l^* = \arg \max_l \sum_t \frac{\kappa(\hat{\mathbf{A}}_t^l, \hat{\mathbf{A}}_{t-1}^l) - \mu_{\hat{\mathbf{A}}_{\kappa,t}}}{\sigma_{\hat{\mathbf{A}}_{\kappa,t}}}, \quad (5)$$

where  $\mu_{\hat{\mathbf{A}}_{\kappa,t}}$  and  $\sigma_{\hat{\mathbf{A}}_{\kappa,t}}$  are the mean and standard deviation of the cosine similarities of activations  $\hat{\mathbf{A}}_t$  at step  $t$ . Finally,  $l^*$  can be used as the frame index for feature reprojection.

#### A.2. Animation Models

We experiment with four different animation models.

**SMPL** Skinned Multi-Person Linear [45] is a skinned mesh-based human model that supports various body shapes and human poses. Vertices are deformed based on forward kinematics and linear blend skinning:  $\mathbf{u}_i^l = \sum_b w_{b,i} \mathbf{T}_b^l \mathbf{u}_i^{\text{init}}$ , where  $\mathbf{T}_b^l \in \mathbb{R}^{4 \times 4}$  is the roto-translation of bone  $b$  at time step  $l$  and  $w_{b,i}$  the skinning weight determining how strongly vertex  $\mathbf{u}_i$  is attached to  $b$ .  $\mathbf{T}_b^l$  is defined recursively by its parent bone transformation according to a kinematic hierarchy.

**SMAL** SMAL [103] is another skinned model that can represent various quadrupedal animals, namely lions, cats, dogs, horses, cows and hippos. It follows the sample approach of forward kinematic and linear blend skinning for reposing as SMPL. We make use of the SMALify [3] implementation in our work.

**FLAME** FLAME [38] also adopts the SMPL formulation but expands it by articulation of the jaw, and the eyes. It utilizes blend-shapes to model facial expression offsets for all vertices in the mesh:  $\mathcal{U}_{exp} = \sum_n \vec{\psi}_n \mathbf{E}_n$ , where  $\vec{\psi}_n$  denotes the  $n$ 'th expression coefficient,  $\mathcal{E} = [\mathbf{E}_1, \dots, \mathbf{E}_{|\vec{\psi}|}] \in \mathbb{R}^{3N \times |\vec{\psi}|}$  is the orthonormal expression basis, and  $\mathcal{U}_{exp}$  contains the vertex expression offset for each  $\mathbf{u}_n$ . We further find it beneficial to scale the expression coefficient  $\vec{\psi}_n$  by a factor of 5 in FLAME

Note that we keep the shape parameters fixed for SMPL, SMAL, and FLAME. Please refer to the corresponding work for more details.

**Neural Jacobian Fields (NJF)** Our method also supports arbitrary meshes that are neither rigged nor have blendshapes. To animate these types of meshes we make use of NJF [1]. In NJF, the deformation is obtained by indirectly optimizing the per-triangle Jacobians  $J_i \in \mathbb{R}^{3 \times 3}$  for each face  $f_i$ , instead of directly regressing the displacement for each vertex. To retrieve the deformation map  $\Phi^*$ , a Poisson problem is solved:  $\Phi^* = \min_{\Phi} \sum_{f_i} |f_i| \|\nabla_i(\Phi) - J_i\|_2^2$ , where  $\nabla_i(\Phi)$  is the Jacobian of  $\Phi$  at triangle  $f_i$  and  $|f_i|$  represents the area of the triangle. We follow the implementation of Gao et al. [14], and initialize the Jacobians with identity matrices. Besides the Jacobians, we additionally optimize root rotation, center of rotation, and a global translation vector. We also make use of the Jacobian regularization [14] to avoid diverging too far from the initial geometry. Consequently, we expand our full optimization objective with an additional term

$$\mathcal{L}_j = 1/(2M) \sum_i (\|J_i - I\|_2 + \|J_i - I\|_1),$$

where  $M$  is the number of triangle faces. Therefore, for NJF, we minimize  $\mathcal{L}' = \mathcal{L} + w_j \mathcal{L}_j$ , where  $w_j = 0.5$ .

Our requirements for the inputs mesh are entirely dependent on the animation model. For NJF, we assume a mesh with a single connected component. For multi-component meshes, we adopt the preprocessing from Wang et al. [89] and transform the mesh representation into an SDF and re-sample the mesh based from this SDF. We additionally decimate faces through Quadric edge collapse [15] to reach 8 000 vertices. In practice, we observe that this procedure is robust even for meshes that are not perfectly watertight nor 2-manifold.

For all animation models, we scale the global translation vector  $\mathbf{t}$  by 0.1.

## B. User Study

### B.1. Baseline methods

**DG4D** We use the original implementation provided by Ren et al. [68] but adapt two hyperparameters such that the model can be trained with only 24 GB of VRAM. Namely, we reduce the batch size from 14 to 8 and the number of views per step ( $n_{\text{views}}$ ) from 4 to 2. In its original setup, DG4D automatically removes the background of the input image before passing it to a VDM with a tool *Rembg*<sup>1</sup>. However, as we show in Appendix D.1, VC produces better results for input images with background. Therefore, for VC, we remove the background after the video generation instead

<sup>1</sup><https://github.com/danielgatis/rembg>

by applying Rembg to each video frame. When using DC, the pipeline of DG4D is unaffected.

**MDM-MT** We observe a lack of class-agnostic end-to-end pure motion generators. Therefore, we combine a human-specific motion generator with a general motion transfer method while accepting that the performance of such solution will depend on morphological and semantic proximity of the source and target shape class. To this goal, we first use a pre-trained author’s implementation of the text-conditioned motion diffusion by Tevet et al. [83] to generate a unique 2D skeletal human motion sequence for each example in our study. We adapt the motion text prompts used for our method (Tbl. 4) to the human domain using a template “a person is [ACTION]” e.g., “a horse is walking”  $\rightarrow$  “a person is walking”. Next, we use a 2D-to-3D human body pose uplifting method adapted from the code of Zuo et al. [104] to obtain sequence of SMPL [45] meshes. Finally, we follow the procedure and code of Liao et al. [41] to retarget the SMPL animations to our target meshes. We apply this step consistently even for the SMPL target mesh. Finally, we render the first 16 frames of the resulting mesh sequences in the same way as for our own method.

### B.2. Stimuli

In our study we utilize 10 different shape-prompt pairs (2 SMAL, 2 FLAME, 2 SMPL, and 4 Neural Jacobian Field combinations) and combine them each with 2 different VDMs resulting in 20 unique videos for each evaluated method. We compare pairwise to 2 methods (DG4D and MDM-MT) for the first 3 questions, and we similarly compare to 2 methods (DG4D and VDM) for the last additional question. In total this produces  $3 \times 2 \times 20 + 1 \times 2 \times 20 = 160$  study trials.

**Meshes** We extract the surface models for SMPL [45], FLAME [103] and SMAL [38] from their official implementations. For SMPL, we opt to lower the arms to 45 degrees from the original T-pose, while we use the default “zero” pose parameters for others. For NJF [1], we use the 4 open assets listed in Tbl. 3.

**VDM Target Motion Prompts** Tbl. 4 lists VDM prompts used to generate the motion sequences for the stimuli in our study.

**Single-View Texturing Prompts** Tbl. 5 shows the positive and negative prompts used for Single-view Texturing as an input for the ControlNet diffusion model [98] for each shape in our experiments and the prompt for the Stable Diffusion XL [62] background inpainting.

Table 3. Mesh assets used to evaluate our method with NJF.

Shape	Author	License	URL
Bunny	Stanford	Stanford Public	<a href="http://graphics.stanford.edu/data/3Ds">http://graphics.stanford.edu/data/3Ds</a>
Lego truck	Mildenhall et al. [51]	MIT License	<a href="https://github.com/bmild/nerf">https://github.com/bmild/nerf</a>
Raptor	Gatzegar	TurboSquid Standard	<a href="https://www.turbosquid.com/3d-models/raptor-dinosaur-model-1538088">https://www.turbosquid.com/3d-models/raptor-dinosaur-model-1538088</a>
Palm tree	mr_zaza	TurboSquid Standard	<a href="https://www.turbosquid.com/3d-models/3d-tropic-palm-tree-model-2090490">https://www.turbosquid.com/3d-models/3d-tropic-palm-tree-model-2090490</a>

Table 4. Prompts used to generate stimuli in our study.

