# Appendix: Pippo: High-Resolution Multi-View Humans from a Single Image

## A. DiT Architecture and Training

**Architecture.** We use a Diffusion Transfomer with 28 DiT + ControlMLP blocks (depth) in Pippo operating at 1536 dimension. The DiT blocks account for 1.3B parameters and ControlMLP blocks for 248M parameters respectively. We use latent autoencoder which provides 8x spatial downsampling, and 4-channel latent space. It contains 101M parameters. We find the placement of the ControlMLP block modulation in DiT to be crucial, specifically, being placed *before* the scale, shift, and gate is important for the control signal to work well.

**Training Stages.** We pre-train (P1@256) our model on 512 A100 GPUs for 1M iterations, with batch size of 24 per GPU and at $256 \times 256$ resolution. 1M on base model, and 700K iterations on 512x512 resolution. During pretraining Pippo, we use image-only conditioning via DinoV2 [52] using ViT-L/14 based models to provide weak supervision and alignment for target generations. We mid-train our model on Full-body and Head-only datasets at $128 \times 128$ resolution for nearly 100K steps, and post-train our models with spatially-aligned conditions at $512 \times 512$ and $1024 \times 1024$ for nearly 50K steps. We pre-train, mid-train, and post-train for roughly 2:1:1 weeks, respectively.

**Optimizer and Hyperparameters.** We use LR of $1e$-4 during all stages and AdamW optimizer with beta values set to $[0.9, 0.98]$ and epsilon to $1e$-6. We use linear warmup for initial 1000 steps starting at $1e$-6. We conduct training at full precision (float32), and run inference at half precision (bfloat16).

## B. Handling Varying Number of Views at Inference

As discussed in Sec. 3.2, during training we jointly denoise a fixed number of views. Specifically, 24 views for mid-training at $128 \times 128$ , and 12 or 2 views for post-training at resolutions $512 \times 512$ and $1024 \times 1024$ respectively. This choice is largely motivated to avoid GPU out-of-memory errors during training. However, during inference, we wish to scale the number of views much further to generate smooth turnaround videos. This is feasible because we can run inference at half precision (using bfloat16) and do not need the backprop computation graph to be stored.

We find that simply scaling the number of views (or tokens) during inference beyond 2x of the number of views during training leads to blurry and degraded generations. We find these degradations to be most significant in regions unspecified in the input, for example, the back of the head or ears as shown in Fig. 8. We investigate this issue next, and introduce Attention Biasing to remedy it.

**Attention Biasing.** Let $\mathbf{X} \in \mathbb{R}^{N \times d}$ denote the sequence of tokens denoised jointly, where $N, d$ represent number of tokens and the token dimension, respectively. In the total number of tokens in the sequence is proportional to number of views since the VAE and Patchify jointly compress the image by 16x in height and width. Within each DiT block, we compute the $\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{AV}$, where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are query, key, value matrices and $\mathbf{A}$ are attention scores computed via taking a row-wise softmax as follows:

$$\mathbf{A}_{i,j} = \frac{e^{\lambda \mathbf{Q}_i \mathbf{K}_j^\top}}{\sum_{j'=1}^{N} e^{\lambda \mathbf{Q}_i \mathbf{K}_{j'}^\top}}, \tag{3}$$

Where the scaling factor $\lambda$ was proposed to be set to $1/\sqrt{d}$ by the original work [82] to stabilize the softmax operation as $d$ increases and avoiding biased (sharp) softmax distributions. We can quantify the sharpness of the attention by computing its entropy. Following previous works in NLP [2, 23, 104], and in vision [35] we define entropy of attention as:

$$\text{Ent}(\mathbf{A}_i) = -\sum_{j=1}^{N} \mathbf{A}_{i,j} \log(\mathbf{A}_{i,j}), \tag{4}$$

By substituting the equation (3) in equation (4), authors of prior work [35] meticulously derive that entropy of attention grows logarithmically with number of tokens as follows: $\text{Ent}(\mathbf{A}_i) \propto \log N$. Furthermore, the authors show that this growth in entropy can be offset during inference by growing the scaling factor $\lambda$ as follows:

$$\lambda = \sqrt{\frac{1}{d}} \cdot \sqrt{\frac{\log N_i}{\log N_t}} \tag{5}$$

Where $N_t, N_i$ denote the number of tokens during training and inference respectively. For more details, please refer to Sections 3.1 and 3.2 of the original work [35]. Empirically, we find that having a slightly faster growing $\lambda$ alleviates the degradation better. Hence, we propose a hyperparameter $\gamma$ (growth factor) that is tuned between the range $[1.0, 2.0]$ to

control the growth of $\lambda$ as follows:

$$\lambda_{\texttt{ours}} = \sqrt{\frac{1}{d}} \cdot \sqrt{\frac{\gamma \cdot \log N_i}{\log N_t}} \tag{6}$$

We follow the above Equation (6) with our growth factor hyperparameter $\gamma$. In Fig. 7, we plot the entropy (Y-axis) during an inference pass of Pippo across varying number of views (tokens) being denoised and demonstrate the corresponding growth in entropy. Furthermore, we also show the attenuation in the entropy under increasing growth factors (X-axis).

