Generative Blocks World: Moving Things Around in Pictures

Anonymous ICCV submission

Paper ID 20

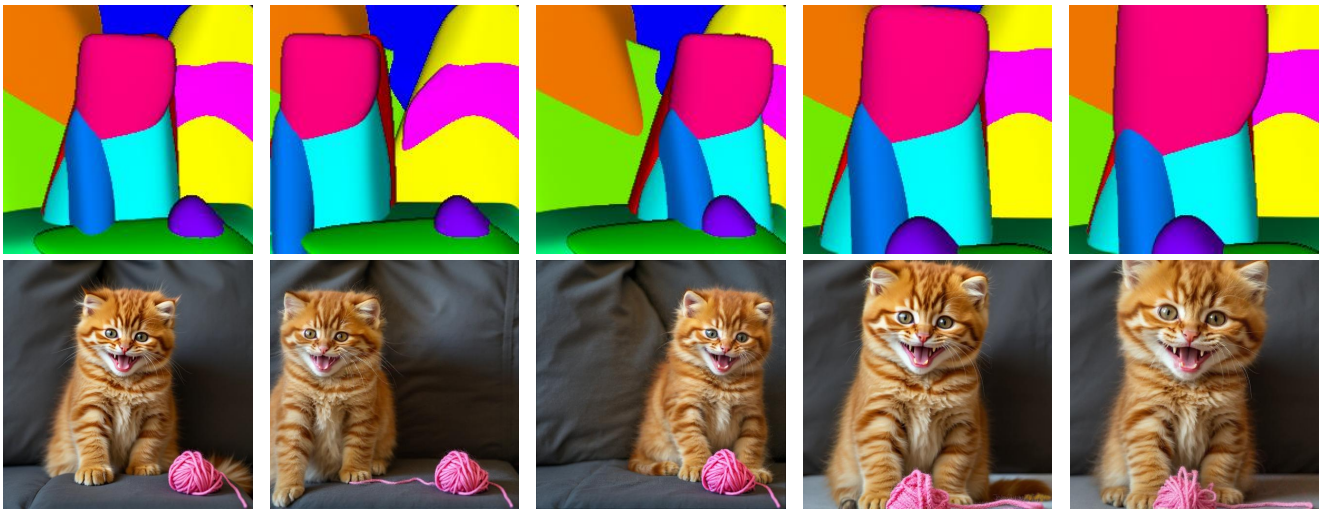

Figure 1. **Generative Blocks World.** Given an input image (bottom left), we extract a set of 3D convex primitives (top left) that provide an editable and controllable representation of the scene. These primitives are used to generate new images that respect geometry, texture, and the text prompt. The first column shows the original input and its primitive decomposition. Subsequent columns show sequential edits: translating the cat to the left (second column), translating it to the right (third column), moving the yarn in front of the cat and shifting the camera toward the scene center (fourth column), and scaling up the cat's head (burgundy primitive; fifth column). Our method enables semantically meaningful, 3D-aware image editing through intuitive manipulation of these learned primitives.

Abstract

We describe Generative Blocks World to interact with the scene of a generated image by manipulating simple geometric abstractions. Our method represents scenes as assemblies of convex 3D primitives, and the same scene can be represented by different numbers of primitives, allowing an editor to move either whole structures or small details. Once the scene geometry has been edited, the image is generated by a flow-based method which is conditioned on depth and a texture hint. Our texture hint takes into account the modified 3D primitives, exceeding texture-consistency provided by existing techniques. These texture hints (a) allow accurate object and camera moves and (b) preserve the identity of objects. Our experiments demonstrate that our approach outperforms prior works in visual fidelity, editability, and compositional generalization.

1. Introduction

Modern large generative models can generate realistic-looking images from minimal input, but they offer limited control. Recent works have shown that intrinsic scene properties essential for rendering—such as normals, depth, albedo, and illumination—emerge within the learned representations of these large generative models [3, 4, 12, 52]. Yet despite these emergent capabilities, modifying geometry, lighting, or viewpoint often disrupts appearance or object identity. Traditional rendering systems offer precise control through explicit geometric representations and physically based shading models but require extensive authoring effort and technical expertise.

Our goal is to bring the control of traditional rendering to modern generative models without the overhead of explicit modeling. The system described here enables an author to modify the camera viewpoint of a scene while preserving its

content, and to relocate objects or parts while maintaining their high-fidelity appearance (see Fig. 1). Achieving this, however, requires addressing two fundamental challenges in view synthesis and editing.

At a high level, these operations should be simple. For many pixels, accurate camera moves are easy: acquire an accurate depth map, project texture onto that map, then reproject into the new camera. Similarly, moving objects or parts is conceptually straightforward: project texture onto the depth map, adjust the depth map, then reproject. But this idealized pipeline breaks down in practice due to two key obstacles: (*i*) many target pixels are not visible in the source view, so texture must be extrapolated; and (*ii*) editing depth maps directly is very difficult and unintuitive.

To address these challenges, we propose representing scenes as small assemblies of meaningful parts or primitives. This idea has deep roots. Roberts' *Blocks World* [40] viewed simple scenes as a handful of cuboids. Biederman [6] suggested that humans recognize and reason about objects as compositions of primitive parts. For our purposes, such assemblies must approximate the scene's depth map well enough to enable view-consistent texture projection. Primitive decompositions have been widely studied in computer vision for recognition, parsing, and reconstruction [17, 20–22, 32, 46, 47], but their application to content generation has been limited. Moreover, reliable primitive fitting is a very recent phenomenon [47, 49]. Our work exploits these advances to control modern generative models, enabling precise, structured, and editable image synthesis.

We represent scenes using convex geometric primitives and use them to control image synthesis, allowing edits such as camera moves, object moves, and detail adjustments, while maintaining structure and appearance. As a nod to computer vision history, we call our framework **Generative Blocks World**, *though our learned primitives are richer than cuboids*. Generative Blocks World decomposes an input image into a sparse set of convex polytopes using an extension of a recent convex-decomposition procedure [11, 47]. These convex primitives provide sufficient geometric accuracy to enable view-consistent texture projection. A final rendering using a pretrained depth-conditioned Flux DiT [28] preserves textures that should be known and inpaints missing textures. Our primitives are accurate enough that we don't need to train the generative depth-to-image model on the particular statistics of our primitives.

