Abstract

EXPLAINABLE 3D RECONSTRUCTION USING DEEP GEOMETRIC PRIOR

Mattan Serry, Microsoft and Tel Aviv University, maserry@microsoft.com
Dov Danon, Tel Aviv University, dov84d@gmail.com
Hagit Schechter, Rafael Advanced Defense Systems, hagit.schechter@gmail.com
Amit H. Bermano, Tel Aviv University, amberman@tauex.tau.ac.il

Keywords. Computer vision, artificial intelligence, ethics.

Reconstructing 3D objects from a single image is a notoriously difficult task, with many different proposed approaches and settings. In this paper, we investigate a unique variant: fitting cuboids to silhouettes. In other words, we ask how strong geometric priors can benefit texture-less binary silhouettes based reconstruction. While more challenging, using silhouettes enables training on purely synthesized perfectly labeled data. For the investigation, we look at street-level images of buildings, since they hold rigorous geometric structure, and their silhouettes are easily obtained, for example through instance-level segmentation. Given a noisy, partially occluded, segmentation mask as input, we present a three-step network that first generates a cleaner version for the mask, then moves to a heat-map estimation of the cuboid corners, and finally extracts the actual, geometrically coherent, vertex positions. Even though jointly trained, each of these steps produces human-legible intermediate results instead of a latent code, which serve both in guiding the training process, but also in providing *explainability*—a pillar of modern ethical AI systems. Finally, we evaluate our approach through street level images and ablation studies

1. Introduction

Reconstructing a 3D object from a single image has been one of the core challenges of computer vision since the dawn of the field. It is useful in countless applications, spanning everything from medical diagnosis and manufacturing control to robot navigation and autonomous driving. This task, however, is ill-posed and extremely difficult in the general case, since it requires vast semantic knowledge and profound reasoning about physics, optical phenomena, and the interaction between objects and people. Hence, many different approaches have been proposed to tackle it, leveraging a variety of assumptions and scene settings in order to make the problem feasible.

In this paper, we look at the problem of reconstruction from silhouettes. Silhouettes are binary masks, indicating the area the target object is occupying in the input image. Using silhouettes offers a lot of flexibility. A system that relies on silhouettes only is more robust, in the sense that it is unaffected by lighting conditions, small changes to geometry, or unusual textures such as drawings. Even more importantly, silhouettes are simple to generate, meaning one could train such a system using only synthetic data. Curating quality datasets is one of the biggest bottlenecks of modern-day algorithms, and has become a stage with practically the same importance as designing the method itself. Hence, the ability to generate perfectly labeled training examples at no cost is significant.

Of course, using only silhouettes, on the other hand, means ignoring a lot of useful visual information, such as inner details and geometric visual cues. Hence, some restrictions must be applied in order to make the problem feasible, as previously mentioned. One of the most powerful assumptions that can be made is restricting the reconstructed geometry to a predefined model. Using a strong *geometric prior* is an efficient way to extend the information seen in an image to its occluded dimensions. Hence, we turn to the problem of primitive fitting – where simple geometries are positioned, oriented, and stretched in order to fit the visible information. Even though our method is not restricted to specific geometries or scenes, in this work we solely use cuboids to estimate buildings from a single street-level silhouette. Buildings can be very versatile in appearance, and datasets regarding their structure are hard to come by. On the other hand, it is well established that buildings can be represented by the construction of simple primitives (see Section 3), which are most commonly cuboids buildings, hence our method is the right choice for the task. Moreover, successfully fitting cuboids to buildings is a useful capability that could be applied, for example, to localize oneself accurately within an urban scene.

Our method is comprised of three jointly trained neural networks, corresponding to three algorithmic steps. The input is a silhouette of a building, typically obtained by an off-the-shelf semantic segmentation process or manual annotation. Such masks are noisy at the boundaries, and usually reflect partial occlusions, such as trees, cars, or other buildings positioned between the target and the camera. Our first step is to heal the input binary mask, by completing holes and reducing noise. This step reduces the diversity that the following network has to deal with and hence improves overall performance. Then, we generate heat maps, estimating the 2D image positions of every vertex of the cuboid. Using points of interest as a means of orientation estimation, instead of the more intuitive approach of explicitly evaluating model parameters such as dimensions and rotations, has been proven useful several times in the past in **[1][2][3]**. Lastly, explicit vertex coordinates are extracted from the heat maps, estimating the final cuboid position.

Imposing intermediate results on a network prevents it from freely finding a latent space on its own, potentially restricting its expressiveness. We chose to go through these intermediate steps, however, since they guide the training process towards the desired results. In addition, they provide explainability — giving insight into how and why a specific choice was made in test time. Since our training is purely performed on synthetic data, we propose a specifically tailored training scheme, combining training each part separately on its generated intermediate result for bootstrapping, with training all the steps together.

In the following, We show how the network is built and trained (Section 3), and its performance (Section 4). Then, we evaluate how different design choices contribute to the method's performance through ablation studies.

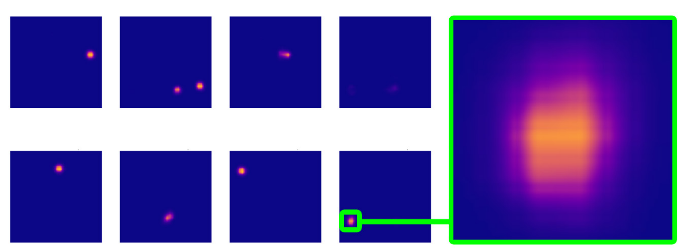

Figure 1: Example of single-box 8 semantic keypoints heat maps estimation, an output from the Estimator, the second network in our model.

2. Related work

Our work focuses on the monocular reconstruction of cuboid objects where the image is already segmented into a binary image. Reconstructing a 3D object from an image or set of images is a fundamental task in computer vision. Hence our work can be cast in several different ways, according to respective lines of work. As far as we know, it is the first work that applies outdoor cuboid—fitting without any human interaction and supervision.

2.1. Monocular reconstruction

One way to cast our problem is through the eyes of object reconstruction from a single image. The general problem is one of the most fundamental goals in computer vision, with countless proposed approaches. A seminal work in the field extracts an occupancy grid from a given image, or in other words, reconstructs a voxel representation of the target object like in [4]. The system is trained using 3D mesh data and corresponding images. More recent publications like [5][6][7] use a single image to obtain 3D information of objects, producing voxel or accurate mesh representations while relying on the camera parameters and object appearance. This line of work, while producing impressive results, is aiming at a general case of objects without any prior on their shape. For this reason, such methods require intensive training, extensive supervision, are typically sensitive to object texture, and are restricted to indoor scenes, which offer some control over scene lighting. To alleviate lighting and appearance limitations, many proposed method reconstruct objects from their silhouettes alone. This could be done using several images of the same object from different angles, or through a differentiable renderer, which offers supervision and loss estimation, like in [8]. For a more thorough review, we refer the reader to a recent state-of-the-art report covering this field in [9]. Cuboid-fitting was also studied previously in [10][11], but so far only for indoor scenes. Similar work to ours but on other outdoor objects, cars, was also studied before in [12]. In general, these approaches aim at a very wide variety of reconstructed objects, which imposes great training and supervision efforts. By using a very simplified and lightweight model of specific object classes, i.e. cuboids, we were able to devise a method that is completely supervision free and requires no labeling or images or 3D data.

