Lecture 11:

Generating 3D Objects and Scenes

Visual Computing Systems
Stanford CS348K, Spring 2026

Today

- Broad overview of state-of-the-art capabilities of AI models for generating 3D objects and 3D scenes
- Recall last time: enhancing lower-quality renderings of 3D scenes into near-photorealistic images
- Next time: ditching traditional 3D representations completely to generate interactive worlds

Demo:
Text/image to textured 3D object

Training data?

Far less available 3D object data than image data.

LAION-5B: A NEW ERA OF OPEN LARGE-SCALE MULTI-MODAL DATASETS

by: Romain Beaumont, 31 Mar, 2022

We present a dataset of 5,85 billion CLIP-filtered image-text pairs, 14x bigger than LAION-400M, previously the biggest openly accessible image-text dataset in the world - see also our NeurIPS2022 paper

See our update on the LAION-5B dataset.

Large image-text models like ALIGN, BASIC, Turing Bletchly, FLORENCE & GLIDE have shown better and better performance compared to previous flagship models like CLIP and DALL-E. Most of them had been trained on billions of image-text pairs and unfortunately, no datasets of this size had been openly available until now. To address this problem we present LAION 5B, a large-scale dataset for research purposes consisting of 5,85B CLIP-filtered image-text pairs. 2,3B contain English language, 2,2B samples from 100+ other languages and 1B samples have texts that do not allow a certain language assignment (e.g. names). Additionally, we provide several nearest neighbor indices, an improved web interface for exploration & subset creation as well as detection

LAION-5B: public dataset of 5B text-image pairs

Objaverse-XL

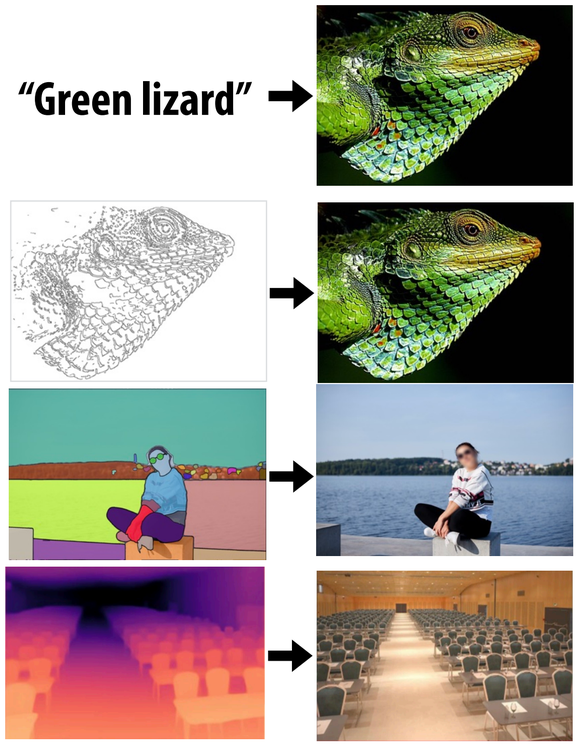
A Universe of 10M+ 3D Objects

arXiv | Google Colab | GitHub | Hugging Face

3D object diffusion

Recall our guided diffusion example

- Given paired C1 -> X2, train a diffusion model…



$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\ \mathbf{z}_i, \quad i = 0, 1, \cdots, T$$

("score function")

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x})$$

(Unguided score function)

Modify image x so that image is more likely [to come from the BILLION IMAGE training set]

(Prompt guidance)