Good primitive decompositions have very attractive properties. They are *selectable*: individual primitives can be intuitively selected and manipulated (see Fig. 1). They are *object-linked*: a segmentation by primitives is close to a segmentation by objects, meaning an editor is often able to move an object or part by moving a primitive (Fig. 1; Fig. 3; Fig. 4). They are *accurate*: the depth map from a properly constructed primitive representation can be very close to the original depth map (Sec. 3.1), which means primitives can be used to build texture hints (Section 3.2) that support accurate camera moves (Fig. 2; Fig. 5). They have *variable scale*: one can represent the same scene with different numbers of primitives, allowing an editor to adjust big or small effects (Fig. 7; Fig. 8; Fig. 12).

Contributions.

- We describe a pipeline that fuses convex primitive abstraction with a SOTA flow-based generator, FLUX. Our pipeline uses a natural texture-hint procedure that supports accurate camera moves and edits at the object-level, while preserving identity.
- We provide extensive evaluation demonstrating superior geometric control, texture retention, and edit flexibility relative to recent state-of-the-art baselines.

2. Related Work

Primitive Decomposition. Early vision and graphics pursued parsimonious part-based descriptions, from Roberts' *Blocks World* [40] and Binford's generalized cylinders [7] to Biederman's geons [6]. Efforts to apply similar reasoning to real-world imagery have been periodically revisited [5, 20, 32] from various contexts and applications. Modern neural models revive this idea: BSP-Net [8], CSG-Net [42], and CVXNet [11] represent shapes as unions of convex polytopes, while Neural Parts [46], SPD [58], and subsequent works [30] learn adaptive primitive sets. Recent systems extend from objects to scenes: Convex Decomposition of Indoor Scenes (CDIS) [47] and its ensembling/Boolean refinement [49] fit CVXNet-like polytopes to RGB-D images, using a hybrid strategy. CubeDiff [25] fits panoramas inside cuboids. Our work leverages CDIS as the backbone, but (i) improves robustness to in-the-wild depth/pose noise and (ii) couples the primitives to a Rectified Flow (RF) renderer, enabling controllable synthesis rather than analysis alone.

Conditioned Image Synthesis. Conditional generative networks such as Pix2Pix [23], CycleGAN [57], and SPADE [37] pioneered layout-to-image translation. Diffusion models now dominate; seminal works include Stable Diffusion [41], ControlNet [55], and T2I-Adapter [34]. Subsequent work has shown that multiple spatial controls can be composed for restoring images [48], and recent methods can exert local and global color edits [50]. In this work, we use a pretrained depth-conditional FLUX model using depth maps derived from primitives.

Point-Based Interactive Manipulation. Point-based manipulation offers direct, intuitive control over 2D image

attributes. DragGAN [35] allows users to deform an object's pose or shape by dragging handle points on a 2D generative manifold. This concept was subsequently adapted to more general diffusion models by methods like DragDiffusion [43], DragonDiffusion [33], StableDrag [10], DiffusionHandles [36], and Dragin3D [19], which improved robustness, controllability, and fidelity. Diffusion Self-Guidance can exert layout control and perform object-level edits [14]. However, these methods fundamentally operate by deforming pixels or lack a true understanding of 3D scene structure. They can perform in-place pose/shape edits and even simple translations but struggle to perform 3D-consistent manipulations, such as moving objects within a scene or moving the camera while respecting perspective, occlusion, and texture. In contrast, we show promising results in such scenarios and offer flexible control of primitives, providing both fine-grained control when using a large number of primitives and coarse, object-level control when using a smaller number of primitives.

Object-Level and Scene-Level Editing. Many recent works embed 3D priors into generative editing but focus on single objects: StyleNeRF [18], SJC [51], DreamFusion [38], Make-A-Dream [45], and 3D-Fixup [9]. Methods like Obj3DiT [31] use language to guide transformations (e.g., rotation, translation) by fine-tuning a model on a large-scale synthetic dataset. In contrast, Generative Blocks World generalizes to complex editing tasks that are not easy to describe precisely in text form. An alternative paradigm, seen in Image Sculpting [54] and OMG3D [56], offers precise control by first reconstructing a 2D object into an explicit 3D mesh, which is then manipulated and re-rendered using generative models. While offering high precision, these multi-stage pipelines can be complex and are often bottlenecked by the initial reconstruction quality. Our method provides a more streamlined approach by operating on abstract primitives, avoiding the complexities of direct mesh manipulation while still providing strong geometric control.

Primitive-Based Scene Authoring. Recently, LooseControl [2] showed how to train LoRA weights on top of a pretrained Depth ControlNet, enabling box-like primitive control of image synthesis. The LoRA weights bridge the domain gap between box-like primitive depth maps and standard depth maps as one might obtain from, e.g., DepthAnything [53]. In contrast, this paper demonstrates primitive fits that do not require fine-tuning diffusion models because the underlying primitive representation is highly accurate. We similarly adopt a depth-to-image generator, but our conditioning signal is *structured geometry*—a set of editable primitives rather than dense maps—yielding stronger semantic correspondence and causal behavior. More recently, Build-A-Scene [13] uses the same primitive generator and image synthesizer as LooseControl; thus, it suffers from the same problems in depth accuracy. Generative Blocks World differs by (i) decomposing each object into a handful of convex polytopes, giving finer yet still abstract control; (ii) supporting camera moves; and (iii) allowing new scenes to be *authored* via primitive assembly.

3. Method

Generative Blocks World generates realistic images conditioned on a parsimonious and editable geometric representation of a scene: a set of convex primitives. The process consists of four main stages: (i) primitive extraction from any image via convex decomposition (Sec. 3.1), (ii) generating an image conditioned on the primitives (and text prompt), (iii) user edits the primitives and/or camera, and (iv) generates a new image conditioned on the updated primitives, while preserving texture from the source image (Sec. 3.3). We describe each component in detail below. See Fig. 2 for an overview.