Shape	Prompt
SMPL	“A person jumping up”
SMPL	“A person walking forward”
Horse (SMAL)	“A horse walking”
Horse (SMAL)	“A horse jumping”
FLAME	“A person laughing”
FLAME	“A person being very angry”
Bunny	“A bunny shaking its ears”
Lego truck	“A yellow truck moving its shovel up and down”
Raptor	“A raptor jumping”
Palm tree	“A palm tree swaying in the wind”

Table 5. Prompts used for our Single-View Texturing and background inpainting.

	Prompt	Negative Prompt
<b>SMPL</b>	“A photo of a clothed person wearing pants and tshirt in front of a <background>, photorealistic, 4k, DLSR”	“grey, gray, monochrome, distorted, disfigured, naked, nude”
<b>FLAME</b>	“A portrait photo a face in front of a <background>, photorealistic, 4k, DLSR, bokeh”	“grey, gray, monochrome, distorted, disfigured, render, teeth, hat”
<b>SMAL</b>	“A photo of a <animal> in front of a <background>, photorealistic, 4k, DLSR”	“grey, gray, monochrome, distorted, disfigured, render”
<b>Others</b>	“A photo of a <object> in front of a <background>, photorealistic, 4k, DLSR”	“grey, gray, monochrome, distorted, disfigured, render”
<b>Inpainting</b>	“Background image of a <background>”	“Person, face, animal, object”

### B.3. Instructions

Fig. 9 shows the instructions as presented to each participant before the start of the study. Fig. 10, Fig. 11, Fig. 12, Fig. 13 show screenshots of our study interface for each of the four distinct questions (3 questions in the main part and one additional question). The questions were presented in four blocks sequentially always in the same order. There was an instruction screen displaying the next question shown at the beginning of each block. The order of blocks was fixed but the order and layout of the trials was randomized for each participant.

### B.4. Detailed Results

Here, we present a break-down of the results from our user study separately for the human stimuli (Fig. 14), where the human-specific MDM-MT baseline performs well and for

the remaining stimuli (Fig. 15), where our class-agnostic method dominates. We also offer a detailed breakdown in Tbl. 6.

## C. Pose Optimization Experiment Details

### C.1. Data

We select the first 20 frames from 20 randomly selected human dancing motion sequences in the AIST++ dataset [37]. Since our goal is not to reproduce the original camera poses, we use a single fixed camera  $\mathcal{C}$  and position the first-frame SMPL mesh into the center of its viewport. Then we render the rest of the SMPL sequence with a fixed camera. An example can be seen in Fig. 16.

### Study Information

The study will take approximately 15 minutes.

You can drop out of the study whenever you want.

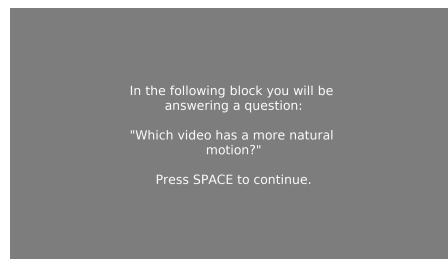
You may voice concerns or ask questions throughout the study.

You may take breaks.

### Study Procedure

You will go through 4 different blocks, each block associated with one question.

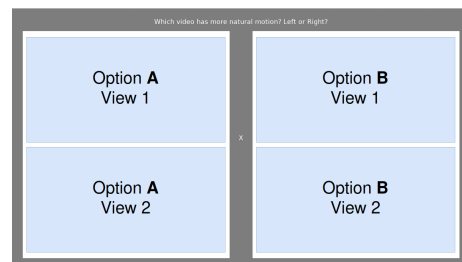
Before the start of each block, the question will be written on the screen and you will have to press SPACE to initiate each block. Example:



The questions are:

- Which video has more natural motion?
- Which video has fewer visual artifacts?
- Which video captures the prompt better?
- Which video do you prefer overall?

After initiating a block, you will see two videos of rendered 3D objects. One on the left and one on the right. Block one, two and three show two views of the scene. Example:



You will have to indicate your preference given the question of the block. For block one, two and three, the question will also be displayed at the top of the screen.

You can choose your preference by pressing the RIGHT or LEFT arrow key.

Figure 9. Study instructions that were read out and explained to our participants before the study.

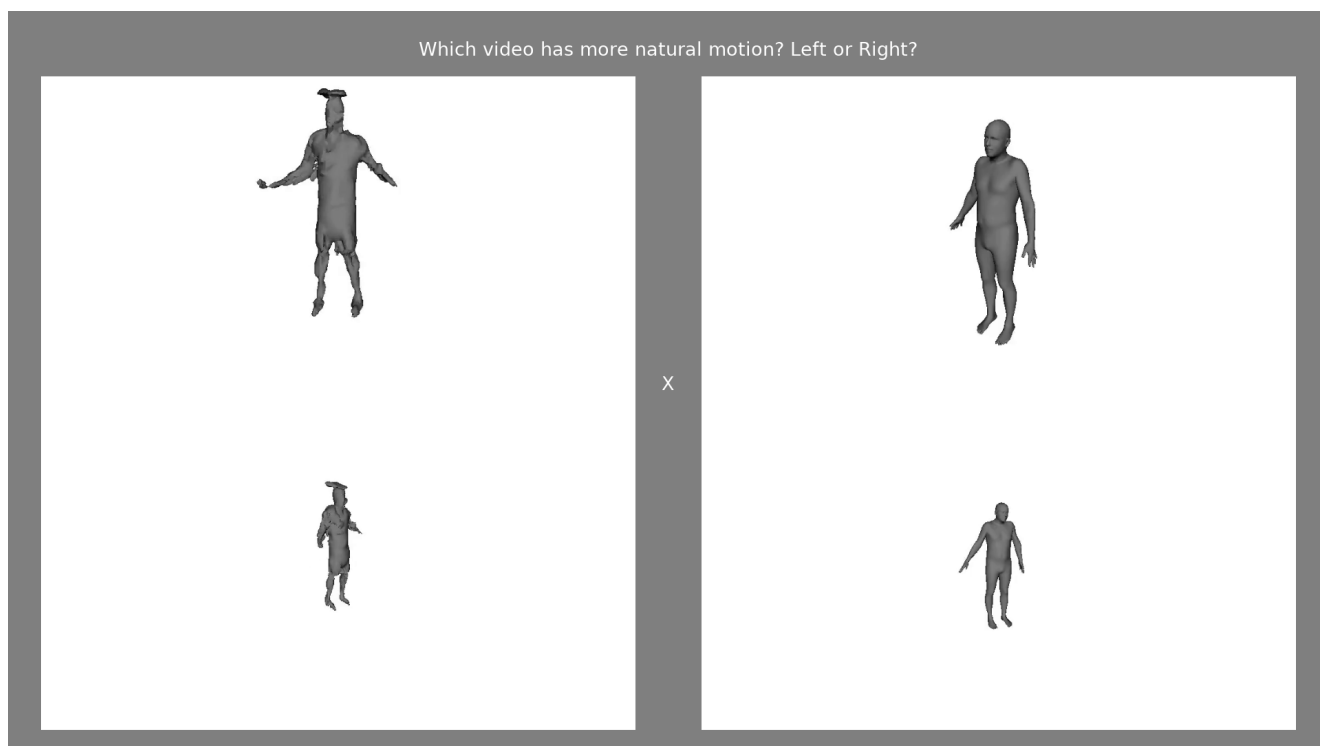


Figure 10. A screenshot of a trial for the 1st question in our user study.