Modify image x to make the guidance signal more likely related to the image

But now we want INPUT → 3D model

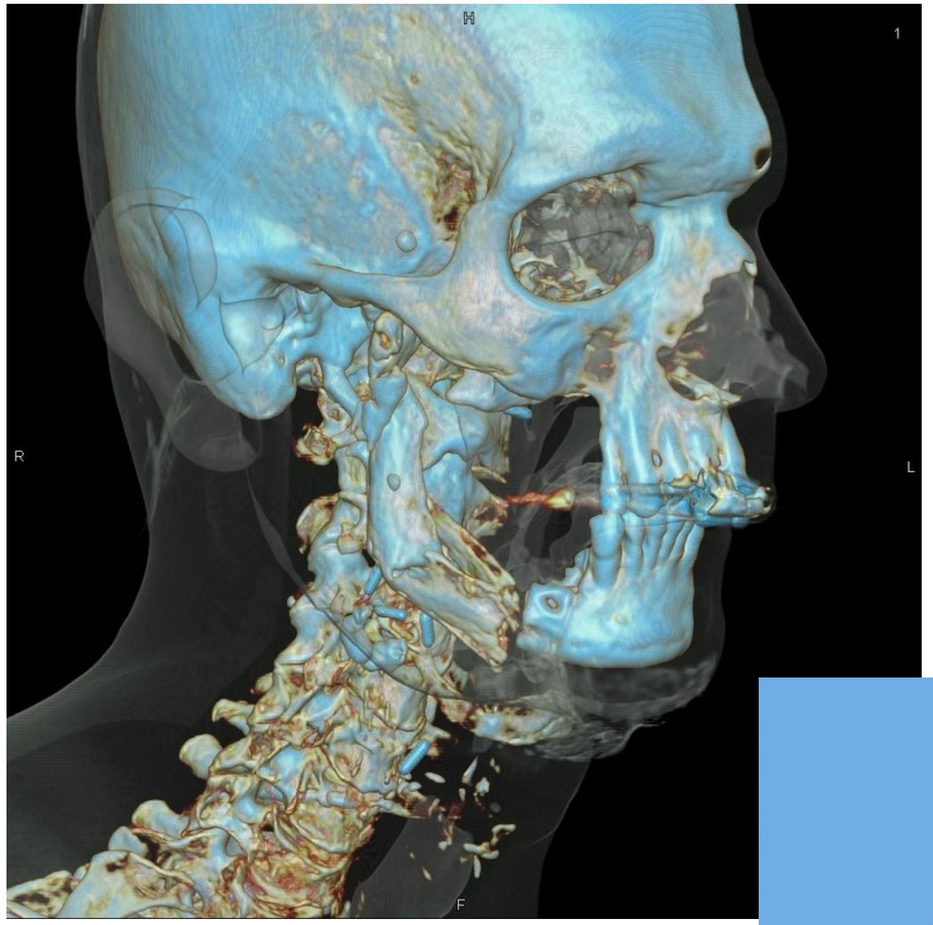
Let's represent 3D objects as volumes

Volume density and "color" at all points in space.

$$\sigma(\mathrm{p})$$

$$c(\mathrm{p}, \omega) = c(x, y, z, \phi, \theta)$$

The reflectance off surface at point p in direction ω



Aside: rendering volumes

Given "camera ray" from point o in direction w….

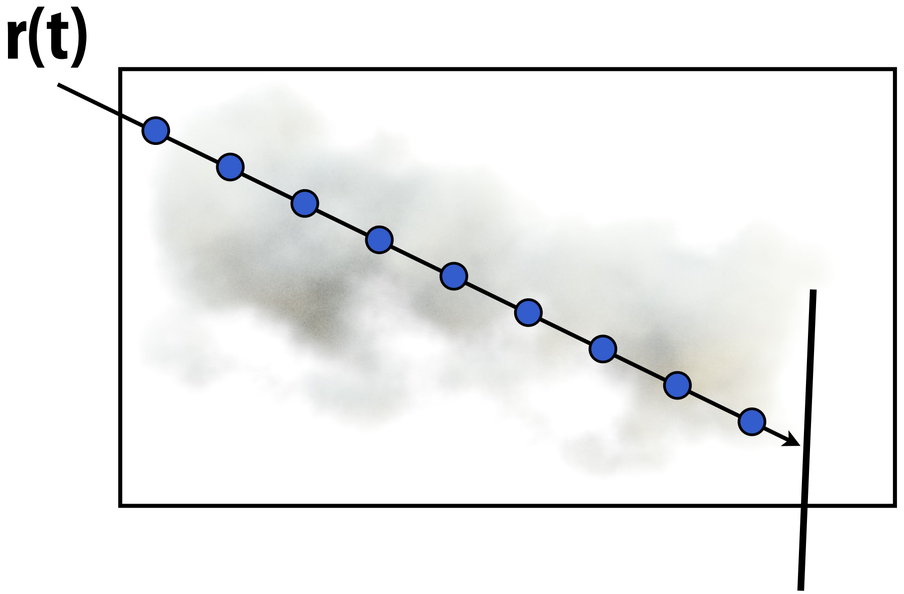
$$\mathbf{r}(t) = \mathbf{o} + t\omega$$

And volume with density and directional radiance.

$\sigma(\mathrm{p})$

$c(\mathrm{p}, \omega)$

Volume density and color at all points in space.



Step through the volume to compute radiance along the ray.

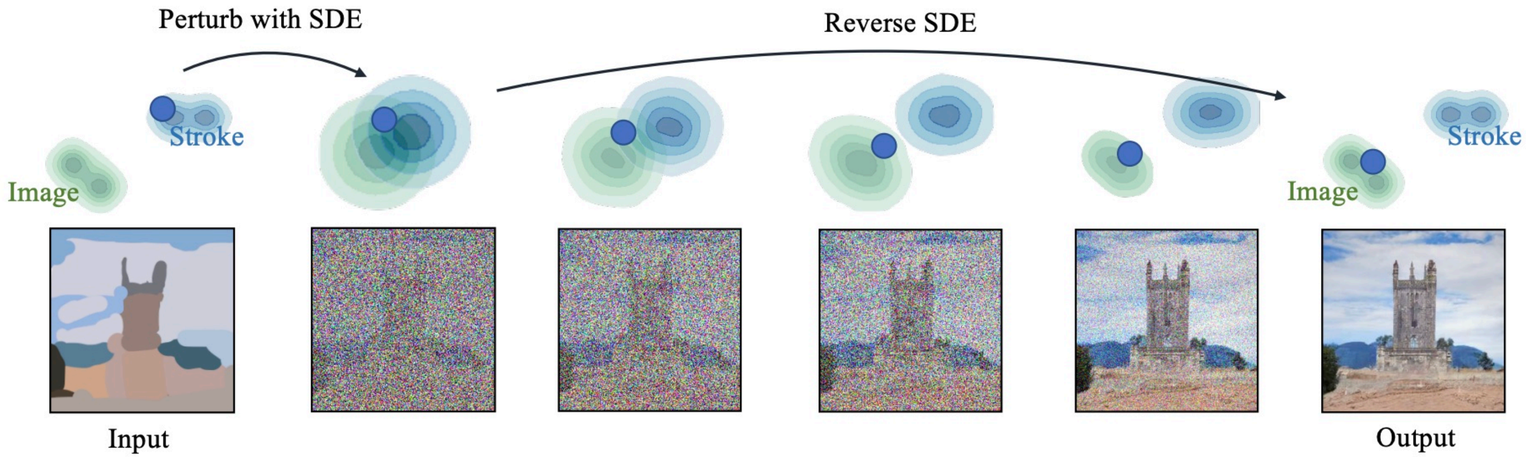
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Distilling a 3D generation model

