Crafting a Miniature Interactive World from a Single Image

Supplementary Material

In the supplemental materials, we present additional details about our PhysGen3D framework App. A, more details of our experimental design App. B.1, more quantitative and qualitative results App. B.3, and various applications of our system App. B.4. Furthermore, we invite the reviewers to check a local webpage in the supplemental materials accessed by `index.html`, to see our generated videos.

A. Additional Details of PhysGen3D

We provide additional details about our framework, specifically on how we handle multiple object occlusions during the mesh generation stage, how we address background completion concerning objects and their shadows, the detailed prompt used in physics reasoning, and further specifics about the physical simulator utilized in our approach.

A.1. Mesh Generation

To reconstruct a 3D foreground object, we require a complete and clearly segmented object image o^i. For scenarios with multiple object occlusions, we employ an iterative inpainting and segmentation strategy, as illustrated in Fig. 1. We first identify all the target objects using GPT-4o. In cases where occlusions are detected, the objects are segmented and inpainted sequentially, progressing from the foreground to the background. Each subsequent segmentation step builds upon the removal of previously processed objects, ensuring accurate and unobstructed reconstruction.

A.2. Background Handling

Shadow significantly impacts the quality of background inpainting if not masked properly. Existing shadow removal methods [4–6] typically detect and remove all shadows indiscriminately. However, our goal is to remove only the shadow related to a specific object. To achieve this, we adopt a straightforward method: we first segment regions where brightness values fall below a certain threshold to identify shadows. For each object, we determine the largest connected component that includes both the object and its shadow. Then, we dilate this mask with a kernel of size 50 and apply inpainting. Developing more adaptable, per-object shadow removal techniques is left as future work.

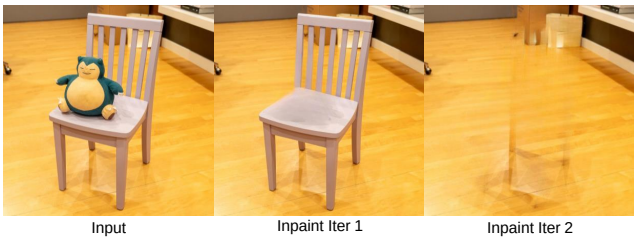

Figure 1. **Iterative Inpainting.** Left: Input image. Middle: Inpainting result after 1 iteration, where the toy is masked and inpainted. Right: Inpainting result after 2 iterations, where the chair is masked and inpainted. The second result is used as background.

A.3. Physics Reasoning

We use GPT-4o to reason the physical parameters for each object and the surface. The prompt and an example answer are as follows.

Listing 1. Prompt used for GPT-4o physics reasoning

```
Answer each question for each object in the picture, using
     one word or number, separated by commas. For numbers,
     do not use scientific notation.
Provide answers in the following format for each object:
'Object number, name, density in kg/m^3, Young's modulus
     (soft/medium/hard), size in meters, requires internal
     filling (yes/no).'
If there are multiple objects in the picture, respond for
     each object on a new line in the specified format.
What is each object's name (one word)?
What is its density in kilograms per cubic meter?
What is its Young's modulus in Pa? (Choose from: Soft:
     Materials like plush toys, foam, or fabric. Medium:
     Materials like rubber or soft plastic. Hard:
     Materials like wood, metal, or hard plastic)
What is its size in meters?
Does the object require internal filling for MPM
     simulations (yes/no)?

Estimate the roughness of the supporting surface in the
     picture, such as tables, floors, or any other
     horizontal surfaces that can act as supports. Provide
     answers in roughness value (0 to 1, where 0 =
     perfectly smooth and 1 = extremely rough)'.
```

Listing 2. Example answers from GPT-4o

```
1, camera_model, 200, soft, 0.15, yes
2, camera, 2700, hard, 0.20, no
0.2
```

In our observation, GPT-4 often provides unstable results for the exact value of Young's modulus, with discrepancies spanning several orders of magnitude. To address this, we defined three categories—**soft**, **medium**, and **hard**—to guide GPT's classification. In the simulator, the elasticity E does not directly correspond to the real Young's modulus. Based on experience, we associate the three categories with $E = 5 \times 10^4$, $E = 5 \times 10^5$, and $E = 5 \times 10^6$, respectively.