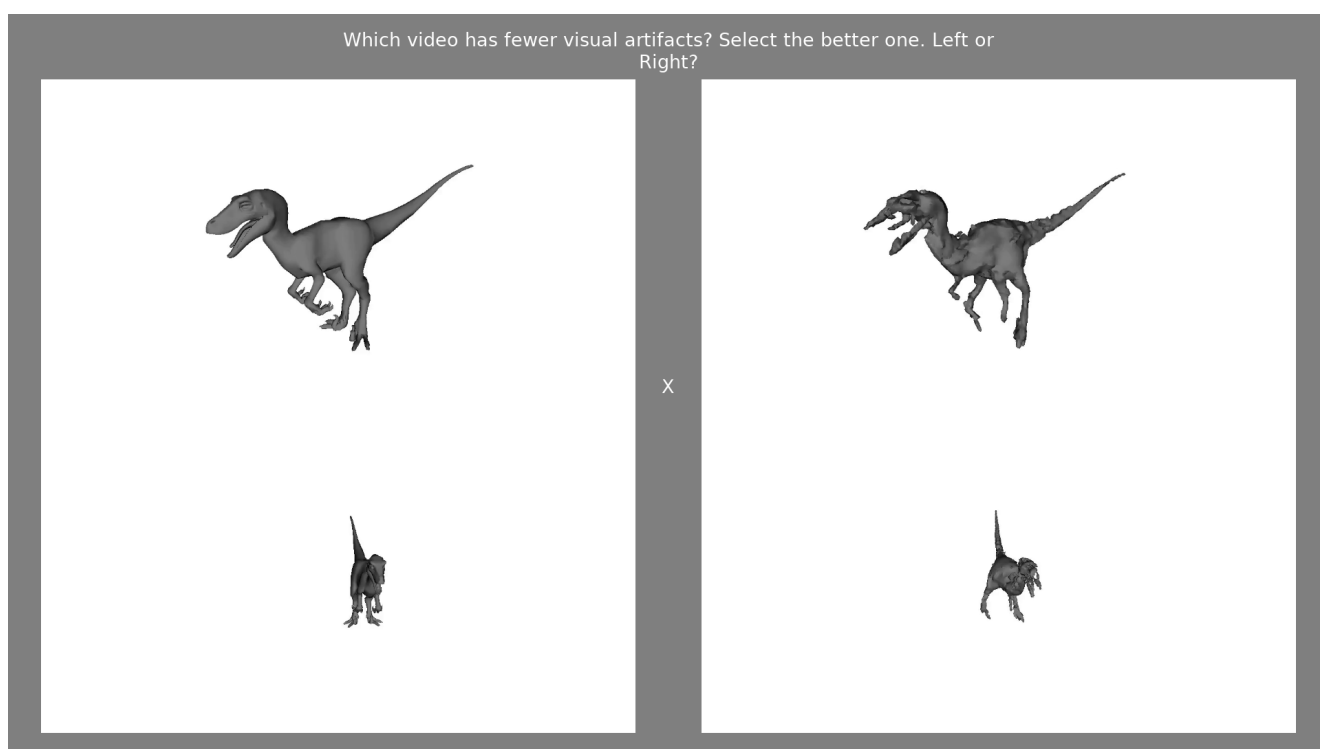


Figure 11. A screenshot of a trial for the 2nd question in our user study.

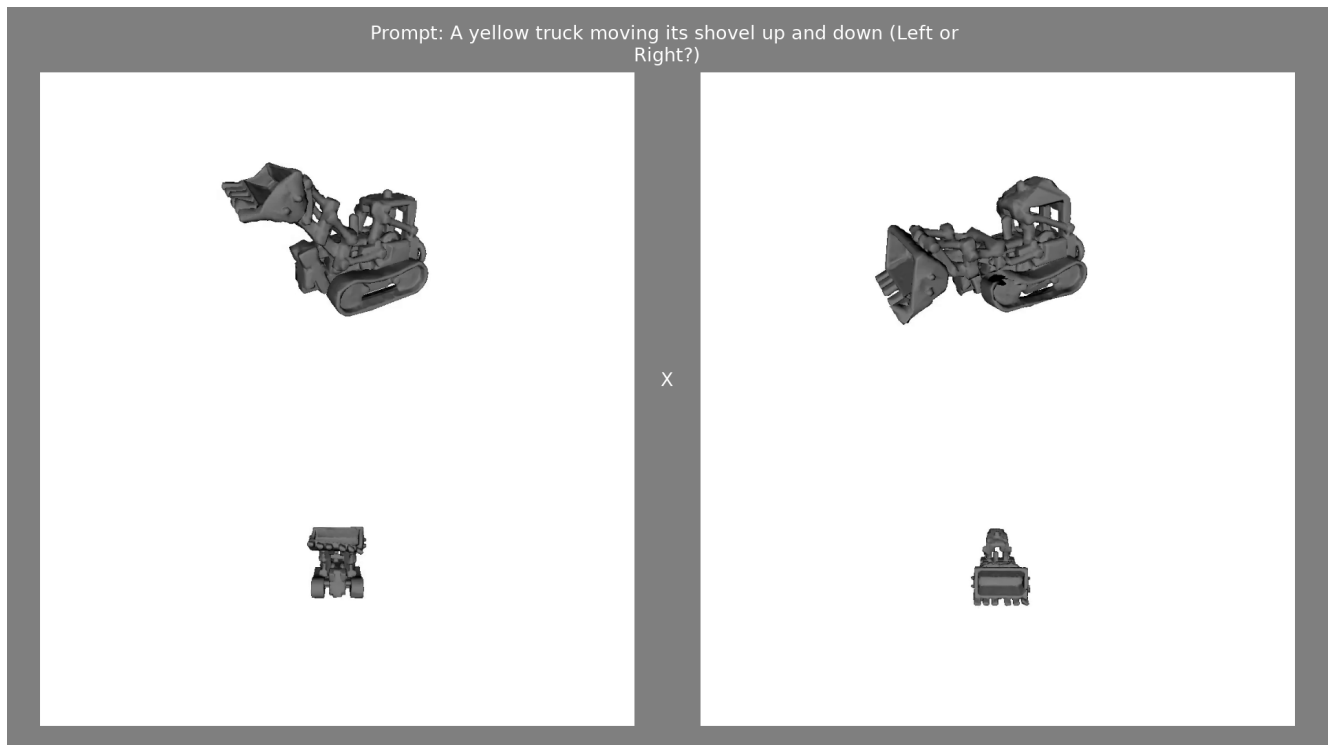


Figure 12. A screenshot of a trial for the 3rd question in our user study.

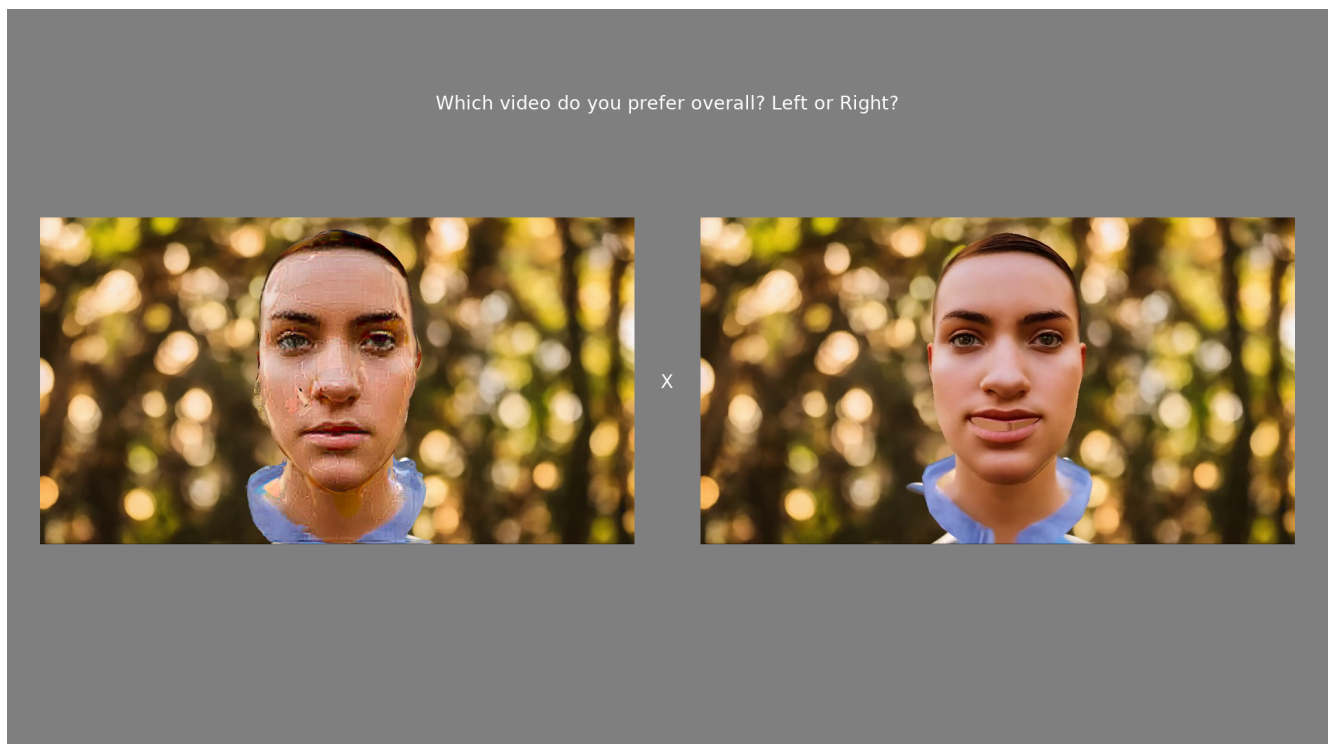


Figure 13. A screenshot of a trial for the additional 4th question in our user study.