2.2. Inverse procedural modeling

Another way to look at our problem is through its similarities to procedural modeling. This field, which is commonly used for buildings and city modeling, defines a grammar with which a system or a user can describe the target object. This approach offers a very lightweight description of a building or another object, which is faster and easier to optimize for. Even though easier to handle, these methods still require laborious efforts for elaborate modeling. Therefore, interactive approaches for procedural modeling have been proposed, like in [13]. To further automate this process, inverse procedural modeling methods have been proposed. These methods try to estimate the grammar parameters from a given image, similar to our task. A work proposed in [14] shares some similarities with ours, as it predicts 2D junctions/corners first from an input image, then reconstructs 2D primitives like boxes. For aerial images of buildings, the task becomes somewhat easier, as all information is visible, and the problem becomes one of finding the 2D shape of the roof, like in [15]. For street-level images, the proposed methods detect the repetitive patterns often seen in building facades to formulate the structure's grammar. One of the most dominant works in this aspect, which is similar to ours in goals, is proposed in [16]. While producing impressive results, this work, similar to others along with this approach like [17], still relies on user interaction and data labeling for supervision, or parameters such as camera intrinsics, or even 3D point locations in inference time, specifically for buildings (see [18]). Again, in contrast to these works, our approach is trained on purely synthetic data and can extract the cuboid shape without any prior knowledge about the camera or any other detail in the scene.

2.3. Pose estimation

Arguably, our method is most similar to the pose estimation field. The common goal for the publications in this area is to estimate the configuration of a known model according to a given image. A prominent direction is the estimation of human poses, with the seminal work of the field, named OpenPose by [2], which estimates the configuration of the human pose in 2D. They do this in a "bottom-up" manner, where first potential locations of joints are estimated, followed by a step to connect them into skeletons correctly. We adopt this concept in our work as well. Our work, however, is more aligned with rigid pose estimation. The latter is addressed using many different approaches in the literature. [19] looks for the orientation of bounding boxes of objects seen in a given image. They suggest estimating the required box dimension and its orientation is correlated, and hence should be deduced by the same network. In addition, they propose a combined discrete-continuous loss, which estimates the parameters in several bins at the same time. While we take some of these concepts into our work, this work uses full supervision — with given 3D objects, images that correspond to them, along with their orientation, which is in complete contrast to our supervision free approach. It has been further shown that given enough training data, the same concept can be applied to more elaborate, fine-detailed, shapes, like in **[20]**. **[3]** adds the concept of synthesizing data in order to improve performance. They do this by estimating correspondences between vertices of given 3D models of several classes, to 2D locations in an image. Once these correspondences are in place, the established PnP method like in **[21]** is employed to extract the 3D orientation of the object accordingly. Similarly, other works have added estimation of camera intrinsic parameters to the process, like **[22][1]**, or propose finding the correct translation and rotation of the target object through concepts of disentanglement, like **[23]**. While these works demonstrate impressive, state-of-the-art results for several classes of objects, they still require mixing between real and synthetic data during training, and rely on finding distinct visual features in the images. For buildings, approaches relying on visual features for correspondence points can prove unreliable since facades have typically very repetitive patterns.

2.4. Explainable artificial intelligence

Explainable AI (XAI) is a rising field in the machine learning community, and the deep learning community in particular **[24]**. The purpose of XAI is to enable ways to interpret models that were previously considered uninterpretable (black-box models), without significant loss of performance. In **[25]**, the connections between explainability and fairness, and explainability and accountability, are discussed. This is why XAI is in high demand in critical decision making systems, such as medical AI systems. The discussion on XAI in those systems and its implications are thoroughly addressed in recent literature, for example in **[26][27][28][29]**. XAI is also common in recommendation systems, for ethical reasons **[30]**. However, XAI is less common in computer vision systems. One similar work to ours is **[31]**, where explainable 3D classification is discussed. Another similar work is **[32]**, where explainable 3D object detection for autonomous vehicles is proposed. Finally, a work that is most similar to ours is **[33]**, where a pose estimation from single images system has explainable components. However, to the best of our knowledge, this is the first work on 3D reconstruction of outdoor objects with explainability.

3. Method

3.1. Overview

Our method seeks to estimate the 2D vertex positions of a cuboid model, which best describes the geometry of a given segmentation mask of a single building. For clarity, we lay out the method using a single cuboid model, however, our method is not restricted to this case alone. In Section 5 we demonstrate another model, of two adjacent cuboids. Many other options could also be conceived, but we found these two to be quite inclusive for urban buildings, and leave using other primitives and combinations to future extensions.

Our solution consists of three neural networks, as can be seen in Figure 2. Our mission is to identify 2D keypoint coordinates of 3D shapes, from images. We assume the image was segmented to instances, and we operate over the segmentation mask of the shape instance. In our work, we trained different models for different classes of shapes. Thus, a decision for the class of the object is needed at an early stage. This decision can indeed be taken early by some heuristics (or a classifier). Alternatively, the decision can be taken after all class predictors were used, by some leveraging of their confidence scores. The first part of our pipeline, the Refiner (R), is introduced to alleviate our finding that interpretation from noisy input is difficult. The Refiner is designed to improve binary segmentation maps by filling in occlusions or holes and reducing noise. Working with cleaner segmentation maps, that are closer to the actual silhouettes, has improved the performance of the next steps, as we demonstrate in Section 5.

Next, instead of explicitly pinpointing desired vertex positions, we follow recent work **[2][34]** and calculate probabilistic heat maps for every vertex of the model — a task more natural for neural networks. Our second network, the Estimator (S), is designed to calculate these heat maps from binary segmentation data. Using points of interest as a means of orientation estimation, instead of the more intuitive approach of explicitly evaluating

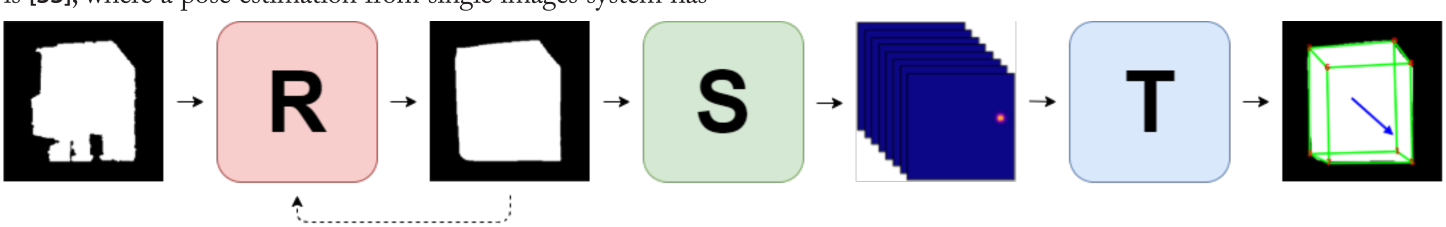

Figure 2: Flow chart of the entire model pipeline. Noisy and occluded segmentation maps are fed to the Refiner (we also suggest that this step repeats until maximal refinement). Clean segmentation maps are fed to the Estimator. Keypoint heat maps are fed to the Tabulator. 3D shape structure is computed.

model parameters such as dimensions and rotations, has been proven useful several times in the past [1][2][3]. In addition, it is well established that networks perform better when trained to produce probability maps instead of being asked to make clear cut choices. This is the reason for the existence of our third step.