3.1. Convex Decomposition for Primitive Extraction

Our primitive vocabulary is blended 3D convex polytopes as described in [11]. CVXnet represents the union of convex polytopes using indicator functions $O(x) \rightarrow [0, 1]$ that identify whether a query point $x \in \mathbb{R}^3$ is inside or outside the shape. Each convex polytope is defined by a collection of half-planes.

A half-plane $H_h(x) = n_h \cdot x + d_h$ provides the signed distance from point x to the h-th plane, where n_h is the normal vector and d_h is the offset parameter.

While the signed distance function (SDF) of any convex object can be computed as the maximum of the SDFs of its constituent planes, CVXnet uses a differentiable approximation. To facilitate gradient learning, instead of the hard maximum, the smooth LogSumExp function is employed to define the approximate SDF, $\Phi(x)$:

$$\Phi(x) = \text{LogSumExp}\{\delta H_h(x)\}$$

The signed distance function is then converted to an indicator function $C : \mathbb{R}^3 \rightarrow [0, 1]$ using:

$$C(x|\beta) = \text{Sigmoid}(-\sigma\Phi(x))$$

The collection of hyperplane parameters for a primitive is denoted as $h = \{(n_h, d_h)\}$, and the overall set of parameters for a convex as $\beta = [h, \sigma]$. While σ is treated as a hyperparameter, the remaining parameters are learnable. The parameter δ controls the smoothness of the generated convex polytope, while σ controls the sharpness of the indicator function transition. The soft classification boundary created

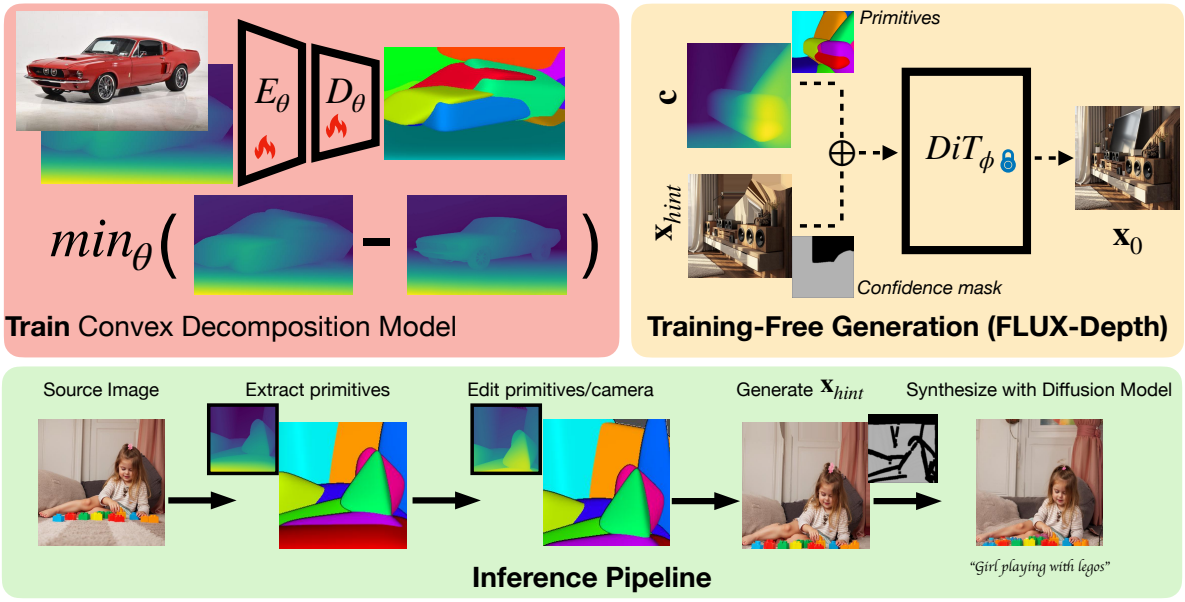

Figure 2. **Pipeline Overview. Top left:** We use pretrained convex decomposition models [49] to extract primitives from an input image at multiple scales. **Bottom:** Users can manipulate these primitives and the camera to define a new scene layout. We render the modified primitives into a depth map and generate a texture hint image. These serve as inputs to a pretrained depth-to-image model [28], which requires no fine-tuning (**Top right**). The resulting image respects the modified geometry, preserves texture where possible, and remains aligned with the text prompt.

by the sigmoid function facilitates training through differentiable optimization. The neural architecture of our primitives model is the standard ResNet-18 Encoder E_θ followed by 3 fully-connected layers that decode into the parameters of the primitives D_θ. While the model is lightweight, the SOTA of primitive prediction requires a different trained model for each primitive count K.

Recent work has adapted primitive decomposition to real scenes (as opposed to isolated objects, such as those in ShapeNet [47]). These methods combine neural prediction with post-training refinement: an encoder-decoder network predicts an initial set of convex polytopes, which is followed by gradient-based optimization to align the primitives closely to observed geometry. This approach is viable because the primary supervision for primitive fitting is a depth map (with heuristics that create 3D samples, and auxiliary losses to avoid degenerate solutions). Note that ground truth primitive parameters are not available (as they could be in many other computer vision settings e.g., segmentation [26]). This is why the losses encourage the primitives to classify points near the depth map boundary correctly instead of directly predicting the parameters.

Rendering the primitives. We condition the RF model on the primitive representation via a depth map, obtained by ray-marching the SDF from the original viewpoint of the scene. Depth conditioning abstracts away potential 'chatter' in the primitive representation from e.g. over-segmentation, while simultaneously yielding flexibility in fine details (depth maps typically lack pixel-level high-frequency details). Depth-conditioned image synthesis models are well-established e.g. [55]. Because **it's hard to edit a depth map, but easy to edit 3D primitives**, our work adds a new level of control to the existing image synthesis models. As we establish quantitatively in Table 2, our primitive generator is extremely accurate, and our evaluations show that we get very tight control over the synthesized image via our primitives. This means that whatever domain gap there is between depth from primitives and depth from SOTA depth estimation networks is not significant.