Table 6. Breakdown of our study results showing a relative preference of our method in %. Q1: Which video has more natural motion? Q2: Which video has fewer visual artifacts? Q3: Which video captures the prompt better? Q4: Which video do you prefer overall?

	Q1		Q2		Q3		Q4	
	DG4D	MDM-MT	DG4D	MDM-MT	DG4D	MDM-MT	DG4D	VDM
<b>SMPL</b>	91.7	39.6	100.0	47.9	91.7	89.6	87.5	70.8
<b>SMAL</b>	54.2	64.6	100.0	100.0	77.1	64.6	85.4	31.3
<b>FLAME</b>	97.9	97.9	100.0	100.0	93.8	95.8	62.5	29.2
<b>Others</b>	93.8	55.2	100.0	89.6	84.8	47.9	95.8	56.3
<b>All</b>	86.25	62.5	100.0	85.4	86.3	69.2	85.4	48.8

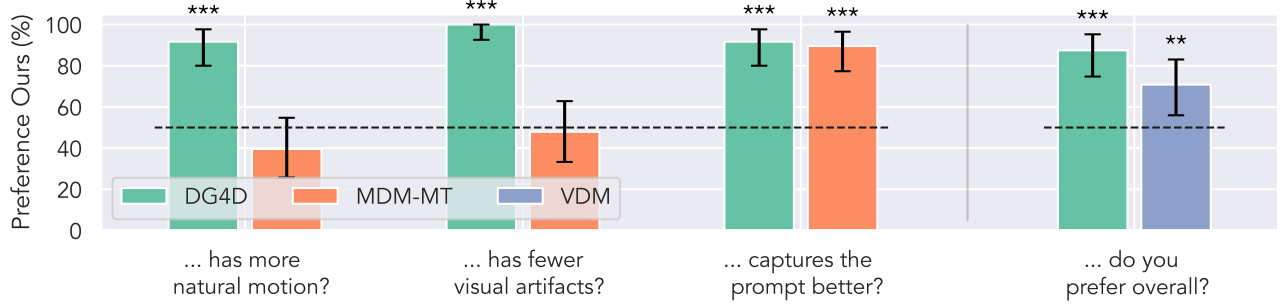


Figure 14. Study results for *SMPL* scenes only.

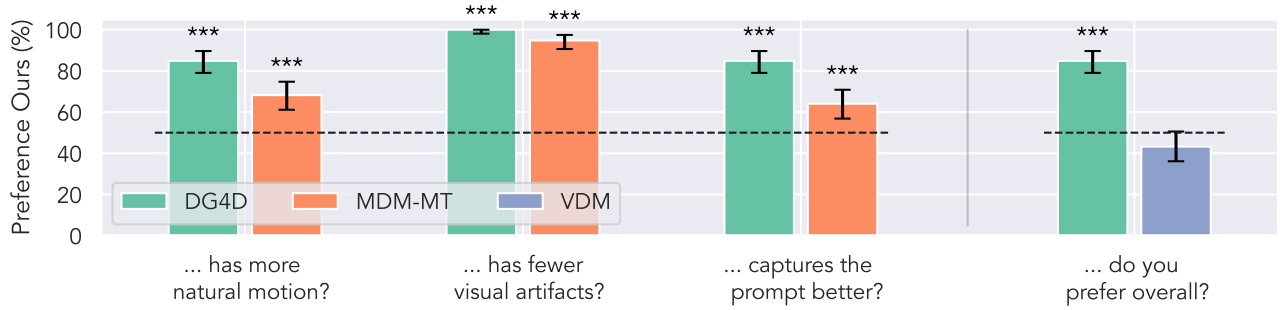


Figure 15. Study result *without SMPL* scenes.



Figure 16. Example of rendered AIST++ scenes. On the top: The untextured models. On the bottom: Models preprocessed by our single-view texturing.



Table 7. Evaluating different alignment strategies for WHAM.

	MPJPE	PA-MPJPE	PVE	Accel
<b>Textured (default)</b>				
WHAM <sub>align</sub>	.092 ± .038	.057 ± .015	.125 ± .047	8.0 ± 9.2
WHAM <sub>copy</sub>	.092 ± .038	.057 ± .015	.112 ± .046	8.0 ± 9.2
WHAM <sub>full align</sub>	<b>.059 ± .029</b>	<b>.042 ± .016</b>	.090 ± .039	<b>7.9 ± 9.0</b>
WHAM <sub>copy&amp;align</sub>	<b>.059 ± .029</b>	<b>.042 ± .016</b>	<b>.075 ± .036</b>	<b>7.9 ± 9.0</b>
<b>Untextured</b>				
WHAM <sub>align</sub>	.091 ± .037	.054 ± .014	.122 ± .043	7.4 ± 9.1
WHAM <sub>copy</sub>	.091 ± .037	.054 ± .014	.109 ± .044	7.4 ± 9.1
WHAM <sub>full align</sub>	<b>.057 ± .028</b>	<b>.039 ± .015</b>	.086 ± .038	<b>7.4 ± 9.1</b>
WHAM <sub>copy&amp;align</sub>	<b>.057 ± .028</b>	<b>.039 ± .015</b>	<b>.070 ± .035</b>	<b>7.4 ± 9.1</b>

## C.2. Methodology

We follow Tang et al. [80] to extract semantic features  $\hat{\mathbf{A}}$  from our rendered videos. First, we add noise corresponding to a diffusion inference step  $t$  to the encoded the rendered video  $\mathbf{x}$ :  $\mathbf{z}_t = \alpha_t \text{Enc}(\mathbf{x}) + \sigma_t \epsilon$ , where  $\text{Enc}()$  is a latent encoder for Latent Diffusion Models [71] or identity for RGB models. Then, we use  $\mathbf{z}_t$  as an input to the VDM denoiser  $f_\theta$  and obtain  $\hat{\mathbf{A}}$  as the U-Net activations in the same manner as in our main method (Sec. 4).

Note that we utilize MSE loss when using RGB features for optimization, as this results in better performance compared to the cosine distance.

## C.3. WHAM baseline

To offer a fair comparison, we evaluate four different alignment strategies for WHAM [75], because our method starts with the known pose  $\mathbf{p}_{\text{init}}$ . Results for either strategy can be found in Tbl. 7. In strategy *align*, we find the rotation and translation to align the wham output with the ground truth:

$$\hat{R} = R_{\text{wham}}^{0^T} R_{\text{gt}}^0, \text{ and } \hat{T} = \text{diag}(t)_{\text{wham}}^{0^{-1}} \text{diag}(t)_{\text{gt}}^0. \quad (6)$$

The transformations are then applied to the consecutive frames  $l$ :  $\tilde{R}_{\text{wham}}^l = \hat{R}^l R_{\text{wham}}^l$ , and  $\tilde{T}_{\text{wham}}^l = \hat{T}^l T_{\text{wham}}^l$ , where  $\tilde{R}$  and  $\tilde{T}$  are the new aligned root rotation and translation.

In *copy*, we copy ground truth root rotations and translations, i.e., we set  $\tilde{R}_{\text{wham}}^l := R_{\text{gt}}^l$  and  $\tilde{T}_{\text{wham}}^l := T_{\text{gt}}^l$ . In *full align*, we transform not only root rotations, like in Eq. 6, but every bone rotation. Lastly, in *copy&align*, we copy all root rotations and translation vectors from the ground truth and also transform the bone rotations as in *full align*. Note that the WHAM prediction is in full correspondence at  $l = 0$  in *full align* and *copy&align*. We find that *copy&align* performs the best for WHAM and, therefore, adopt this alignment strategy in Sec. 5.2.