Finally, we turn to extract the final 2D coordinates. Ideally, each heat map will be activated in exactly one pixel, or at least should be centered around one pixel, which is the true coordinate. In practice, however, the extraction task poses a twofold problem: First, the generated heat maps are imperfect, due to occlusions, ambiguities, or unclear boundaries. Second, the vertex positions should collectively represent a projected cuboid, hence their extraction should take this geometric prior into account, as opposed to extracting each vertex independently. With these challenges in mind, we design our third network, the Tabulator (T).

3.2. Architecture

The model's architecture is composed of three deep neural networks implemented in PyTorch [35]. The source code is available in the TorchVision library [36]. The networks' architecture is based on ResNet–18 [37], either the convolutional (CNN) variant, which maps tensors to vectors, or the fully-convolutional (FCN) [38] variant, which maps tensors to tensors. The networks are optimized using Adam [39] with the L2 penalty [40]. Each network specializes in a specific, human-interpretable task.

Refiner The **R**efiner (**R**) is an FCN, with the purpose of segmentation map refinement, i.e. removing noise and occlusions. Its input and output are a single channeled $H \times W$ image. The final output layer of this network is activated by a *tanh* function, to force the output to be in the same range as the input, where the pixel value of 1 represents the object, and –1 represents the background.

Estimator The E**s**timator (**S**) is also an FCN, with the purpose of estimating probability heat maps of semantic keypoints from the refined segmentation maps. In our design, we use a single estimator network for N points, where N is the number of vertices in the fitted model. This is in contrary to employing N estimators processing a single point each. The single network configuration facilitates learning the dependencies between the different points and hence outputs a coherent, geometrically correct estimation.

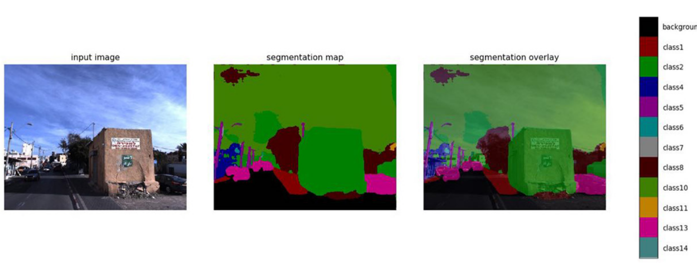

Figure 3: Example of semantic segmentation of an image, with classes used in the autonomous driving dataset Cityscapes.

The Estimator's input is again a single channeled $H \times W$ image, but its output is N images of this size. The output of this network is followed by a sigmoid activation function, where values closer to 1 mean higher probability, and values closer to 0 mean lower probability.

Tabulator The **T**abulator (**T**) is a CNN, with the purpose of learning exact keypoint locations from heat maps. Again, in our design, we use a single tabulator network for N points, contrary to N tabulators for a single point, for internal coherence. The Tabulator's input consists of N images of dimensions $H \times W$, and its output dimensions are $N \times 2$. The output of this network is activated using a sigmoid function and is afterwards multiplied by (H, W), so the final values are in the range of the original segmentation map's dimensions. As shown in Figure 4, the Tabulator holds the critical extracting vertices that are geometrically valid. For example, should a produced map consist of two distinct

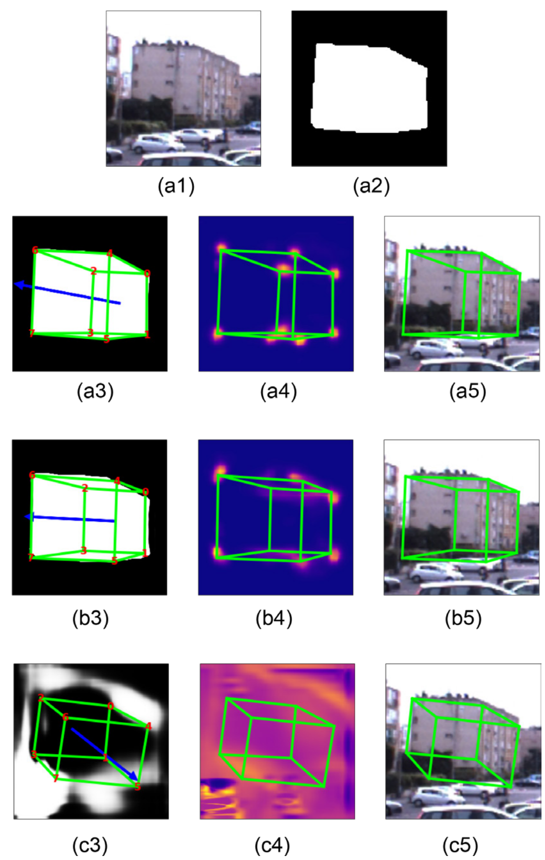

Figure 4. Header: (a1) a natural image; (a2) its binary segmentation map; **Top row:** (a3) refined segmentation map, with the final result overlaid; (a4) the sum of all generated heat maps, overlaid with the final result; (a5) the final result laid over the original image. We also show (d), a visualization of the cropped input segment.
Middle row: restriction-less fine tuning ablation study result — undergoing additional training that did not penalize for the intermediate results.
Bottom row: end-to-end ablation study result — training with no forced representational meaning for intermediate steps.

blobs, the Tabulator decides between them, in a way that resembles a projected box the most. This is possible since the Tabulator makes a holistic decision, looking at all N maps and producing all N positions at the same time.

3.3. Training

3.3.1. Training data

Our unique problem setting enables training our model on synthetic data only. This means producing unlimited amounts of perfectly balanced and labeled data, eliminating bottleneck costs of annotated data collection. To generate a synthetic data sample, we start by constructing the vector of vertex positions that defines our shape. For a box, we construct a canonical cube, having its 8 vertices at locations $(\pm 0.5, \pm 0.5, \pm 0.5)$, where the order is significant (i.e. the front-facing top left vertex is always in index 0, etc.). See Table 1 for definition, For the shape of two adjacent boxes, for example, the structure is composed of 14 points. See Table 2 for definition. Note that one could consider removing the order significance of the vertices, however, this would impose reflection and symmetry considerations during training, making the process more cumbersome, and also less accurate, as it turns out from our experiments.

We then perform a series of random 3D transformations on these vectors. These transformations consist of a translation, a rotation, and a non-uniform scale. We limit the range of the random parameters to be consistent with the task we intend to learn. For example, for the case of street-level images of buildings, one can safely assume that the camera position is not below the object. For our experiments, we uniformly sample the parameter space defined by the ranges found in Appendix A. Once the shape has been positioned in 3D, we project it to the 2D plane using perspective projection with a random projection plane distance. The latter is varied to enable the support of different cameras. Then, the only step left is rasterizing the binary image according to the resulting 2D coordinates, to form the segmentation map.