We put generated visuals before and after using the suggested attention biasing in Fig. 8. We put visuals sweeping over more values of the growth factor in Appendix Fig. 9. Similar techniques have been explored in LLMs for handling and generating text at longer contexts [56, 83], where the scaling factor $\lambda$ mentioned above is analogous to the inverse of the temperature scale.

Additionally, we also found that using a bump function instead of constant Classifier-free Guidance during generation leads to fewer artifacts. We discuss this trick further in Appendix C.

## C. Diffusion Model and Inference Settings

We use classifier-free guidance(CFG) [30] during training, with 20% dropout probability on input reference image. We use 50 DDIM steps for sampling. During inference, we find that lower CFG scales between $3-5$ work better at higher resolution and generating viewpoints that are closer to input view, this observation is inline with Stable Video Diffusion [5] which proposed to linearly increase CFG scale after the first frame generation. Inference speed with Pippo depends on the resolution and number of views to be denoised; we report few combinations in Tab. 8.

| # | Method (Stage @ Resolution) | Views | Speed (sec) ↓ |
|---|---|---|---|
| 1. | Pretrained (P1@256) | 1 | 2.51 |
| 2. | Pretrained (P1@512) | 1 | 2.59 |
| 3. | Mid-trained (M2@128) | 4 | 6 |
| 4. | Mid-trained (M2@128) | 48 | 14 |
| 5. | Post-trained (P3@512) | 4 | 40 |
| 6. | Post-trained (P3@512) | 48 | 490 |
| 7. | Post-trained (P3@1K) | 4 | 185 |
| 8. | Post-trained (P3@1K) | 12 | 622 |

Table 8. **Inference Speed of Pippo.** We show inference speed without any optimizations (using bfloat16) against varying resolution and number of views being generated.

**Attention Biasing.** To compute the entropy (shown in Fig. 7), we use the first, middle, and last blocks of Pippo; and aggregate over all attention heads, conditional and unconditional inference passes, and DDIM steps. In Fig. 9, we show visuals for increasing strength of the scaling growth factor ($\gamma$) when generating 60 views simultaneously at 512x512 resolution. It is evident that using this attention biasing is quite crucial in making diffusion models generalize across many views (long context sequences). Growth factor ($\gamma$) greater than 1.0 helps mitigate the entropy buildup; however, increasing $\gamma$ beyond 1.6 leads to over-saturation artifacts somewhat akin to ones caused by high CFG scale.

**Varying Classifier-free Guidance (CFG) using a bump function.** Classifier-free Guidance guidance enables a trade-off between diversity and realism [30]; with higher values of CFG resulting in diverse generations (i.e., higher Inception Score) and lower values of CFG leading to phtorealistic generations (i.e., higher FID) In our setup, the single image to multi-view task involves faithfully preserving known content while also hallucinating diverse possibilities of unseen regions. The amount of known content and unknown content varies for each view that is being generated; for example, the back of the heads are often unseen; however, central parts of the face, such as the nose, are generally specified in the frontal input image. We can use this information to rescale the CFG weight for each view separately. Specifically, we find that using a lower weight for regions where content copying is required prevents saturation artifacts, whereas increasing the CFG scale of unseen regions leads to more stable and diverse generations. Thus, we increase the CFG linearly starting from the front facing view at $0°$ azimuth until $90°$ azimuth (side-view) where it reaches its peak value. Then we keep the CFG scale fixed at this peak value until the azimuth reaches $270°$ (opposite side-view), and finally decrease it linearly back to starting value at azimuth of $360°$. This is a bump function and we find that starting with CFG scale between $[7.0, 9.0]$, and having a peak CFG scale between $[15.0, 19.0]$ results in reduced artifacts and diverse generations especially in the unseen regions. A similar trick is also used in image-to-video work of SVD [5].

