CAD-Estate: Large-scale CAD Model Annotation in RGB Videos

Kevis-Kokitsi Maninis
Google Research

Stefan Popov
Google Research

Matthias Nießner
TUM

Vittorio Ferrari
Google Research

Figure 1. Example annotations of the CAD-Estate dataset. We create globally consistent 3D representations from RGB videos, by retrieving CAD models and estimating their poses. We visualize them by overlaying them on the video frames (left, mid), and from a top-view (right).

Abstract

We propose a method for annotating videos of complex multi-object scenes with a globally-consistent 3D representation of the objects. We annotate each object with a CAD model from a database, and place it in the 3D coordinate frame of the scene with a 9-DoF pose transformation. Our method is semi-automatic and works on commonly-available RGB videos, without requiring a depth sensor. Many steps are performed automatically, and the tasks performed by humans are simple, well-specified, and require only limited reasoning in 3D. This makes them feasible for crowd-sourcing and has allowed us to construct a large-scale dataset by annotating real-estate videos from YouTube. Our dataset CAD-Estate offers 101k instances of 12k unique CAD models placed in the 3D representations of 20k videos. In comparison to Scan2CAD, the largest existing dataset with CAD model annotations on real scenes, CAD-Estate has 7× more instances and 4× more unique CAD models. We showcase the benefits of pre-training a Mask2CAD model on CAD-Estate for the task of automatic 3D object reconstruction and pose estimation, demonstrating that it leads to performance improvements on the popular Scan2CAD benchmark. The dataset is available at https://github.com/google-research/cad-estate .

1. Introduction

Semantic 3D scene understanding from images and videos is a major topic in 3D scene understanding, crucial for many computer vision applications, ranging from robotics to AR/VR scenarios. The final goal is to detect all objects in the scene, recognize their class, reconstruct their 3D shape, as well as their pose within the overall scene coordinate frame. With the advances of scalable deep learning techniques, the field has progressed from reconstructing the 3D shape of one object in a simple image with trivial background [32, 50, 40, 33, 8, 17], to limited reasoning about object arrangements in simple multi-object scenes [36, 18, 23], and finally to unrestricted multi-object 3D reconstruction in complex real-world scenes [49, 30, 42, 12]. This evolution has been dependent on the availability of ever larger and more diverse data sets for training and evaluation [3, 10, 6, 44, 47, 7, 27, 15, 16]

Existing datasets for Semantic 3D scene understanding

Dataset	Type of data	Sensor type	Multi-object	Annotation type	Requires 3D reasoning	Total # objects
SUN RGB-D [44]	image	RGB-D	✓	3D box	yes	64.6k
PASCAL 3D+ [51]	image	RGB	∼	CAD	yes	36k
IKEA [28]	image	RGB	✗	CAD	limited	759
Pix3D [47]	image	RGB	✗	CAD	limited	10k
ABO [9]	image	RGB	✗	CAD	no	6.3k
Objectron [1]	video	RGB	✗	3D box	yes	17k
CO3D [37]	video	RGB	✗	object point cloud	no	19k
Replica [46]	video	RGB-D++	✓	labels on scene point cloud	yes	∼3k
Matterport3D [6]	video	RGB-D	✓	labels on scene point cloud	yes	50.8k
Scan2CAD [3]	video	RGB-D	✓	CAD	yes	14.2k
CAD-Estate (Ours)	video	RGB	✓	CAD	limited	101k

Table 1. Real 3D scene understanding datasets and their attributes. 'Multi-object': whether there is more than one annotated object in the same image/video. 'Annotation type': what constitute the annotation for an object. 'Requires 3D reasoning': whether annotators need to reason in 3D. 'Total': number object instances with annotations.

fall broadly in two categories: synthetic and acquired from real images/videos. The former [27, 15, 16, 45, 41] feature artificial 3D scenes that are manually designed by human artists, and then rendered into synthetic images. While these datasets are relatively large, their images/videos expose a domain gap to real imagery [52, 39, 48, 19, 38, 35].

Acquired datasets [10, 3, 44, 46, 6] annotate 3D objects on real images and videos (Table 2). Such datasets have been limited in size and diversity so far, partly due to limitations in their annotation process. They rely on specialized equipment to capture depth images (RGB-D) in order to get a high-quality 3D point cloud reconstruction of the scene. Humans then annotate objects on this 3D point cloud. However, it is very expensive and cumbersome to go and physically acquire RGB-D videos in the real world, which limits the number of scenes captured, as well as their variety (e.g. RGB-D sensors struggle outdoors due to sunlight, fail on glossy surfaces, and they have limited depth range). Moreover, annotating on 3D point clouds requires expert annotators able to reason in 3D.