## D. Additional Results

### D.1. Effect of Texturing and Background for Different VDMs

In Fig. 17 and Fig. 18, we compare different image input variants for the three different considered VDMs: VideoComposer [91] (VC), DynamiCrafter [93] (DC), and Stable Video Diffusion [5] (SVD). We observe that VC struggles to produce coherent output for images without background images, as the object often either disappears (Fig. 17 top) or gets distorted (Fig. 18 top). DC exhibits resilience to this problem and performs well both with and without a background image. Therefore, we opted to use images without background for DC, since it makes the videos more similar to the typical inputs of the DG4D baseline [68]. Finally, the publicly accessible SVD model is conditioned by image only without any text prompt input. We observe that the motion produced by SVD for our image inputs often results in a global camera motion with no object motion. This is particularly prominent if no background is used. For this reason, we excluded SVD from our other experiments.

### D.2. Qualitative Comparison to Consistent4D

We further compare our method to Consistent4D [28]. Since this is a computationally significantly more expensive method (50 minutes in its low VRAM setup versus less than 3 minutes for ours), which ended-to-end produces both the shape and the animation, we consider it a separate category from method which is better suited for quick motion prototyping and iterative animation development. This is why we did not include Consistent4D in our user study and instead provide a general discussion and a qualitative comparison here. A similar method DG4D was included in our study instead.

As seen in Fig. 19 and Fig. 20, our method generally produces more plausible motion given the underlying geometry in a faster manner. The evaluation of Consistent4D take on average 32 minutes per object in its low VRAM setup, while our motion fitting takes on average under 3 minutes on an NVIDIA RTX 3090. Note that we exclude the time it takes to generate the driving videos in both cases. Consistent4D’s slower runtime can be explained by its adoption of SDS which necessitates encoding the rendered RGB image into the latent space repeatedly. In contrast, our method remains in the semantic feature space.

Consistent4D utilizes a K-Plane [13] for their 4D representation which allows for a higher flexibility when modeling geometry distortions produced by the VDM. One such example can be seen in the raptor sequence in Fig. 19. Here, the raptor’s right and left legs rapidly alternate between the foreground and background, creating a sense of motion, but disrupting the raptor’s underlying topology. While the ability to fit such non-physical effects can be beneficial in some



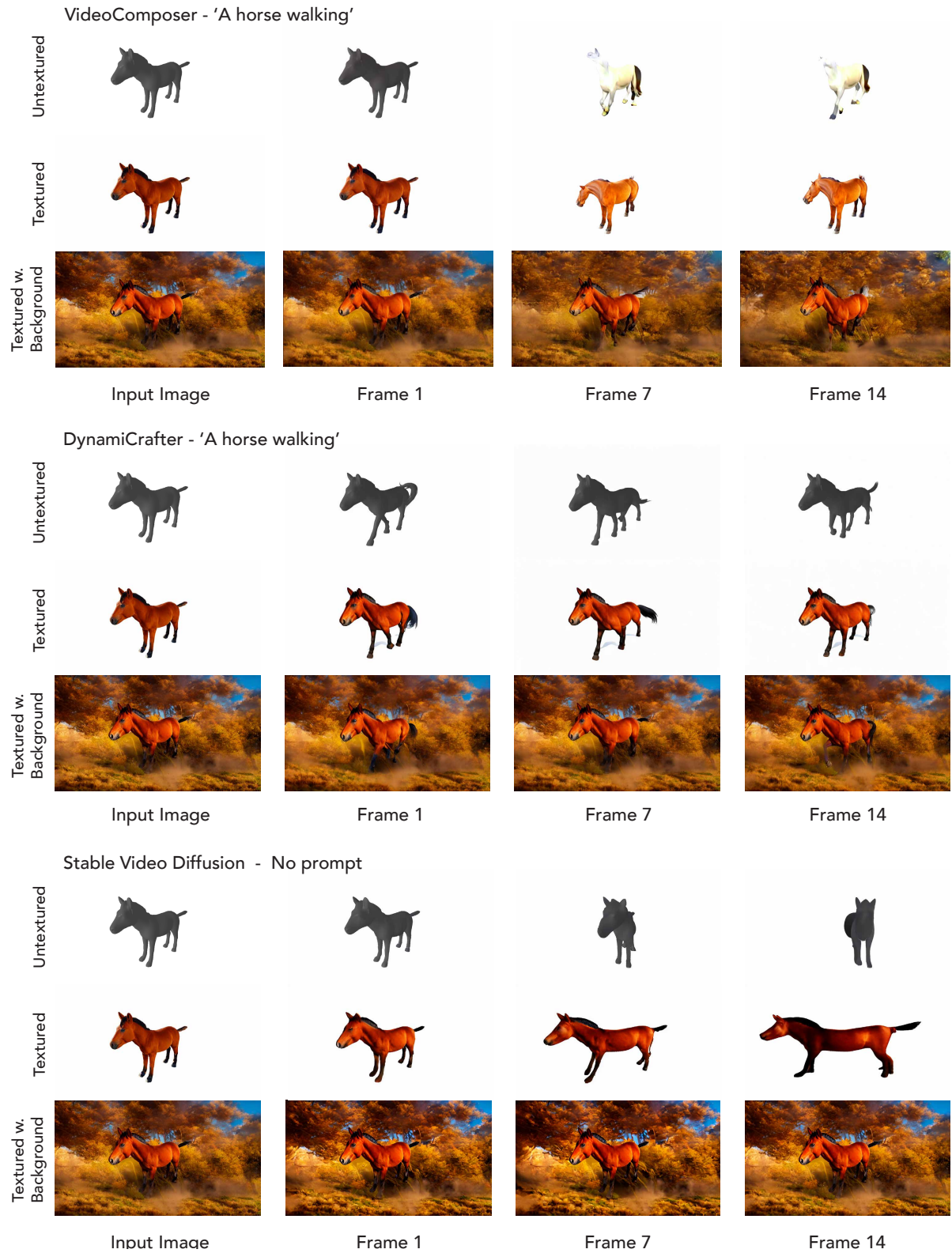


Figure 17. Comparison of 3 output video frames (columns 2–4) for 3 VDMs considered for our experiments given the same *Horse* 3D mesh (1st column) rendered (from top to bottom) as an untextured shaded image, single-view textured image and a single-view textured image with a synthesized background  $\mathbf{B}$  (Sec. 4.1 for details of the texturing process).



Figure 18. Comparison of 3 output video frames (columns 2–4) for 3 VDMs considered for our experiments given the same FLAME 3D mesh (1st column) rendered (from top to bottom) as an untextured shaded image, single-view textured image and a single-view textured image with a synthesized background **B** (Sec. 4.1 for details of the texturing process).

scenarios, our explicit representation is more robust to fitting to VDM artifacts and thus reduces the consequent visually implausible transformations.

Furthermore, the K-plane representation entangles shape and motion and hence it cannot be easily integrated into common computer-graphics pipelines. This is in contrast to our method which deforms an explicit canonical mesh, where the time-dependent vertex deformations can be easily exported.

Lastly, Consistent4D (as well as DG4D) adopts an image-to-3D model Zero-1-to-3 [44] for 3D-Uplifting. However, Zero-1-to-3 requires input images without background which contradicts our observations that VDMs benefit from context in the background for better results (see in Appendix D.1). To tackle this, Consistent4D creates masks in an automated fashion for videos with background which can, however, introduce additional errors. In comparison, our method does not rely on any such masks.

### D.3. Comparing Semantic Featuring against RGB for Optimization

Here, we complement our Pose estimation experiment from the main paper and compare the RGB and semantic features end-to-end in our full pipeline. We find that modeling temporal deformations with NJF in combination with RGB features results in unstable optimization. Therefore, we only showcase the kinematic models in Fig. 21. Similarly to pose fitting, semantic features lead to superior results.

