arXiv:2411.19149v4 [cs.CV] 30 Jul 2025

Counting Stacked Objects

Corentin Dumery[1], Noa Etté[1], Aoxiang Fan[1], Ren Li[1], Jingyi Xu[2], Hieu Le[1], Pascal Fua[1]

[1]EPFL, [2]Stony Brook University

{name.surname}@{epfl.ch, stonybrook.edu}



Figure 1. **3D Counting (3DC).** From multiple views of objects to be counted and their container, we estimate both the total volume they occupy and the fraction of this volume taken up by the objects. Combining these estimates yields the total number of objects.

Abstract

Visual object counting is a fundamental computer vision task underpinning numerous real-world applications, from cell counting in biomedicine to traffic and wildlife monitoring. However, existing methods struggle to handle the challenge of stacked 3D objects in which most objects are hidden by those above them. To address this important yet underexplored problem, we propose a novel 3D counting approach that decomposes the task into two complementary subproblems - estimating the 3D geometry of the object stack and the occupancy ratio from multi-view images. By combining geometric reconstruction and deep learning-based depth analysis, our method can accurately count identical objects within containers, even when they are irregularly stacked. We validate our 3D Counting pipeline on large-scale synthetic and diverse real-world datasets with manually verified total counts. Our datasets and code and can be found at https://corentindumery.github.io/projects/stacks.html

1. Introduction

Visual object counting—the task of quantifying the number of instances in a scene—serves as a fundamental building block for numerous real-world applications and autonomous decision-making systems. This ranges from cell counting in biomedical imaging [27] to traffic [12] and wildlife [1] monitoring. However, these methods [10, 11, 18, 21, 29] can only count visible objects such as apples spread across a table or people in a crowd. The problem becomes significantly harder when objects are stacked on top of each other, as in Fig. 1: Only a subset of them is visible, making counting much more difficult. In fact, our experiments show that this task is truly challenging even for humans. Nevertheless, solving it would have significant applications in industrial and agricultural settings, where precise quantification of items—such as products on a pallet or fruits in crates—not only prevents stock and quality errors but also enhances operational efficiency and logistics.

Overcoming the above-mentioned challenges requires inferring the presence and quantity of hidden instances from limited visual cues. This means not only detecting visible object features but also reasoning about hidden ones through contextual understanding. The challenge is further amplified by variations in stacking patterns, object orientations, and irregular arrangements, making traditional counting approaches inadequate.

At the heart of our proposed solution is a key insight: the fraction of space occupied by objects, which we will refer to as the *occupancy ratio*, can be accurately inferred from a depth map computed by a monocular depth estimator from a view in which enough objects of interest are clearly visi-

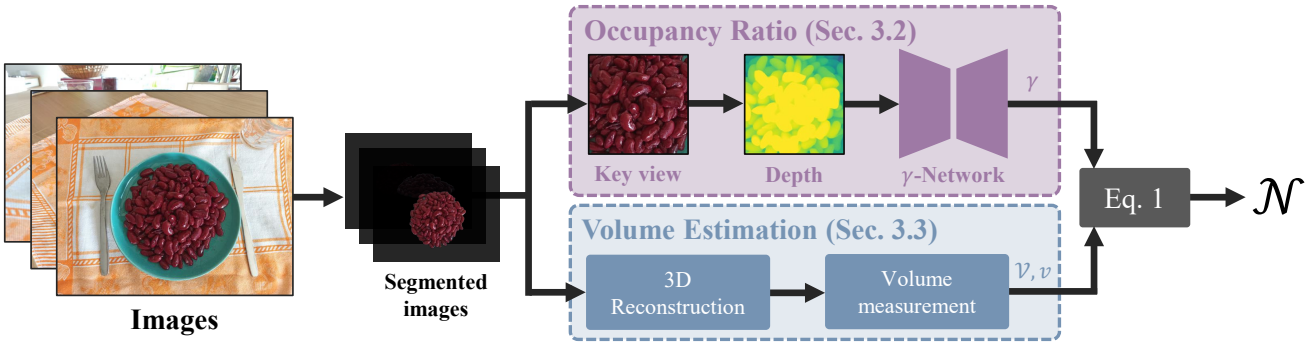


Figure 2. **3DC pipeline.** We decompose the counting task into estimating the volume of the objects to be counted and then estimating the occupancy ratio within that volume. The first is done on the basis of geometry reconstructed from segmentations in multiple images.The second uses as input a depth-map computed by a monocular depth estimator and regresses an occupancy ratio from it.

ble. In most cases, such a view is one where the container is seen roughly from above, without having to be strictly vertical. To exploit it, we break down the problem into two complementary tasks: estimating the 3D geometry of the object stack and estimating the occupancy ratio within this volume, as depicted by Fig. 2. This decomposition enables us to solve the 3D counting problem through a combination of geometric reconstruction for volume estimation and deep learning-based depth analysis for occupancy prediction, both of which can be solved efficiently.

We validate *3DC* through extensive experiments on real-world and synthetic datasets. Our real-world evaluation leverages a diverse collection of scenes depicting objects stacked in containers or still in their packaging, as shown in Fig. 1. To further assess the reliability of 3DC, we also constructed a large-scale synthetic dataset with precisely annotated ground-truth counts. This dataset, along with our code, will be made available upon publication.

Thus, our contributions are:

- A complete pipeline for 3D counting of overlapping, stacked objects, a novel and challenging computer vision task not previously addressed in the literature.
- A network designed to infer the occupancy ratio, which embodies a novel idea and forms a key component of the architecture.
- An extensive new 3D Counting Dataset comprising 400,000 images from 14000 physically simulated and rendered scenes with precise ground-truth object counts and volume occupancy computed programmatically.
- A complementary real-world validation dataset consisting of 2381 images from 45 scenes captured with accurate camera poses and manually verified total counts.
- A human baseline derived from 1485 annotations on real images, representing estimates from 33 participants.

In particular, the latter shows that this task is truly hard and that humans perform poorly. This indicates that training a network to predict stacked counts from images in a single step may not be feasible and that our decomposing the problem into simpler subproblems is critical to success, as demonstrated in Sec. 4.

2. Related work

Counting aims to estimate the number of instances of a specific object category in a scene. Most methods focus on counting visible objects from a single image. A few methods leverage multiple images to enhance counting accuracy. Our method extends this to the more challenging scenario where many of the objects to be counted are hidden. We summarize those related work below.

Single-View Counting. Most recent counting methods focus on single-view scenarios where they train a network specialized in counting a single object-category such as for crowd-counting [3, 9, 22, 24, 26, 28, 35, 36], counting cars [12], or penguins [1]. These methods are widely applicable [2], such as when counting cells or other anatomical structures in medical imaging [4] or counting trees or building in satellite images [33]. The proposed algorithms address challenges such as scale variation, perspective distortions, and occlusions. Common approaches are to learn robust feature representations [39], to perform density map estimation [23], or to leverage multi-scale features [17]. Apart from the traditional setting in which objects of a single category are counted, class-agnostic counting [11, 14] enables counting an arbitrary category at test time, given a few image samples of the class [31], a few bounding boxes [18], or just the class name [29]. Like class-agnostic counting methods, our approach is not restricted to specific categories but can generalize to any object type at test time. For the most part, counting methods only deal with visible objects. The only related attempt to ours is the approach of [5] that infers counts of occluded objects by using LiDAR data. However, their algorithm only handles a specific setup of counting different beverages on shelves for a specific set of categories with known volumes, such as "*Coca-Cola 20oz bottle*" or "*milk carton*", while our method does

not require LiDAR and generalizes to diverse scenes, object types, and geometries.