In this paper, we present the CAD-Estate dataset, which annotates real videos of complex scenes from Real Estate 10k [53] with globally-consistent 3D representations of the objects within them. For each object we find a similar CAD model from a database, and place it in the 3D coordinate frame of the scene with a 9-DoF pose transformation. We designed a semi-automatic approach which works on commonly-available RGB videos, without requiring a depth sensor, thereby opening the door to annotating many videos readily available on the web. In our approach many steps are performed automatically, and the tasks performed by humans are simple, well-specified, and require only very limited reasoning in 3D. This makes them feasible for crowd sourcing, enabling to distribute work to a large pool of annotators. In turn, this has allowed us to construct a truly large-scale data set.

CAD-Estate contains 100,882 instances of 12,024 unique CAD models, covering 19,512 videos (Sec. 4). The models span 49 categories, 28 of which with more than 100 objects annotated. In comparison, the largest existing dataset with CAD model annotations on real multi-object scenes (Scan2CAD [3]) has 7× fewer objects (14,225), 4× fewer unique CAD models, 2× fewer categories with more than 100 objects (14) and 13× fewer videos (1,506).

In our experiments, we show that pre-training a modern model for automatic 3D object reconstruction and pose estimation [23] on CAD-Estate improves performance on the popular Scan2CAD benchmark [3]. Moreover, we establish baseline performance on our own test set, and provide ablation experiments to validate various choices of our annotation pipeline.

2. Related Work

Synthetic scene understanding datasets. Datasets of 3D object assets (without their poses on images) include ShapeNet [7], 3D-FUTURE [16], ABC [22] and ABO [9]. Most recently, Objaverse [13] released a large dataset of 818k 3D assets. Other synthetic datasets contain 3D objects placed in artificial 3D scenes designed by artists, usually indoor rooms [27, 45, 41], and then rendered into images.

Synthetic datasets are large scale (up to 818k objects of [13]), but require extra efforts to bridge the domain gap for applications on real imagery [52, 39, 48, 19, 38, 35].

Real 3D scene understanding datasets. Several datasets have objects annotated on individual images (Table 1, top block). Sun RGB-D [44] provides image-depth pairs from an RGB-D sensor along with objects annotated with 3D bounding-boxes (no 3D shapes). PASCAL-3D+ [51] aligns simple CAD models to images by manually specifying the object pose and the focal length of the camera. They focus on simple images with fewer than 2 instances on average. IKEA Objects [28] and Pix3D [47] annotated one object per image by aligning a 3D CAD model on it. Moreover, their scale is limited by the requirement for having CAD models exactly matching the objects in the images, which

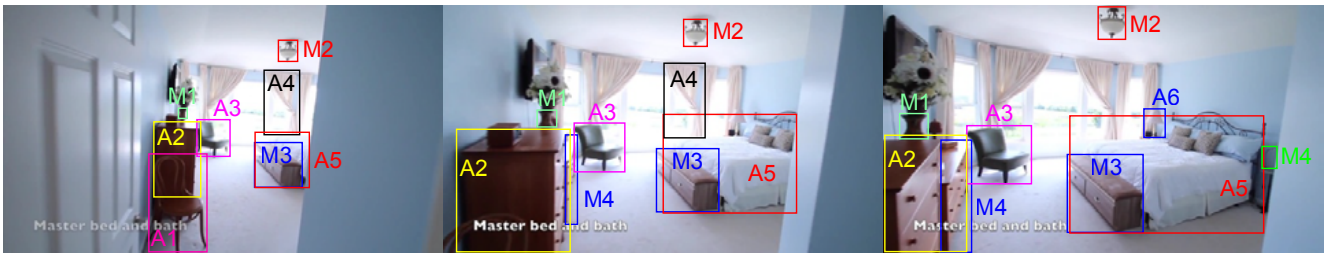

Figure 2. Automatic (A) and manual tracks (M). Automatic tracking can miss objects if they are truncated, occluded (like M4), or small (like M1). We complete such tracks manually, on the validation and test sets.

are difficult to find. More recently, ABO [9] automatically estimated 3D object poses for part of their 3D assets, on automatically retrieved images (6.3k images with one object annotated in each).

Other datasets annotate objects on videos (Table 1, bottom block). CO3D [37] and Objectron [1] have videos mostly featuring one object each, and provide either a reconstructed point cloud of the object [37] or a 3D bounding box [1]. Several works [46, 6, 10] use an RGB-D sensor to capture videos of rooms with multiple objects, then reconstruct a 3D point cloud scan of the scene by fusing the acquired depth maps. They then label this 3D scan with object class and instance labels, resulting in incomplete object shapes. Closer to our work, Scan2CAD [3] goes a step further, building on [10] by manually annotating posed CAD models on the 3D scan. These datasets heavily rely on a depth sensor, which limits their scale and applicability. In contrast, we propose an annotation method which works on RGB videos, enabling annotating videos readily available on the web. Moreover, our human annotation tasks are very simple, and require little reasoning in 3D. These two features make our approach more scalable. We construct CAD-Estate, which annotates 101k objects with clean CAD models and full 9-DoF poses on pure RGB videos. This is larger than any other dataset of real imagery, and is 7× larger than Scan2CAD, which also offers posed CAD models (on RGB-D video).