**Rescaling diffusion timesteps under varying resolution.** Prior works [10, 19] suggest that as resolution increases the noise scale has to be shifted to ensure same level of corruption. Based on this Stable Diffusion 3 [19] derive a noise reweighing scheme by assuming degradation to a constant-pixel image and ensuring that the uncertainity under degradation for each pixel stays constant. Since SD3 uses conditional flow-matching objective and we train Pippo using the DDPM [31] objective, we cannot use their reweighing scheme directly. Here, we provide a derivation for an equivalent reweighing scheme for DDPM objective. We can define forward process as:

$$z_t = \sqrt{\alpha_t} z_0 + \sqrt{1-\alpha_t}\epsilon,$$

where $\alpha_t$ is a monotonically decreasing function of $t$. The uncertainty in DDPM is governed by the variance of the

forward process, which depends on $\alpha_t$ and $1-\alpha_t$. Consider a constant image $z_0 = c\mathbb{1}$, where $c \in \mathbb{R}$ and $\mathbb{1} \in \mathbb{R}^n$. The forward process in DDPM produces:

$$z_t = \sqrt{\alpha_t} c\mathbb{1} + \sqrt{1-\alpha_t}\epsilon,$$

where $\epsilon \sim \mathcal{N}(0, I)$. The uncertainty in estimating a constant-valued image $c$ is, where $n$ are total number of pixels in the image:

$$\sigma(t, n) = \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}} \cdot \frac{1}{\sqrt{n}}.$$

To map a timestep $t_n$ at resolution $n$ to a timestep $t_m$ at resolution $m$ such that the uncertainty $\sigma(t_n, n) = \sigma(t_m, m)$, we solve:

$$\frac{\sqrt{1-\alpha_{t_n}}}{\sqrt{\alpha_{t_n}}} \cdot \frac{1}{\sqrt{n}} = \frac{\sqrt{1-\alpha_{t_m}}}{\sqrt{\alpha_{t_m}}} \cdot \frac{1}{\sqrt{m}}.$$

Rearranging, we get the resolution-dependent timestep mapping for DDPM isolates $\alpha_{t_m}$ as:

$$\alpha_{t_m} = \frac{\alpha_{t_n}}{\frac{m}{n} + \alpha_{t_n}\left(1 - \frac{m}{n}\right)}.$$

We use the above reweighing to rescale noise steps when training models at a higher resolution of $512 \times 512$ and $1024 \times 1024$ during post-training. In practice, we set $m/n$ ratio to be slightly lower than the actual value following SD3 [19].

## D. Humans-3B : Filtering and Stats

We run image metadata filtering to keep images whose short edges are at least 720 pixels and file sizes are at least 120 KB. We run Detectron2 [92] (pose detection) to keep images containing one clearly detected person (detection score at least 0.9 and the secondary clear person detection score is at most 0.4) with heads at least partially visible, and the long edge of the detected bounding box is at least 300 pixels. We also use a custom person realism classifier to drop computer-generated or computer-processed imagery. We provide rough statistics of our curated human-centric dataset in Fig. 10. We bucket these attributes in bins along X-axis and plot their respective sizes (normalized between $0-1$) on Y-axis.

## E. Visuals Webpage [Link]

**Webpage.** We upload a supplementary webpage which contains 360° turnaround videos from our model generated for Full-body, Head-only and Face-only settings. Additionally, we put visuals where we provide as input a monocular video (framewise), and generate frames independently at each timestep. We find Pippo preserves the known details while hallucinating plausible unseen parts well. We put the visualization of our spatial anchor and corresponding generations.

## F. Why the name Pippo?

**Filippo Brunelleschi.**

Our model is named after Filippo di ser Brunellesco di Lippo Lapi (1377 – 15 April 1446), widely recognized as Filippo Brunelleschi and affectionately known as Pippo by Leon Battista Alberti. Brunelleschi was an Italian architect, designer, goldsmith, and sculptor. He pioneered the application of vanishing points in artwork to achieve accurate perspective vision. Similarly, our model employs a single 3D spatial anchor to produce consistently improved images.

## G. Visuals from Pretrained Model (P1 )

In Fig. 11, we showcase qualitative visuals related to the findings in Table 2. It is evident that the model trained with filtered data and using an image-conditioned objective produces high-quality human figures.

## H. Frequently Asked Questions

**Ablation with Missing or Inconsistent spatial anchor.** Spatial Anchor acts as a placement signal for Pippo since we do not provide it with intrinsics or extrinsics of the input image. In Fig. 14, we run inference on the Head-only P1@512 model with missing spatial anchor, or when it is inconsistent with input head pose (rotated downwards at the floor by 90°). When the spatial anchor is missing, Pippo often generates an empty image because in our training data it implies that the subject's head is not visible in the generated view. We find that Pippo is robust to anchor rotations; this suggests Pippo relies on the spatial anchor only for placement control and infers head-pose from the input image.