A.4. Dynamics Simulation

For simulation stability, we fix the size of the simulator to 2 and the resolution to 256. Since the target object's scale varies from several centimeters to tens of meters, we align the object with the reconstructed scene and fit it into the simulator. To simulate real physics, we scale the physical parameters accordingly. Suppose the reasoned real size of the object is s_0, and the scaled mesh has size s'. Then, the scaling factor is $k = \frac{s'}{s_0}$. In the simulator, we set gravity to $g' = k \times g_0 = k \times 9.8$. The elasticity of each object is also scaled: $E_i' = \frac{E_i}{k}$. (According to dimensional analysis, Young's modulus is inversely proportional to the scale of length.)

We use Taichi Elements [1–3] for Material Point Method (MPM) simulations and modify it to support inhomogeneous materials. MPM is a computational technique used to simulate the behavior of continuum materials. The governing equation of motion is:

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f}_{\text{ext}},$$

where:
- ρ: Density of the material,
- $\mathbf{v}$: Velocity field,
- $\boldsymbol{\sigma}$: Cauchy stress tensor,
- $\mathbf{f}_{\text{ext}}$: External forces per unit volume.

To be specific, MPM combines the strengths of Lagrangian and Eulerian methods by representing materials as discrete particles while performing computations on a background grid. The key steps of MPM are particle-to-grid (p2g) and grid-to-particle (g2p) transfers.

Particle-to-Grid (p2g) Transfer. This step transfers particle properties (mass, momentum, etc.) to the grid.

Mass Transfer. Grid mass is computed by distributing particle mass m_p to nearby grid nodes using weighting functions w:

$$m_i = \sum_p w(x_p - x_i) m_p,$$

where:
- $m_p = \rho_p V_p$: Particle mass (density ρ_p, volume V_p),
- w: Quadratic kernel for interpolation.

Momentum Transfer. Momentum is transferred to the grid using the same weight:

$$\mathbf{v}_i = \frac{\sum_p w(x_p - x_i)\mathbf{v}_p m_p}{m_i},$$

where:
- $\mathbf{v}_i$: Grid velocity,
- $\mathbf{v}_p$: Particle velocity.

Stress Contribution. The stress tensor $\boldsymbol{\sigma}$ contributes force to the grid momentum. Using the deformation gradient F, the stress is defined as:

$$\boldsymbol{\sigma} = 2\mu(F - \mathbf{R})F^\top + \lambda J(J-1)\mathbf{I},$$

where:
- μ and λ: Lamé parameters,
- F: Deformation gradient,
- $\mathbf{R}$: Rotation matrix from SVD ($F = \mathbf{RS}$),
- $J = \det(F)$: Determinant of F,
- $\mathbf{I}$: Identity matrix.

The Lamé parameters λ and μ are computed from Young's modulus E and Poisson's ratio ν as follows:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

$$\mu = \frac{E}{2(1+\nu)}$$

where:
- E: Young's modulus, which describes the material's stiffness,
- ν: Poisson's ratio, which defines the ratio of lateral strain to axial strain.

Grid Velocity Update. The grid force due to stress is given by:

$$\mathbf{f}_i = -\sum_p w'(x_p - x_i) V_p \boldsymbol{\sigma}_p.$$

Newton's second law updates grid velocities:

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \Delta t \frac{\mathbf{f}_i}{m_i},$$

where Δt is the time step.

Grid-to-Particle (g2p) Transfer This step interpolates updated grid data back to particles and updates their states (e.g., velocity, deformation).

Velocity Interpolation. Particle velocities are updated by interpolating grid velocities:

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \sum_i w(x_p - x_i)\mathbf{v}_i^{n+1}.$$

Affine Velocity Field. Affine velocity updates capture velocity gradients from the grid:

$$\mathbf{C}_p = \sum_i 4\frac{w(x_p - x_i)}{\Delta x}\mathbf{v}_i \otimes (\mathbf{x}_i - \mathbf{x}_p).$$

Deformation Gradient Update. The deformation gradient F_p evolves based on the velocity gradient:

$$F_p^{n+1} = (\mathbf{I} + \Delta t \mathbf{C}_p)F_p^n,$$

where $\mathbf{I}$ is the identity matrix.