Multi-object 3D reconstruction Many works tackle multi-object 3D reconstruction from a single image [18, 36, 23, 24]. They are either trained on synthetic data [36], or on small real datasets [18, 23, 24]. Similarly, recent learning-based approaches reconstruct a scene from a video [30, 25, 26, 49, 42], and use Scan2CAD as their main evaluation benchmark. Our CAD-Estate dataset can benefit all of these works as it offers new, large-scale, diverse, real video data with annotated complex spatial arrangements of 3D objects into scenes. In Section 5 we show that pretraining on CAD-Estate boosts the results of [23] on the original dataset it has been trained for [3].

3. Dataset construction

Given a video of a static scene, our goal is to create a globally-consistent 3D representation that contains all its objects. To achieve this, we propose a semi-automatic system that relies on a large database of CAD models. For each object in the scene, we find a similar-looking CAD model from the database and place it in the 3D coordinate frame of the scene by estimating its 9-DoF pose (i.e. 3D translation, 3D rotation, and 3D scale, allowing for independent scaling along each axis).

We design the system so that many steps are performed automatically. We leave only a few, simple and well-specified tasks for human annotators. These are all decomposed over individual objects, removing the complexities of considering the whole scene, and involve only very limited reasoning in 3D. These characteristics make the tasks feasible for crowd sourcing, enabling to distribute work to a large pool of annotators, as opposed to few in-house experts [10, 3]. This enables constructing a truly large dataset.

We annotated videos of RealEstate10K [53], which show multiple rooms of real estate properties. The videos are split into shots, and camera poses have been extracted using an SfM pipeline [43]. We use ShapeNet [7] as our CAD model database, which contains 51k objects over 55 classes.

System overview. Our system receives an RGB video as input, with camera parameters for each frame (typically derived using SfM [43]). The output is the class, 3D pose (rotation, translation, scale), and 3D shape of each object in the video (represented as a CAD model from a database).

The system amounts to a sequence of 5 stages:
(1) We start by detecting objects in the video and tracking them over time, either automatically or with the help of humans (Sec. 3.1). Each track corresponds to one physical object in the scene and forms the unit of annotation. All further stages operate on one track at a time with the goal of reconstructing its pose, shape, and class.
(2) For each track, we automatically select a few similar-looking CAD model candidates from the database, and then ask humans to choose the best match (Sec. 3.2).
(3) We ask humans to annotate $3D \leftrightarrow 2D$ point correspon-

dences between the chosen CAD model and the object in the video, on a few key-frames (Sec. 3.3).
(4) We use the annotated correspondences together with the camera parameters of the key-frames to automatically estimate the 9-DOF pose of the object (Sec. 3.4).
(5) Finally, we ask humans to verify the estimated pose for quality control (Sec. 3.5).

3.1. 2D Object detection and tracking

In this first stage we detect objects in the video and track them over time. Each track then corresponds to one physical object and forms the unit of annotation for all subsequent stages. We apply somewhat different procedures for the training and val/test sets of our dataset, in order to strike a good trade-off between automation (hence reducing human effort) and completeness of annotation (we want to capture all objects in the val/test set).

Train set. We detect objects in each frame automatically using a SpineNet-based model [14].

We also extract an appearance descriptor for each detection box, by applying a Graph-Rise-based [20] model.

Next, we associate detections over time, as common in tracking-by-detection approaches [4, 11, 2]. We compute various similarity scores between two detections in different video frames, including the similarity between their appearance descriptors, the difference in their class labels, and the spatial continuity of the box positions in adjacent frames. Then we cluster all detections across all frames into tracks based on these similarity scores using the Clique Partitioning approach of [31].

Val/Test sets. Automatic detection and tracking models can sometimes miss objects as they do not work perfectly. Since for the validation and the test sets we strive for a high degree of completeness, we annotate missing object tracks manually (in addition to the automatic ones). For this we developed an efficient custom interface that allows annotators to draw a whole object track in time, i.e. drawing a bounding-box [34] on each key-frame where a particular physical object appears. For efficiency, we automatically focus work on 6 frames regularly-spaced in time. The annotators see all current tracks already found by the automatic approach, and only draw missing ones.

Note how we apply this manual annotation procedure only to a rather small subset of the data (val/test sets have fewer videos than train, Table 2).

3.2. Selecting a CAD model

The second stage is to select a suitable CAD model for a tracked object. We first select 10 candidates automatically from the database. We then ask a human to chose the one that looks the closest to the object in the video. This removes the need for annotators to search through the large database.

Finding candidates automatically. We find candidate CAD models for an object track by considering both appearance similarity and class label similarity cues. During pre-processing, we render the CAD models in the database from 10 random viewpoints and compute an appearance descriptor for each view (the same as in Sec. 3.1). We then compute the appearance similarity between an object box in a frame of the object track and a CAD model view as the cosine similarity of their descriptors.