**Can we reconstruct Pippo outputs?** Yes, we can reconstruct Pippo's generations using a NeRF or GSplat.

**Why use spatial-control in post-training only?** We find that skipping pixel-aligned controls in mid-training helps us to train faster and allows training jointly on a greater number of views. Additionally, since we mid-train at the low resolution of $128 \times 128$ , we find that pixel-aligned control needs to be re-injected during post-training again.

**Is Reprojection Error (RE) pairwise, and can it handle occlusions?** Yes, we compute *the mean* Reprojection Error over pairs of images. We divide the generated images randomly into non-overlapping pairs and compute pairwise RE. We re-project the triangulated 3D points back to each of the two images to compute the RE. We only use high-quality correspondence matches by setting a threshold of $> 0.2$ in SuperGlue, and reject image pairs that have fewer

than $< 5$ matches detected. This filtering helps to avoid spurious correspondences in distant and occluded views. We will release the code for metric.

**Trends in overfitting experiments (in Sec. 3.3) may change under largescale training.** Empirically, we found that the ability to generalize to novel viewpoints of a single scene under overfitting is correlated with a greater ability to steer the camera viewpoint and place the subject precisely after post-training. For example, in Tab. 4, Row 5 we can see that removing the spatial anchor drops the 3D consistency of the post-trained model the most and is correlated with the overfitting result in Tab. 1, Row 6. The distribution of cameras in overfitting remains similar to that of full training. Thus, we use overfitting as a proxy to compare existing spatial control modules cheaply.


Figure 5. **High Resolution Multi-view Generation of Unseen subjects.** Pippo enables generation of high-resolution 1K images given only a single image as input (left most in each block, separated with a dashed line). First row LHS shows generation from mobile captured photo, while the RHS shows unseen studio subject. Second row shows mobile captured face-only generations. Third and fourth row shows unseen studio subjects. Last two rows demonstrate simultaneous generation of 10 novel views given unseen studio subject.


Figure 6. **Pippo can handle occluded inputs.** We show Pippo's generations given incomplete input images – such as partially or fully occluded faces, unobserved t-shirt designs on test split of Full-body dataset. We show corresponding ground truth separated with blue dotted line – it can be seen that Pippo faithfully follows the known content while auto-completing unseen segments.

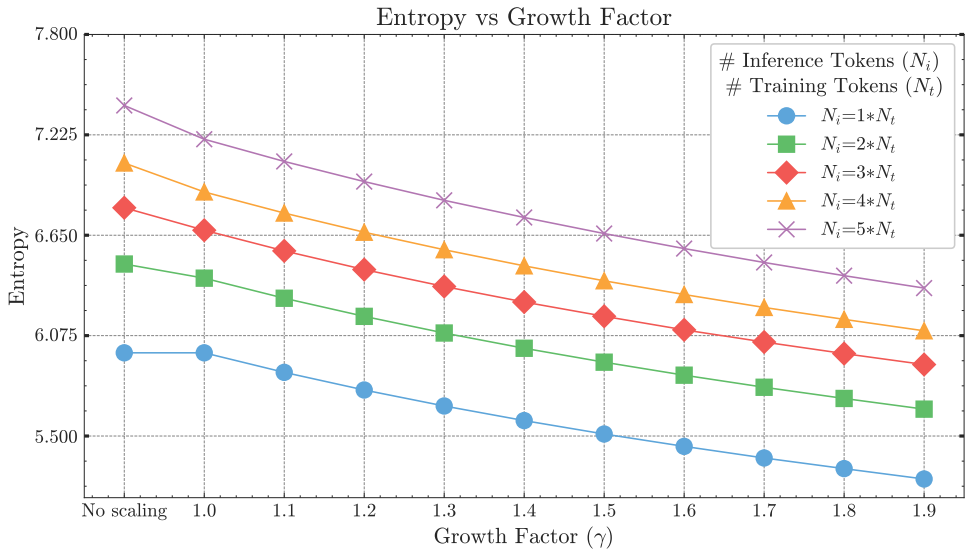


Figure 7. **Entropy vs Growth Factor ($\gamma$) for varying number of views (tokens) (Appendix B).** We present the entropy results (Y-axis) from our Attention Biasing technique inspired from [35] for varying number of tokens (individual line plots), and across different scaling growth factor $\gamma$ introduced in Eq. (6) (X-axis). On X-axis, "No scaling" refers to the default attention formulation [82] and $\gamma = 1.0$ refers previous work [35] formulation. Empirically, we find that a slightly higher value of $\gamma = 1.4$ leads to best visuals.

