R3DS: Reality-linked 3D Scenes for Panoramic Scene Understanding Supplemental Materials

In this supplement, we provide additional examples and statistics for our R3DS dataset (Appendix A) and details on our annotation interface (Appendix B).

A R3DS dataset examples and statistics

We show a histogram of the region types covered by our dataset (Figure 1), and histograms of the object categories (Figures 2 and 3a to 3d). We first show a histogram of the 20 most commonly occurring coarse object categories in Figure 2 and then fine-grained category distributions for some broader object categories such as 'Chair', 'Sofa', 'Table' and 'Lighting' in Figures 3a to 3d. We also present a box plot of the physical size distribution (measured by volume in m^3) per category in Figure 4.

We show additional qualitative examples of scenes in our R3DS dataset in Figures 5 and 6.

We also provide statistics of object support relations to architecture elements (Figure 7) and other objects (Figure 8). As expected, we see that chairs typically go on floors, while curtains are supported by walls. From Figure 8, we see that cushions are typically found on beds, chairs, and couches while towels are typically found on shelves. Similarly, in Figure 9 we show the object-to-region statistics. We see that some object categories tend to appear more frequently in a particular region (i.e. room) type. For example, couches are more frequently found in living rooms than in bedrooms.

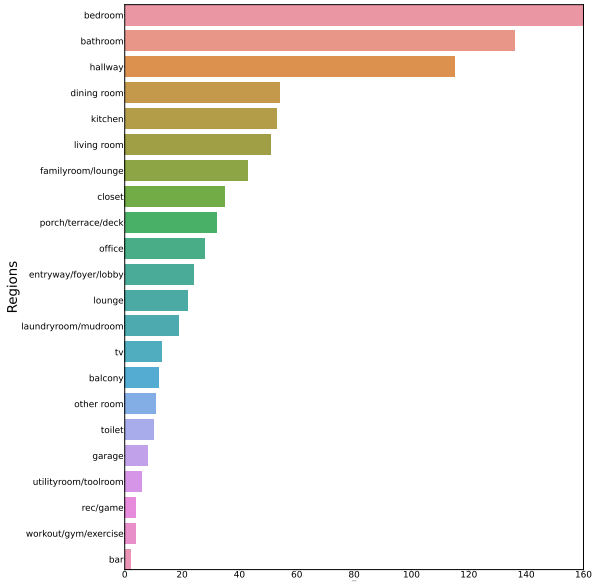


Fig. 1: Histogram of all region (i.e. room) types in our R3DS dataset. The three most common region types are bedroom, bathroom and hallway, but there is a long-tail distribution with many other region types.

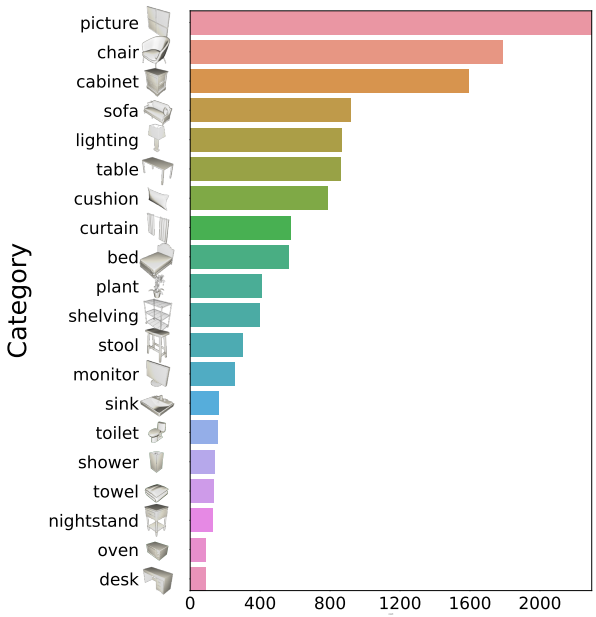


Fig. 2: Coarse Object Categories. Histogram of the 20 most common object categories in R3DS. We see that the scenes in our dataset exhibit a long-tail distribution over common object categories occurring in real-world scenes.