INDEX	DEPTH	WIDTH	HEIGHT
0	BACK	RIGHT	TOP
1	BACK	RIGHT	BOTTOM
2	BACK	LEFT	TOP
3	BACK	LEFT	BOTTOM
4	FRONT	RIGHT	TOP
5	FRONT	RIGHT	BOTTOM
6	FRONT	LEFT	TOP
7	FRONT	LEFT	BOTTOM

Table 1: Semantic IDs of keypoints in single-cube estimation, agnostic to pose, assuming that the box's front plane is somehow fronting the camera.

This is done by marking with the inner points that are limited by the shape's convex hull. If the object is not convex but composed of convex sub-objects, for example in the two-boxes case, we do this procedure for each sub-object and then union the maps pixel-wise, for a single binary segmentation map. A segmentation map is a square matrix, where the pixels in the polygon filled by these points with 1 (object), contrary to the background pixels, which we set as -1.

Finally, we augment the clean image produced to better match the behavior of real-life segmentation maps. To simulate occlusions, we remove the silhouettes of random objects from the produced image. For each binary segmentation map, we randomly choose an occlusion element from a predefined library of RGBA images. In our example, these are 2D cut-outs of urban elements such as trees, cars, pedestrians, and signs. extracted from an existing dataset of similar scenes — in our case, Cityscapes [41]. We then choose the random size, orientation, and position for this element, and subtract it from the segmentation map (see example in Figure 15). To account for inaccuracies of segmentation networks, we add noise of random amount to each of the generated segment's contour. In Figure 9, we show that our model performs well even if the occlusion type is of a different kind than what it was trained on.

Additionally, for every 2D coordinate, we generate the corresponding heat map. Each heat map contains a single, low variance Gaussian blob around the respective vertex coordinate (see Figure 1). These heat maps serve as the ground truth for the heat map prediction produced by S.

INDEX	DEPTH	WIDTH	HEIGHT (SIDE)
0	BACK	RIGHT	TOP (R)
1	BACK	RIGHT	BOTTOM
2	BACK	MIDDLE	TOP (R)
3	BACK	MIDDLE	BOTTOM
4	FRONT	RIGHT	TOP (R)
5	FRONT	RIGHT	BOTTOM
6	FRONT	MIDDLE	TOP (R)
7	FRONT	MIDDLE	BOTTOM
8	BACK	MIDDLE	TOP (L)
9	BACK	LEFT	TOP (L)
10	BACK	LEFT	BOTTOM
11	FRONT	MIDDLE	TOP (L)
12	FRONT	LEFT	TOP (L)
13	FRONT	LEFT	BOTTOM

Table 2: Semantic IDs of keypoints in two-cubes estimation, agnostic to pose, assuming that the boxes' front planes are somehow fronting the camera. TOP (L) and TOP (R) are the locations of the top of the building on the left hand side and the building on the right hand side, respectively.

3.3.2. Training scheme

Since we synthetically constructed our data, we have full knowledge of its parameters, and in particular, the 2D location of the N keypoints. For this reason, we have unlimited labeled data for training. Unorthodoxly, in this work, the data creation and model training are linked. A pipeline for data generation was described in Section 4.3.1. That pipeline is executed on the CPU, enabling the GPUs to train without competition for resources.

When invoking the training phase, data folder and data size must be specified. Training samples are generated on the CPU, then they serialize and replace older data files on the disk. When the limit of data size is reached, the oldest files are replaced again with new samples. This process runs continuously, and essentially provides an infinite amount of data for training. As a result, the training data folder is constantly refreshed with new samples. One may specify data size of value 0, to retain the data in the folder. In our work, we found that a limit of 10,000 for number of samples in the data folder was sufficient.

At the same time, the model's training on the GPUs begins, and every epoch we sequentially load B data samples from the training data folder. The loading process is enumerated, and is much faster than the generation process, which guarantees that every data sample will be loaded for training at least once. The samples consist of: 1) tuples of noisy and occluded segmentation maps, 2) clean segmentation maps, arrays of heat maps, and 4) vectors of 2D coordinates. We denote the batch of data samples by (x_1, x_2, x_3, x_4).

We define three loss terms, D_R, D_S and D_T, where each term corresponds to a different network.

- $D_R(z_R) = |x_2 - z_R|_1$. The L_1 distance between z_R (an output of the network R) and x_2.
- $D_S(z_S) = |x_3 - z_S|_2$. The L_2 distance between z_S (an output of the network S) and x_3.
- $D_T(z_T) = |x_4 - z_T|_2$. The L_2 distance between z_T (an output of the network T) and x_4.

If we train all networks jointly, then the model is equivalent to a large single network, which yields poor performance and the loss of the interpretability of intermediate results. Instead, We guide the training process and avoid learning uninterpretable representations by introducing a two-phase training scheme, taking a bottom-up approach. The scheme starts with the first phase of training each network separately, according to the intermediate results existing in our synthesized data (see Section 4.3.1). The second phase trains the networks together, ignoring intermediate losses. This improves final performance in terms of accuracy, but at the cost of a possible decrease in interpretability. The training phases are described in more detail below:

Phase 1. We start by training each network separately. We do this for N_1 epochs. In other words, the noisy and occluded segmentation maps, x_1, are fed to R, and the clean segmentation maps, x_2, are expected to be produced. Similarly, the clean segmentation maps x_2 are fed to S, and heat maps, x_3, are the ground truth labels for this network. Finally, the heat maps x_3 are fed to T, which is expected to produce the final vector of vertex coordinates, x_4. In this configuration, we employ three independent loss terms: $D_R(R(x_1))$, $D_S(S(x_2))$, and $D_T(T(x_3))$.

Phase 2. In this phase, we train our 3 networks as a single end-to-end unit, for N_2 epochs. In every epoch, x_1 is fed to R, resulting in $\hat{x_2}$. Then $\hat{x_2}$ is fed to S, resulting in $\hat{x_3}$, which in turn is fed to T, resulting in $\hat{x_4}$. The purpose of this stage is to let networks adapt to the needs of the other networks through joint training. Of course, we still don't want to lose the explainability and guidance we offer in the form of the intermediate results. Therefore, we do not use only the loss $D_3(T(S(R(x_1))))$. Considering only the final result will cause degradation of the intermediate maps, and ultimately the overall performance, as is indicated by our ablation studies (Section 5.1). Instead, we train the network to work as a whole, while preserving the semantics of the intermediate results by considering all of our losses. Namely, the loss for this phase is

$$L = \alpha_R \cdot D_R\left(\hat{x_2}\right) + \alpha_S \cdot D_S\left(\hat{x_3}\right)) + \alpha_T \cdot D_T\left(\hat{x_4}\right)) \quad (1)$$

In our work, the coefficients are $\alpha_R = \alpha_S = \alpha_T = 1$.