For the class label similarity we need to take special care, as the label spaces of the CAD model database and the object detector are different and feature multi-way relationships (e.g. the CAD "cabinet" matches the detector's "filing cabinet", "wardrobe", and "chest of drawers"). Hence, we embed each class label name into a common semantic space using the Universal Sentence Encoder [5], and compute the cosine similarity between any two class labels in this space. This is a general solution that can work with any label space. We combine the appearance and class similarity scores with a simple product.

To compute the overall similarity between an object track and a CAD model, we aggregate the combined appearance-class similarity over all pairs of frames and CAD model views. We use this overall similarity score to rank CAD models and select the top 10 as candidates for an object track. In practice the class similarity act as a soft filter for the appearance similarity, so the best CAD models are the most similar-looking ones to the object in the track, among those that have a similar class label.

Selecting the best candidate with a human. We ask annotators to choose the best matching candidate. We show them the detected object on a set of evenly spaced key-frames, next to the rendered CAD model candidates. Annotators can navigate between key-frames, to see the object from multiple views. Annotators can declare that none of the candidates are similar enough to the tracked object (hence that track is not passed on to the later stages).

3.3. 3D $\leftrightarrow$ 2D point correspondences

We now ask humans to annotate point correspondences between the 3D surface of the CAD model and the video frames of the tracked object (Fig. 3). As for the CAD candidate selection case, the interface enables annotators to navigate between key-frames. We show the selected CAD model next to the key-frames. For each key-frame, we ask annotators to annotate 4-6 point correspondences between the CAD model and the frame. To make the task easier, they can rotate and flip the CAD model in 3D, in order to

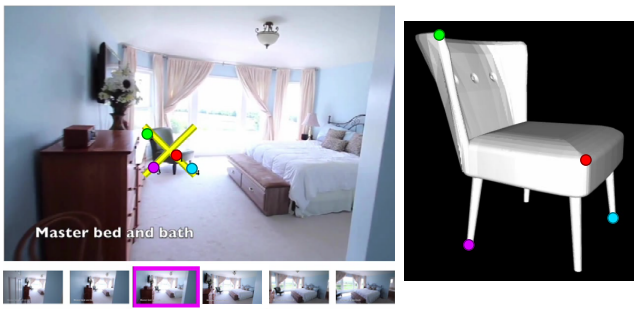

Figure 3. We ask humans to annotate point correspondences between the 3D surface of the CAD model and the video frames of the tracked object.

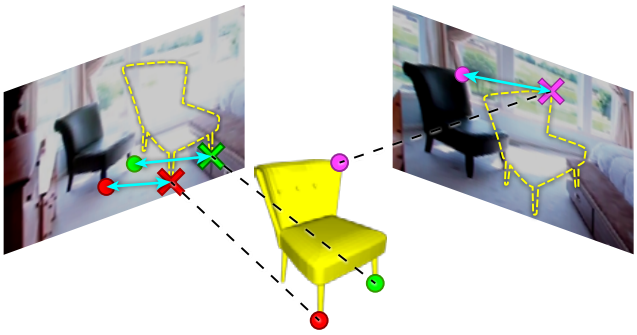

Figure 4. Point re-projection objective. We project the annotated points on the CAD model to the video frames given a candidate pose (T, R, S), and penalize the displacement with respect to their 2D correspondences (arrows in cyan). We minimize this objective over poses (along with two others).

roughly match the orientation of the object in the frame. We will use these correspondences to recover the 9-DOF object pose in the next stage.

Our approach consists of steps that are easy to understand and easy to master. Annotators control rotation with Orbit Controls [29], which translates 2D mouse movements to view-local object rotation in 3D in an intuitive way. Afterwards, clicking on CAD-to-image point correspondences is very easy and is similar to other familiar 2D annotation tasks. Most importantly, this approach is object-centric and requires no reasoning in 3D in the global coordinate frame of the scene. Instead, this harder task is done automatically in the pose estimation stage of our system. Finally, annotating point correspondences is decoupled between frames: the annotator is free to pick different points in every frame. This makes it easy even for objects with complex shapes.

3.4. Object 3D pose estimation

We use the $3D \leftrightarrow 2D$ point correspondences to automatically estimate a global 9-DOF pose for the object. We apply a non-linear optimization method, which integrates evidence from all views in a track, and consists of multiple objectives.

We express the object pose as a 9-DOF transformation that brings the CAD model from its canonical pose to the world coordinate frame of the scene.

The transformation has 3 components: 3D translation T, 3D rotation R, and anisotropic 3D scale S (i.e. we allow independent scaling along each axis). The goal is to recover this unknown transformation (T, R, S). We setup below several objectives, which are functions of (T, R, S), and combine them into an overall objective. Finally we minimize that overall objective over (T, R, S).

Point re-projection objective. We know the extrinsic and intrinsic camera parameters at each video frame. Given a potential (T, R, S) we can use it along with the camera parameters to project the 3D points on the surface of the CAD model to the video frame. Therefore, we setup a point re-projection objective $L_{repr}(T, R, S)$ which measures the L1 distance between the projected 3D points and their corresponding 2D points in each frame (and sum over all frames, Figure 4). The correspondences are given by the $3D \leftrightarrow 2D$ annotations from Sec. 3.3, and we also take into account whether the annotator flipped the CAD model.