Advection. Finally, particles are advected using updated velocities:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1}.$$

B. Additional Details of Experiments

Our experiments are designed to compare with the most competitive baselines using multiple evaluation metrics, including human evaluation and GPT-based evaluation. Due to page limitations in the main paper, we provide detailed information about the experimental settings, evaluation metrics, and additional results here.

B.1. Experiments Settings

In the comparative experiment between our method and baseline generative models, we tried our best to ensure they shared the same generation goal. For our method, we manually assigned an initial 3D velocity to each object. To "interpret" this into text, we described the corresponding dynamics and converted them into prompts such as, *"The elephant hops up and falls onto the ground"* or *"The book falls and the orange rolls forward."* All three baseline models were prompted with the same text. Additionally, Kling supports "motion brush" inputs, which were provided alongside the textual prompt. Fig. 2 illustrates examples of "motion brush" inputs, where we manually set the stable parts, movable parts, and their trajectory.

B.2. Evaluation

In our main paper, we only present the quantitative results of human evaluation. Here, we conduct further experiments using GPT-4 and provide the details.

Human Evaluation. We designed a questionnaire to conduct human evaluation, as illustrated in Fig. 3. A total of 31 participants were recruited to complete the 27-page questionnaire. At the beginning, we provided an explanation of video generation models to ensure that participants had a clear understanding of the task. Each page of the questionnaire contains an initial reference image, accompanied by a text prompt describing the expected behavior in the video (e.g., "Red apple rolls on the table"). Four videos are presented on each page in a random order, all corresponding to the same initial condition and text prompt. Participants are instructed to assess each video based on three dimensions. This design ensures a fair, consistent, and comprehensive evaluation process.

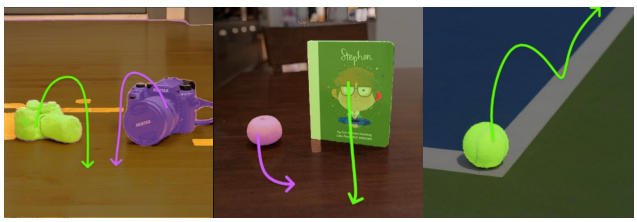

Figure 2. **Motion brush input for Kling.** In all cases, we manually define the motion for each object by identifying the movable part and drawing its trajectory. Additionally, we specify the stable part of the object.

GPT-4o Evaluation. To assess the quality of the generated videos, we also conducted evaluations using GPT-4o for both our results and the baselines. The prompt is as follows:

Listing 3. Prompt used for GPT-4o evaluation

```
I would like you to evaluate the quality of four generated
     videos based on the following criteria: physical
     realism, photorealism, and semantic consistency. The
     evaluation will be based on 10 evenly sampled frames
     from each video. Given the original image and the
     following instructions: '{instructions}', please
     evaluate the quality of each video on the three
     criteria mentioned above.
Note that: Physical Realism measures how realistically the
     video follows the physical rules and whether the
     video represents real physical properties like
     elasticity and friction. To discourage completely
     stable video generation, we instruct respondents to
     penalize such cases. Photorealism assesses the
     overall visual quality of the video, including the
     presence of visual artifacts, discontinuities, and
     how accurately the video replicates details of light,
     shadow, texture, and materials. Semantic Consistency
     evaluates how well the content of the generated video
     aligns with the intended meaning of the text prompt.
Please provide the following details for each video,
     scores should be ranging from 0-1, with 1 to be the
     best:
Video 1: Physical Realism Score: [a score]; Photorealism
     Score: [a score]; Semantic Consistency Score: [a
     score]
Video 2: Physical Realism Score: [a score]; Photorealism
     Score: [a score]; Semantic Consistency Score: [a
     score]
Video 3: Physical Realism Score: [a score]; Photorealism
     Score: [a score]; Semantic Consistency Score: [a
     score]
Video 4: Physical Realism Score: [a score]; Photorealism
     Score: [a score]; Semantic Consistency Score: [a
     score]
Note that your output should strictly follows the above
     format, with a ';' after each score. Do not give any
     other explanations.
The first image is the input image.
# input image
Here are 10 evenly spaced frames from the generated video
     number {idx + 1}.
# generated frames
```

B.3. Additional Results

show that both methods introduce unrealistic deformations. DragAnything sometimes fails to maintain a stable background, even when manually set. MOFA demonstrates better motion control but lacks realism as well. See the table below for quantitative results. We provide additional quantitative and qualitative results of our experiments.