3.4. Inference

In inference time, binary segmentation maps are fed to the concatenation of the three networks, producing semantic 2D keypoint estimations. The intermediate results (refined segmentation maps and probabilistic heat maps) can be observed, helping to describe the decision process of the whole system. For example, Figure 1 demonstrates a result of the probabilistic heat maps. As can be seen, the network has identified two possible locations for vertex number 1 (the second maps from top left) and has a rather unclear decision, represented by a smeared blob for vertex number 2 (the third map). These difficulties in identifications become immediately intuitive when looking at human legible heat maps, which could potentially help fine-tune the network and data we feed to it. Figure 4 demonstrates the entire pipeline, including the keypoint estimation from heat maps. In addition, further manipulations can be applied to intermediate products. For example iterative refinement (if the trained R is not stationary) or amplifying heat maps' values. In Figure 7 we show how re-entering the refined segmentation maps to the Refiner, gradually removes more occlusions, ultimately improving the final keypoints estimation.

4. Evaluation

We begin the evaluation of our method by collecting binary segmentation maps. These maps could be segmented manually, or automatically through a semantic segmentation network (or an instance segmentation network). We present results from multiple sources.

For the task of building identification, we have acquired a few examples of outdoor street-level RGB images and their corresponding man-made segmentation maps. We extracted segments of objects that are from the domain of our task (buildings that are composed of a single box or two boxes) and placed each segment in the middle of a new 256×256 binary image.

A similar process was made for automatically segmented images. For semantic segmentation, we use an off-the-shelf TensorFlow **[42]** DeepLab network with Xception 71 backbone **[43]**, trained on the Cityscapes dataset **[41]**. An illustration of it on one of our test images can be seen in Figure 3.

From these segmentation maps, we infer their portrayed shapes by feeding them to the system, as described in section 4.4. An example result from a natural image and a manually annotated segment can be seen in Figure 4, top row. In (a3), each keypoint is marked by its index in red, and the wireframe formed by the keypoints is depicted in green. The orientation from the box center to the middle of the front face of the box is depicted by the blue arrow. In (a4) we observe that resulting coordinates are not necessarily at the maxima of the heat maps — as implicit geometric correctness was learned. Note that we do not have the true keypoint coordinates for the images from both of these sources. Hence, quantitative results can only be calculated on synthetic data, as there is no real-world labeled data of buildings images, their silhouettes, and their keypoint 2D coordinates, to the best of our knowledge. Table 3 indicates the average distance between ground truth 2D coordinates and the inferred ones. As can be seen, this distance for synthetic data is minute (second line, 0.04 pixels).

4.1. Ablation studies

4.1.1. Alternative training scheme

One of our main claims is that a model composed of a few networks, where each one tackles a different, separable, and human legible sub-task of the problem, will outperform a single network model with similar architecture or capacity. In this section, we demonstrate how the proposed training scheme guides the networks for a better solution while providing explainability.

We start by using the same architecture with no phases and no intermediate losses, and simply train all networks in an end-to-end fashion for semantic keypoints estimation.

In Figure 4, bottom row, we demonstrate intermediate and final results of such a model, trained on equivalent terms to our original network. As can be expected, the intermediate results are meaningless to humans. Less predictably, the final keypoints location estimation has also lost accuracy, though the general geometric correctness is preserved. This result demonstrates how our intermediate steps guides the network, for an otherwise difficult optimization problem.

Furthermore, we make a stronger claim: a trained end-to-end model with similar architecture, initialized with the weights of our proposed models, would not surpass our reported performance. In Figure 4, middle row, we show the outputs of such a model, which is trained in an end-to-end manner, with not intermediate losses, after our proposed training scheme. As expected, we see degradation in explainability, as the blobs in the heat maps become smeared. Also, even though the model is not limited to the meaningful intermediate representation, the final result still displays degradation in accuracy. We speculate this is due to the high accuracy achieved with the initialized weights, keeping the parameters in a near space of its initialization.

4.1.2. Alternative architectures

In this experiment, we explore the possibility of replacing the Tabulator network with classic computer vision approaches. One could argue that keypoint locations can be inferred directly from probabilistic heat maps, by taking their center of mass as the point location. This approach tends to collapse in some cases, for example, when there are two major centers on a single heat map (see an example in Figure 1). In this case, the center of mass would not be close to either of the dominant blobs.

Another approach is choosing the maximal value location in the heat map as the point location. This approach can also fail when there are two or more equally dominant centers if the wrong center is chosen. This could happen since this approach works on each heat map locally, when sometimes a global view is needed, to handle outliers, or, importantly, to enforce geometric correctness.

However, global validity can be enforced using classic computer vision algorithms. A prominent example is Point Distribution Model, or PDM **[44][45]**. PDM constructs a linear subspace of the geometric shapes it was initialized with, and it can learn the statistics and the variability of geometric shapes. Once initialized, it enables the refinement of inaccurate shapes by projecting them to its subspace and reconstructing from it. We show an example of a failed heat maps estimation attempt, and a PDM correction, in Figure 5.

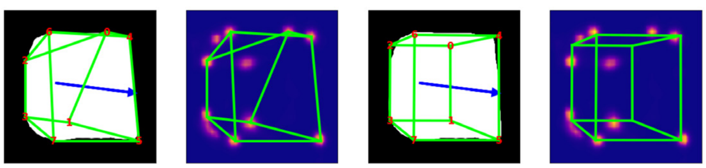

Figure 5. Example of semantic keypoints coordinates estimation chosen as the maximum value from the heat maps, and fixed with PDM. From left to right: (a) refined segmentation map, output of the Refiner, and keypoint locations and structure, estimated from maximal value locations of heat maps; (b) keypoint locations from image (a) over the sum of heat maps, output of the Estimator; (c) refined segmentation map and keypoint locations and structure, estimated from PDM projection and reconstruction over maximal value locations of heat maps; (d) keypoint locations over the sum of heat maps, output of the Estimator; keypoint locations from image (c) over the sum of heat maps.

In our method, we initialize a PDM instance with 1000 ordered 2D octets, generated as described in Section 4.3.1. The instance learns the general statistical shape of a valid building, and the semantic meaning of each point separately. During inference, we extract the maximal value locations of the heat maps, and pass them to the PDM, to assure their global validity. This approach was used in **[1]**. From our experiments, on synthetic data, this approach yields similar results to the Tabulator, but on real-world data, it performs less accurately. Furthermore, it has some disadvantages:

- PDM has a constant parameter called active components, which is analogous to the size of the projected-to-subspace, or the percentage of variance to be kept. Usually, for alignment and noise removal, not all of the subspace is needed, similarly to projecting and reconstructing from PCA. The number of active components needs to be decided in advance, and may not generalize well from problem to problem.
- PDM is not a differentiable algorithm, which prevents backpropagation through it to the previous networks in the training scheme.
- If the semantic label of the points is wrong, then PDM generally fails. An example can be seen in Figure 6.

In Table 3 we summarize the average pixel error over the validation set, for all the discussed approaches. The average pixel error with random weights networks is given as a reference to the results of the followed approaches. It can be seen that learning without representational meaning, i.e., as a single network model, decreases the accuracy of the model. The conclusion is that representational meaning learning is not only enabling explainability but also better accuracy.