Multi-View Counting. Multi-view counting approaches improve accuracy by combining information across multiple camera views, often projecting feature maps onto a common ground plane to generate precise density maps for crowd counting [37, 38] or segmentation maps of fruits [13]. However, these methods also assume that all objects are visible from at least one view and are restricted to a specific class of objects, making them inadequate in many practical scenarios. In contrast, our method targets the largely unexplored area of counting occluded objects, without any restriction on the nature and shapes of these objects.

3. Method

3DC aims to estimate the total number $\mathcal{N}$ of objects in a container solely from a set of 2D images, which is a challenge even for humans and has not been attempted before as far as we know. Our approach is predicated on the idea that, even though we cannot faithfully recover the exact arrangement of all invisible objects at the bottom of the container, the occupancy ratio can be estimated from a single image, provided that enough objects are visible in it.

To localize the objects in cluttered scenes that may contain several stacks, we rely on a segmentation of the objects S_o and their container S_c in the first frame. We then use SAM2 [19] to propagate this segmentation as $S_{o,f}$ and $S_{c,f}$ to all subsequent frame f, so that the frame in which the most objects are visible can be automatically identified.

In the remainder of this section, we first formalize our approach. We then introduce our occupancy ratio estimator, followed by our approach to estimating volumes.

3.1. Problem Statement

Assuming the average volume occupied by a single object is v, and the total volume of the container is $\mathcal{V}$, it would be tempting to compute the number of objects as $\mathcal{N} = \frac{\mathcal{V}}{v}$. However, this fails to account for gaps between objects. If we consider that the objects are stacked in such a way that only a fraction γ of the volume $\mathcal{V}$ is actually taken up by the objects themselves and that the rest is empty space, then the previous estimate becomes

$$\mathcal{N} = \frac{\gamma \mathcal{V}}{v} \tag{1}$$

Our key insight is that this volume usage rate γ over the whole container can be estimated with high accuracy from the visible elements only. In practice, the density within the container may not be strictly uniform. However, if the container is large enough, the variations tend to compensate each other over the whole volume and using an average value is warranted.

Assumptions. In this work we assume that the objects are stacked uniformly in bulk and approximately identical. Some objects are expected to be partially visible, so that the occupancy ratio γ can be estimated.

Applicability. The above assumptions hold in the real-world scenes of Fig. 1. Furthermore, they are weak enough to also hold in many realistic scenarios across various industries. In warehousing and retail, our proposed setup can be used to automate the inventory process by accurately counting stacked items, reducing the need for manual labor and easing restocking. In manufacturing, our method can enhance quality control by ensuring that shipped containers include a sufficient number of items. It can also provide 3D scene understanding to autonomous systems for robotic tasks like pick-and-place and sorting.

3.2. Occupancy Ratio Estimation

The most critical step in our approach is estimating the occupancy volume ratio γ of Eq. 1 from a single image in which enough target objects can be seen, typically one taken from above even though this is not a strict requirement.

Formally, we seek to learn a function $\Phi : \mathcal{D} \rightarrow \gamma \in [0, 1]$ that takes as input a depth map and predicts an occupancy ratio γ. Φ learns the relationship between depth maps and occupancy ratio, capturing a key fact: if objects deeper in the stack remain visible, then the gaps between objects are large and the percentage of the volume occupied by them is low. This does not depend on the exact shape of the objects being observed, and should, in theory, apply even to new inputs with shapes not seen during training. In the results section, we will confirm this to be true in practice.

Network Architecture. To implement Φ, we use an encoder-decoder architecture that first computes rich image features and then decodes them into our target γ. For the encoder, we use DinoV2 [15], a foundation model trained on many real-world images to help with generalizability while increasing convergence speed.

The decoder has to aggregate feature values into a single scalar representing the volume usage, predicted for the whole image. To this end, we use consecutive convolutional layers to gradually decrease the resolution of the features, along with ReLU activations, reducing the encoded feature image to a single pixel with 64 channels, and a final linear layer to predict a single scalar from the output of the last convolutional layer. We refer the reader to our supplementary material for additional details.

Training Data. To train Φ, the simplest is to minimize the squared error between the estimated occupancy ratio and ground-truth one over an annotated dataset. Unfortunately, no such dataset exists and we therefore synthesized our own. It comprises 400,000 images spanning 14,000

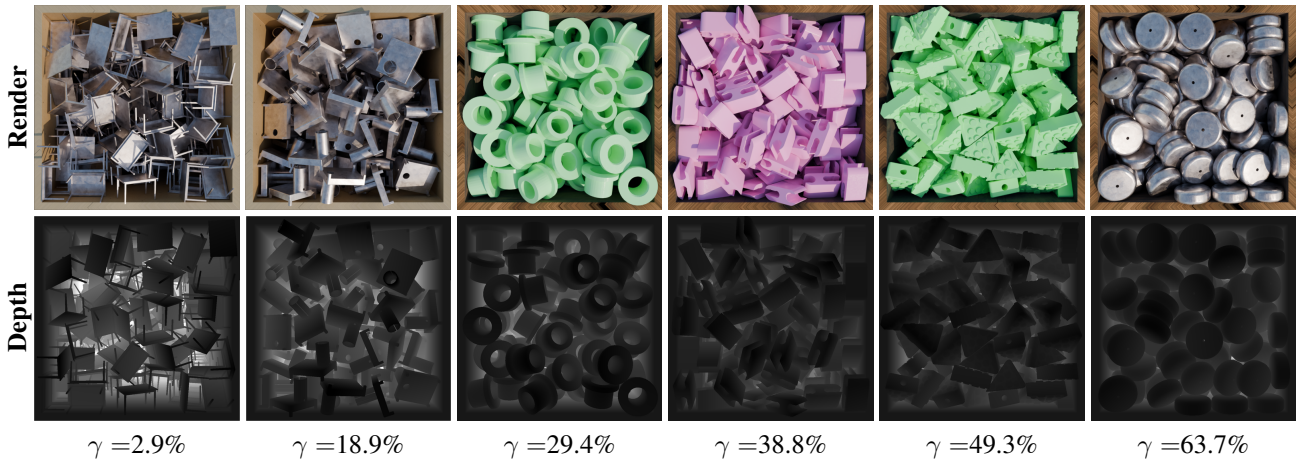


Figure 3. **Dataset samples.** We visualize generated scenes in ascending order of occupancy ratio, with ground-truth depth maps.

scenes containing various objects in different containers. Fig. 3 depicts some of them.