Up-axis objective. Most objects in our videos are usually placed vertically in an upright position. We reflect this by imposing an $L1$ objective that penalizes 3D rotations that change the "up"-axis of the object with respect to the world. We do this directly on the target rotation matrix R by applying the additional objective $L_{up}(R)$. This objective is applied to object classes that are usually found in upright position (e.g. chairs, tables, cabinets, etc.), whereas other classes such as pillows are excluded. For this objective to be applied, we need to know the up-axis for the objects in our CAD database, and in the world coordinate frame (which we do for ShapeNet and RealEstate10K).

Front-of-camera objective. We encourage object pose transformations that place all annotated 3D points in front of their respective cameras (rather than behind), by penalizing 3D points that have a negative depth in the coordinate frame of that camera.

Special scale parameterization for co-planar 3D points. Sometimes, all 3D points chosen in Sec. 3.3 by the annotator on the CAD model are co-planar. This typically happens when the video shows only a planar part of the object, e.g. a table seen only from the top, or a cupboard seen only frontally. Co-planar 3D points prevent resolving all three dimensions of the target scaling transformation S. We detect such cases automatically during annotation. We then resolve them during pose estimation by constraining

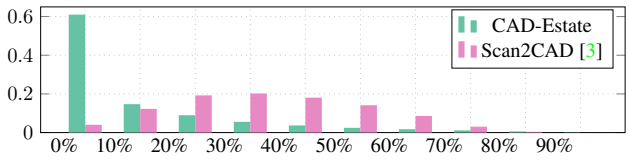

Figure 5. Truncation histogram for CAD-Estate and Scan2CAD. We measure the degree of object truncation as the fraction of a CAD model surface that projects outside of the video frame.

the scaling factor perpendicular to the annotated plane to be the average of the other two scale factors. This reduces the DOF of the scaling transformation S down to 2.

Special rotation/scale parameterization for symmetric objects. In many cases the retrieved CAD models are symmetric, which typically leads to inconsistent point correspondence annotations across frames (e.g. an annotator picking a particular 3D point on a rotation-symmetric lamp corresponds to a point in the video in a frame, but then picking a different 3D point in a different frame, as these are equivalent up to symmetry). We handle these cases by optimizing for a rotation w.r.t any of the symmetries of the object in the reprojection objective. We consider the same symmetries as in Scan2CAD, i.e. 2-way (e.g. a rectangular table), 4-way (e.g. a square table), and 36-way (e.g. a round table). We detect symmetries automatically directly on each CAD model. For fully symmetric objects (36-way symmetric), we further constrain the two scaling factors around the up-axis to be identical.

Optimization We combine the above objectives in an overall one:

$$L_{pose}(T,R,S) = L_{repr}(T,R,S) + \alpha \cdot L_{up}(R) + \beta \cdot L_{front}(T,R,S) \quad (1)$$

We minimize this objective over (T,R,S) with Adam [21]. α and β are hyperparameters set empirically.

3.5. Pose verification by humans

In this last stage, we verify whether the pose computed in the previous stage matches the image contents in the video. This is necessary as pose estimation can fail for several reasons, including limited/degenerate camera motion, occlusion, and truncated objects.

We render the CAD model as overlay on top of the video frames in a track, using the camera parameters and the estimated object pose (T,R,S). We then ask human annotators to judge whether the rendered CAD aligns well with the object in the video. If it aligns well in all key-frames, we mark the pose as correct.

	CAD-Estate			
Dataset	Train	Val/Test	Total	Scan2CAD
#Scenes	16713	2799	19512	1506
#Posed objects	77832	23050	100882	14225
#Classes	49	40	49	35
#Classes >1000	11	5	13	4
#Classes >100	24	22	28	14
#Objects per scene	4.7	8.2	5.2	9.4
#CAD models	10358	6192	12024	3049
#Frames per scene	138	139	138	1604
Source	RGB	RGB	RGB	RGB-D

Table 2. General statistics of CAD-Estate and Scan2CAD.

4. Dataset analysis

General statistics. Table 2 compares general statistics of CAD-Estate to the closest existing video dataset Scan2CAD [3]. We further split the stats of our dataset into training set and val/test test sets.

CAD-Estate is an order of magnitude larger than Scan2CAD (20k vs. 1.5k scenes, and 101k vs. 14.2k posed objects). The annotated objects cover more classes (49 vs. 35 in Scan2CAD). Figure 6 shows the distribution of annotated objects over classes. Despite the long tail, there are many more classes that have a large number of objects (13 classes with $>$ 1000 objects vs 4 in Scan2CAD, and 28 classes with >100 objects vs 14).

CAD-Estate also offers greater diversity of object 3D shapes. It is annotated with 12k CAD models vs 3k for Scan2CAD (noting that in both datasets the CAD shapes are a close match rather than exactly matching the shape of the object in the image).