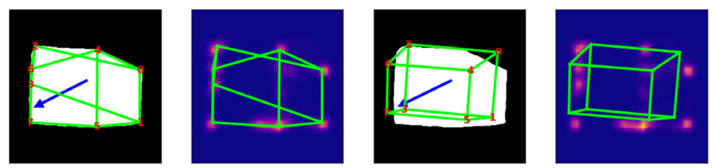

Figure 6. Example of wrong semantic keypoints coordinates estimation chosen as the maximum value from the heat maps, and the failed attempt of it to be fixed by PDM, an approach similar to the one used in **[1]**. From left to right: (a) refined segmentation map, output of the Refiner, and keypoint locations and structure, estimated from maximal value locations of heat maps. It can be seen that two points have their wrong semantic label; (b) keypoint locations from image (a) over the sum of heat maps, output of the Estimator; (c) refined segmentation map and keypoint locations and structure, estimated from PDM projection and reconstruction over maximal value locations of heat maps. Because PDM is extremely sensitive to label noise, its reconstruction yields bad results; (d) keypoint locations over the sum of heat maps, output of the Estimator; keypoint locations from image (c) over the sum of heat maps.

MODEL CONFIGURATION	AVERAGE PIXEL ERROR
RANDOM WEIGHTS	17.9
TRAINED WITH REP. MEANING	0.6
TRAINED WITH REP. MEANING, T REPLACED WITH PDM	0.6
PRE–TRAINED WITH REP. MEANING, TRAINED WITHOUT REP. MEANING	0.8
UNINITIALIZED, TRAINED WITHOUT REP. MEANING	1.0

Table 3. Average pixel location estimation errors on the synthetic validation dataset, per model configurations, based on our architecture. In this context, representational meaning is the enabling of each network to learn its interpretable task.

In these experiments, no noise and occlusions were used. For studying the effect of the level of noise, see Figure 14.

4.2. Other shapes

We continue to investigate the strength of our model by introducing a harder shape class, of two adjacent boxes. An example of our method for such a class can be seen in Figure 9. Furthermore, we claim that our framework is flexible, and can be extended for any cuboid-based class if the users feel that their environment requires it. The definitions of the keypoint indices of the single box and two box classes are in Table 1 and Table 2. Some real-world examples of both classes can be seen in Figure 16.

5. Application: Localization

Following the work we presented, we continue to investigate applications based on it. In particular, we introduce the problem of localization by visual information and our suggestion of its improvement.

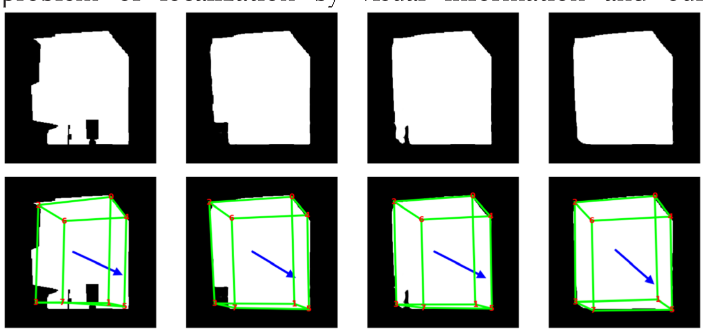

Figure 7. Effect of iterative refinement (re-entering refined binary segmentation maps to the Refiner) on final estimation. Top: from left to right, an occluded segments and results of refinement steps. Bottom: Keypoints estimation corresponding to the segmentation maps above.

Figure 8. Example of single-box segment completion and 8 semantic keypoints coordinates estimation, outputs from the Refiner and the Tabulator. In this example, the Refiner demonstrates imperfect denoising, due that the occlusion type is different than what is was trained on. The Tabulator, however, manages to overcome the imperfect segment and successfully reconstruct the object.

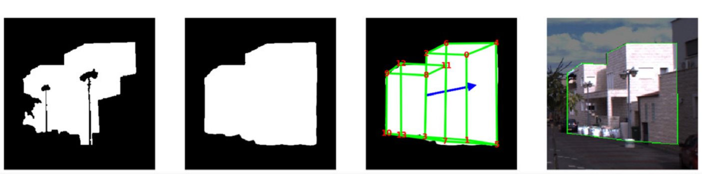

Figure 9. Example of processing a manually annotated segment using the two-boxes, 14 semantic keypoints model. The figure layout is identical to the one in Figure 4.

Localization is the ability to locate an object geographically with computational means. Many environment signals can be used for localization: visual, audial, and most commonly, GPS. GPS is specifically used for localization, however, it is notoriously known for inaccuracy. Under the open sky, GPS enabled smartphones are typically accurate to within a 4.9 meters radius, and worse near bridges, buildings and trees. In this section, we suggest a way to enhance localization by exploiting visual information.

5.1. Formulation

We focus on a specific scenario: understanding the location on a street. We assume the street is known (perhaps by more general systems such as GPS), and all of its buildings are known for their dimensions.

If a picture of one of the buildings is taken and segmented, then we can locate its 2D corners. It is practically impossible to identify a building by a single segment of it, as many similar buildings on that street can also match it. We offer some relaxation to the problem:

- Instead of extracting exact dimension, we extract dimension ratios. This enables us to disregard distance from building ambiguity. The ratios are extracted as another output of the Tabulator in Section 4.2.
- Photos of a sequence of consecutive buildings were taken, not a single building.

We conducted the following experiment: N = 100 noisy and occluded segments of buildings, and their ratios ($\frac{\text{depth}}{\text{height}}$ and $\frac{\text{width}}{\text{height}}$), were sequentially drawn randomly according to the distribution in Table 4. These segments passed the inference process in Section 4.4, and predicted ratios were calculated. Next, we observed all sub-sequences of predicted ratios of length m, for all $m \in [1, M]$, for $M = 30$. For each sub-sequence, we ranked how much it matches to each true sub-sequence of length m. For example, say $m = 3$ and we observe the predicted ratios of buildings (12,

Parameter	Min.	Max.
Scale (X)	×0.5	×2
Scale (Y)	×0.5	×2
Scale (Z)	×0.5	×2
Translation (X)	−2	+2
Translation (Y)	0 Px	+2 Px
Rotation (X)	−10°	10°
Rotation (Y)	−10°	+10°
Rotation (Z)	−5°	+5°
Focal Length	2 Px	10 Px
Flip Probability	0.5	

Table 4: Range of parameters for random data generation.

13, 14). Then we calculate its distance from all true triplets of buildings, (1, 2, 3), (2, 3, 4), etc. The chosen distance is the L_2 distance between the vectors of the ratios. We define accuracy as:

$$acc(m,l) = \frac{1}{N-m+1} \sum_{i=1}^{N-m+1} \mathbb{1} r[i:i+m] \sim^{l} \hat{r}[i:i+m] \quad (2)$$

In other words, the accuracy for sub-sequence length m and top-ranking l is the average number of predicted sub-sequences of length m that match their true sub–sequences, where matching is considered if it is among the top l possibilities (a possibility is ranked higher if the distance is lower).

5.2. Results

In Figure 11 partial tables of accuracy scores, for a sequence of 10 and 100 buildings, are presented. Few insights can be extracted from it. For example, statistically, localization in a 100–buildings sequence, with predicting building rations of only 8 buildings, is enough to have more than 50% probability to identify the correct sub-sequence. When also considering the second most probable match (this can be useful if they are very far geographically and we can eliminate one of them by another localization method), only 3 buildings are enough to achieve 50% accuracy. In Figure 12, we also demonstrate localization in a 10–buildings sequence with a sub-sequence of length 4.