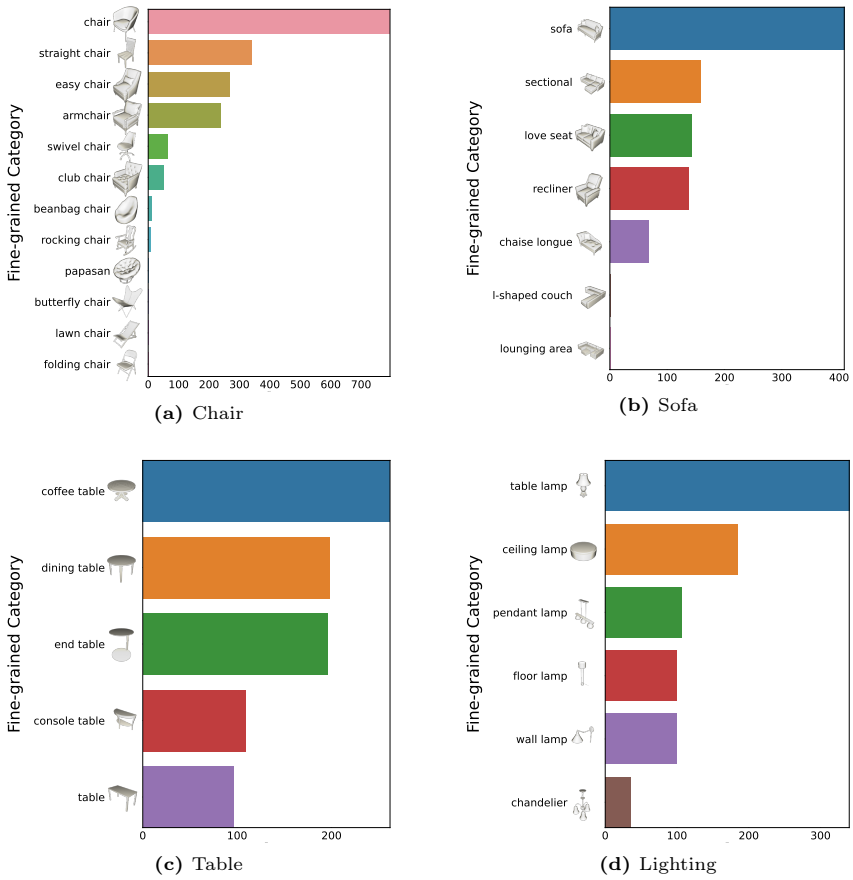


Fig. 3: Histograms of the fine-grained categories for 'Chair', 'Sofa', 'Table' and 'Lighting'. There are a variety of object instances in our scenes. Note that lighting objects are also found in dramatically different support relations with the architecture and other objects (e.g., table lamp vs ceiling lamp vs floor lamp).

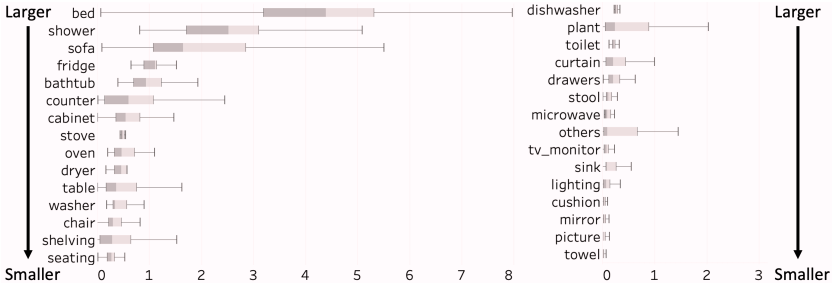


Fig. 4: Box plot of the physical size distribution (measured by volume in m^3) per category, showing a broad spectrum of sizes with several small categories (right side).

Fig. 5: Top-down overviews of various annotated scenes with objects colored by instances. Our dataset covers several region types including kitchen, bedroom, bathroom, office, lounge room and more. The object arrangements are dense, with objects supported by other objects (e.g., pillows on beds and couches) and by architectural elements (e.g., paintings on walls and curtains on windows).