Camera framing. There is a qualitative difference between the video captures of Scan2CAD (from ScanNet [10]) and CAD-Estate (from RealEstate10K [53]). The videos of [10, 3] were captured with an RGB-D sensor, taking close-up views which facilitates acquiring good quality depth maps. Instead, the videos of CAD-Estate are captures of real estate properties with more distant views that depict a larger part of each room, as the goal was to showcase the space for selling it. The video shots are also shorter (138 frames per video in CAD-Estate vs. 1.6k in Scan2CAD).

As a consequence of the more distant views, several key statistics are different in CAD-Estate, compared to Scan2CAD: (1) More objects are visible in one video frame at the same time: on average, 7.9 in CAD-Estate vs 3.3 in Scan2CAD. (2) More objects are further way from the camera and thus appear smaller on the images: on average, the bounding-box of a CAD-Estate object covers 7.5% of the image area vs. 16.5% in Scan2CAD. (3) The dynamic range of the Z position of objects is larger: in CAD-Estate the farthest object is $4.5\times$ farther from the camera than the nearest one, vs. $2.3\times$ in Scan2CAD. (4) Object truncation is much

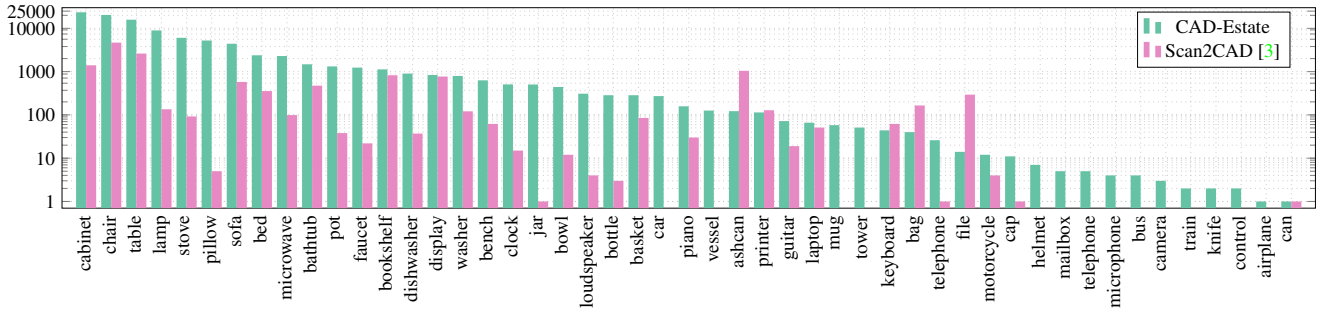

Figure 6. Class histogram of CAD-Estate vs. Scan2CAD [3]. We annotate more classes with many more objects. Note the logarithmic scale of the vertical axis.

higher in ScanNet compared to CAD-Estate, where most objects are completely visible (Figure 5). This is also a consequence of the capture process, as ScanNet needs close-up captures due to the range of the depth sensor.

The camera framing statistics above highlight how CAD-Estate poses a different challenge than Scan2CAD for automatic scene understanding methods, as they need to handle more complex views with more objects visible at the same time, many smaller objects, a higher variability of their distance to the camera, but also less truncated by the image frame.

5. Experiments

We first perform several experiments by training a learning-based method for CAD model alignment [23] on CAD-Estate (Sec. 5.1), demonstrating that it leads to performance improvements on the Scan2CAD test set, and establishing that our test set offers a harder challenge. Then in Sec. 5.2 we provide ablation experiments for the different components of our annotation pipeline, showing their relative merit and demonstrating that they are all necessary to achieve high quality.

5.1. Training Mask2CAD on CAD-Estate

In this section, we showcase how CAD-Estate can be used to train Mask2CAD [23], a deep learning method for single-image 3D object reconstruction and pose estimation. We start by studying the benefits of having a large training set by pre-training [23] on CAD-Estate and then fine-tuning and evaluating on Scan2CAD [3] (where Mask2CAD was originally benchmarked on). Then we establish baseline results for Mask2CAD trained and tested on CAD-Estate.

From CAD-Estate to Scan2CAD. Mask2CAD has been extensively evaluated [23] by training and testing on the Scan2CAD dataset [3], whose training set consists of 9.5k objects over 19k frames on 1194 scenes. We run the same experiment, but first pre-train Mask2CAD on a much larger training set of 45k objects over 150k frames sampled from 11k scenes of CAD-Estate's trainval. Then we fine-tune on the train set of Scan2CAD, and evaluate on the test set with the popular metrics AP_{mesh}, AP^{50}_{mesh}, and AP^{75}_{mesh} [23, 18].

Table 3 presents the results on all 3 metrics above, and additionally per-class AP_{mesh}. As the results show, pre-training on our large dataset improve the performance of Mask2CAD significantly, for almost all classes. We observe that the improvement is greater for classes for which CAD-Estate has many objects (cabinet, table, bed).