Scaling to in-the-wild scenes. We collect 1.8M images from LAION to train our primitive prediction models. To obtain ground truth depth supervision, we use DepthAnythingv2 [53]. To lift a depth map $D \in \mathbb{R}^{H \times W}$ to a 3D point cloud using the pinhole camera model, each pixel (u, v) with depth $d_{u,v}$ maps to a 3D point (X, Y, Z) as:

$$X = \frac{(u - c_x) \cdot d_{u,v}}{f_x}, \quad Y = \frac{(v - c_y) \cdot d_{u,v}}{f_y}, \quad Z = d_{u,v}$$

where (c_x, c_y) is the principal point (typically $W/2, H/2$), and (f_x, f_y) are the focal lengths along the image axes. DepthAnythingv2 supplies a metric depth module with reasonable camera calibration parameters. These 3D samples are required to supervise primitive fitting. In fact, at test-time, we can directly optimize primitive parameters using the training losses since these 3D samples are available.

Primitive fitting details. We use the standard ResNet-18 encoder (accepting RGBD input) followed by 3 fully-connected layers to predict the parameters of the primitives. We train different networks for different primitive counts $K \in \{4, 6, 8, 10, 12, 24, 36, 48, 60, 72\}$, and allow the user to select their desired level of abstraction. Alternatively, the ensembling method of [49] can automatically select the appropriate number of primitives. Depending on the primitive count, the training process takes between 40-100 mins on a single A40 GPU, and inference (including generating the initial primitive prediction, refinement, and rendering) can take 1-3 seconds per image. While traditional primitive-fitting to RGB images fits cuboids [27], we find that polytopes with more faces and without symmetry constraints yield more accurate fits. Thus, we use $F = 12$ face polytopes. We do not use a Manhattan World loss or Segmentation loss; the former helped on NYUv2 [44] but not on in-the-wild LAION images and the latter showed an approximately neutral effect in the original paper [47].

3.2. Depth-Conditioned Inpainting in Rectified Flow Transformers

Here, we describe our image synthesis pipeline. We build upon the SOTA FLUX, a rectified flow model [15, 28].

Forward Noising Process. In the forward process, a clean latent representation $\mathbf{x}_0$ (derived from an input image via a variational autoencoder, VAE) is progressively noised over T timesteps to produce a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. The noise schedule is defined by sigmas σ_t, typically linearly interpolated from 1.0 to $\frac{1}{T}$. The forward process is governed by:

$$\mathbf{x}_t = \sqrt{1 - \sigma_t^2}\mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where σ_t controls the noise level, and $\boldsymbol{\epsilon}$ is Gaussian noise. For conditional inputs like a depth map, the control image is encoded into latents via the VAE and concatenated with the noisy latents $\mathbf{x}_t$ during the reverse process.

Adding Spatial Conditions. Older ControlNet implementations [55] train an auxiliary encoder that adds information to decoder layers of a base frozen U-Net. Newer implementations, including models supplied by the Black Forest Labs developers, concatenate the latent $\mathbf{x_t}$ and condition (e.g., depth map) $\mathbf{c}$ as an input to the network, yielding tighter control. `FLUX.1 Depth [dev]` re-trains the RF model with the added conditioning; `FLUX.1 Depth [dev] LoRA` trains LoRA layers on top of a frozen base RF model. Both options give tight control and work well with our primitives, though LoRA exposes an added parameter $lora_{weight} \in [0, 1]$ tuning how tightly the depth map should influence synthesis. This is helpful when the primitive abstraction is too coarse relative to the geometric complexity of the desired scene.

Reverse Diffusion Process. The reverse process starts from a noisy latent $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively denoises to approximate $\mathbf{x}_0$. The DiT-RF model uses a transformer architecture with: *Double-stream layers*: process image tokens (noisy latents and control image latents) and text tokens (prompt embeddings) separately with cross-attention. *Single-stream layers*: jointly process all tokens to capture interactions. The model predicts noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{c}, \mathbf{p})$, where $\mathbf{c}$ is the control image and $\mathbf{p}$ includes text embeddings and pooled projections. The scheduler updates the latents:

$$\mathbf{x}_{t-1} = \text{SchedulerStep}(\mathbf{x}_t, t, \boldsymbol{\epsilon}_\theta),$$

using RF techniques to optimize the denoising trajectory.

Role of Hint and Mask. A core contribution of this work is an algorithm to generate a "hint" image to initialize the image generation process, as well as a confidence mask (see Sec 3.3). The hint and mask influence the generation within timesteps $t_{\text{end}} \leq t \leq t_{\text{start}}$, which are hyperparameters. The mask $\mathbf{m} \in [0, 1]$ specifies regions where the hint should guide the output. The hint is encoded into latents $\mathbf{x}_{\text{hint}}$ via the VAE. During denoising, the latents are updated as:

$$\mathbf{x}_t = (1 - \mathbf{m}) \cdot \mathbf{x}_{\text{hint},t} + \mathbf{m} \cdot \mathbf{x}_t,$$

where $\mathbf{x}_{\text{hint},t}$ is the noised hint latent at timestep t:

$$\mathbf{x}_{\text{hint},t} = \text{SchedulerScaleNoise}(\mathbf{x}_{\text{hint}}, t, \boldsymbol{\epsilon}).$$

Thus, the hint image is *noised* to match the current timestep's noise level before incorporation, ensuring consistency with the denoising process. Outside $[t_{\text{end}}, t_{\text{start}}]$, the hint and mask are ignored.