6. Discussion

In this paper, we have proposed a method for a completely unsupervised fitting procedure to texture-less maps of cuboid and bicuboid buildings. The only form of supervision our method incorporates is in the geometric prior of the fitted model. The paper gives an interesting insight into how much of the information required for fitting is actually in the silhouette of the object, and how far can one get with only implicit supervision through the use of a geometric prior. Even though we have demonstrated the effectiveness of our method only for restrictive models (one or two adjacent cuboids), the method should have no conceptual limitation in supporting any simple polygon primitive or composition of primitives, including prismal and pyramidal geometries, and even geodesic domes. Non-polygonal shapes, like conical, cylindrical and hemispherical geometries, cannot be solved by this method without significant expansion, since their faces are defined by the infinite number of vertices. Also, their pose cannot be properly described. In the future, it would be interesting to examine these polygonal and non-polygonal shapes, which would expand the

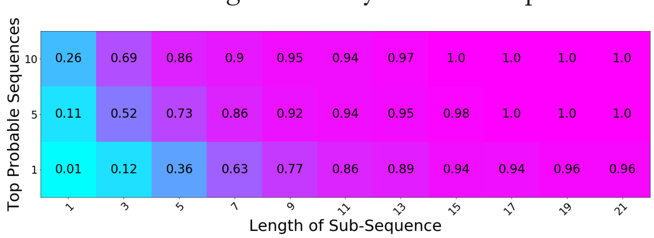

Figure 10: Accuracy scores for sub-sequences matching. For a sequence of 100 random buildings, each cell describes the probability to correctly locate a sub–sequence by building ratios estimation.

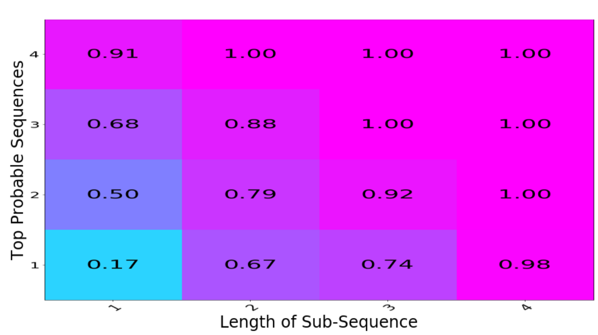

Figure 11: Accuracy scores for sub-sequences matching. For a sequence of 10 random buildings, each cell describes the probability to correctly locate a sub–sequence by building ratios estimation.

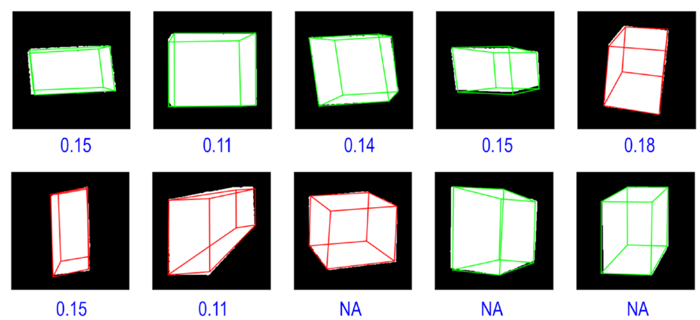

Figure 12: Example of matching a sub-sequence on 4 buildings in a 10 buildings sequence. We attempt to locate the starting building for the predictions of the buildings in red. For each building, we present the probability of it being the first building in the sub-sequence, by calculating the negative distance between predicted and ground truth ratios, and softmaxing it. It can be seen that the highest probability was received for the correct building: the first building in red.

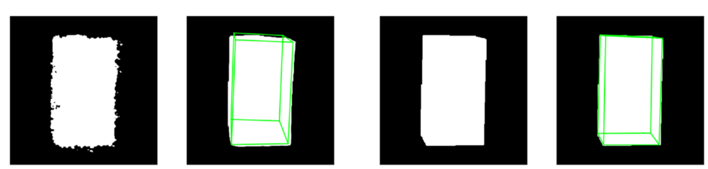

Figure 13: Effect of segmentation noise on 3D reconstruction. From left to right: (a) High level of noise on segmentation map. (b) Failed attempt of 3D reconstruction. (c) No noise on segmentation map. (d) Successful attempt of 3D reconstruction.

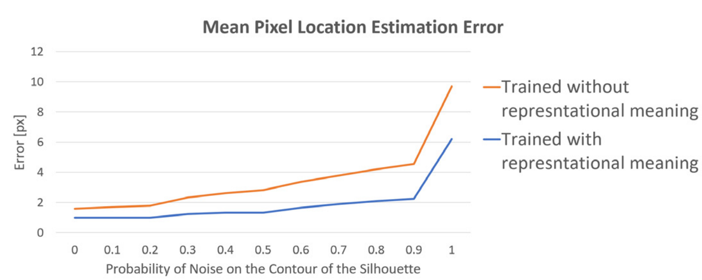

Figure 14: Effect of different levels of noise around the contour of the silhouette in the segmentation map, on the average pixel location estimation error. Although there is a denoising step in the pipeline, the graph shows how high levels of noise in the initial segmentation map can harm the final result.

types of buildings we can support, and see how the relaxation in the model's rigor affect accuracy, for better or maybe for worse.

Another important aspect of our method is *explainability*. We have shown how guiding the training towards specific intermediate results has not only produced human legible milestones, which help in understanding the network's decision making, but also improves performance. This is in contrast to the popular assumption that a network should be free to find its own intermediate representation and warrants further investigation between the trade-offs of a network's expressiveness and the concepts of curriculum learning.

Finally, we have also demonstrated how the method could be applied to the problem of small scale localization — an important and complementary problem to ubiquitous GPS solutions. We believe that carrying a lightweight representation of a region, and using a few simple camera shots to localize oneself accurately could have powerful implications on modern day urban navigation experiences. Furthermore, other fitting approaches have demonstrated that 3D reasoning is attainable using similar concepts to ours. Successfully incorporating this concept into our method would produce a power scheme, which may be able to populate entire cities with 3D building schematics just from sparse street-level images within its streets. These two examples demonstrate how much potential such a system may have, and therefore we hope to see it continue growing in the near future.

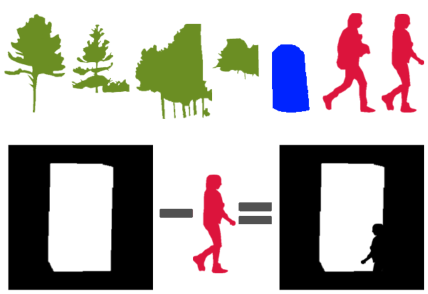

Figure 15: Above: some common urban occluding elements extracted from the Cityscapes dataset. These elements are used in the training process as occlusion augmentation to the binary segmentation maps. Below: subtracting the top right element from a building silhouette. Note how the element was flipped and resized before the subtraction.