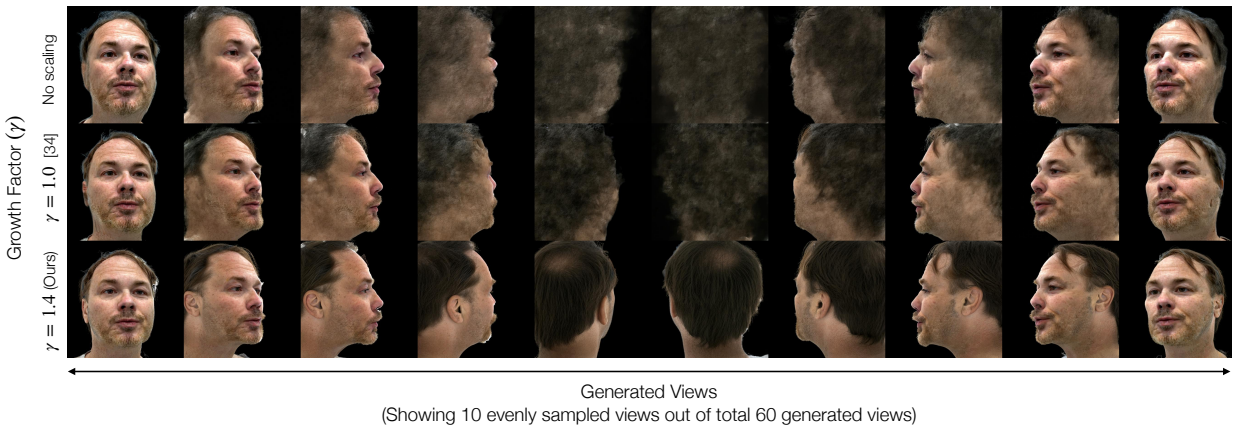


Figure 8. **Generations under varying strengths of growth factor $\gamma$ (Appendix B).** On each row we show the generated views across vanilla attention [82] (No scaling), prior work [35] and our formulation Eq. (6). It can be seen that growth factor ($\gamma$) greater than 1.0 is crucial to mitigate the entropy buildup. We show only 10 views per row subsampled evenly from 60 views generated at $512 \times 512$ resolution. The model was trained to jointly denoised only 12 views ($N_i = 5 * N_t$).