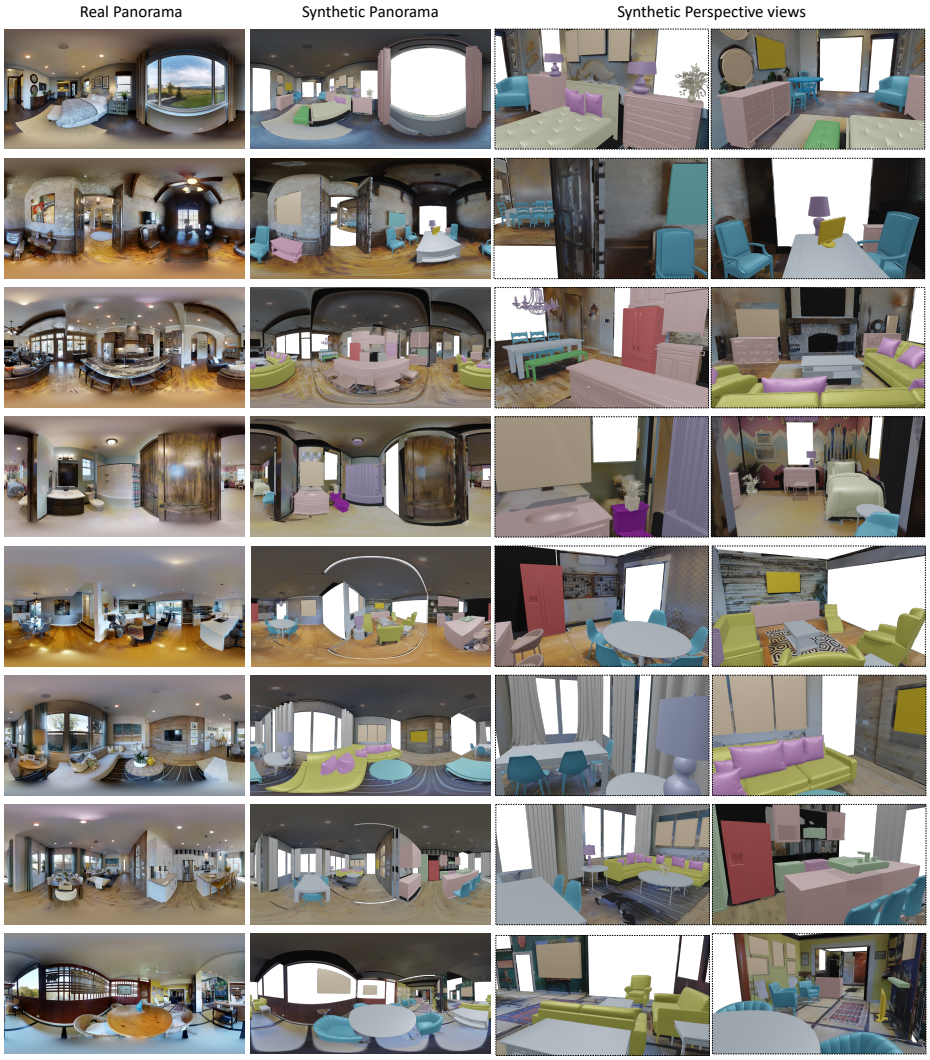


Fig. 6: Examples of real-world panoramas and corresponded synthetic panoramas. The synthetic scenes are rendered using the annotated textured 3D architecture and all placed synthetic objects, from the same camera pose as the original panorama. The rightmost two columns show additional sampled perspective views within each scene. Objects are colored according to semantic category label. The scenes are densely populated with objects plausibly arranging on other objects and with respect to the architecture.

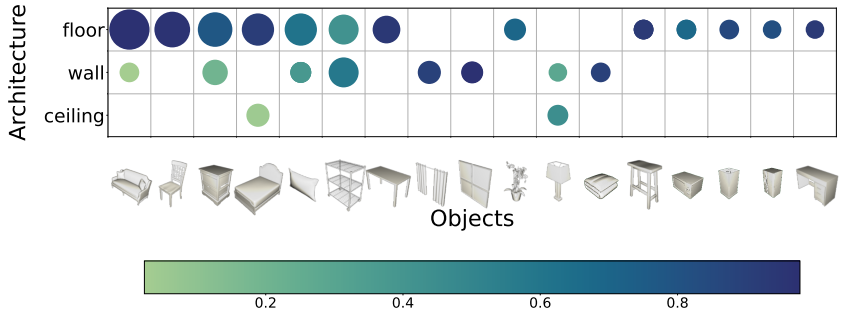


Fig. 7: Object-to-architecture support. The plot shows support between objects and architecture elements (floor, wall, ceiling). The circle radius indicates the average number of the object type (ranges from 5 to 24) supported by a given arch type per panorama, whereas the color indicates the number of times the object type is supported by that arch type out of the total number of times the object type appears. For example, we see that on average, same number of shelves can be found on the floor and the wall in a panorama (circle radius), but overall shelves appear more on the wall than on the floor (color).