- The takeaway: we can make an image from a 3D representation (e.g., a volume) using a differentiable rendering function

$$\mathrm{x} = R(\theta)$$

- And, with a image diffusion model, we can push that image closer to the distribution of real images



1. Start with a guide image (a target)
2. Add "small" amount of noise
3. Iteratively denoise to produce sample from target image distribution

Distilling a 3D generation model

- The takeaway: we can make an image from a 3D representation (e.g., a volume) using a differentiable rendering function

$$\mathrm{x} = R(\theta)$$

- And, with a text-conditioned image diffusion model, we can push image closer to the distribution of real images associated with a given prompt. That diffusion denoting step produces…

$$\Delta \mathrm{x}$$

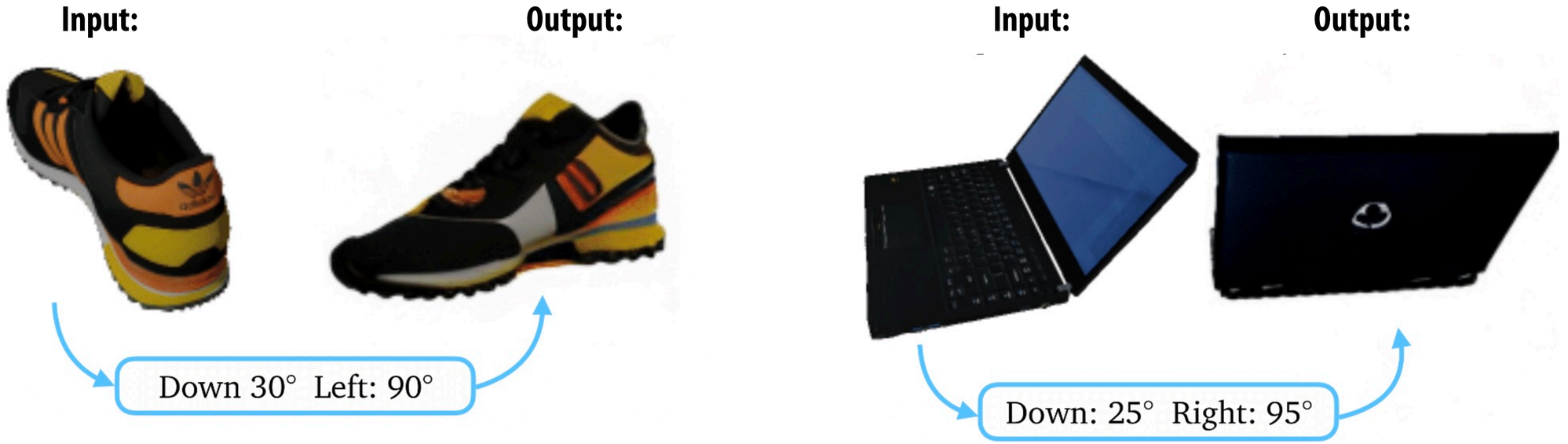
- Now, given $\nabla R(\theta)$ (recall R() was differentiable), optimize the parameters θ of the 3D representation to produce…

$$\mathrm{x} + \Delta \mathrm{x}$$

- In other words, we've converted the score function of a text-conditioned image diffusion model into a training procedure for a text-conditioned 3D diffusion model

Image-conditioned 3D diffusion

- Now let's say we want to condition 3D generation based on an image, not text:
- How about a simpler image editing problem: given a reference image X, and camera change parameters (rotation, translation), produce a novel view of the object in the image



Take a (pretty big) dataset of objects, fine-tune conditioned image diffusion model on pairs

Rendered from viewpoint 1

Rendered from viewpoint 2

Input (conditioning)

Output

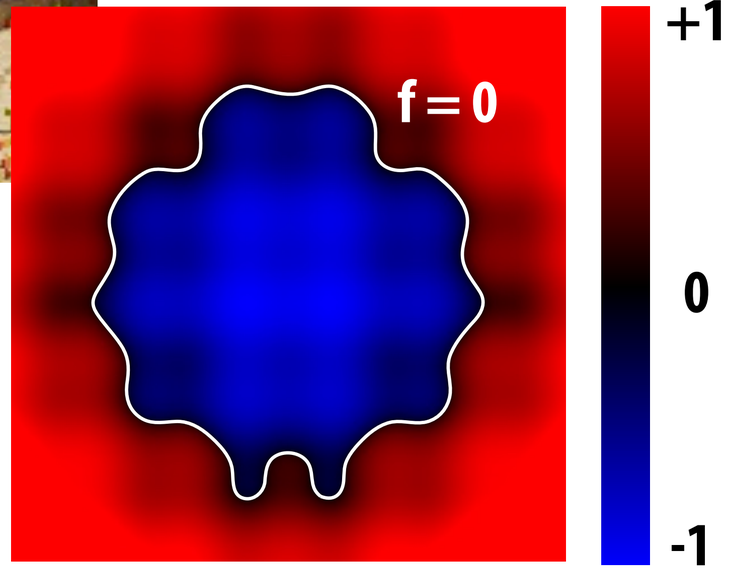
(X,R,T)