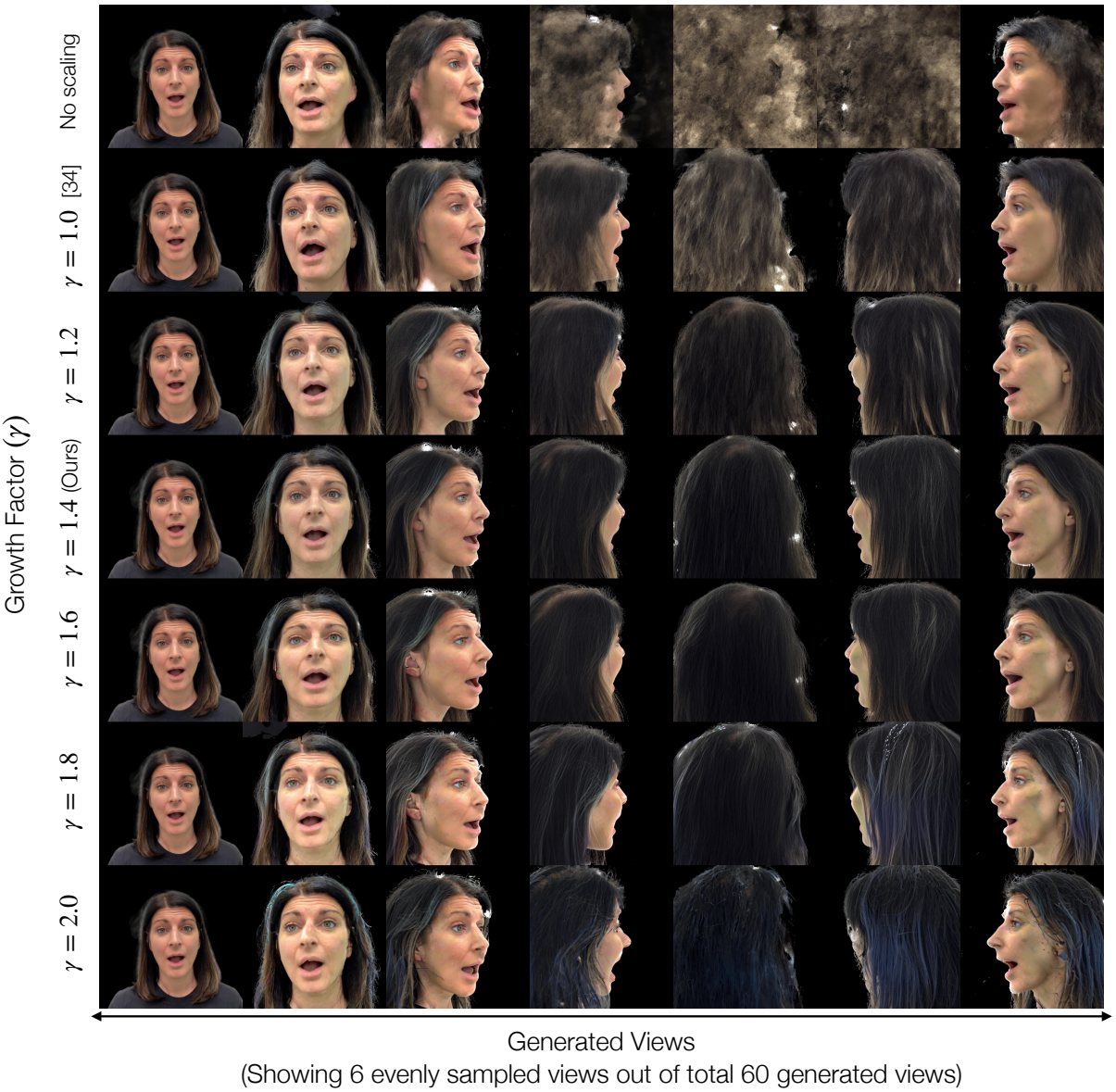


Figure 9. **Generations under varying strengths of growth factor $\gamma$ (Appendix B)**. On each row we show the generated views across vanilla attention [82] (No scaling), prior work [35] and our formulation Eq. (6). Growth factor ($\gamma$) greater than 1.0 helps mitigate the entropy buildup, however increasing $\gamma$ beyond 1.6 leads to oversaturation artifacts (somewhat akin to high CFG scale). We show only 6 views per row subsampled evenly from 60 views generated at $512 \times 512$ resolution. The model was trained to jointly denoised only 12 views ($N_i = 5 * N_t$).

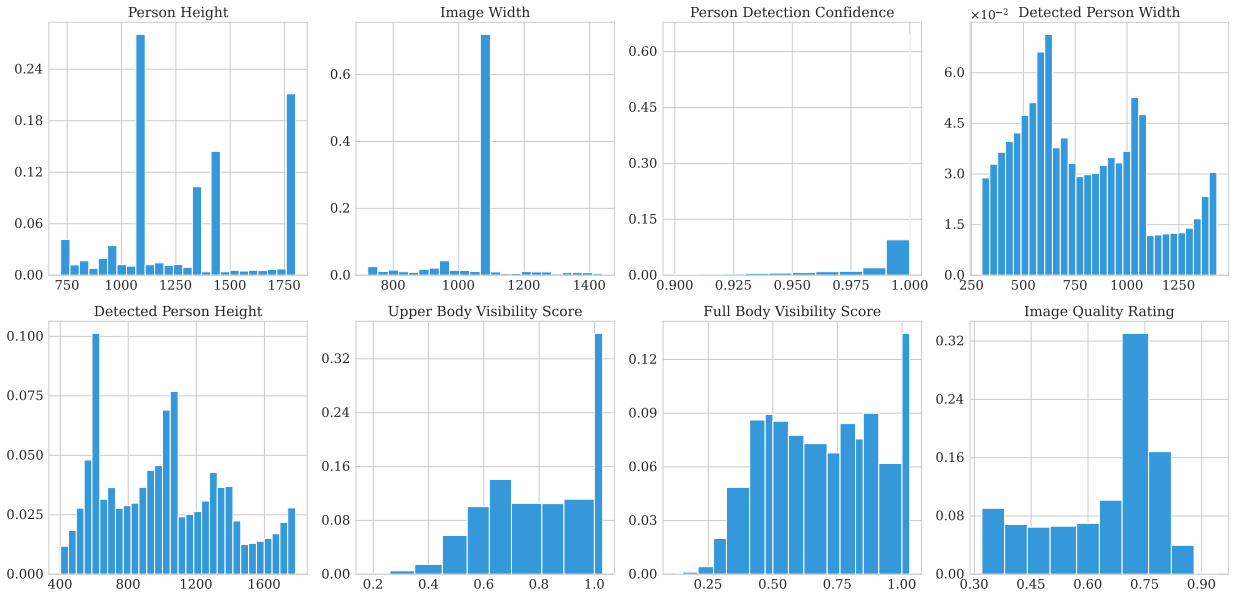


Figure 10. **Statistics of our curated Humans-3B dataset**. We bucket these attributes in bins along X-axis and plot their respective sizes normalized between $[0, 1]$ on Y-axis. Filtering with these statistics enable us to retain images with detected-person confidence and image quality.