Human Evaluation Results. We analyze the human evaluation scores further in Fig. 4. The distribution of scores indicates that participants generally agree that most of our results are both physically realistic and semantically consistent.

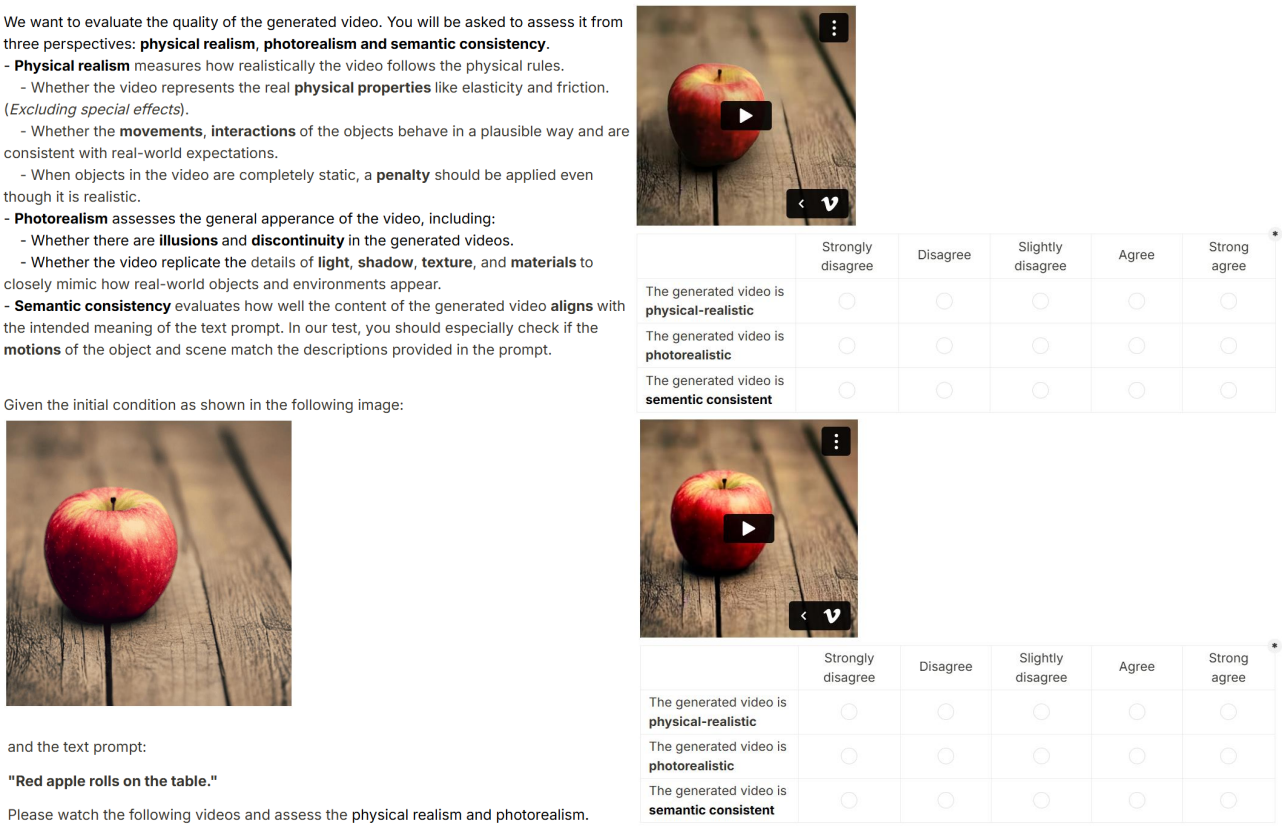

Figure 3. **An example page of human evaluation questionnaire.** In each page of the questionnaire, we explain the criteria in detail. We provide the input image, the text prompt and four generated videos in a random order. Each video is followed by a evaluation matrix on a five-point scale, from strongly disagree (1) to strongly agree (5).