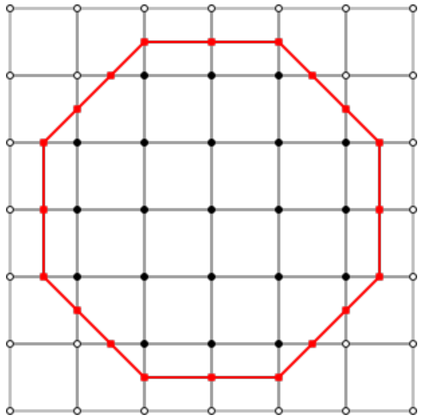
Now let's apply the same distillation trick

- We can now train an image conditioned 3D generation model using a similar process as described before in lecture

One note:

- Conversion from occupancy grid or signed distance function representations to 3D mesh via techniques like marching cubes / dual-contouring / flexi-cubes





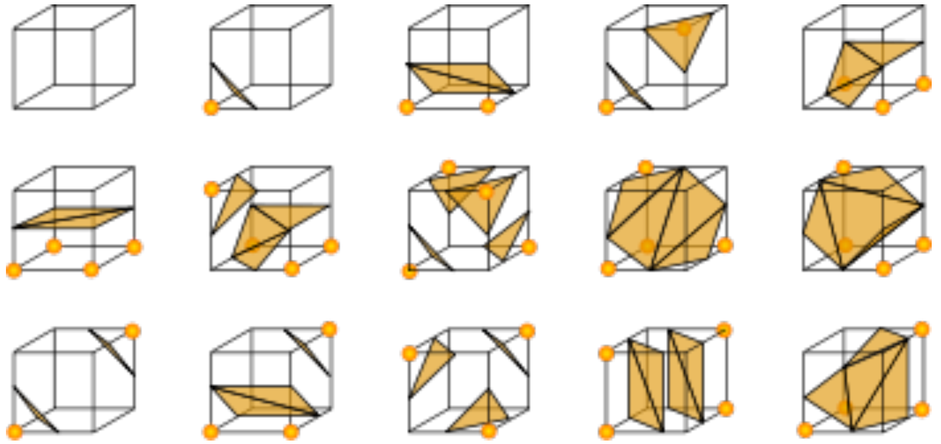
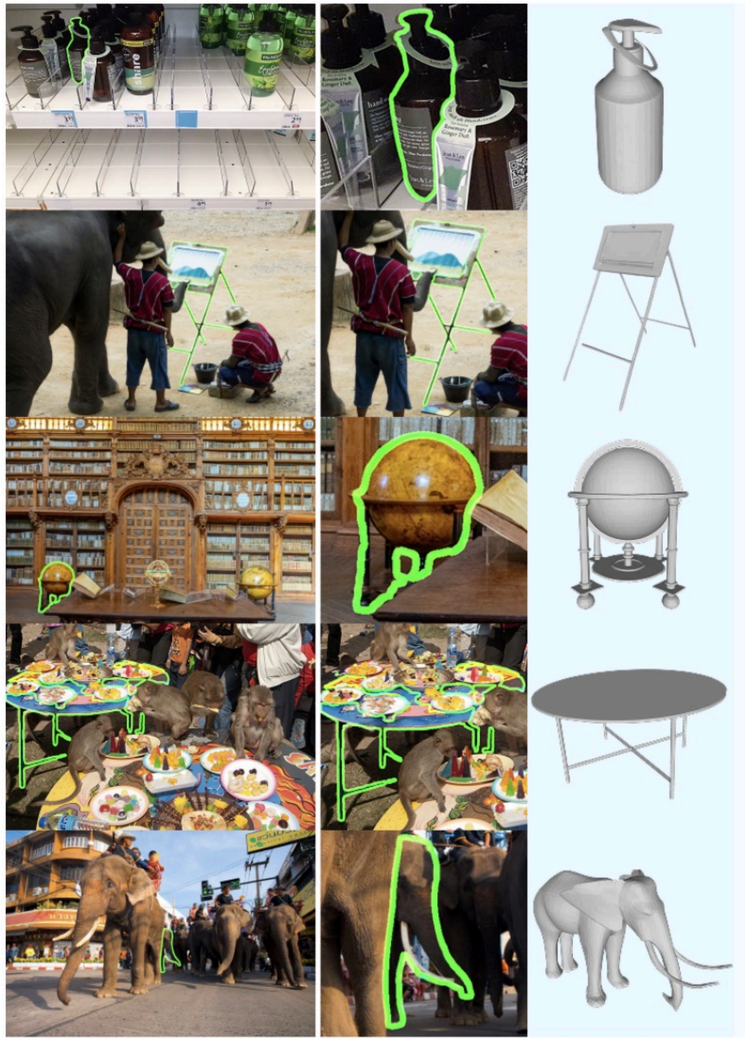