Train and test on CAD-Estate. We now establish baseline results for Mask2CAD on CAD-Estate (training on our trainval set, and evaluating on the test set). We use the same classes as Scan2CAD for this experiments, and the same evaluation metrics, enabling approximate comparisons across datasets.

The results in Table 4 show that Mask2CAD achieves considerably lower performance on CAD-Estate than on Scan2CAD. Especially on the strict IoU threshold AP^{75}_{mesh}, the performance is much lower (5.7 vs. 2.4). This indicates that our test set might offer a harder challenge. CAD-Estate provides more complex scenes that are difficult to reconstruct, and objects are in general further away from the camera, which makes pose estimation harder.

5.2. Optimization objectives for 3D pose estimation

We study the influence of the object pose optimization objectives of Section 3.4 on pose estimation quality. We evaluate by asking annotators to verify the poses produces by different versions of the pose estimator (as in Sec. 3.5, but on a subset of the data). A higher percentage of positively verified object poses indicates a better pose estimator.

Starting from 52.2%, the percentage of positively verified poses improves steadily as we add the special parameterization of the re-projection objective for handling coplanar 3D points (57.6%), the one for handling symmetric objects (62.9%), and the up-axis objective (74.9%). This

Figure 7. Annotated scenes from CAD-Estate, overlaid on video frames (left, mid), and shown from a top view (right).

Pretraining on CAD-Estate	AP_{mesh}	AP^{50}_{mesh}	AP^{75}_{mesh}	bed	sofa	chair	cabinet	bin	display	table	bookshelf
no (original [23])	8.4	23.1	4.9	14.2	13	13.2	7.5	7.8	5.9	2.9	3.1
no	8.2	23.1	4.9	14.0	12.7	12.9	7.1	7.6	6	2.5	3.0
yes	**9.4**	**25.0**	**5.7**	15.1	13.2	14.5	9.0	7.4	7.8	4.0	4.5

Table 3. Performance of Mask2CAD on Scan2CAD's test set. Top row: results reported by [23] by training on Scan2CAD train set; Second row: our reproduction of that experiment, which reaches nearly identical performance. Bottom row: pre-training on CAD-Estate train set, then fine-tuning on Scan2CAD train set. Performance improves thanks to our additional training data.

	AP_{mesh}	AP^{50}_{mesh}	AP^{75}_{mesh}	bed	sofa	chair	cabinet	bin	display	table	bookshelf
Maks2CAD on CAD-Estate	7.5	21.2	2.4	13.4	10.2	10.7	10.3	2.0	4.1	5.2	4.2

Table 4. Mask2CAD results on CAD-Estate.

demonstrates that all of them contribute to the quality of our dataset, as they enable to estimate a correct pose for a greater number of objects. The largest contribution is made by the up-axis objective, as it affects many objects. Instead, 27.8% of all objects in our dataset are symmetric, and only 15.5% received co-planar 3D point annotations.

6. Conclusions

We introduced a new way to annotate 9-DoF pose of CAD models on monocular RGB videos. As a result of our method, we obtained the CAD-Estate dataset, which features 101k instances of 12k unique CAD models placed in the 3D representations of 20k videos. This dataset is an order of magnitude larger than existing CAD annotation efforts facilitated by our new annotation method. We have shown experimentally that the quantity and diversity of such data significantly benefits the modern CAD alignment technique Mask2CAD, leading to improved performance on Scan2CAD. However, we believe that this is only a first step, and CAD-Estate is an important stepping stone towards leveraging CAD priors for 3D scene reconstruction and understanding in the context of a wide range of downstream tasks.

Acknowledgements: We thank Prabhanshu Tiwari, Sweety Chaudhary, Abha Dwivedi, Ashlesha Shantikumar, Umesh Vashisht, Mohd Adil for coordinating the annotation process, and Weicheng Kuo who helped us with running Mask2CAD on our dataset.

References

[1] Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *CVPR*, pages 7822–7831, 2021. 2, 3

[2] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, pages 1–8. IEEE, 2008. 4

[3] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2CAD: Learning cad model alignment in RGB-D scans. In *CVPR*, 2019. 1, 2, 3, 6, 7

[4] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *ICCV*, pages 941–951, 2019. 4

[5] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *EMNLP*, pages 169–174, Brussels, Belgium, Nov 2018. Association for Computational Linguistics. 4

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017. 1, 2, 3

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint*, arXiv:1512:03012, 2015. 1, 2, 3

[8] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *ECCV*, 2016. 1

[9] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. ABO: Dataset and benchmarks for real-world 3D object understanding. In *CVPR*, pages 21126–21136, 2022. 2, 3

[10] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 1, 2, 3, 6

[11] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *CVPR*, pages 4660–4669, 2019. 4

[12] Anton Konushin Danila Rukhovich, Anna Vorontsova. ImVoxelNet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In *WACV*, 2022. 1

[13] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv:2212.08051*, 2022. 2

[14] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. In *CVPR*, pages 11592–11601, 2020. 4

[15] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *ICCV*, pages 10933–10942, 2021. 1, 2