To create it, we used the ABC dataset [8] that features a wide variety of *computer-assisted design* (CAD) models. We retained only watertight objects with a single connected component and rescaled them to fit in a cube of side 0.05. We then generated a virtual 3D scene with a container, and used a physics-based simulator to drop an initial batch of 100 identical objects in that box. We repeated this step a random number of times or until the box was full, that is to say, the union of objects reached the space above the box after the physical simulation has converged. In each scene, the container is given a random shape and scale. We also include some scenes without any container where objects are directly stacked on the floor, and some scenes where boxes are partially full, as is often the case inreal-life.

Once the physical simulation was complete, we computed analytically the occupancy ratio as well as the total number of objects in the container. To each object and container, we randomly assigned a realistic material that could be metallic, wood, or plastic. In some scenes, we selected a different material for each individual object. Finally, we used a ray-tracing engine to render multiple realistic images of the container and objects seen from several different angles to allow 3D reconstruction, as shown in Fig. 5. Crucially, realistic rendering causes objects at the bottom of the boxes to appear darker due ambient occlusion.

We repeated this for over 14,000 scenes and set aside a subset of 100 to use as a test set of shapes unseen during training. For each one of the 4800 shapes used, we generate a scene with a unit-cube container to reliably measure the ground truth γ as well as two additional scenes with random containers. This dataset allows us not only to train Φ on the top view, but also to run our complete pipeline on the multi-view images in order to measure the accuracy of our count estimate $\mathcal{N}_{est}$, as performed in Sec. 4. We train our model on depth maps produced by the depth estimator, but our dataset also includes ground-truth depth maps, which we employ in an ablation study in our experiments to assess the requirement of accurate depth maps of our method. We report additional statistics that highlight the diversity of the proposed dataset in Fig. 12 of our supplementary.

Inference. On real-world data, it remains to determine which image to use as input to the γ-network. We will refer to this image as the *key view*. To produce the best results, this view should be as close as possible to the depth maps seen during training, as illustraed in Fig. 4. We automatically select the view that has the largest object segmentation, ensuring the objects are clearly visible, and crop the image to include only the masked content. We then employ Depth Anything V2 [30] to compute a depth map. Note that we also train Φ on depth maps predicted by that model instead of the ground-truth depth maps, which further reduces the domain gap as demonstrated in our experiments.

3.3. Volume Estimation

Given an estimate of the occupancy ratio γ obtained as discussed above, we still need to compute the total volume of the stack $\mathcal{V}$ to derive the total number of objects from Eq. 1, and the unit volume v if it is not known. When the cameras are uncalibrated, we use COLMAP [20] to compute their poses and adjust their scale using a real-world reference measurement. In industrial scenarios with fixed camera setups known *a priori*, simpler methods could be be used.

Inferring the volume $\mathcal{V}_{est}$ of the stacked objects from multiple images is a well-understood problem. In our specific implementation, we start from the segmented images and extract the container and objects by adding the mask as alpha channel. We then optimize 3D Gaussian Splats [6] from these images, ensuring our reconstruction covers the



(a) (b) (c)

Figure 4. **Reducing the domain gap.** Instead of estimating the occupancy ratio γ from synthetic (top) and real images (bottom) (a), we identify a *key view* (b) and train a network to predict γ from their depth maps (c), which are indistinguishable. Top row: synthetic, $\gamma_{gt} = 62.4\%$. Bottom row: *chocolates*, $\gamma_{est} = 53.5\%$, $\mathcal{N}_{est} = 119, \mathcal{N}_{gt} = 131$.

container and objects only. To compute the volume the splats enclose, we adapt the voxel carving algorithm. We initialize a voxel grid from the bounding box of the gaussians. Then, given the masks generated previously and depth maps rendered from 3DGS, a voxel is carved out if its projection on any given view falls outside of the mask or if its projected depth is less than the reconstructed depth map. This procedure successfully reconstructs objects and their containers, including partially filled boxes.

To remove the container from the reconstruction, we estimate its thickness t and erode the voxels on all sides except the top by t, thereby refining the estimated volume to represent only the contents. The value of t is predicted by an additional decoder Ψ that takes the same encoded image features as our previous network Φ in Sec. 3.2. To this end, we use dilated convolutional layers [34], increasing the network's receptive field with no additional parameters. To make this prediction scale independent and easily predictable from 2D images alone, we predict thickness as a ratio of the container's size. We supervise Ψ with ground-truth thickness from our dataset and at inference we average the estimation of Ψ over all images. Taken together, these algorithms yield a good estimate of the volume $\mathcal{V}$ spanned by the stacked objects.

Unit volume v. In most applications, the unit volume v of Eq. 1 is known exactly because the object has been manufactured to a precise specification, v can be computed from the simple geometry of the objects, or v can be obtained from existing reference data, particularly for food items. When the unit volume of an object v is not readily available, we estimate its value using the method described above and

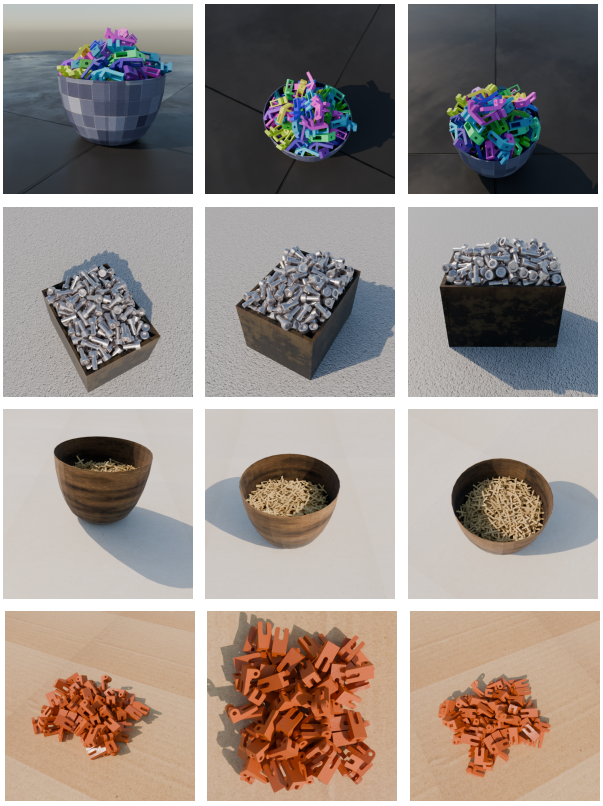


Figure 5. **Multi-view images.** We generate 30 views from arbitrary angles for each of the scenes in our large-scale synthetic dataset.

from a set of images of a template object. This task is made easier by the absence of a container, and this is shape but not scene specific. For a new shape, one exemplar is selected in a single frame and we can then use SAM2 [19] to generate a segmentation on all frames at once. The unit volume v computed this way can then be used across all scenes containing this object.