3.3. Texture Hint Generation for Camera and Object Edits

A number of methods have been proposed to preserve texture/object identity upon editing an image. A common and simple technique is to copy the keys and values from a style image into the newly generated image (dubbed "style preserving edits"). For older U-Net-based systems, this is done in the bottleneck layers [2]. For newer DiTs, this is done at selected "vital" layers [1]. In our testing, key-value copying methods are insufficient for camera/primitive moves (see Fig. 6). Further, because of our primitives, we have a geometric representation of the scene. Here we demonstrate a routine to obtain a source "hint" image $\mathbf{x}_{hint}$ as well as a confidence mask $\mathbf{m}$ that can be incorporated in the diffusion process. The hint image is a rough approximation of what the synthesized image should look like using known spatial correspondences between primitives in the first view and the second. The confidence mask indicates where we can and cannot trust the hint, commonly occurring near depth discontinuities. We rely on the diffusion machinery to essentially clean up the hint, filling gaps and refining blurry

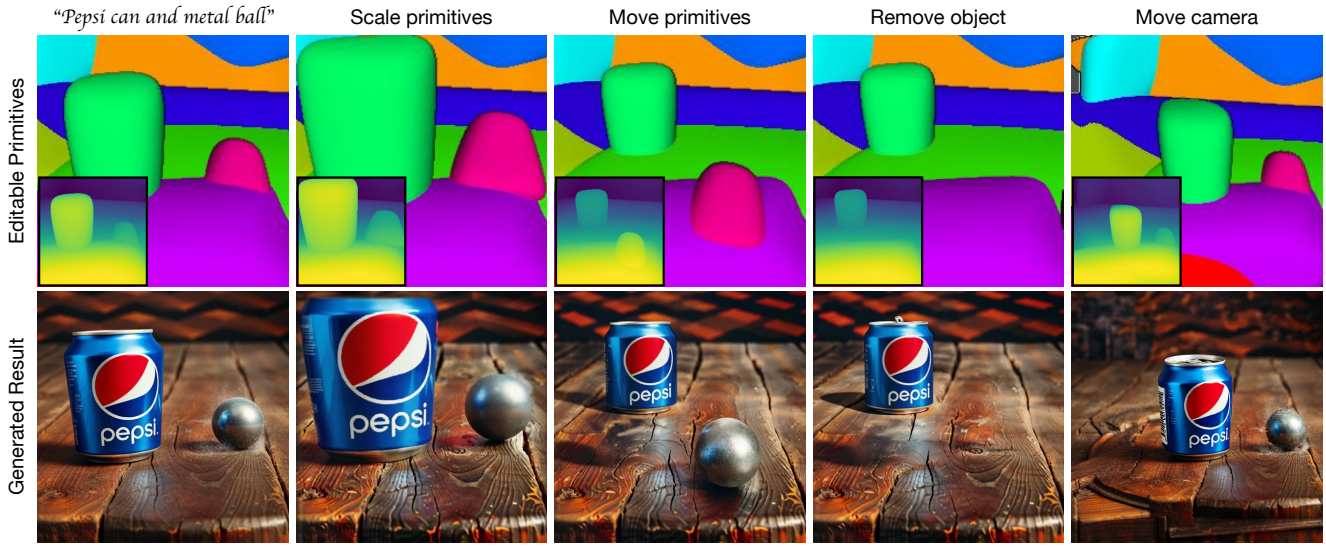

Figure 3. **Editable Primitives as a Structured Depth Prior for Generative Models.** Our method uses 3D convex primitives as an editable intermediate representation from which depth maps are derived. These depth maps (shown as insets in the top row) are used to condition a pretrained depth-to-image generative model. The top row shows primitive configurations after sequential edits—translation, scaling, deletion, and camera motion—alongside their corresponding derived depth maps. The bottom row shows the resulting synthesized images. Unlike direct depth editing, which is unintuitive and underconstrained, manipulating primitives offers a structured, interpretable, and geometry-aware interface for controllable image generation.

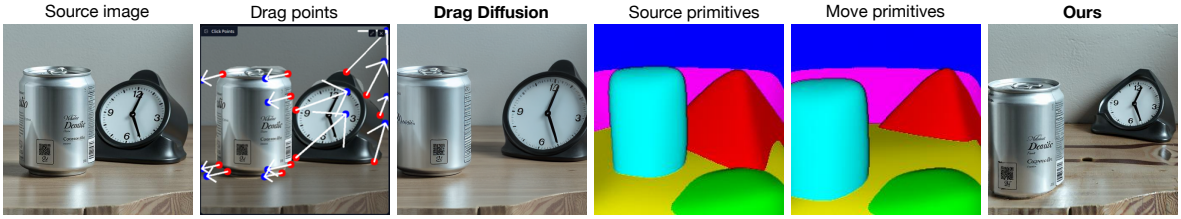

Figure 4. **Comparison with Drag Diffusion [43].** Given a scene (first column), we attempt to reposition objects using a recent point-based image editing method by drawing drag handles (second column). However, drag points are ambiguous: it is unclear whether the intended operation is translation or scaling. As a result, the output lacks geometric consistency (third column). E.g., the clock changes shape, and pushing it deeper into the scene fails to reduce its size appropriately; fine details on the can are lost. In contrast, Generative Blocks World infers 3D primitives (fourth column) that can be explicitly manipulated (fifth column), producing a plausible image that respects object geometry, scale, positioning, and texture (last column).

projected textures so it looks like a real image. The result of our process is an image that respects the text prompt, source texture, and newly edited primitives/camera.

Creating point cloud correspondences We develop a method that accepts point clouds at the ray-primitive intersection points, a convex_map integer array indicating which primitive was hit at each pixel, a list of per-primitive transforms (such as scale, rotate, translate), and a hyperparameter $max\_distance$ for discarding correspondences. This procedure robustly handles camera moves because the input point clouds are representations of the same scene.

Creating a texture hint Given a correspondence map of each 3D point in the new view relative to the original view, we can apply this correspondence to generate a hint image that essentially projects pixels in the old view onto the new view. This is the $\mathbf{x}_{\mathbf{hint}}$ supplied to the image generation model, taking into account both camera moves and primitive edits like rotation, translation, and scaling. The point cloud correspondence ensures that if a primitive moves, its texture moves with it. In practice, this hint is essential for good texture preservation (see Fig. 6). Correspondence and hint generation take about 1-2 seconds per image; 30 denoising steps of FLUX at 512 resolution take about 3 seconds on an H100 GPU.