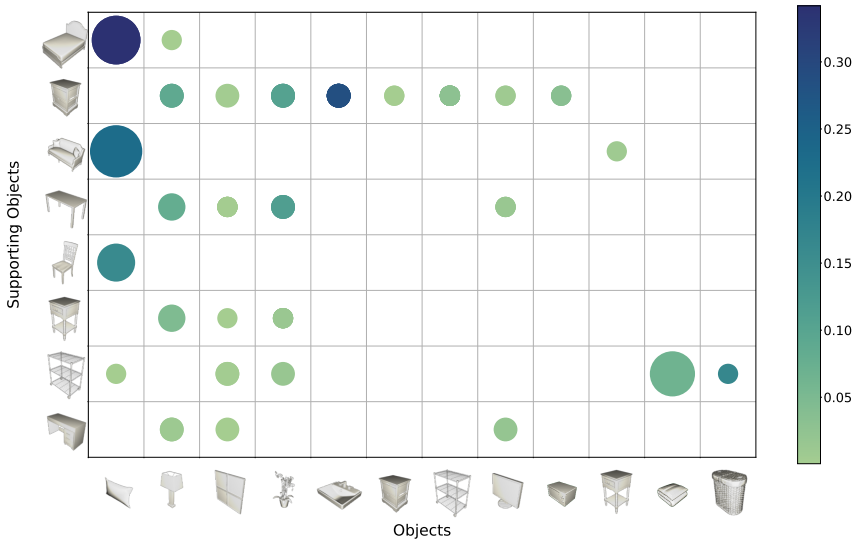


Fig. 8: Object-to-object support. The plot shows support between various objects in R3DS. The circle radius indicates the average number of the object type (ranges from 1 to 10) supported by the parent object type per panorama, whereas the color indicates the number of times the object type is supported by the parent object type out of the total number of times the object type appears in the dataset. For example, we see that on average, more pillows are found on a couch than on a bed in a panorama (circle radius), but overall pillows appear more frequently on the bed in the dataset (color).