4. Experiments

We evaluate our method in two ways: measuring the accuracy of 3D counting as a whole and of the occupancy ratio estimation, in Sec. 4.2 and Sec. 4.3, respectively. These evaluations are performed over two datasets. The first comprises 100 scenes representing physically simulated shapes from the ABC dataset [8]. These scenes were isolated after their generation and were not seen during the training of our occupancy ratio network. The second is made of 2381 real images spanning 45 real scenes that were captured with a regular smartphone's RGB camera, and no additional sensor. These captures offer multiple views around stacks of objects in a container, lying flat on a table or still enclosed in their packaging. We count the ground-truth number of units manually for all scenes below 500, or infer it from the

	NAE ↓	SRE ↓	MAE ↓	sMAPE ↓
BMNet+ [21]	0.91	0.87	320.50	158.87
SAM+CLIP [7, 16]	0.73	0.61	259.22	102.77
CNN	0.66	0.48	235.74	98.44
ViT+H	0.42	0.24	149.90	47.36
Ours	**0.22**	**0.09**	**79.48**	**27.65**

Table 1. **Counting evaluation on 100 synthetic scenes.**

	NAE ↓	SRE ↓	MAE ↓	sMAPE ↓
BMNet+ [21]	0.93	0.98	966.76	131.44
SAM+CLIP [7, 16]	0.94	0.99	980.33	124.31
CNN	0.95	0.93	992.06	97.09
ViT+H	0.94	0.93	979.29	91.45
Human	0.79	0.84	823.23	76.85
Human-Vote	0.60	0.30	621.46	57.91
LlamaV 3.2	1.00	1.00	1037.5	190.48
Ours (Color)	0.57	0.27	607.98	74.33
Ours	**0.36**	**0.06**	**382.59**	**53.31**

Table 2. **Counting evaluation on real-world scenes.**

	MAE ↓	RMSE ↓	sMAPE ↓	R^2 ↑
DepthExtrapolated	0.36	0.38	77.43	-6.04
DepthCorrected	0.10	0.12	34.80	0.28
Mean Estimator	0.12	0.14	41.25	0.00
Ours	**0.06**	**0.07**	**29.18**	**0.79**

Table 3. **Occupancy ratio estimation.** We evaluate our method against three additional baselines that are tasked with prediction the occupancy ratio γ from a depth map.

weight for even larger counts. In Fig. 7, we provide intermediate results to help our readers form a better intuition about our method's behavior, and additional qualitative results in Fig. 6 and Fig. 1.

4.1. Metrics

We use several metrics to assess the accuracy of object counting and occupancy ratio estimation. The object counts vary significantly across scenes, ranging from 19 to 20063. Thus, the Mean Absolute Error (MAE)—defined as MAE $= \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, where y_i is the ground truth count and $\hat{y}_i$ is the predicted count for each scene—tends to amplify the importance of scenes with a high counts. To mitigate this, we also report normalized metrics. We use the Normalized Absolute Error (NAE), Squared Relative Error (SRE), and Symmetric Mean Absolute Percentage Error (sMAPE), which scale errors relative to the ground truth. The NAE provides a measure of the absolute error normalized by the total ground truth count across scenes, SRE emphasizes larger errors and penalizes significant deviations in high-count scenes, and , sMAPE offers a normalized percentage error. The exact formulas for all metrics can be found in our supplementary.



(a) $\mathcal{N}_{est} = 38, \mathcal{N}_{gt} = 36$ (b) $\mathcal{N}_{est} = 2133, \mathcal{N}_{gt} = 1830$

(c) $\mathcal{N}_{est} = 338, \mathcal{N}_{gt} = 397$ (d) $\mathcal{N}_{est} = 261, \mathcal{N}_{gt} = 300$

Figure 6. **Additional qualitative results.**

4.2. Counting Evaluation

As far as we know, there is no previous work on counting from multiple images which does not either implicitly assume all objects to be visible or require additional sensors such as LiDARs. Thus, we compare *3DC* against BMNet+ [21]. It predicts a density map over all pixels of an image, and the estimated count is then inferred by summing over all pixels. Additionally, we compare against a combination of SAM [7] and CLIP [16], where SAM is used to generate a large number of masks from an input image and CLIP uses a set of negative and positive text prompts to identify masks that represent an object of interest. The final count is then taken to be the number of these masks. In Tables 1 and 2, we compare the performance of these two baselines against that of our method. We outperform them in both cases.

In early experiments, we attempted to directly predict object count from images. To this end, we trained different networks, coined *ViT+H* and *CNN*. We also report the results in Tables 1 and 2. They perform poorly, especially on the real-world dataset, which is what prompted us to look into decomposing the problem into occupancy and volume estimation.

We also sought to estimate how good humans are at this counting task. To this end, we organized a contest and encouraged participants to make accurate guesses on the 45 real scenes. The contest registered 1485 guesses from 33 participants. We define the *Human* baseline as the average of error metrics of participants, and *Human-Vote* as the error of the average guess across all participants. This second baseline should be stronger as the errors of participants tend

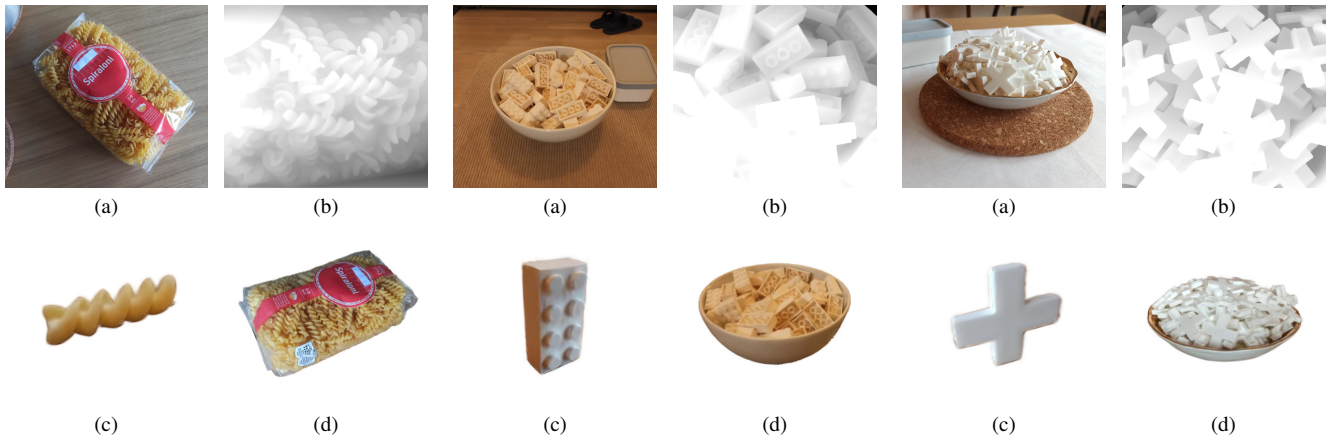


Figure 7. **Intermediate results.** From images (a), we find the key viewpoint and compute its depth (b) to estimate the occupancy ratio γ. Using the unit shape previously reconstructed from images of a template (c), and the overall reconstruction (d), we deduce the final count. (*Pasta*: $\gamma_{est} = 30.5\%$, $N_{est} = 509$, $N_{gt} = 588$. *Bricks*: $\gamma_{est} = 31.8\%$, $N_{est} = 73$, $N_{gt} = 100$. *Crosses*: $\gamma_{est} = 29.6\%$, $N_{est} = 88$, $N_{gt} = 116$)

to cancel-out and that is what we observe in Tab. 2. However, the results are still much worse than what our approach delivers. Interestingly, participants who spent a longer time did not perform better than their peers, highlighting the difficulty of this task. When asked about the method they used, a majority of participants reported counting the number of objects on each axis, and multiplying these values together. This proved to be ineffective, however.