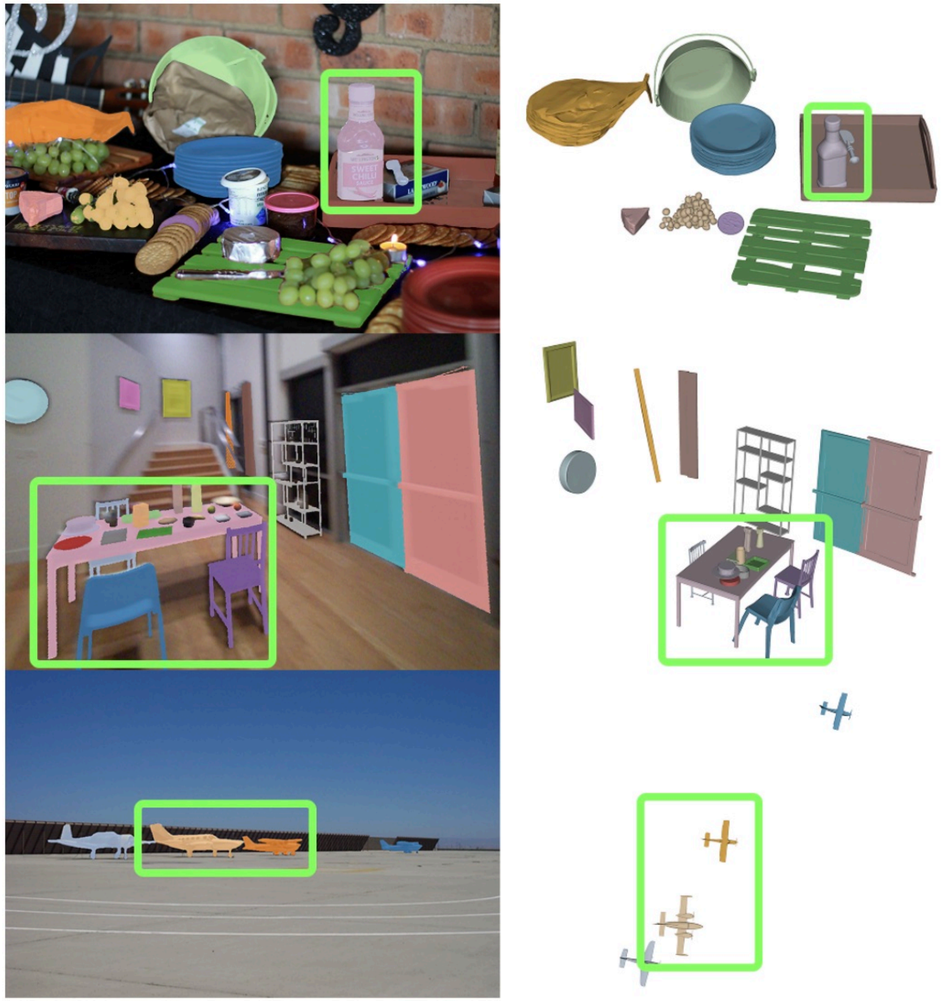
Image credits: BorisTheBrave.com, Credit: Voxel Ville NFT (voxelville.io)

3D from real-world scenes (SAM 3D)

- Take a photo of a real scene (object in real world context) and extract both 3D geometry and layout



Input Image Zoom-In SAM 3D (Ours)



Input Image w/ Overlaid Mask SAM 3D (Ours)

SAM 3D data pipeline (tonight's reading)

- Presents model with "curriculum" of increasingly difficult 3D extraction tasks generated synthetically

Segmented 3D meshes

- To be useful in interactive experiences, 3D content often needs to be segmented into distinct articulatable parts (e.g., parts must move relative to body, a door must swing open)

Generating 3D Scenes

Assume: library of individual 3D assets

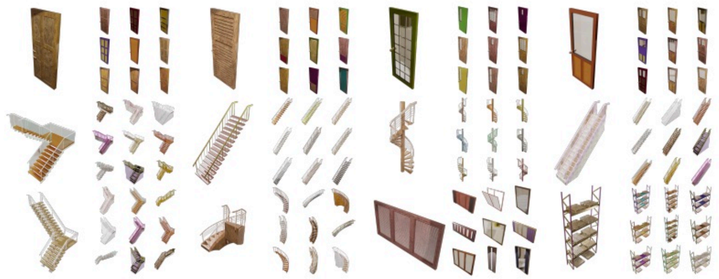


Figure 3. Random samples of procedurally generated doors (top), staircases (middle/bottom) and windows/warehouse shelving (bottom-right).