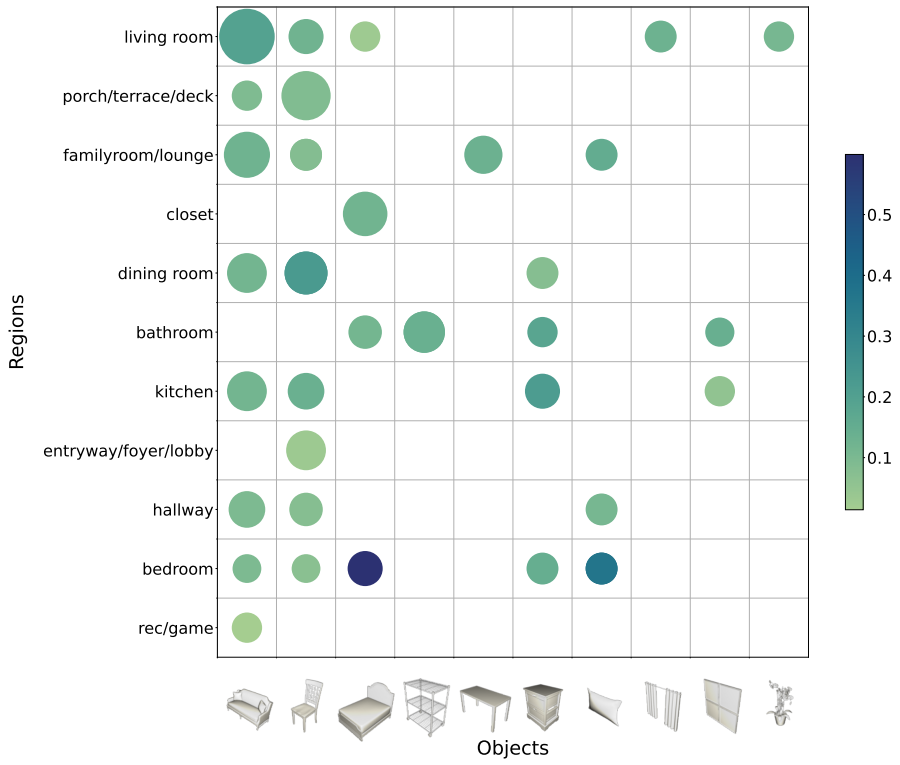


Fig. 9: Object-to-region. The plot shows objects found in various regions in R3DS. The circle radius indicates the average number of the object type (ranges from 7 to 24) found in a region per panorama, whereas the color indicates the number of times the object type occurs in that region out of the total number of times the object type appears in the dataset. For example, we see that on average, similar number of chairs are found in dining room and lobby (circle radius), but overall chairs appear more frequently in the dining room in the dataset (color).

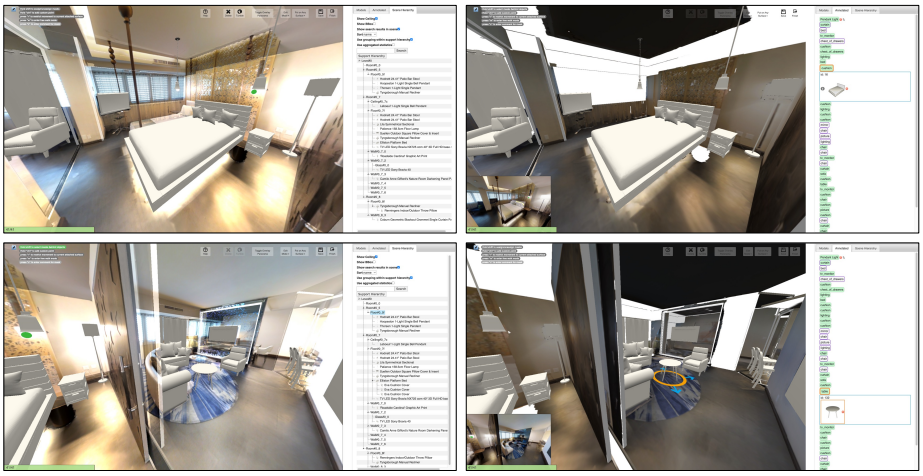


Fig. 10: The R3DS interface allows users to select 3D CAD models and place them in the scene. Users can see the alignment of the object against the panorama. Camera controls allows the user to rotate the camera and to see different perspective views of the scene. The user can also toggle the overlaid panorama on (left) or off (right), to get a better view of the underlying 3D architecture and all objects placed into the scene thus far using the interface.

B Annotation interface details

Our annotation interface consists of a web interface developed using `three.js` that allows users to insert 3D assets into the scene while *visually overlaid* on the panorama. To achieve this, our interface assumes that there is a set of panoramas with corresponding camera poses, and an 3D architecture on which the objects can be placed. We implement two viewing modes, panorama mode and architecture mode, to let users switch between overlaid panorama and underlying 3D scene.

Data Assets. We construct a parametric 3D architecture for 20 Matterport3D scenes. We take the region annotations that specify wall segments to create the initial 3D architecture. We then project annotations for the labels relating to windows and doors to get an initial estimate for the placement of doors and windows on the architecture. Next, we create a textured architecture by rendering the reconstructed scene onto the estimated surfaces of each architecture element plane (wall, floor, ceiling). Using a 3D interface that shows the architecture, we manually refining the wall boundaries and the placement of doors and windows on the walls to correct any prominent errors. The projection of door and window annotations onto the walls is often noisy due to open doors, inaccurate windows, and noises in the annotation. We obtain each RGB panorama by stitching 6 skybox images from the same camera viewpoint. During the data preprocessing stage, we also parse panoramas into semantic object instance masks to provide reference objects during annotation. We get these instance masks by