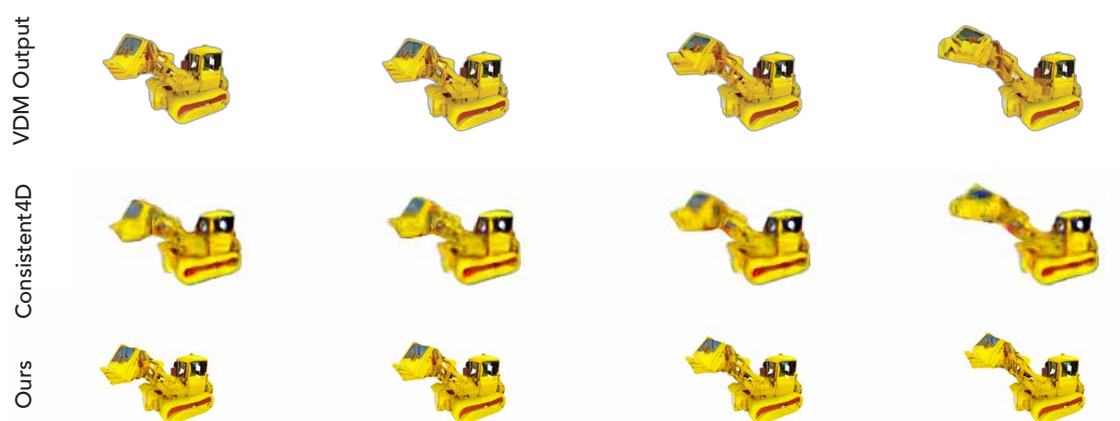
### D.4. Ablation of Number of Vertices

Fig. 22 and Fig. 23 show additional results when varying the number of vertices in our method with NJF. We find that the output quality degrades gracefully and predictably, when scaling down from 4000 to 500 vertices. Notice that the VDM output slightly differs, due to the variations in the conditioning input image when varying the number of vertices.

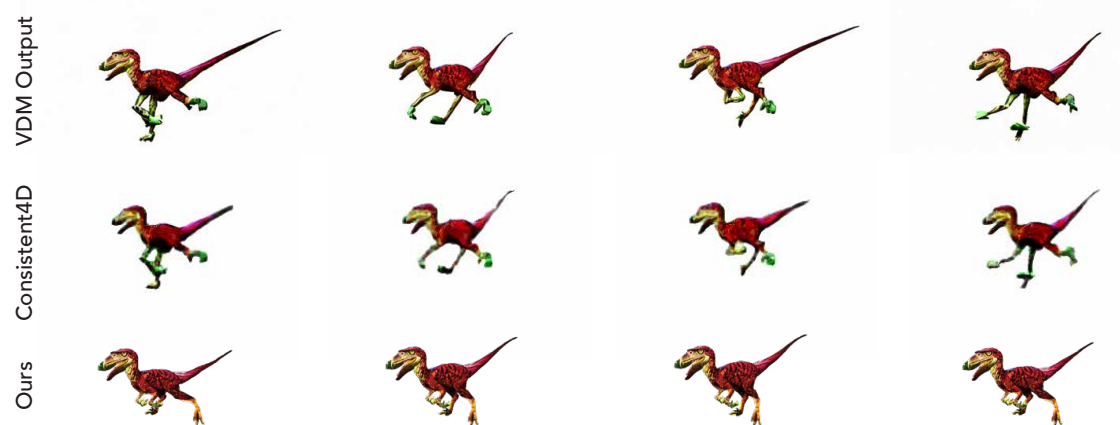
### D.5. Results with Stable Video Diffusion

As reported in Appendix D.1, SVD produces camera motion rather than object motion. For completeness sake, we show that our method produces plausible results in Fig. 24 when fitting to the SVD semantic features.

'A truck moving its shovel up and down' (NJF)



'A raptor walking' (NJF)



'An orc laughing' (NJF)

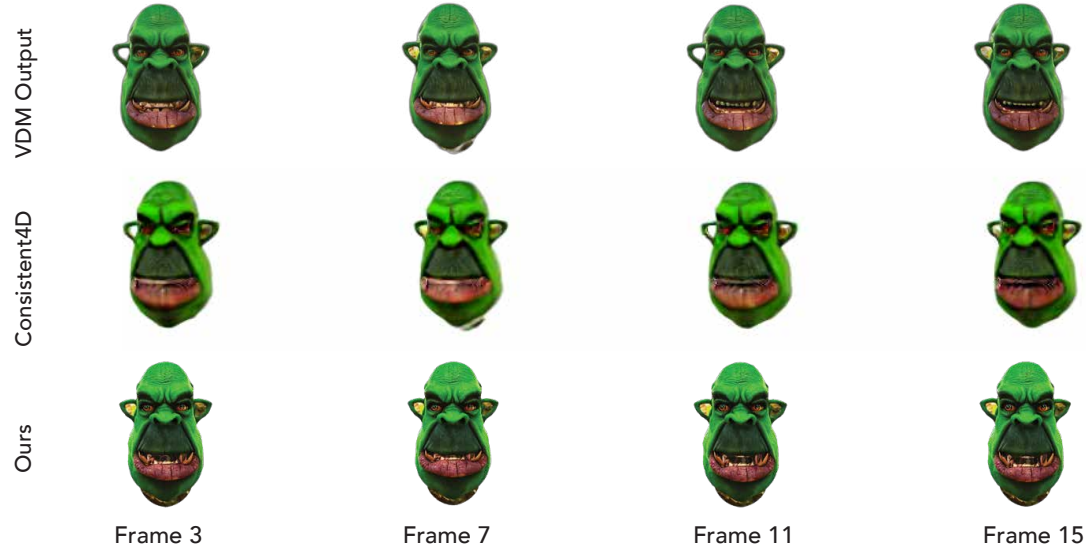


Figure 19. Comparing Consistent4D against our method.