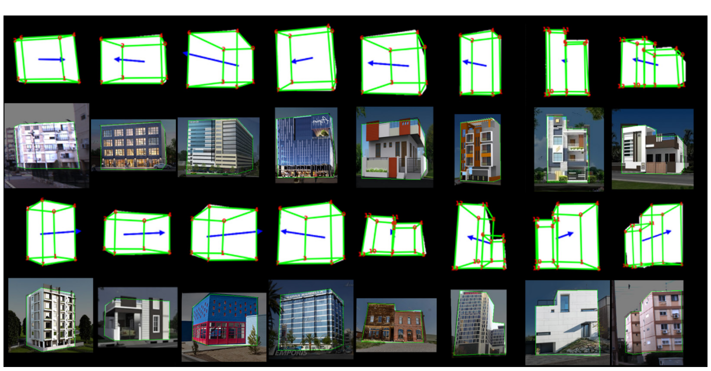

Figure 16: More results on photographs and illustrations of single and double cuboids.

7. Appendix

Listing 1. Random box generator in homogeneous coordinates

```
from random import uniform
import numpy as np
def get_points():

    def rand():
        return uniform(0.25, 1.0)

    def rand2():
        return -rand(), rand()

    BACK, FRONT = rand2()
    RIGHT, LEFT = rand2()
    TOP, BOTTOM = rand2()

    points = np.array(
        [
            [BACK,  TOP,    RIGHT, 1],
            [BACK,  BOTTOM, RIGHT, 1],
            [BACK,  TOP,    LEFT,  1],
            [BACK,  BOTTOM, LEFT,  1],
            [FRONT, TOP,    RIGHT, 1],
            [FRONT, BOTTOM, RIGHT, 1],
            [FRONT, TOP,    LEFT,  1],
            [FRONT, BOTTOM, LEFT,  1],
        ]
    ).T

    depth = FRONT - BACK
    width = LEFT - RIGHT
    height = BOTTOM - TOP
    dims = (depth, width, height)

    return points, dims
```

References

[1] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, “6-DoF object pose from semantic keypoints,” 2017.

[2] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, “Openpose: Realtime multi-person 2D pose estimation using part affinity fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[3] P. P. Busto and J. Gall, “Joint viewpoint and keypoint estimation with real and synthetic data,” in *German Conference on Pattern Recognition*, pp. 107–121, Springer, 2019.

[4] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, “3D-r2n2: A unified approach for single and multi-view 3D object reconstruction,” in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 628–644, Springer International Publishing, 2016.

[5] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, “Occupancy networks: Learning 3D reconstruction in function space,” 2018.

[6] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, “Pixel2mesh: Generating 3D mesh models from single RGB images,” 2018.

[7] G. Gkioxari, J. Malik, and J. Johnson, “Mesh R-CNN,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9785–9795, 2019.

[8] S. Liu, W. Chen, T. Li, and H. Li, “Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction,” 2019.

[9] X. Han, H. Laga, and M. Bennamoun, “Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[10] J. Xiao, B. Russell, and A. Torralba, “Localizing 3D cuboids in single-view images,” in *Advances in Neural Information Processing Systems*, pp. 746–754, 2012.

[11] D. Dwibedi, T. Malisiewicz, V. Badrinarayanan, and A. Rabinovich, “Deep cuboid detection: Beyond 2d bounding boxes,” *arXiv preprint arXiv:1611.10010*, 2016.

[12] M. Hejrati and D. Ramanan, “Analyzing 3D objects in cluttered images,” in *Advances in Neural Information Processing Systems*, pp. 593–601, 2012.

[13] G. Nishida, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and A. Bousseau, “Interactive sketching of urban procedural models,” *ACM Transactions on Graphics*, vol. 35, July 2016.

[14] C. Liu, J. Wu, P. Kohli, and Y. Furukawa, “Raster-to-vector: Revisiting floorplan transformation,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2195–2203, 2017.

[15] N. Nauata and Y. Furukawa, “Vectorizing world buildings: Planar graph reconstruction by primitive detection and relationship inference,” 2019.

[16] G. Nishida, A. Bousseau, and D. G. Aliaga, “Procedural modeling of a building from a single image,” in *Computer Graphics Forum*, vol. 37, pp. 415–429, Wiley Online Library, 2018.

[17] C. Deng, J. Huang, and Y.-L. Yang, “Interactive modeling of lofted shapes from a single image,” *Computational Visual Media*, pp. 1–11, 2019.

[18] H. Zeng, J. Wu, and Y. Furukawa, “Neural procedural reconstruction for residential buildings,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 737–753, 2018.

[19] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, “3D bounding box estimation using deep learning and geometry,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7074–7082, 2017.

[20] Y. Wang, X. Tan, Y. Yang, X. Liu, E. Ding, F. Zhou, and L. S. Davis, “3D pose estimation for fine-grained object categories,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0, 2018.

[21] G. Nakano, “A versatile approach for solving PnP, PnPf, and PnPfr problems,” in *European Conference on Computer Vision*, pp. 338–352, Springer, 2016.

[22] A. Grabner, P. M. Roth, and V. Lepetit, “GP2C: Geometric projection parameter consensus for joint 3D pose and focal length estimation in the wild,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2222–2231, 2019.

[23] Z. Li, G. Wang, and X. Ji, “CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7678–7687, 2019.

[24] W. Samek, T. Wiegand, and K.-R. Müller, “Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models,” *arXiv preprint arXiv:1708.08296*, 2017.

[25] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, “Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible AI,” *Information Fusion*, vol. 58, pp. 82–115, 2020.

[26] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[27] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, "What do we need to build explainable AI systems for the medical domain?," *arXiv preprint arXiv:1712.09923*, 2017.

[28] A. J. London, "Artificial intelligence and black-box medical decisions: accuracy versus explainability," *Hastings Center Report*, vol. 49, no. 1, pp. 15–21, 2019.

[29] A. Singh, S. Sengupta, and V. Lakshminarayanan, "Explainable deep learning models in medical image analysis," *Journal of Imaging*, vol. 6, no. 6, p. 52, 2020.

[30] Y. Zhang and X. Chen, "Explainable recommendation: A survey and new perspectives," *arXiv preprint arXiv:1804.11192*, 2018.

[31] N. Kwon, C. Liang, and J. Kim, "3D4ALL: Toward an inclusive pipeline to classify 3D contents," *arXiv preprint arXiv:2102.12606*, 2021.

[32] H. Pan, Z. Wang, W. Zhan, and M. Tomizuka, "Towards better performance and more explainable uncertainty for 3D object detection of autonomous vehicles," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7, IEEE, 2020.

[33] F. Manhardt, G. Wang, B. Busam, M. Nickel, S. Meier, L. Minciullo, X. Ji, and N. Navab, "CPS++: Improving class-level 6D pose and shape estimation from monocular images with self-supervised learning," *arXiv preprint arXiv:2003.05848*, 2020.

[34] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3D hand tracking from monocular RGB," in *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024– 8035, Curran Associates, Inc., 2019.

[36] "Resnet implementations by pytorch's torchvision." https://github.com/pytorch/vision/ blob/v0.9.1/ torchvision/models/ resnet.py, 2021.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[38] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[41] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems." https://www. tensorflow. org/, 2015. Software available from tensorflow.org.

[43] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, 2017.

[44] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[45] T. F. Cootes, C. J. Taylor, *et al.*, "Statistical models of appearance for computer vision," 2004.