3.4. Evaluation

We seek error metrics to establish (1) geometric consistency between the primitives requested vs. the image that was

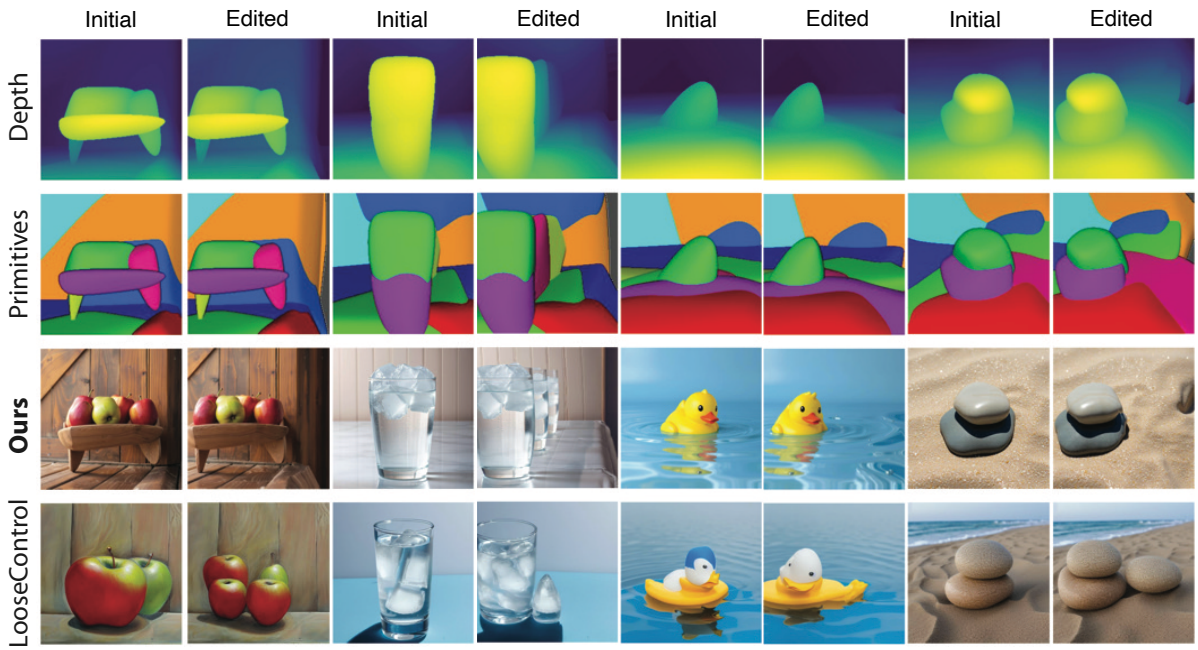

Figure 5. **Comparison with LooseControl [2]**. Camera moves present serious problems for existing work. Four scenes (**left** side of each pair), synthesized from the depth maps shown. In each case, the camera is moved to the right (**right** side of each pair), and the image is resynthesized. Note how, for LooseControl, the number of apples changes (first pair); the level of water in the glass changes and there is an extra ice cube (second pair); the duck changes (third pair); an extra rock appears (fourth pair). In each case, our method shows the same scene from a different view, because the texture hint image is derived from the underlying geometry, and strongly constrains any change.

synthesized and (2) texture consistency between the source and edited image. For (1) we compute the AbsRel between the depth map supplied to the depth-to-image model (obtained by rendering the primitives) and the estimated depth of the synthesized image (we use the hypersim metric depth module from [53] to get linear depth). Consistent with standard practice in depth estimation, we use least squares to fit scale and shift parameters onto the depth from RGB (letting the primitive depth supplied to the DM be GT).

To evaluate texture consistency, we apply ideas from the novel view synthesis literature and our existing point cloud correspondence pipeline. Given the source RGB image and the synthesized RGB image (conditioned on the texture hint), we warp the second image back into the first image's frame using our point cloud correspondence algorithm. If we were to synthesize an image in the first render's viewpoint using the second render, this is the texture hint we would use. In error metric calculation, the first RGB image is considered ground truth, the warped RGB image from the edited synthesized image is the prediction, and the confidence mask filters out pixels that are not visible in view 1, given view 2. This evaluation procedure falls in the category of cycle consistency/photometric losses that estimate reprojection error [16, 24, 29, 39].

3.5. Hyperparameter selection

There are a number of hyperparameters associated with our procedure, and we perform a grid search on a held-out validation set to find the best ones. When generating correspondence maps between point clouds, we let `max_distance`$= 0.005$. In our confidence map, we dilate low-confidence pixels with a score less than $\tau = 0.01$ by 9 pixels, which tells the image model to synthesize new texture near primitive boundaries that are often uncertain. We set (t_{start}, t_{end}) to $(1000, 500)$ by default, though t_{end} can be tuned per test image by the user. Applying the hint for all time steps can reduce blending quality near primitive boundaries; not applying the hint for enough time steps could weaken texture consistency. Allowing some time steps to not follow the hint enables desirable super resolution behavior e.g. when bringing a primitive closer to the camera. See supplementary for detailed algorithms for creating the hint and confidence mask.

Inpainting the hint After warping the source image to the new view, we inpaint low-confidence regions of the hint $\mathbf{x}_{hint}$ before supplying it to the image model. We considered several possibilities, including `cv2_telea` and `cv2_ns` from the OpenCV package, as well as simply leaving them as black pixels. We find that Voronoi inpainting, a variation of nearest neighbor inpainting, worked well. The `voronoi_inpainting` function performs image in-

Method	$\text{AbsRel}_{\text{src}} \downarrow$	$\text{AbsRel}_{\text{dst}} \downarrow$	PSNR ↑	SSIM ↑
Ours	**0.072**	**0.076**	**18.7**	**0.874**
LooseControl [2]	0.143	0.146	6.65	0.670

Table 1. Comparison of image reconstruction and generation metrics between our method and LooseControl. $\textbf{AbsRel}_{\text{src}}$ and $\textbf{AbsRel}_{\text{dst}}$ are absolute relative errors evaluating how well the generated images adhere to the requested primitive geometry (source and modified, respectively). PSNR and SSIM are evaluated by reprojecting the second synthesized image back to the original camera viewpoint (see Sec 3.4 and measuring texture consistency with the source. Observe how our procedure simultaneously offers tight geometric adherence to the primitives while preserving the source texture. Results obtained by averaging 48 test images with random camera moves. Because [2] does not offer primitive extraction code, we supply our own primitives to both methods for evaluation. We use $K = 10$ parts for this evaluation.

painting by filling in regions of low confidence in a hint image using colors from nearby high-confidence pixels, based on a Voronoi diagram approach.