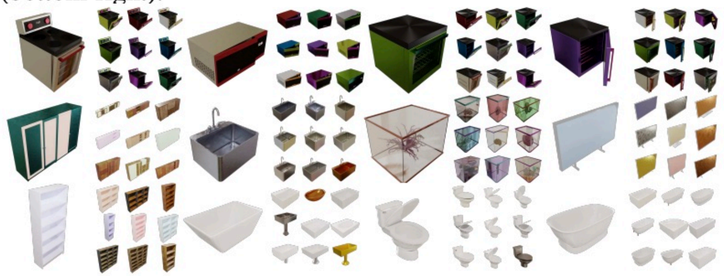
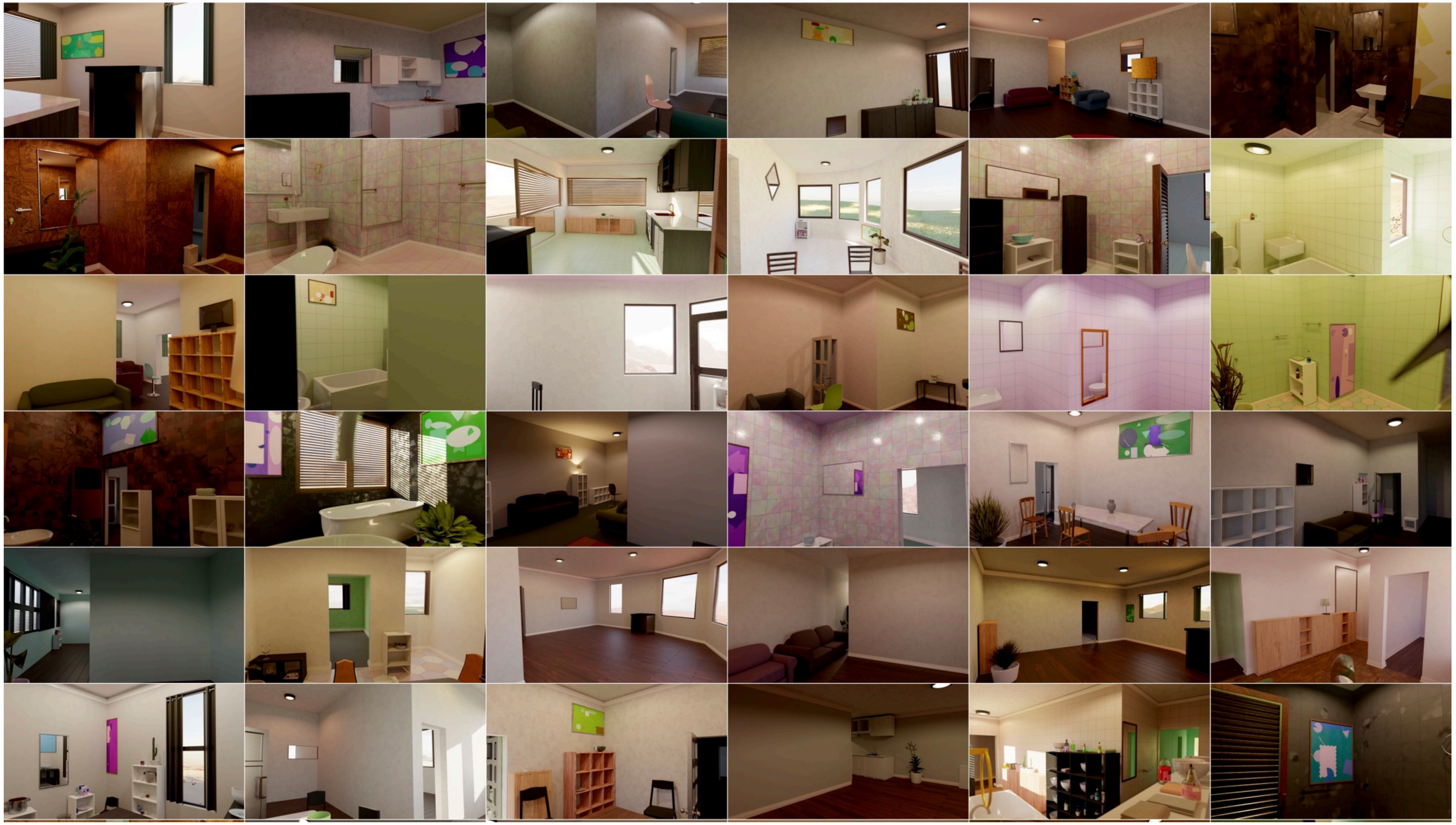


Figure 4. Random samples of procedurally generated ovens, dishwasher and sinks (top/middle), living-room furniture (middle) and bathroom fixtures (bottom).

Figure 5. Random samples of procedurally generated furniture, including sofa, chairs, and beds (top), tables (middle/bottom), and shelves (bottom).

How do we assemble the assets into a good scene layout?



Spatial layout constraints

- Simple language for describing constraints on spatial layout
- Iterative solver that attempts to satisfy constraints

```
tables = scene.with_semantics(Tag.Chair)
chairs = scene.with_semantics(Tag.Table)
on_floor = cl.StableAgainst(Tag.Bottom, Tag.Floor)
constraints = cl.forall(
    rooms[Tag.DiningRoom],
    r,
    (tables.related_to(r, on_floor).count() == 1,
     chairs.related_to(r, on_floor).count().in_range(2, 10)),
)
score = (
    chairs.related_to(rooms[Tag.DiningRoom], on_floor)
    .mean(c, cl.reflection_symmetry(c, tables) + \
                cl.min_distance(t, tables) * (-1))
    .maximize(weight=2)
)
```

Constraints for proximity, symmetry, alignment, accessibility, object volume, etc.

Our solver allows both discrete and continuous moves:

Addition - Adds a procedural object to the scene.
Deletion - Deletes an object from the scene.
Relation Plane Change - Assigns an object to another plane.
Resample - Regenerates an object with new parameters.
Reinitialize Pose - Samples a new random pose for an object.
Translate - Translates the object within its DoF plane.
Rotate - Rotates the object around its DoF axis.

Floor plan layout objective function is a weighted combination of terms:

- Shortest path to entrance
- Room aspect ratio
- Room wall conciseness
- Room collinearity
- Exterior length by room
- Staircase occupancies
- Typical room area
- Room convexity
- Functional room area
- Narrow passages
- Exterior corners by room
- Staircase IOU with rooms