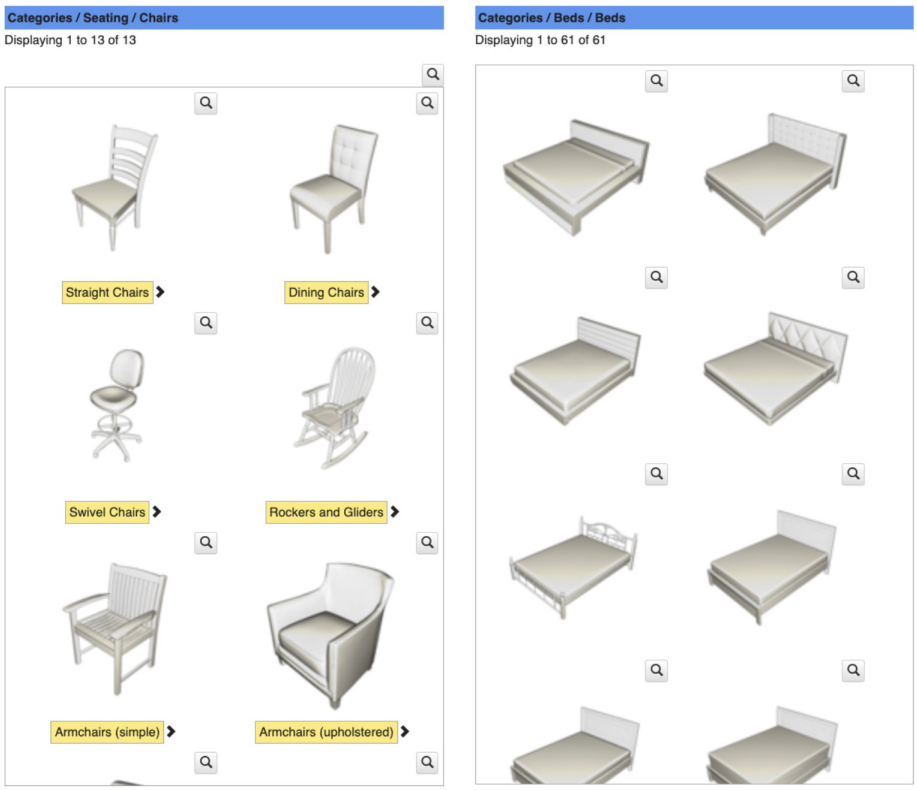


Fig. 11: The user sees a list of candidate CAD models that is filtered depending on the semantic object category of the mask that was clicked in the panorama view. The list is hierarchical, allowing the user to refine the category into finer-grained categories such as the examples of chair types on the left side, and then select an appropriate instance of a chair within the finer category.

rendering segmentations from Matterport3D's annotated object instance house meshes.

Annotation process. We describe a typical annotation workflow starting with an empty scene (see Figure 10). A user freely pans the camera to explore the whole scene while the overlay is kept in sync. After clicking an object to be annotated in the panorama, a list of candidate 3D shapes of the same category is shown in a side panel (see Figure 11). The user is instructed to identify the best matching 3D shape (see Figure 14). The inserted 3D shape is automatically placed at the location in the scene where the user initially clicked. The user can further manipulate the position, scale, and orientation of objects so that the object is aligned to the image (see Figure 12). The placement is attached to a specific surface already in the scene, thus creating a scene support hierarchy by construction.

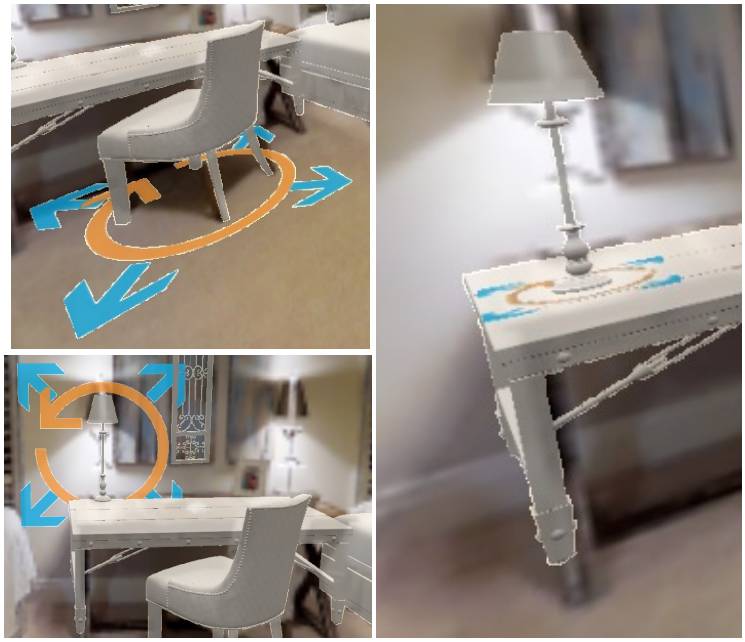
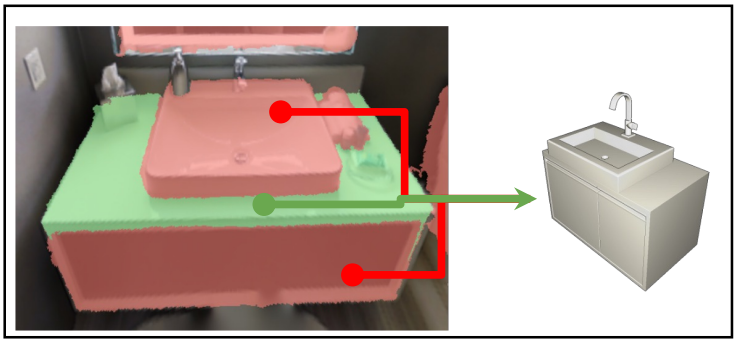