This process leverages a KD-tree for efficient nearest-neighbor searches, ensuring that each pixel adopts the color of the closest reliable pixel, thus preserving color consistency in the inpainted result.

For **FLUX image generation** we begin with the default settings from the diffusers FLUX controlled inpainting pipeline [1]. We set the `strength` parameter (controlling starting noise strength) to 1.0 and `guidance` to 10. We use 30 `num_steps` for denoising. In comparative evaluation, we use the default settings from the authors.

4. Results

In Fig. 4, we show how users can manipulate depth map inputs to depth-to-image synthesizers with our primitive abstractions. We can use these primitives to edit images, as shown in Fig. 5. Notice how we can get precise control over the synthesized geometry while respecting texture, which existing methods struggle to do. We quantitatively evaluate this property in Table 1, demonstrating we hit both goals conclusively. Existing texture preservation methods for multi-frame consistency typically rely on key-value transfer from one image to another. This, unfortunately, does not preserve details very well, only high-level semantics and style. We ablate the advantage of our texture preservation approach in Fig. 6. When there are few primitives, moving one primitive affects a big part of the scene; when there are a lot of primitives, we can make fine-scale edits. We show several such examples in Figs. 7, 8.

[1] https://huggingface.co/docs/diffusers/en/api/pipelines/control_flux_inpaint

5. Discussion

This work demonstrates that we can utilize 3D primitives to achieve precise geometric control over image generation model outputs, and even preserve high-level textures more effectively than existing methods that rely on key-value transfer. A central reason this works is that good primitive decompositions offer several useful properties: they are selectable, allowing intuitive manipulation of scene components; they are object-linked, with boundaries that often correspond to semantic parts; they are scalable, enabling both coarse and fine-grained edits (with fewer and higher source primitive counts); and they are accurate enough to yield depth maps that support high-quality texture projection. Moreover, our pipeline is designed to be user-friendly: since our primitive decomposition is fast, users can easily choose between coarse and fine control by adjusting the number of primitives, and seamlessly switch between decompositions to suit the editing task and scene context.

We believe we have unlocked new interactive controls for image synthesis with our Generative Blocks World. While our method handles problems near primitive boundaries robustly, objects with holes that are not tightly modeled by the primitives (e.g., underneath a chair or a coffee mug handle) are challenging in our current formulation; additional segmentation and masking would be required (or simply using more primitives). Depth-of-field blurring/bokeh may not get resolved or sharpened when bringing out-of-focus objects into focus. Significant object rotations may also fail (see Fig. 10). In an interactive workflow, manually expanding the confidence mask to include problematic regions e.g., unwanted reflections that don't move with a primitive, can fix some issues. Future work that applies our point correspondences within the network layers themselves (e.g., in vital layers) may yield more robust solutions. Our method does not yet account for view-dependent lighting effects and does not enforce temporal consistency across frames for video synthesis.

Our results focus on editing generated images. While we can extract texture hints from real images, our experiments show that edited images should start from the same noise tensor and prompt as the source image to achieve good results. Therefore, good image inverters that work with depth-conditioned diffusion transformers are needed. Additionally, certain extreme edits that are at odds with the text prompt are likely to cause problems (e.g., if the prompt mentions an object is on the right, but a user manipulates the primitives to move the object to the left). Changing the text prompt could work in some circumstances (Fig. 9), or the DiT will inpaint missing regions with content that doesn't harmonize well with the rest of the image. This is due to the delicate link between the text prompt, hint image, initial noise tensor, and depth map.

References

[1] Omri Avrahami, Or Patashnik, Ohad Fried, Egor Nemchinov, Kfir Aberman, Dani Lischinski, and Daniel Cohen-Or. Stable flow: Vital layers for training-free image editing. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 7877–7888, 2025. 5, 12, 14, 20

[2] Shariq Farooq Bhat, Niloy Mitra, and Peter Wonka. Loosecontrol: Lifting controlnet for generalized depth conditioning. In *ACM SIGGRAPH 2024 Conference Papers*, New York, NY, USA, 2024. Association for Computing Machinery. 3, 5, 7, 8

[3] Anand Bhattad, Daniel McKee, Derek Hoiem, and David Forsyth. Stylegan knows normal, depth, albedo, and more. *Advances in Neural Information Processing Systems*, 36: 73082–73103, 2023. 1

[4] Anand Bhattad, James Soole, and David A Forsyth. Stylitgan: Image-based relighting via latent control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4231–4240, 2024. 1

[5] Anand Bhattad, Konpat Preechakul, and Alexei A. Efros. Visual jenga: Discovering object dependencies via counterfactual inpainting, 2025. 2

[6] I Biederman. Recognition by components : A theory of human image understanding. *Psychological Review*, (94):115–147, 1987. 2

[7] TO Binford. Visual perception by computer. In *IEEE Conf. on Systems and Controls*, 1971. 2

[8] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[9] Yen-Chi Cheng, Krishna Kumar Singh, Jae Shin Yoon, Alexander Schwing, Liangyan Gui, Matheus Gadelha, Paul Guerrero, and Nanxuan Zhao. 3D-Fixup: Advancing Photo Editing with 3D Priors. In *Proceedings of the SIGGRAPH Conference Papers*. ACM, 2025. 3