Figure 11. **Qualitiative and ablation visuals from pretrained model**. Consistent with quantitative evaluation, visual quality of generated images improves with using human-centric filtering, and image-conditioned models generate samples which are visually closer to the domain (casual iPhone captures). There is also an obvious boost in quality due to higher resolution.

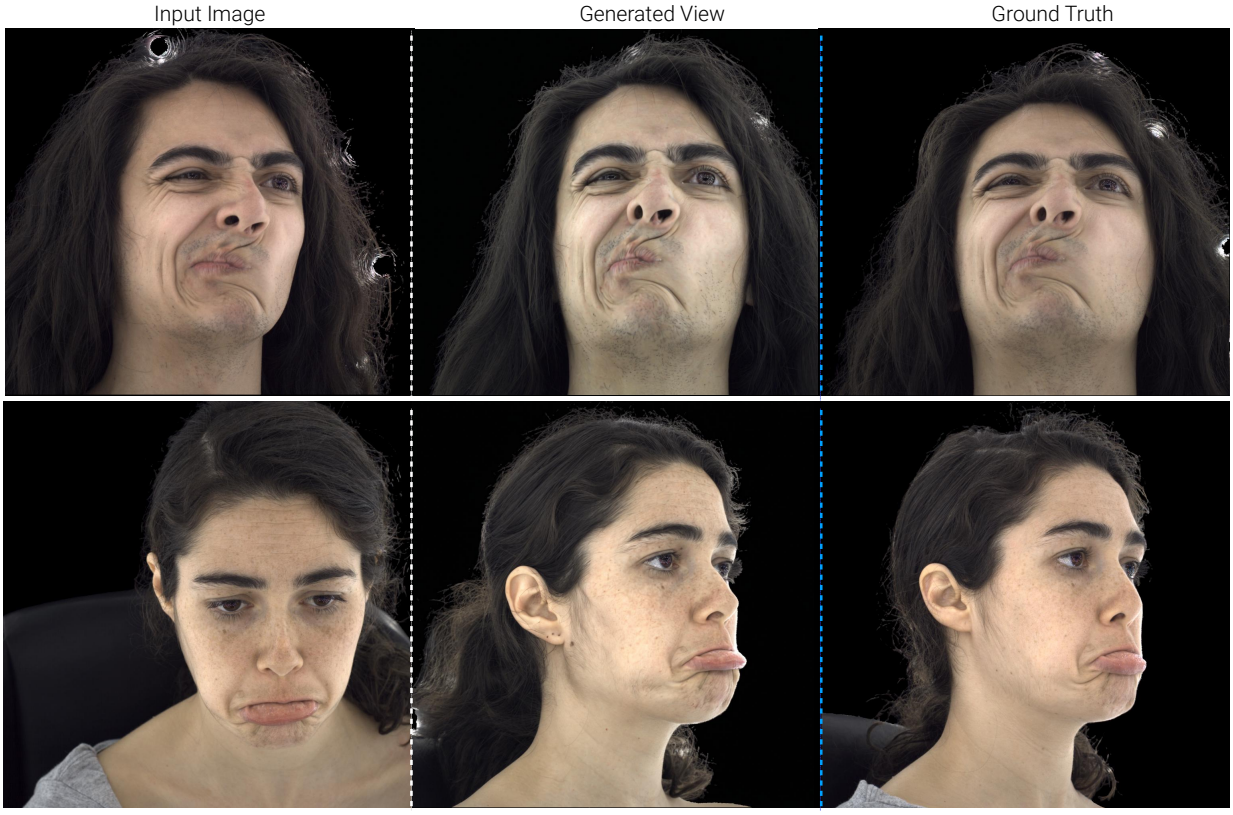


Figure 12. **Pippo can handle extreme facial expressions**. We show generations where reference image comes from unseen Head-only subjects showing extreme facial expression alongside ground truth. We put similar visuals from our Face-only model on webpage.