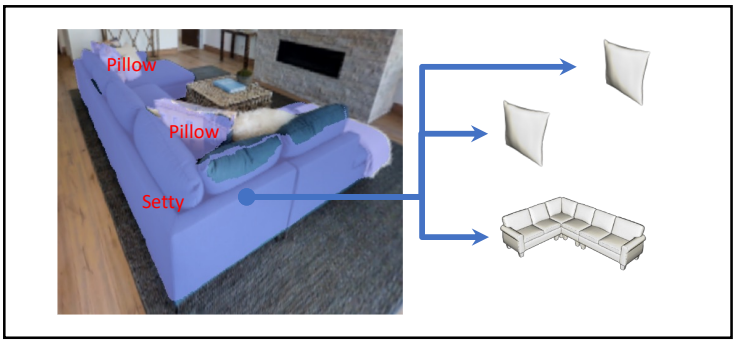
Fig. 12: The interface allows annotators to attach objects to either architecture or other objects. The manipulator is oriented on the attached surface and allows for rotating the object whereas the arrows allow for scaling.

We recruited annotators and instructed them to follow these guidelines: 1) *Completeness*: each mask should be annotated with a 3D model of an object. Some masks may be divided into parts for different objects and some masks may be merged into one (see detailed discussion of "mask-to-object assignment"). If an important object does not have a mask, it can still be added (see discussion of "custom masks"). 2) *Object match*: the categories, shapes and sizes of the placed objects match those observed (see "object selection" criteria) 3) *Spatial accuracy*: object placements and orientations should be as close to those observed in the panorama (see "object selection" criteria). There should be no collisions or floating objects.

Mask-to-Object Assignment. In some cases, it is overly restrictive to assume that there is a one-to-one correspondence between masks and objects. For example, an object may need to be assigned to multiple masks because the two masks correspond to parts of the same object, separated by occlusion. In other cases, we have masks that include multiple objects (see Figure 13). Our system supports these cases such that a user can place multiple models for the same mask by re-selecting a mask that already has a model assigned and inserting an additional model. For cases where a model is shared among multiple masks, the user first inserts the model having selected one of the masks. Then, the user can assign other relevant masks to the already inserted model. Handling of these



(a) Single object mapped to multiple masks



(b) Single mask containing multiple objects

Fig. 13: Mask-to-object assignment. Our annotation strategy allows for objects that need to be assigned to multiple instance masks (e.g., sink at top), and multiple masks needing to be assigned to the same object (e.g., couch and pillows at bottom).

cases enables us to correctly annotate densely cluttered arrangements such as kitchen cabinetry, sink units, and pillows on couches.

Object Selection. We decompose the requirement on semantically-matching objects into 4 sub-aspects (see Figure 14): category, shape, structural, and functional similarity. For example, a category mismatch constitutes a 'chair' being annotated with a 'table', a shape mismatch constitutes 'high-back armchair' being annotated with a 'dining-chair' model, a structural mismatch constitutes a 'single-seater chair' being annotated with a 'double-seater chair' and a functional mismatch constitutes a 'an armchair with no wheels' being annotated with a 'swivel chair with wheels and no arms' (Figure 14). We exclude door and window objects for annotations since they are represented as holes on the walls of the architecture and their placement can be largely automated.