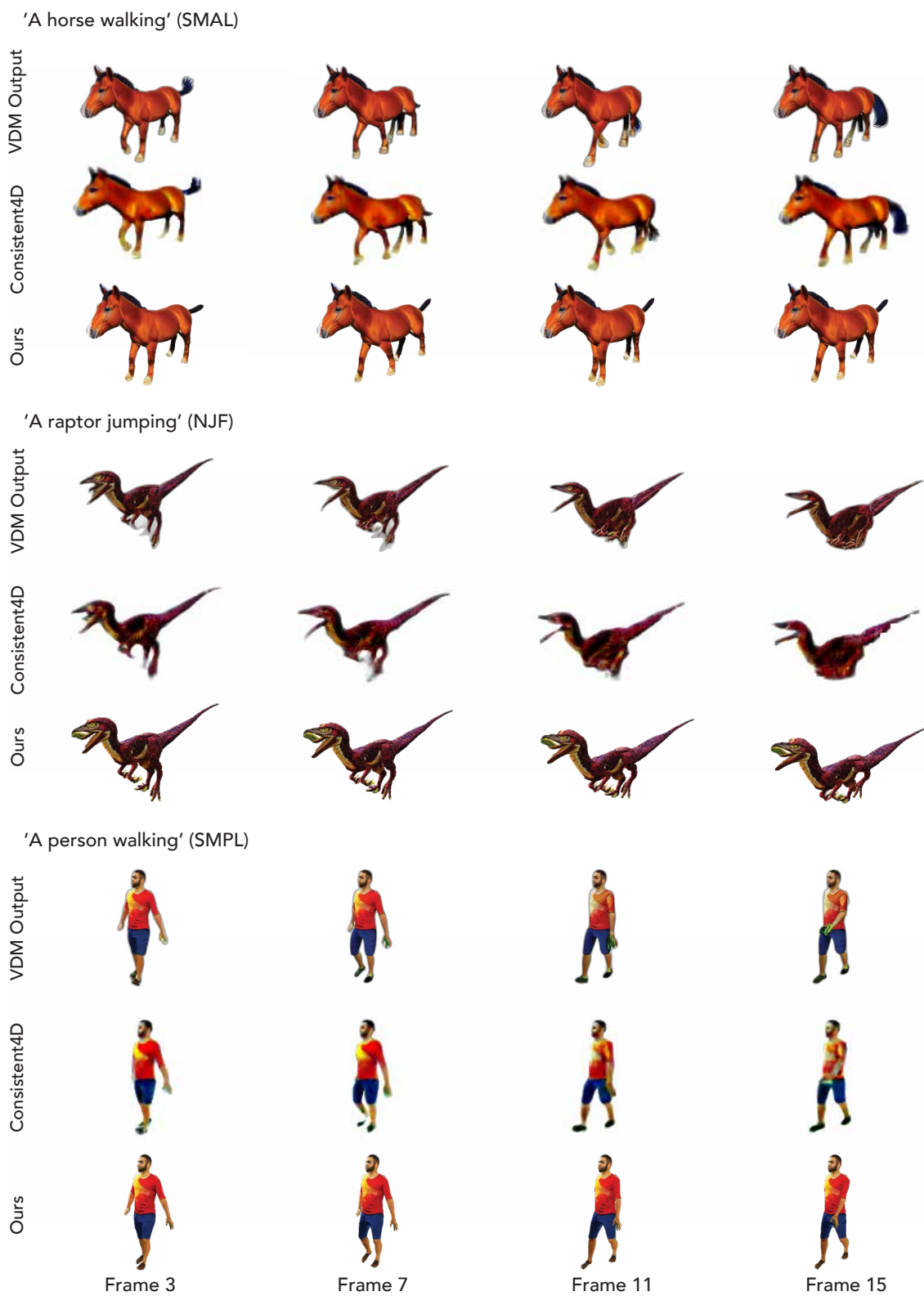


Figure 20. Comparing Consistent4D against our method.

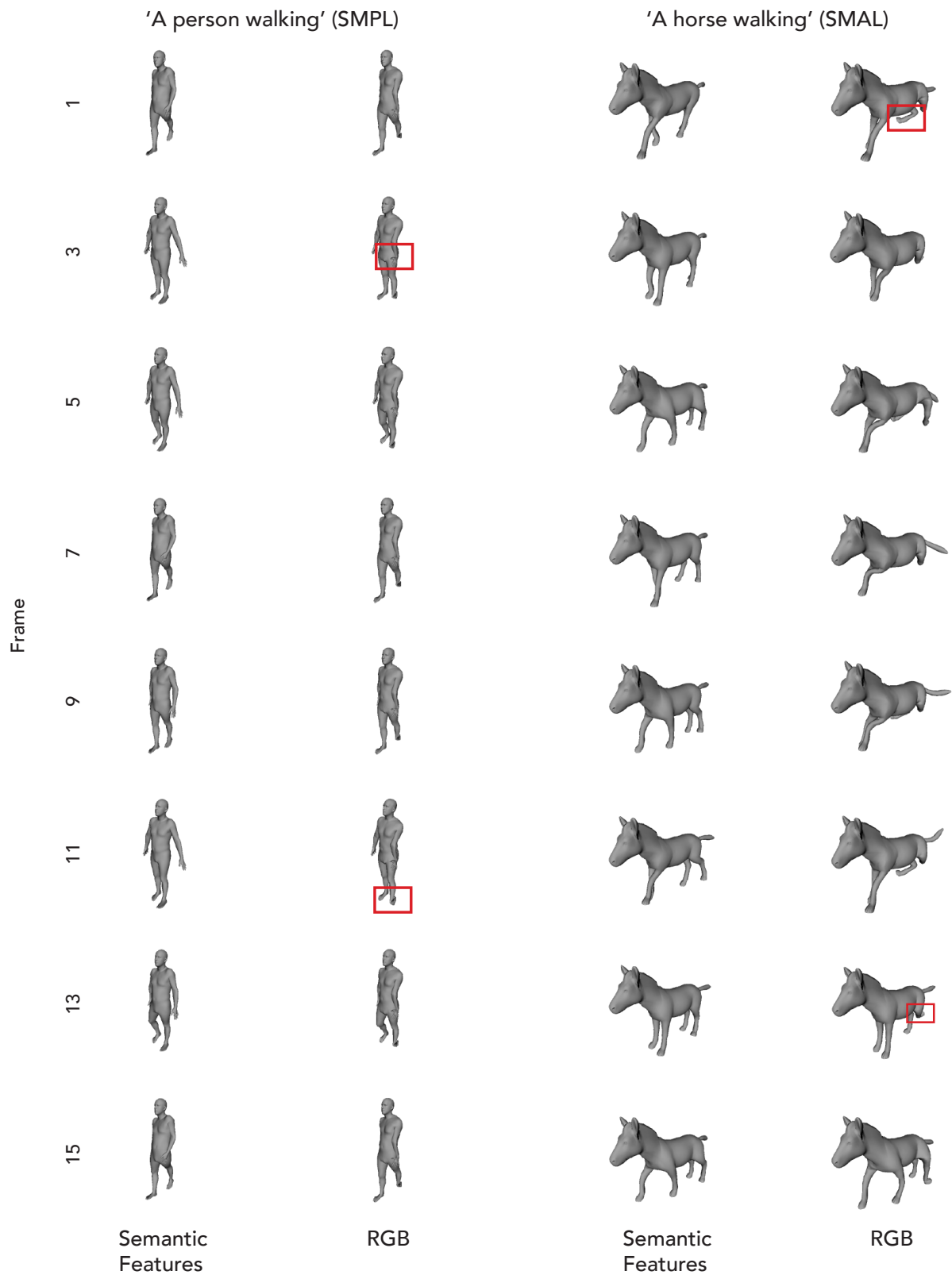


Figure 21. Comparison of semantic features against RGB used for pose optimization. Each column shows frames for a single sequence. Note the red boxes, highlighting errors in the the pose optimization when utilizing RGB: 1) In case of SMPL (human), the hand gets stuck in front of the torso, as the RGB features do not distinguish the body from the hand. 2) In case of SMAL (horse), the limbs of the horse assume less realistic articulation with RGB features.