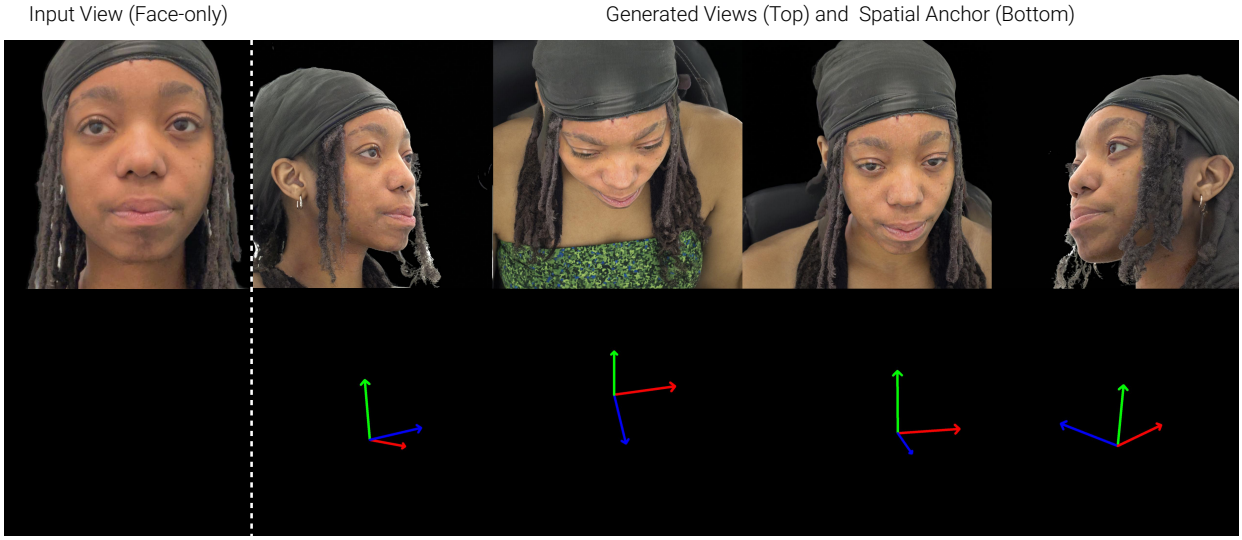


Figure 13. **Visualizing Spatial Anchor**. Spatial anchor is an oriented 3D point in space, which helps anchor the generation by specifying a fixed headpose for generations. We put detailed discussion in Sec. 3, ablation using it in Tab. 6 and more visuals on webpage.

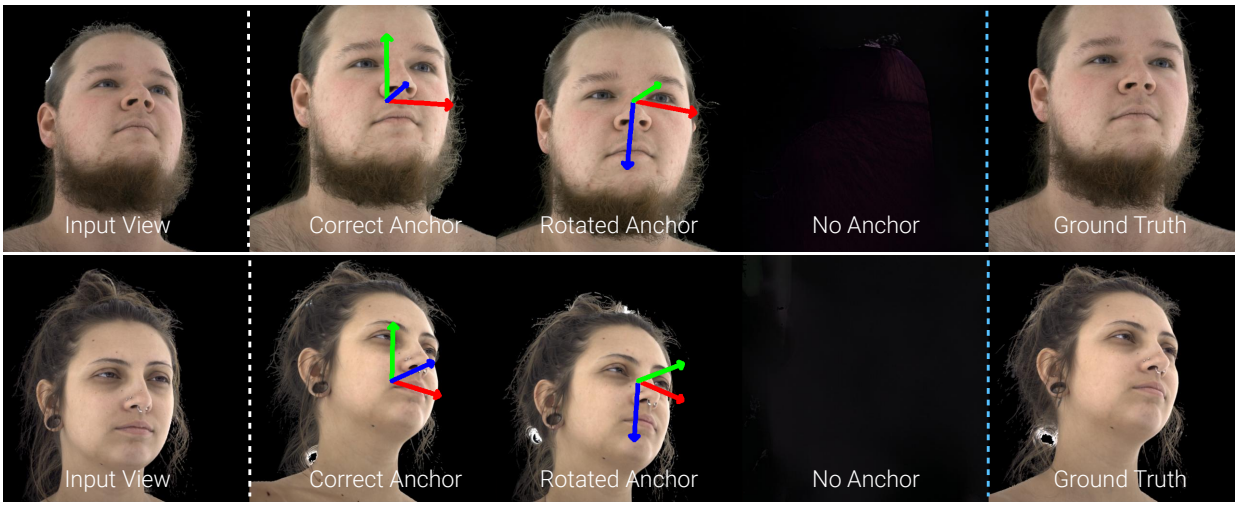


Figure 14. **Ablation with Missing or Inconsistent Spatial Anchors.** We run inference on the Head-only P1@512 model with missing spatial anchor, or when it is inconsistent with input head pose rotated downwards towards the floor by 90°.