Impact of layout constraints

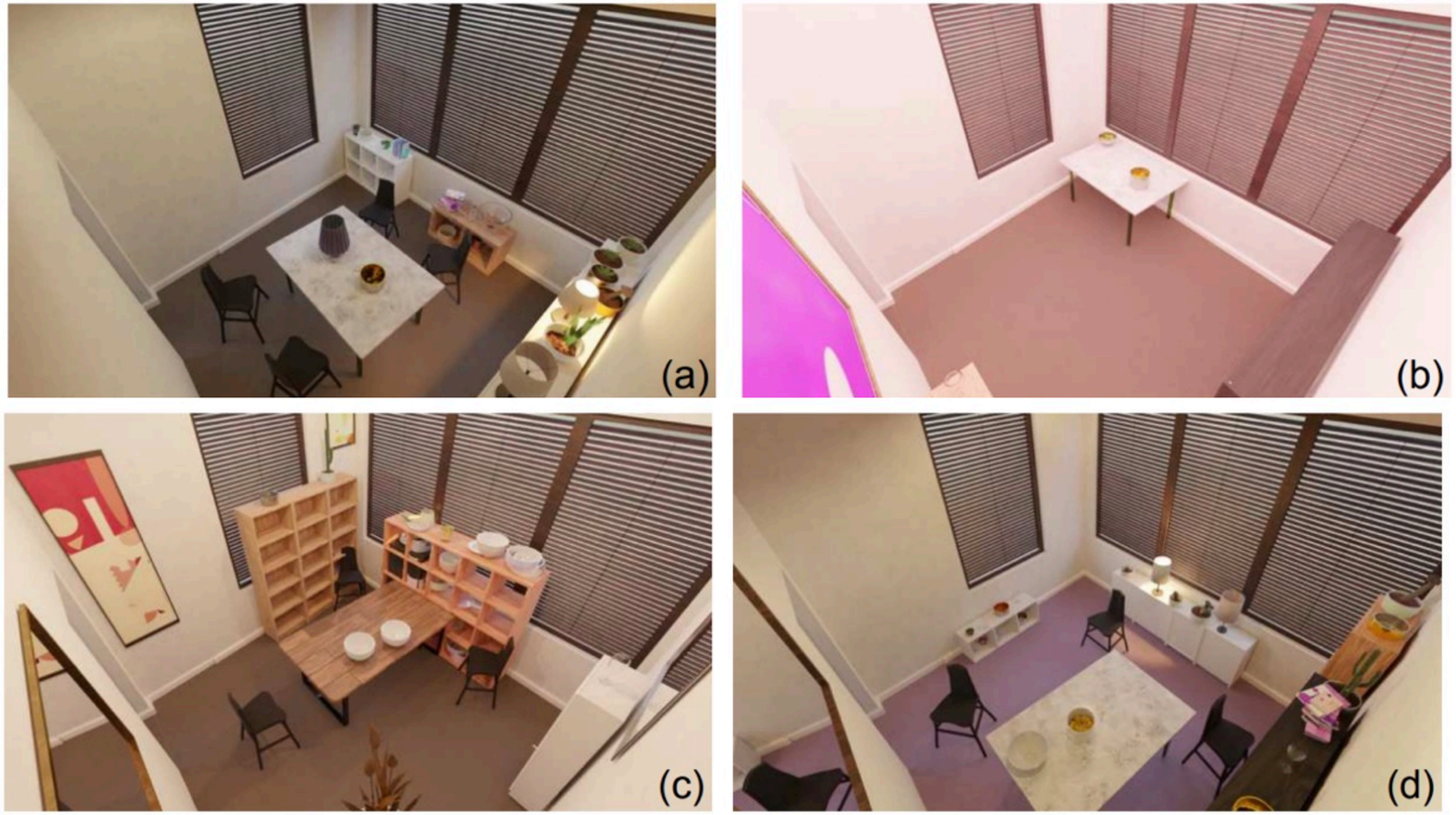
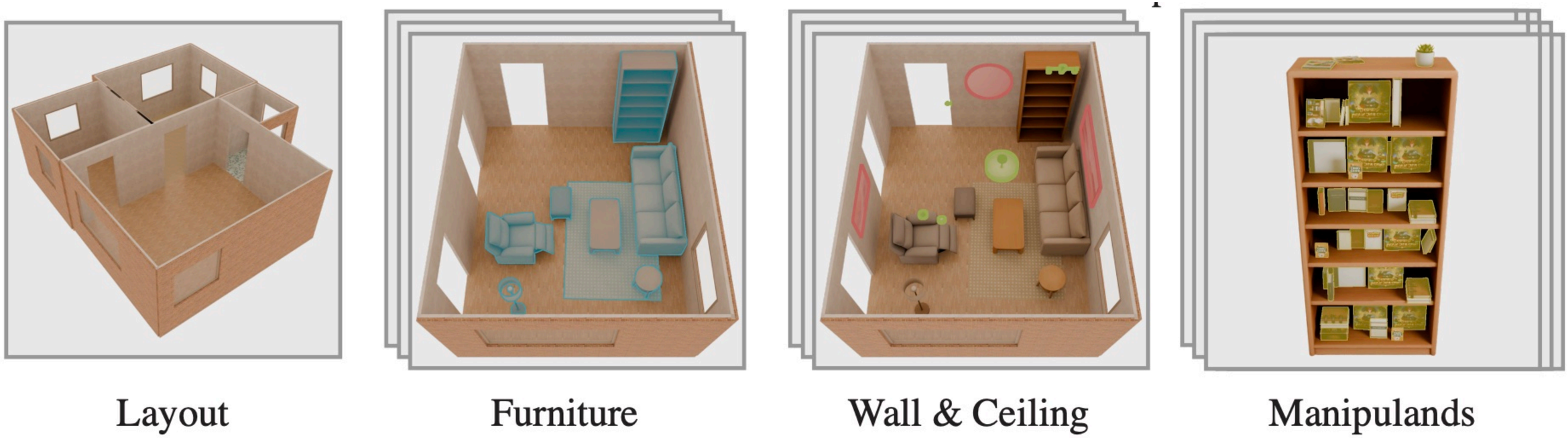