[16] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *IJCV*, pages 1–25, 2021. 1, 2

[17] R. Girdhar, D.F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 1

[18] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *ICCV*, 2019. 1, 3, 7

[19] Haoshuo Huang, Qixing Huang, and Philipp Krahenbuhl. Domain transfer through deep activation matching. In *ECCV*, September 2018. 2

[20] Da-Cheng Juan, Chun-Ta Lu, Zhen Li, Futang Peng, Aleksei Timofeev, Yi-Ting Chen, Yaxi Gao, Tom Duerig, Andrew Tomkins, and Sujith Ravi. Graph-rise: Graph-regularized image semantic embedding. *CoRR*, abs/1902.10814, 2019. 4

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 6

[22] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *CVPR*, pages 9601–9611, 2019. 2

[23] Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. Mask2CAD: 3D shape prediction by learning to segment and retrieve. In *ECCV*, 2020. 1, 2, 3, 7, 9

[24] Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. Patch2cad: patchwise embedding learning for in-the-wild shape retrieval from a single image. In *ICCV*, pages 12589–12599, 2021. 3

[25] Kejie Li, Daniel DeTone, Yu Fan Steven Chen, Minh Vo, Ian Reid, Hamid Rezatofighi, Chris Sweeney, Julian Straub, and Richard Newcombe. Odam: Object detection, association, and mapping using posed rgb video. In *ICCV*, 2021. 3

[26] Kejie Li, Hamid Rezatofighi, and Ian Reid. MOLTR: Multiple object localization, tracking and reconstruction from monocular rgb videos. *RA-L*, 6(2):3341–3348, 2021. 3

[27] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. *arXiv:1809.00716*, 2018. 1, 2

[28] Joseph J Lim, Hamed Pirsiavash, and Antonio Torralba. Parsing ikea objects: Fine pose estimation. In *ICCV*, pages 2992–2999, 2013. 2

[29] Mark Livingston, Arthur Gregory, and William Culbertson. Camera control in three dimensions with a two-dimensional input device. *Journal of Graphics Tools*, 5, 01 2000. 5

[30] Kevis-Kokitsi Maninis, Stefan Popov, Matthias Nießner, and Vittorio Ferrari. Vid2CAD: CAD model alignment using multi-view constraints from videos. *IEEE Trans. on PAMI*, 45(1):1320–1327, 2023. 1, 3

[31] Manuel Jesús Marin-Jimenez, Andrew Zisserman, Marcin Eichner, and Vittorio Ferrari. Detecting people looking at each other in videos. *IJCV*, 106(3):282–296, 2014. 4

[32] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 1

[33] Chengjie Niu, Jun Li, and Kai Xu. Im2Struct: Recovering 3D shape structure from a single RGB image. In *CVPR*, 2018. 1

[34] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017. 4

[35] Xingchao Peng, Ben Usman, Kuniaki Saito, Neela Kaushik, Judy Hoffman, and Kate Saenko. Syn2real: A new benchmark forsynthetic-to-real visual domain adaptation. *CoRR*, abs/1806.09755, 2018. 2

[36] Stefan Popov, Pablo Bauszat, and Vittorio Ferrari. CoReNet: Coherent 3D scene reconstruction from a single RGB image. In *ECCV*, 2020. 1, 3

[37] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV*, pages 10901–10911, 2021. 2, 3

[38] S. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 2

[39] Stephan R. Richter, Hassan Abu Alhaija, and Vladlen Koltun. Enhancing photorealism enhancement. *IEEE Trans. on PAMI*, 45(2):1700–1715, 2023. 2

[40] Stephan R. Richter and Stefan Roth. Matryoshka networks: Predicting 3D geometry via nested shape layers. In *CVPR*, pages 1936–1944, 2018. 1

[41] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. 2

[42] Martin Runz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, et al. Frodo: From detections to 3d objects. In *CVPR*, 2020. 1, 3

[43] Johannes L Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 3

[44] S. Song, S. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015. 1, 2

[45] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, pages 1746–1754, 2017. 2

[46] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv:1906.05797*, 2019. 2, 3

[47] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B. Tenenbaum, and William T. Freeman. Pix3D: Dataset and methods for single-image 3D shape modeling. In *CVPR*, 2018. 1, 2

[48] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, V. Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *CVPR-W*, pages 1082–10828, 2018. 2

[49] Michał J Tyszkiewicz, Kevis-Kokitsi Maninis, Stefan Popov, and Vittorio Ferrari. Raytran: 3d pose estimation and shape reconstruction of multiple objects from videos with ray-traced transformers. In *ECCV*, pages 211–228. Springer, 2022. 1, 3

[50] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*, 2018. 1

[51] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *Proc. WACV*, pages 75–82. IEEE, 2014. 2

[52] Sergey Zakharov, Rares Ambrus, Vitor Campanholo Guizilini, Wadim Kehl, and Adrien Gaidon. Photo-realistic neural domain randomization. In *ECCV*, 2022. 2

[53] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4):1–12, 2018. 2, 3, 6