Finally, we tested *LlamaVision 3.2 11B*, a Large Language and Vision Model, on our counting task. While it was very good at describing the physical appearance of the objects in the real scenes and the composition of the scene, its count estimates were completely off as shown in Tab. 2.

Our approach beats all the aforementioned baselines, providing the first method to estimate stack counts with reasonable accuracy. However, we also note that our perfomance remains better on the synthetic data than on our benchmark of real scenes. This can be attributed to the additional complexity of these real scenes, which often contain thousands of objects and can be more challenging for volume estimation.

4.3. Occupancy Ratio Evaluation

We now focus on the occupancy ratio network alone. There is little work on occupancy ratio estimation, we thus implemented additional baselines to gain insights about the inner working of our approach.

Since we assume that the depth map contains enough information to predict the occupancy ratio, we define a first baseline that we will refer to as *DepthExtrapolated*. Given the top view of the container, we compute the maximal depth using a monocular depth estimator and use it to normalize the depth map. Then, we average the resulting values of the K pixels, yielding the volume fraction estimate

$$\gamma_{est}^{norm} = \frac{1}{K} \sum \frac{d_i}{d_{max}} \, .$$

This first baseline tends to predict values lower than expected. Thus, we defined a second one we dubbed *DepthCorrected*. It uses linear regression to correct γ_{est}^{norm} into a new estimate $\gamma_{est}^{corrected}$. This yields a method able to model the observation that depth maps with high variance tend to correspond to low occupancy ratios. Finally, we also compare with the mean estimator, that predicts the mean percentage of 32.3% occupancy ratio for all inputs.

As can be seen in Tab. 3, our method outperforms these three baselines by a significant margin. *DepthCorrected* is better than the other two baselines, which shows that depth information is indeed useful for this task. However, it does not fully predict the occupancy ratio. We interpret this as a strong clue that the ratio $\frac{d_i}{d_{max}}$ alone is not enough to predict γ, and our proposed network successfully learns to extract meaningful information from the depth maps. We hypothesize that our volume network captures additional geometric information such as the influence of concavities in the final occupation of volume.

4.4. Ablation Study

We ran additional experiments to evaluate the sensitivity of our approach to the depth maps produced by the monocular depth estimator.

Since our synthetic dataset has ground-truth depth maps both for the training images and the validation images, we use them as follows. Recall from Sec. 3.2 that, at training time, we use Depth Anything V2 [30] depth maps, a setting we refer to as $\mathcal{T}-$ for training without ground-truth.

	NAE ↓	SRE ↓	MAE ↓	sMAPE ↓
$(\mathcal{T}-, \mathcal{D}-)$	**0.22**	**0.09**	**79.48**	**27.65**
$(\mathcal{T}+, \mathcal{D}-)$	0.28	0.11	100.12	30.93
$(\mathcal{T}+, \mathcal{D}+)$	0.31	0.12	111.04	35.92

Table 4. **Ablation study on 3D counting.** If ground-truth depth maps are used during training, it is indicated as $\mathcal{T}+$, and $\mathcal{T}-$ otherwise. Similarly, for evaluation purposes if ground-truth depth-maps are used during validation, we indicate it as $\mathcal{D}+$.

	MAE ↓	RMSE ↓	sMAPE ↓	R^2 ↑
$(\mathcal{T}-, \mathcal{D}-)$	**0.06**	**0.07**	**29.18**	**0.79**
$(\mathcal{T}+, \mathcal{D}-)$	0.08	0.11	32.01	0.52
$(\mathcal{T}+, \mathcal{D}+)$	0.10	0.13	37.35	0.32

Table 5. **Ablation study on occupancy ratio estimation.**

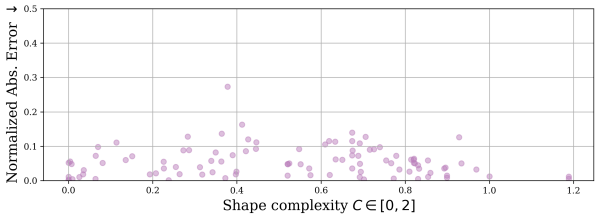


Figure 8. **Complexity Analysis.** Each point represents the γ occupancy ratio error for a shape in the validation set.

Instead, we could use ground-truth depth maps during training, a setting we will refer to as $\mathcal{T}+$. Similarly, at inference time, we can use the estimated depth map. which is what we normally do, or the ground-truth one. We refer these as $\mathcal{D}-$ and $\mathcal{D}+$, respectively. Thus, the standard configuration of our method is $\mathcal{T}-, \mathcal{D}-$. The others are only used for ablation purposes.

We report the results in Tab. 4 and occupancy ratio estimation in Tab. 5. Entirely dropping the ground-truth depth maps and relying *only* on estimated ones, which is our standard operating procedure, is best. We hypothesize that this is due to the slight smoothing in the produced depth maps, which may prevent the model from overfitting to specific shape features in perfect depth maps.

Even though the network is trained on synthetic data, this observation further confirms the generalizability of *3DC* to real data, since in a practical scenario such as the real data of Tab. 2, ground-truth depth maps are not available.

Finally, we also report in Tab. 2 an ablated method *Ours (Color)* where the γ-network takes an RGB image as input instead of a depth map. While this method still outperforms humans, it suffers from a significant performance drop and justifies the use of depth maps in our final approach.

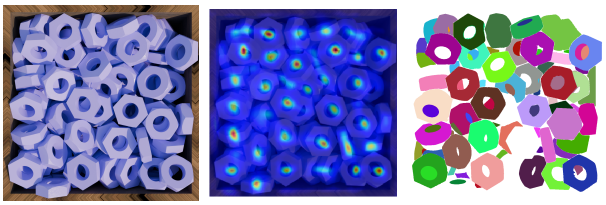
4.5. Limitations and Future Work

To evaluate the robustness of our method to complex shapes, we visualize in Fig. 8 the error in occupancy ratio estimation as a function of shape complexity. The latter is measured by summing a curvature complexity term with the ratio of the shape's volume by the volume of its convex hull:

$$C = \frac{\kappa}{\|x_{max} - x_{min}\|_2 \kappa_0} + \frac{V_{hull} - V}{V_{hull}}$$

where κ is the integrated mean curvature of the shape, $\|x_{max} - x_{min}\|_2$ is a scaling factor and κ_0 is the maximum scaled κ observed in the dataset. We only observe a slight error increase as the shapes become more complex,

Unlike ours, many earlier methods attempt to localize the objects being counted, increasing the interpretability and usability of the results. However, these localizations are often erroneous when objects are stacked together, as illustrated in Fig. 9, greatly limiting their applicability. Thus another possible direction for further enhancements lies in integrating a robust localization of visible instances, and an estimation of a possible configuration of invisible instances.