Object alignment and support. Additionally, the objects can have two types of support structure: i) object-to-object support; and ii) object-to-architecture support. Object-to-object support ensures that two objects are supported by each other properly. For example, a microwave placed on a counter is by construction constrained to be on the counter top, and not to float in midair. Similarly,

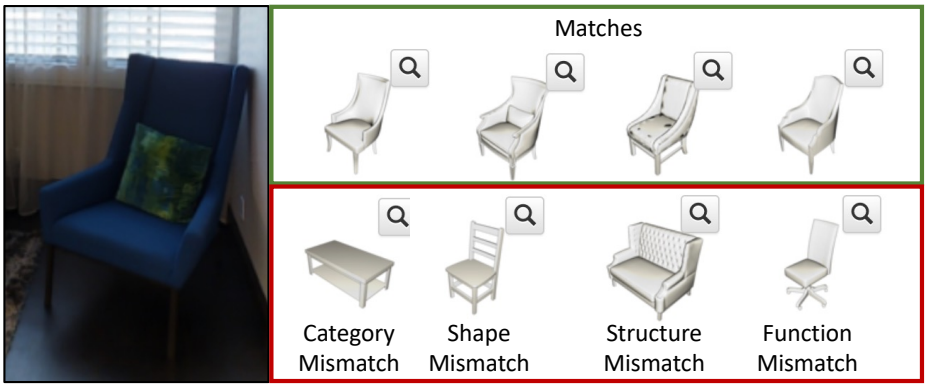


Fig. 14: Object selection in R3DS. Annotators are instructed to select objects to insert in to a scene based on how well they match with the object observed in the panoramic image. Top: shows good object matches that an annotator would select following our instructions. Bottom: shows different types of mismatching objects that annotators are instructed to avoid.

Fig. 15: Object alignment in R3DS. (Left) Objects are closely aligned to the image. (Middle) Objects are properly supported by other objects. (Right) Objects are supported by appropriate architecture elements. In each column, the top is a cropped view from the panorama, the middle highlighted in green is a correctly placed object, while the bottom is an example of an error that annotators avoid using our annotation interface.

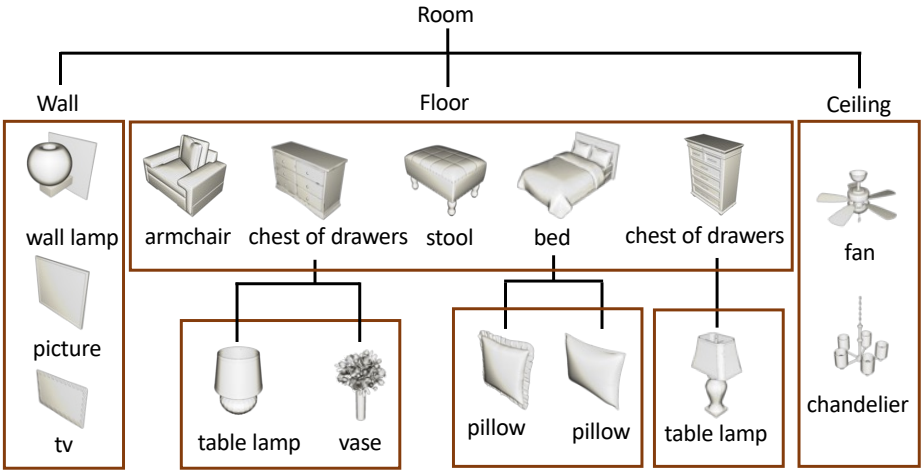


Fig. 16: Scene object support hierarchy. Objects in a scene are supported by either architecture elements or other objects.

in the object-to-architecture support case, an object placed on an architectural element (floor, wall or ceiling) is ensured to be supported by the planar surface of that element. This type of annotation also helps to disambiguate some otherwise physically implausible scenarios. For example, a chest of drawers is typically supported by the floor, and not by the adjacent wall (Figure 15). In Figure 7, we show a concrete example of how different objects are attached to architectural elements and supported by other objects.

Custom Masks. We further allow annotators to insert objects for which there are no existing instance masks to ensure the scenes are densely populated and objects are properly supported. In some cases, the user may decide to leave a mask unannotated. This could be because the mask is invalid or there are no viable models for the object. In this case, the user can mark the object as 'unannotated' and leave comments explaining the reason.