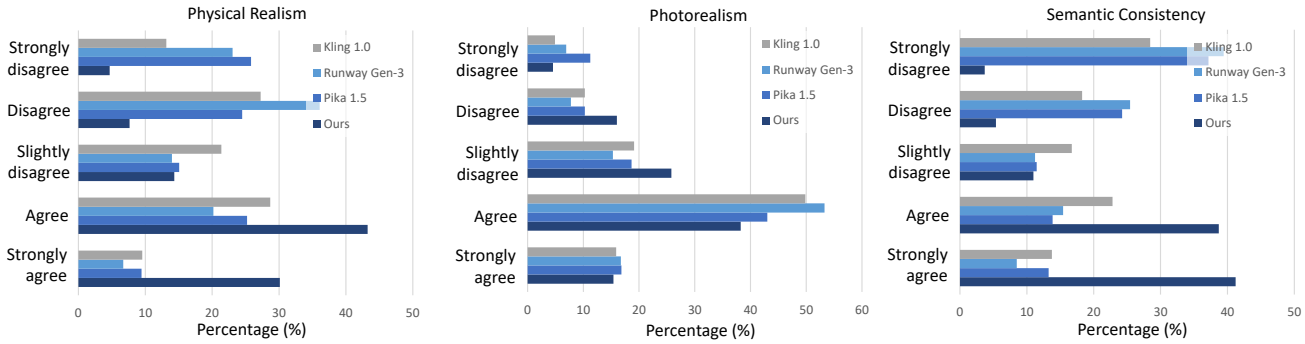

Figure 4. **Human evaluation score distribution.** Score distribution shows our method's superiority in physical realism and semantic consistency, with comparable performance across models in photorealism.

Our method significantly outperforms baseline generative models on these two criteria. However, the four models perform comparably in terms of photorealism.

Additional Qualitative Results. Here, we present additional qualitative results in Fig. 5. The first row demonstrates the "sandy" effect, where the material of the teddy bear is transformed into sand. The last row illustrates a multi-object collision scenario, where three apples collide with one another. More results are available in video format on our supplementary webpage.

Fig. 6 shows the results after VEnhancer's post-processing. Although VEnhancer recovers fine details, it can also introduce hallucinations. This illustrates a fundamental trade-off between photorealism and physical accuracy: integrating diffusion models into the pipeline leverages their strong priors to compensate for reconstruc-

Figure 5. **More qualitative results.** The first row demonstrates the **"sandy" effect**, transforming the teddy bear's material into sand, while the second and third rows showcase **bouncing** and **rolling** effects, respectively. The fourth row illustrates a **multi-object collision** scenario, with three apples colliding with one another, and the final row highlights the system's ability to **generate a video from a painting**.

Figure 6. **Qualitative comparison of VEnhancer.** After post-processing by VEnhancer, more details are recovered and the video appears to be more photorealistic.

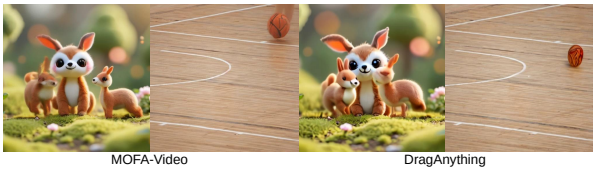

Figure 7. **Qualitative results of MOFA-Video and DragAnything.** These two open-sourced diffusion models fail to keep background consistent and produce unrealistic deformations.

tion and rendering errors, but it cannot guarantee adherence to real-world physics.

Fig. 7 shows the results of two open-sourced diffusion models, MOFA-Video and DragAnything. Both methods introduce unrealistic deformations: DragAnything sometimes fails to maintain a stable background, even when manually set. MOFA demonstrates better motion control but lacks realism as well. Quantitative results of VBench scores in the main paper support these findings.

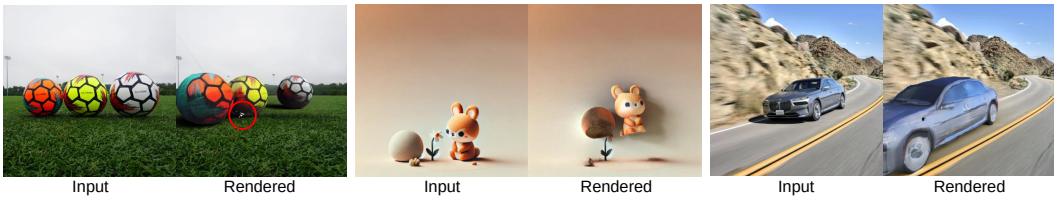

Figure 8. **More On Limitations.** The left two images show simulation failures, where unwanted floating points appear in the final rendering results. The middle two images show reconstruction failures, where the wall is recognized as ground by mistake. The right two images depict texture optimization failures, where the car fails to accurately reproduce the real roughness and metallic properties, resulting in an unrealistic appearance.