(a) Input image (b) BMNet+ [21] (c) SAM[7]+CLIP[16]

Figure 9. **Instance localization.** Previous methods also produce interpretable results, representing a promising direction for future work.

5. Conclusion

In this paper, we introduced a novel method to count sets of stacked nearly-identical objects. In this scenario, occlusions and irregular arrangements make accurate counting difficult. By decomposing the counting task into complementary subproblems—estimating independently the 3D volume of the stacks and the proportion of this volume actually occupied by objects—we were able to propose an effective solution that is easy to implement and far outperforms humans at this highly non-trivial task.

Our experiments show that the performance of our approach can degrade with increased geometric complexity or visually complex scenes. In future work, we will therefore look into training the volume occupancy estimator to overcome these challenges. More generally, we believe our method and proposed datasets will open new applications and encourage future works centered on stacks of 3D objects, including 3D reconstruction, counting, or 3D scene understanding.

6. Acknowledgements.

We would like to sincerely thank all the participants who spent time to play our counting game and provided us with valuable data on human performance at this task. In particular, we would like to congratulate Martin Engilberge, who gave the most accurate estimates and has been awarded a box of chocolates. We would also like to thank Nicolas Talabot, Víctor Batlle and Adriano D'Alessandro for insightful discussions. This work was supported in part by the Swiss National Science Foundation.

References

[1] C. Arteta., V. Lempitsky, and A. Zisserman. Counting in the Wild. In *European Conference on Computer Vision*, 2016. 1, 2

[2] P. Chattopadhyay, R. Vedantam, R.R. Selvaju, D. Batra, and D. Parikh. Counting Everyday Objects in Everyday Scenes. In *Conference on Computer Vision and Pattern Recognition*, 2017. 2

[3] Z. Cheng, J. Li, Q. Dai, X. Wu, and A. G. Hauptmann. Learning Spatial Awareness to Improve Crowd Counting. In *International Conference on Computer Vision*, 2019. 2

[4] T. Falk, D. Mai, R. Bensch, Özgün Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. D. Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox, and O. Ronneberger. U-Net: Deep Learning for Cell Counting, Detection, and Morphometry. *Nature Methods*, 16:67–70, 2018. 2

[5] P. Jenkins, K. Armstrong, S. Nelson, S. Gotad, J. S. Jenkins, W. W. Wilkey, and T. Watts. Countnet3D: A 3D Computer Vision Approach to Infer Counts of Occluded Objects. *IEEE Winter Conference on Applications of Computer Vision*, pages 3007–3016, 2023. 2

[6] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4), 2023. 4, 1

[7] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment Anything. In *arXiv Preprint*, 2023. 6, 8

[8] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo. ABC: A Big CAD Model Dataset for Geometric Deep Learning. In *Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019. 4, 5

[9] C. Liu, X. Weng, and Y. Mu. Recurrent Attentive Zooming for Joint Crowd Counting and Precise Localization. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[10] W. Liu, N. Durasov, and P. Fua. Leveraging Self-Supervision for Cross-Domain Crowd Counting. In *Conference on Computer Vision and Pattern Recognition*, 2022. 1

[11] E. Lu, W. Xie, and A. Zisserman. Class-Agnostic Counting. In *Asian Conference on Computer Vision*, 2018. 1, 2

[12] V. Mandal and Y. Adu-Gyamfi. Object Detection and Tracking Algorithms for Vehicle Counting: A Comparative Analysis. *Journal of Big Data Analytics in Transportation*, 2: 251–261, 2020. 1, 2

[13] A. K. Nellithimaru and G. A. Kantor. ROLS : Robust Object-Level SLAM for Grape Counting. In *Conference on Computer Vision and Pattern Recognition*, pages 2648–2656, 2019. 3

[14] T. Nguyen, C. Pham, K. Nguyen, and M. Hoai. Few-shot object counting and detection. In *ECCV*, pages 348–365, 2022. 2

[15] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P. Huang, H. Xu, V. Sharma, S. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning Robust Visual Features Without Supervision. In *arXiv Preprint*, 2023. 3

[16] A. Radford, J. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PMLR, 2021. 6, 8

[17] V. Ranjan, H. Le, and M. Hoai. Iterative Crowd Counting. In *European Conference on Computer Vision*, 2018. 2

[18] V. Ranjan, U. Sharma, T. Nguyen, and M. Hoai. Learning to Count Everything. In *Conference on Computer Vision and Pattern Recognition*, pages 3393–3402, 2021. 1, 2

[19] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. R"adle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. *ARXIV*, 2024. 3, 5

[20] J.L. Schönberger and J.M. Frahm. Structure-From-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition*, 2016. 4, 1

[21] M. Shi, H. Lu, C. Feng, C. Liu, and Z. Cao. Represent, compare, and learn: A similarity-aware framework for class-agnostic counting. In *CVPR*, 2022. 1, 6, 8

[22] Z. Shi, P. Mettes, and C. G. M. Snoek. Counting with Focus for Free. In *International Conference on Computer Vision*, 2019. 2

[23] V.A. Sindagi and V.M. Patel. Generating High-Quality Crowd Density Maps Using Contextual Pyramid CNNs. In *International Conference on Computer Vision*, pages 1879–1888, 2017. 2

[24] V.A. Sindagi and V.M. Patel. Multi-Level Bottom-Top and Top-Bottom Feature Fusion for Crowd Counting. In *International Conference on Computer Vision*, 2019. 2

[25] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *ACM SIGGRAPH*, 2023. 2, 3

[26] J. Wan and A. B. Chan. Adaptive Density Map Generation for Crowd Counting. In *International Conference on Computer Vision*, 2019. 2

[27] W. Xie, J. A. Noble, and A. Zisserman. Microscopy cell counting and detection with fully convolutional regression

networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 6:283 – 292, 2018. 1

[28] C. Xu, K. Qiu, J. Fu, S. Bai, Y. Xu, and X. Bai. Learn to Scale: Generating Multipolar Normalized Density Maps for Crowd Counting. In *International Conference on Computer Vision*, 2019. 2

[29] J. Xu, H. Le, V. Nguyen, V. Ranjan, and D. Samaras. Zero-Shot Object Counting. In *Conference on Computer Vision and Pattern Recognition*, pages 15548–15557, 2023. 1, 2

[30] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao. Depth anything v2. *ARXIV*, 2024. 4, 7, 3

[31] S. Yang, H.-T. Su, W. H. Hsu, and W.-C. Chen. Class-Agnostic Few-Shot Object Counting. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 869–877, 2021. 2

[32] V. Ye, R. Li, J. Kerr, M. Turkulainen, B. Yi, Z. Pan, O. Seiskari, J. Ye, J. Hu, M. Tancik, and A. Kanazawa. gsplat: An open-source library for gaussian splatting. *ARXIV*, 2024. 3

[33] J. Yi, Z. Shen, F. Chen, Y. Zhao, S. Xiao, and W. Zhou. A Lightweight Multiscale Feature Fusion Network for Remote Sensing Object Counting. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–13, 2023. 2