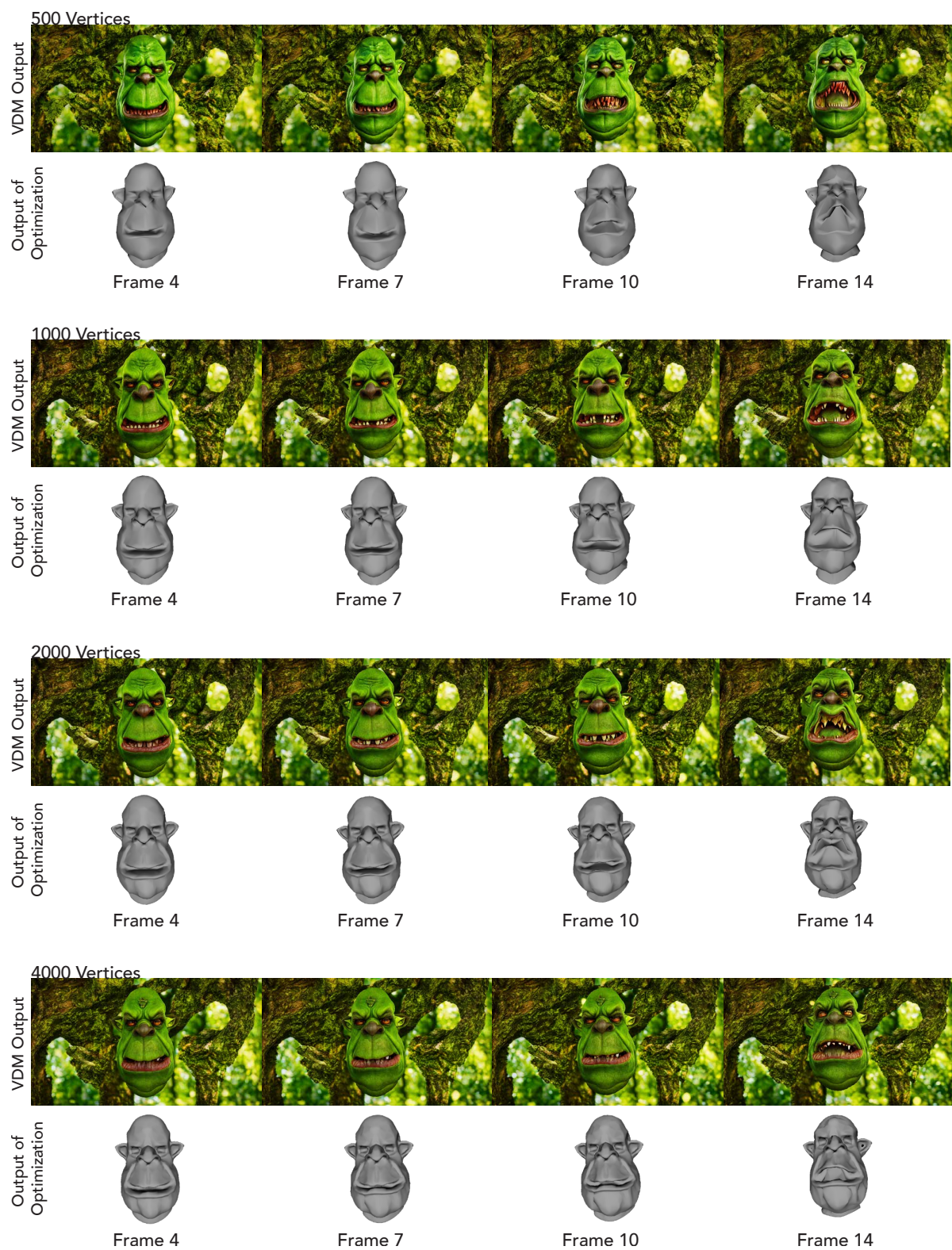


Figure 22. Effect of vertex number on our method when adopting NJF for deformations with VC as VDM backbone. Each row shows the output of our method with an increasing number of vertices. Prompt: 'An orc laughing.'



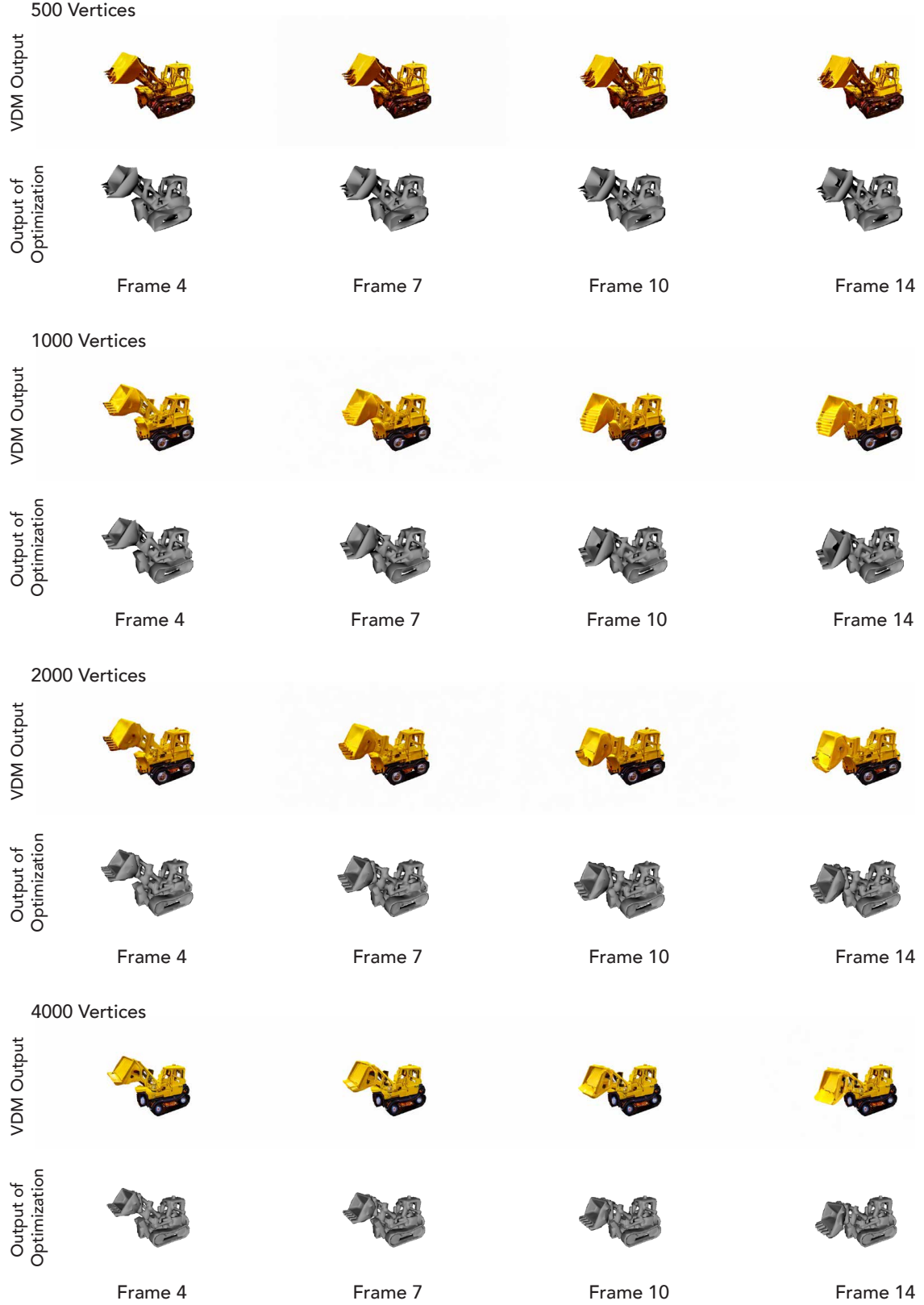


Figure 23. Effect of vertex number on our method when adopting NJF for deformations with DC as VDM backbone. Each row shows the output of our method with an increasing number of vertices. Prompt: 'A truck moving its shovel up and down.'

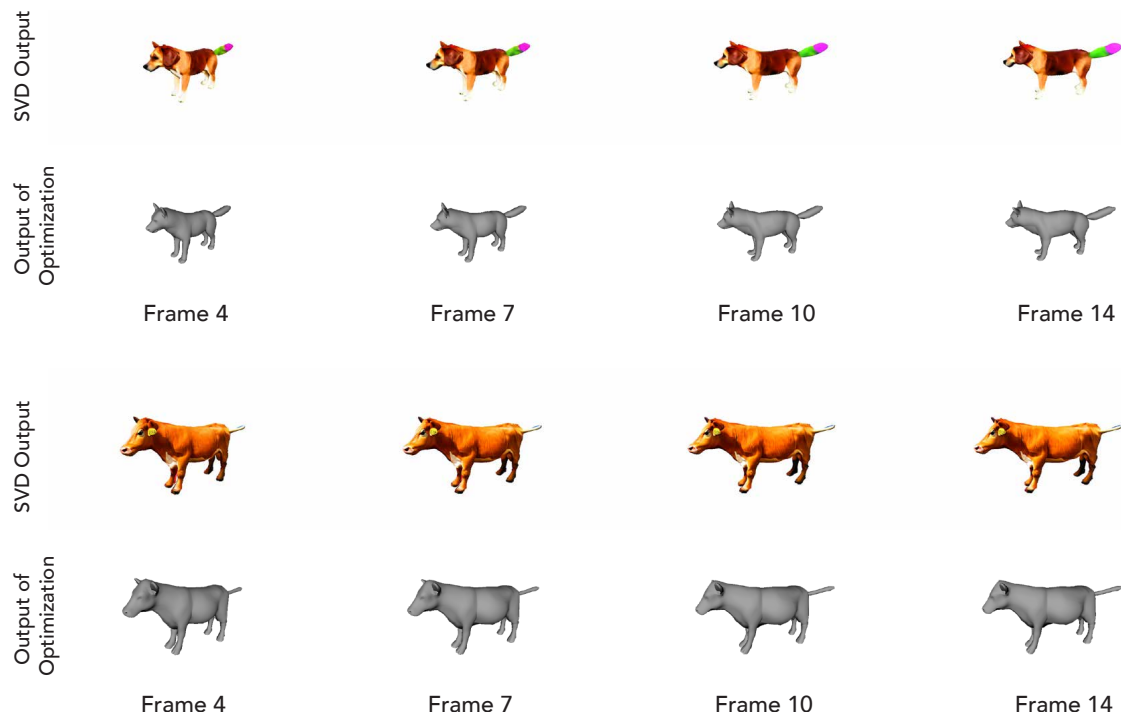


Figure 24. Results when motion fitting using the SVD semantic features. While our method is well suited to work even with SVD features, SVD videos tend to focus on a global camera rotation rather than actual object motion. Consequently, the fitted object motion is often minimal or uninteresting. As a result we omitted SVD in our further studies in favor of other VDMs.