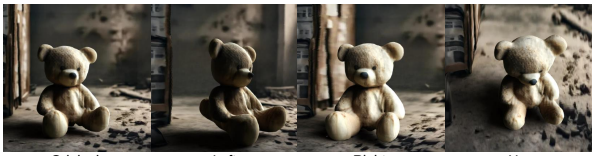

Figure 9. **Camera controls.** We provide a case demonstrating the potential to perform camera controls on above our pipeline. The left one is the only input image. The right three images are generated with outpaiting and reconstruction.

B.4. Applications

Our video generation framework, PhysGen3D, enables a range of exciting applications through its explicit representation. Here are just a few of the compelling use cases our system supports:

Camera controls. PhysGen3D's 3D scene representation inherently supports novel view synthesis. We demonstrate this capability (see figure below) by extending our method with minimal modifications: (1) outpainting and meshing the background and (2) rendering from novel views. Results in Fig. 9 show good consistency across views while maintaining environmental coherence.

Generate Video from Paintings. Thanks to the generalization ability of our interactive 3D world reconstruction pipeline, our method can extend beyond real photos to accommodate other types of inputs, such as generated images and paintings. The final row of Fig. 5 demonstrates the generation of a video from a painting.

C. Limitations

In the main text, we present three failure cases, each highlighting a specific type of error in perception, simulation, and rendering. Fig. 8 illustrates additional failures. One involves incorrectly reconstructed meshes with unwanted floating points. Although we have implemented floating point removal during rendering, some points are too close to the object to be detected. Another failure involves material that is incorrectly estimated. The reflectance behavior of cars poses a challenging optimization target, and inaccuracies in inverse rendering result in unrealistic renderings. Failures or inaccuracies may also occur in depth and light estimation. However, these modules are relatively mature, and such errors are comparatively rare.

Many of these failures stem from the inherently ill-posed nature of the task, as reconstructing the full geometry, physics, and textures from partial scene observations requires substantial prior knowledge.

Currently, we only support a single collider surface, such as the ground or a table. However, our pipeline has the potential to set all stable components as colliders. Additionally, each object is currently homogeneous in density and elasticity. In the future, we may assign different materials to different parts of an object, as demonstrated in [7].

Overall, our method is designed for object-centric scenes, excelling at mimicking real-world physics for rigid and deformable objects. It also supports a variety of edits and effects. However, reconstructing entire scenes for more complex scenarios remains an open challenge.

References

[1] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):201, 2019. 2

[2] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

[3] Yuanming Hu, Jiafeng Liu, Xuanda Yang, Mingkuan Xu, Ye Kuang, Weiwei Xu, Qiang Dai, William T. Freeman, and Frédo Durand. Quantaichi: A compiler for quantized simulations. *ACM Transactions on Graphics (TOG)*, 40(4), 2021. 2

[4] Chenghua Li, Bo Yang, Zhiqi Wu, Gao Chen, Yihan Yu, and Shengxiao Zhou. Shadow removal based on diffusion segmentation and super-resolution models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6045–6054, 2024. 1

[5] Zhihao Liu, Hui Yin, Xinyi Wu, Zhenyao Wu, Yang Mi, and Song Wang. From shadow generation to shadow removal. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4927–4936, 2021.

[6] Florin-Alexandru Vasluianu, Tim Seizinger, Zhuyun Zhou, Zongwei Wu, Cailian Chen, Radu Timofte, Wei Dong, Han Zhou, Yuqiong Tian, Jun Chen, et al. Ntire 2024 image shadow removal challenge report. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6547–6570, 2024. 1

[7] Albert J Zhai, Yuan Shen, Emily Y Chen, Gloria X Wang, Xinlei Wang, Sheng Wang, Kaiyu Guan, and Shenlong Wang. Physical property understanding from language-embedded feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28296–28305, 2024. 6