Figure 15. Qualitative Ablation. From left to right, we show scenes generated by our system with 10K solving steps (a), 1K solving steps (b), with collision checking removed (c) and with symmetry terms disabled (d)

LLM-based layout generation

- Given library of objects (or 3D object generator), ask LLM to directly place objects in a scene
- e.g., place/orient/scale bounding boxes with text descriptions
- Typically with some hierarchical structure to order of generation



Agentic layout generation

- Given access to renderer, VLM to analyze rendered scenes, or evaluation functions (intersection tests, etc) that execute on the scene, iteratively have LLM improve scene's layout
- Generate a layout —> assess layout —> take a next design action

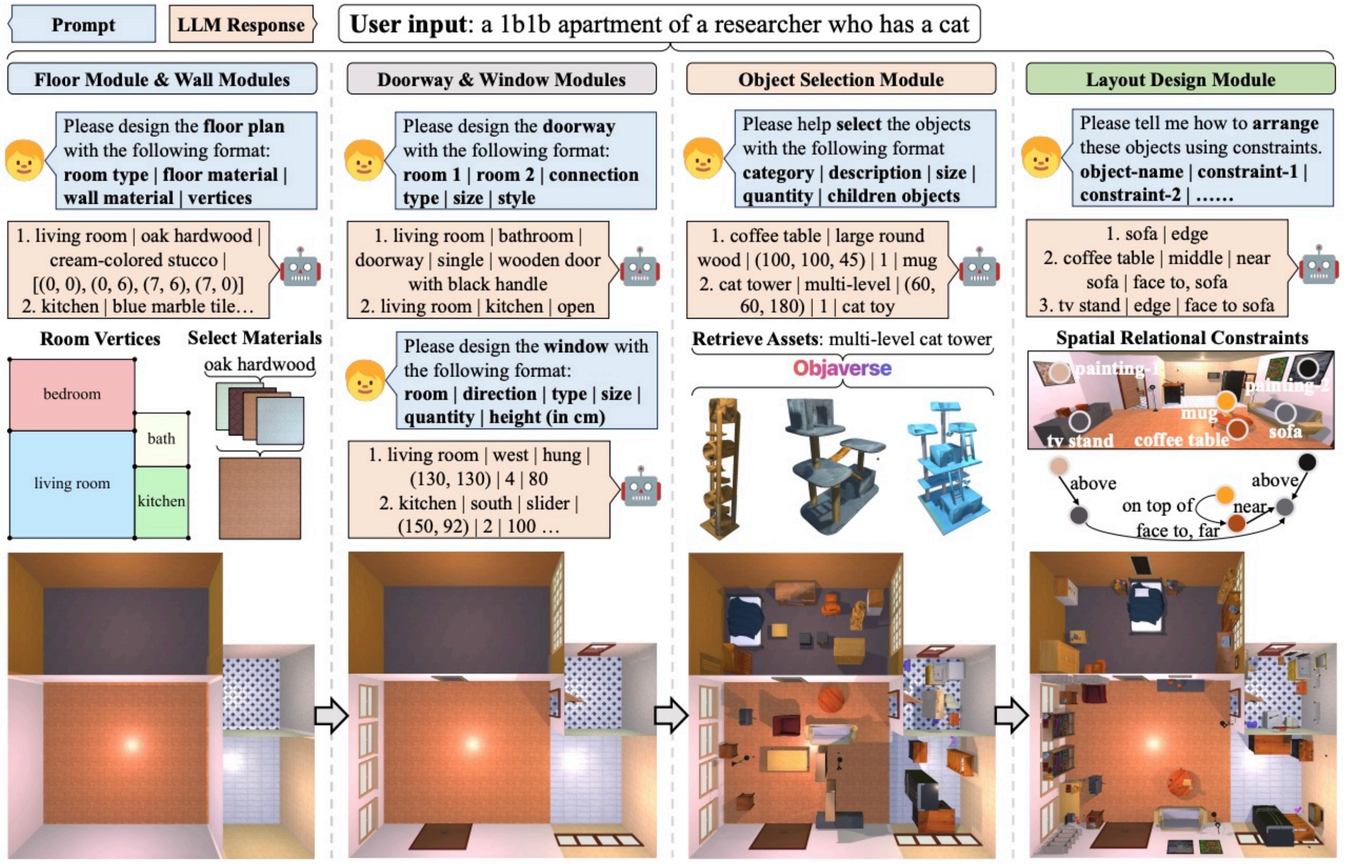


Figure 2. Given a text input, HOLODECK generates the 3D environment through multiple rounds of conversation with an LLM.

Overall trends in 3D scene generation

- Use of LLM agentic flows (given access to 3D IDE tools) to generate scene layouts
  - Conditioned on text descriptions or images
- Domain-specific layout APIs typically of common computer graphics tools aid the LLM
- Hard-coded (or LLM-produced) verifiers or constraints to provide feedback
  - "Default" feedback mechanism — have a VLM look at a rendering of the scene to produce feedback
  - Common division of LLM/VLM work into (generator, critic)
- Use of optimization to minimize cost terms to "refine" layouts

Increasing interest in “simulatable” scenes

- Typically for Robotics use cases
- Articulated objects, or objects with estimated mass

Discussion:
How might you verify/evaluate a scene layout given a scene description (text description or example image)