[34] F. Yu and K. Vladlen. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016. 5

[35] A. Zhang, J. Shen, Z. Xiao, F. Zhu, X. Zhen, X. Cao, and L. Shao. Relational Attention Network for Crowd Counting. In *International Conference on Computer Vision*, 2019. 2

[36] A. Zhang, L. Yue, J. Shen, F. Zhu, X. Zhen, X. Cao, and L. Shao. Attentional Neural Fields for Crowd Counting. In *International Conference on Computer Vision*, 2019. 2

[37] Q. Zhang and A.B. Chan. 3D Crowd Counting via Multi-View Fusion with 3D Gaussian Kernels. In *AAAI Conference on Artificial Intelligence*, 2020. 3

[38] Q. Zhang and A. B. Chan. Wide-Area Crowd Counting: Multi-View Fusion Networks for Counting in Large Scenes. *International Journal of Computer Vision*, 130:1938–1960, 2020. 3

[39] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. In *Conference on Computer Vision and Pattern Recognition*, pages 589–597, 2016. 2

Counting Stacked Objects

Supplementary Material

7. Dataset Details

7.1. 3DC-Real dataset.

We capture 45 real scenes where the objects to count can be any stack of items that are at least partially visible. This includes stacked objects on a table or on the floor, objects in containers such as bowls or boxes, or objects still in their packaging.

Cameras. We use a regular RGB smartphone camera to capture 30-60 pictures of the scene from various angles, forming a semisphere surrounding the objects and their container. These images are downscaled to approximately 600 pixels wide to reduce memory usage and facilitate the processing with COLMAP [20]. Additionally, we take a measurement of an arbitrary object within the scene, allowing us to scale the camera measurements and align the unit distance of the scene with a meter in the real world.

Initially, we experimented with triangulation methods using two pairs of corresponding points across images. This would enable the calculation of a 3D distance and allow us to scale the scene. However, this approach proved to be unstable, as small inaccuracies in point matching led to significant variations in the scaling. Instead, we reconstructed the 3D scene with 3DGS [6] and measured the 3D distance directly within the reconstruction. This measurement allows us to rescale the scene to match the reference measurement. Note that the 3D point cloud generated by COLMAP [20], which is used as an initialization by 3DGS, is also scaled accordingly.

Unit volume. For each scene, we require the unit volume of the object being counted. For complex shapes, we determine the unit volume v by 3D reconstruction, similar to the method described in our main paper. For many common food items, such as kidney beans or corn, this information is readily available online. For other scenes, the volume can be approximated, for example in the case of the beads in Fig. 1, by subtracting the volume of a cylinder from that of a sphere.

Pre and Post-processing. Using this method, we capture 45 scenes consisting of various items in different environments. The scenes vary in complexity, from simple quasi-spherical objects in containers to more challenging configurations, such as complicated shapes still in packaging (e.g., in the *pasta* scene). Some items are used in several scenes, in which case the container and location are modified to create a new setting.

7.2. 3DC-Synthetic Dataset.

To generate our large-scale synthetic dataset, we utilize Blender, a free and open-source 3D creation suite that supports Python scripting. This allows us to implement a fully automated generation pipeline, which is mainly composed of two steps: simulation and rendering.

Simulation. We drop batches of objects, arranged in a $4 \times 4 \times 5$ grid, into a box positioned at $(0, 0, 0.5)$. The box has a side length of 1 and a thickness of 0.04. Once the simulation converges, we check if the union of the objects intersects with an invisible cube placed directly on top of the box. If an intersection occurs, the simulation stops, and objects outside the box are deleted. If no intersection is detected, a new batch of objects is added, and the simulation is performed again.

We use the convex hull to compute collision between objects. Ideally, we would use the triangle mesh itself, however this becomes far too costly when physically simulating thousands of shapes with tens of thousands of triangles. We experimented with using convex hulls first, and then refining with additional frames using the triangle mesh, but this turned out to still be extremely costly and computationally very unstable, leading to objects being ejected outside the box due to the change in collision computation.

Rendering. For rendering, we use a texture randomly sampled from 3 possibilities for the box, five textures for the ground, and a random material for each model chosen from one of the following: a realistic grey metal texture, a red metallic texture, or a plastic material with a randomly selected color.

We always render the first view directly above the box, looking downwards, which we call the *nadir* view. For the validation dataset, we also generate 29 additional views on the unit sphere, each observing the box from different angles. The rendering is performed using Blender's Cycles rendering engine. Additionally, we generate ground-truth depth maps and masks that separate the ground, box, and objects in the images.

Pre and Post-processing. In addition to the simulation and rendering steps, we perform pre-processing to filter out unsuitable meshes, such as those with multiple connected components or excessive size. Since the physical simulation can sometimes be unstable or fail, we also remove a small fraction of results in post-processing. This includes cases where the unit volume is too small or where too few objects remain in the box in the final frame.

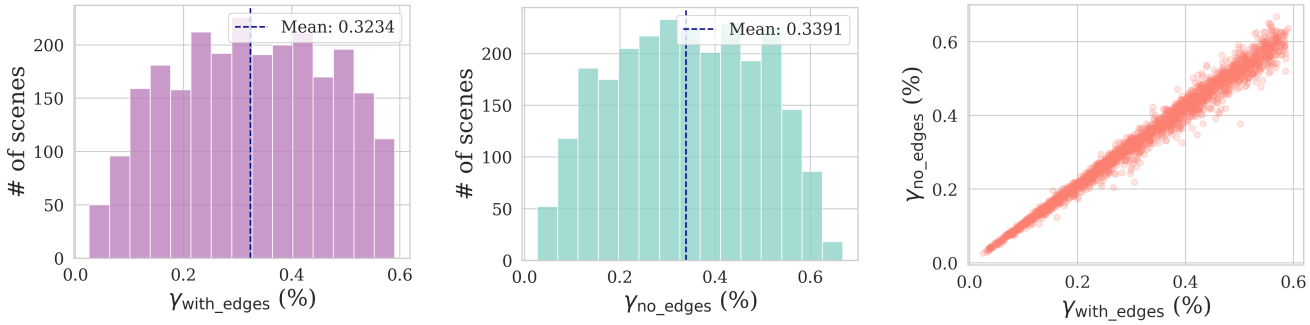


Figure 10. **Border effects.** Measuring the ground-truth γ over the complete box or over only a smaller section makes little difference, indicating that our assumption of uniform γ across the volume is justified.

Finally, we export the calibrated camera parameters in a format compatible with nerfstudio [25]. Since these cameras are not produced by COLMAP, they do not include a 3D point cloud that 3DGS can use as initialization. This poses a challenge, as a fully random initialization may generate distant Gaussian points outside the cameras' range, which are not removed and interfere with the volume estimation. To address this, we generate a set of 100 grey points within the unit cube, centered at $(0, 0, 0.5)$. This simple initialization proves sufficient to quickly produce a faithful 3D reconstruction and resolves the aforementioned issue.