[10] Yutao Cui, Xiaotong Zhao, Guozhen Zhang, Shengming Cao, Kai Ma, and Limin Wang. Stabledrag: Stable dragging for point-based image editing. In *European Conference on Computer Vision*, pages 340–356. Springer, 2024. 3

[11] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 3

[12] Xiaodan Du, Nicholas Kolkin, Greg Shakhnarovich, and Anand Bhattad. Generative models: What do they know? do they know things? let's find out! *arXiv preprint arXiv:2311.17137*, 2023. 1

[13] Abdelrahman Eldesokey and Peter Wonka. Build-a-scene: Interactive 3d layout control for diffusion-based image generation. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[14] Dave Epstein, Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. 2023. 3

[15] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. 5

[16] Qihang Fang, Yafei Song, Keqiang Li, Li Shen, Huaiyu Wu, Gang Xiong, and Liefeng Bo. Evaluate geometry of radiance fields with low-frequency color prior. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2024. 7

[17] David F. Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3d primitives for single image understanding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013. 2

[18] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations*, 2022. 3

[19] Weiran Guang, Xiaoguang Gu, Mengqi Huang, and Zhendong Mao. Dragin3d: Image editing by dragging in 3d space. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 21502–21512, 2025. 3

[20] Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision (ECCV)*, 2010. 2

[21] Varsha Hedau, Derek Hoiem, and David Forsyth. Thinking inside the box: using appearance models and context based on room geometry. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, page 224–237, Berlin, Heidelberg, 2010. Springer-Verlag.

[22] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007. 2

[23] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2

[24] Yoonwoo Jeong, Jinwoo Lee, Seokyeong Lee, Doyup Lee, and Minhyuk Sung. NVS-Adapter: Plug-and-play novel view synthesis from a single image. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 7

[25] Nikolai Kalischek, Michael Oechsle, Fabian Manhardt, Philipp Henzler, Konrad Schindler, and Federico Tombari. Cubediff: Repurposing diffusion-based image models for panorama generation, 2025. 2

[26] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3992–4003, 2023. 4

[27] Florian Kluger, Hanno Ackermann, Eric Brachmann, Michael Ying Yang, and Bodo Rosenhahn. Cuboids revis-

ited: Learning robust 3d shape fitting to single rgb images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 5

[28] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 2, 4, 5

[29] Feifei Li, Qi Song, Chi Zhang, Hui Shuai, and Rui Huang. PoI: Pixel of interest for novel view synthesis assisted scene coordinate regression. *arXiv preprint arXiv:2502.04843*, 2025. 7

[30] Haolin Liu, Yujian Zheng, Guanying Chen, Shuguang Cui, and Xiaoguang Han. Towards high-fidelity single-view holistic reconstruction of indoor scenes. In *European Conference on Computer Vision*, pages 429–446. Springer, 2022. 2

[31] Oscar Michel, Anand Bhattad, Eli VanderBilt, Ranjay Krishna, Aniruddha Kembhavi, and Tanmay Gupta. Object 3dit: Language-guided 3d-aware image editing. *Advances in Neural Information Processing Systems*, 36:3497–3516, 2023. 3

[32] Tom Monnier, Jake Austin, Angjoo Kanazawa, Alexei A. Efros, and Mathieu Aubry. Differentiable Blocks World: Qualitative 3D Decomposition by Rendering Primitives. In *NeurIPS*, 2023. 2

[33] Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models. *arXiv preprint arXiv:2307.02421*, 2023. 3

[34] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(5):4296–4304, 2024. 2

[35] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–11, 2023. 3

[36] Karran Pandey, Paul Guerrero, Matheus Gadelha, Yannick Hold-Geoffroy, Karan Singh, and Niloy J. Mitra. Diffusion handles enabling 3d edits for diffusion models by lifting activations to 3d. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7695–7704, 2024. 3

[37] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[38] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *Proceedings of the International Conference on Learning Representations (ICLR)*. OpenReview.net, 2023. 3

[39] Yuxin Qin, Xinlin Li, Linan Zu, and Ming Liang Jin. Novel view synthesis with depth priors using neural radiance fields and cyclegan with attention transformer. *Symmetry*, 17(1), 2025. 7

[40] L. G. Roberts. *Machine Perception of Three-Dimensional Solids*. PhD thesis, MIT, 1963. 2

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10684, 2022. 2

[42] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[43] Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8839–8849, 2024. 3, 6

[44] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, page 746–760, Berlin, Heidelberg, 2012. Springer-Verlag. 5

[45] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22819–22829, 2023. 3

[46] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[47] Vaibhav Vavilala and David Forsyth. Convex decomposition of indoor scenes. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9142–9152, 2023. 2, 4, 5

[48] Vaibhav Vavilala, Rahul Vasanth, and David Forsyth. Denoising monte carlo renders with diffusion models, 2024. 2

[49] Vaibhav Vavilala, Florian Kluger, Seemandhar Jain, Bodo Rosenhahn, Anand Bhattad, and David Forsyth. Improved convex decomposition with ensembling and boolean primitives, 2025. 2, 4, 5

[50] Vaibhav Vavilala, Faaris Shaik, and David Forsyth. Dequantization and color transfer with diffusion models. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 9630–9639, 2025. 2

[51] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12619–12629, 2023. 3

[52] Xiaoyan Xing, Konrad Groh, Sezer Karaoglu, Theo Gevers, and Anand Bhattad. Luminet: Latent intrinsics meets diffusion models for indoor scene relighting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 442–452, 2025. 1

[53] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *Advances in Neural Information Processing Sys-*

tems, pages 21875–21911. Curran Associates, Inc., 2024. 3, 4, 7

[54] Jiraphon Yenphraphai, Xichen Pan, Sainan Liu, Daniele Panozzo, and Saining Xie. Image sculpting: Precise object editing with 3d geometry control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4241–4251, 2024. 3

[55] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824, 2023. 2, 4, 5

[56] Ruisi Zhao, Zechuan Zhang, Zongxin Yang, and Yi Yang. 3d object manipulation in a single image using generative models, 2025. 3

[57] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2017. 2

[58] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2