8. Additional comparison distinguishing visible and invisible objects.

Our method is, to the best of our knowledge, proposing the first solution for this task. The methods used as comparison in 4 are initially designed to count only visible objects, and perform poorly on our dataset. In this section, we distinguish between visible and invisible objects in stacks and evaluate baseline methods against both counts. To perform this evaluation, we manually annotate the locations of visible objects in real scenes and provide the visible count and locations in our released datasets. Two examples are displayed in Fig. 11.

The results of this experiment can be found in Tab. 6. While the numbers for the 2D counting methods improve, it remains hard for them to distinguish similar objects clumped together. As a result, their performance re,ains much lower on these challenging scenes than on traditional 2D counting benchmarks.

9. Evaluation metrics

In this section, we provide the exact formula behind the metrics used in Sec. 4. As explained, the NAE and SRE are defined as:

$$\text{NAE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} y_i}, \quad \text{SRE} = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} y_i^2}.$$

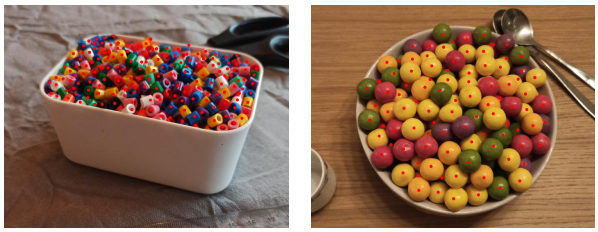


Figure 11. **Localization of visible objects.** Left: $\mathcal{N}_{vis} = 168$, Right: $\mathcal{N}_{vis} = 96$

	NAE ↓	SRE ↓	sMAPE ↓
Visible objects only			
BMNet+ [21]	**0.51**	**0.65**	51.28
SAM+CLIP [7,16]	0.57	0.81	**50.84**
All objects			
BMNet+ [21]	0.93	0.98	131.44
SAM+CLIP [7,16]	0.94	0.99	124.31
Ours	**0.36**	**0.06**	**53.31**

Table 6. **Counting visible and invisble objects separately.**

We also report the Symmetric Mean Absolute Percentage Error (sMAPE), which can be considered a normalized percentage error. It is expressed as follows:

$$\text{sMAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{(y_i + \hat{y}_i)/2},$$

The formula of sMAPE ensures that errors are scaled symmetrically between the prediction and ground truth counts. Finally, the coefficient of determination, R^2, measures the proportion of variance in the ground-truth occupancy ratio γ explained by our predictions

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2},$$

where $\overline{y}$ is the mean of the ground truth counts. High values of R^2 indicate strong agreement between predictions and ground-truth values.

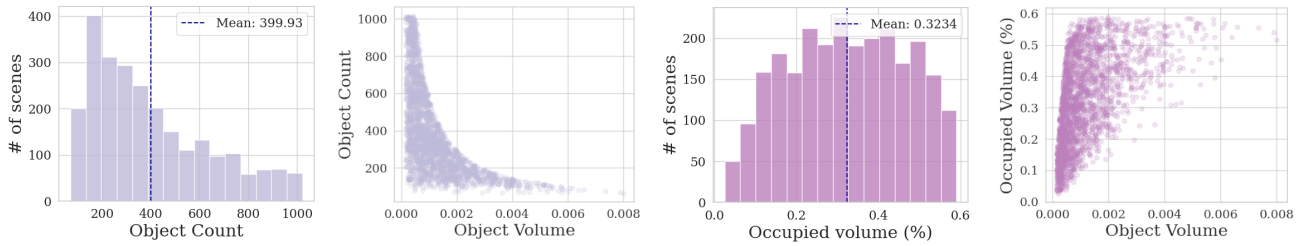


Figure 12. **Dataset statistics.** The histograms represent the distributions of object count and occupancy ratio, respectively, and each bar plots the number of scenes in a given bin. In scatter plots, each point represents a physically simulated 3D scene. In particular, the occupancy ratio γ spans a large range between 1% and 65%

10. Additional Discussion on Border Effects

In our work, we assume that the occupancy ratio, γ, is approximately uniform throughout the container. This assumption generally holds as the number of stacked objects increases. However, it neglects the influence of container borders, where objects tend to occupy less volume due to the boundary.

To quantitatively evaluate the impact of border effects and verify the validity of our uniform γ assumption, we analyze the ground-truth volume ratio in two distinct ways using our large-scale synthetic dataset. First, we compute $\gamma_{\text{with_edges}}$ for the entire unit box, as described in the main paper. Additionally, we compute $\gamma_{\text{no_edges}}$ by measuring the volume ratio in a smaller sub-box of side length 0.5, centered within the unit box. Intuitively, the difference between $\gamma_{\text{with_edges}}$ and $\gamma_{\text{no_edges}}$ reflects the influence of border effects, allowing us to evaluate whether this assumption is justifiable.

Figure 10 presents two histograms comparing the distributions of $\gamma_{\text{with_edges}}$ and $\gamma_{\text{no_edges}}$. The results indicate that both metrics follow highly similar distributions, with their mean values differing by less than 5%. Notably, the mean value of $\gamma_{\text{no_edges}}$ is slightly higher than that of $\gamma_{\text{with_edges}}$, consistent with the intuition that density decreases near borders.

To further investigate the relationship between these two values, we provide a scatter plot of $\gamma_{\text{with_edges}}$ versus $\gamma_{\text{no_edges}}$ in Fig. 10. The plot demonstrates a strong correlation between the two measures, particularly for objects with small volume ratios. For objects with high values of both $\gamma_{\text{with_edges}}$ and $\gamma_{\text{no_edges}}$, minor discrepancies are observed. These differences can be attributed to the relatively large size of these objects compared to the measurement box, which introduces noise in the estimation of γ.

Overall, these analyses confirm that $\gamma_{\text{with_edges}}$ and $\gamma_{\text{no_edges}}$ are highly consistent and can be used interchangeably without significant loss of accuracy. In our experiments, we rely on $\gamma_{\text{with_edges}}$ to train our occupancy ratio estimation network.

11. Implementation details

We use the nerfstudio library [25] for 3D reconstruction, specifically the *splatfacto* method built on top of the gsplat library [32]. We thank the contributors of all the aforementioned libraries.

Our pipeline also uses pretrained models for depth estimation and mask generation. We employ the *vitl* model from Depth Anything v2 [30] for depth estimation and the *sam2.1_hiera_large* model from SAM2 [19] for mask generation. These state-of-the-art models ensure high-quality and robust outputs across diverse scenes.

The dataset is generated using CPUs only, greatly reducing its production cost and environmental impact. Other operations are fairly light and performed locally on a 4080 Mobile GPU, taking up only a few gigabytes of VRAM and being completed in a couple minutes.

12. Architecture details

Our architecture utilizes a DinoV2 [15] encoder model that produces pixel-aligned features. Since DinoV2 downscales the input image by 14, we feed it an image of size 448 x 448 to produce a 32 x 32 x 768 feature image. Specifically, we use the pretrained weights of the *dinov2_vitb14* model and freeze them during all subsequent learning.

To predict a scalar value from the $32 \times 32 \times 768$ feature image produced by DinoV2, we employ a series of convolutional layers to progressively reduce both the spatial dimensions and the number of channels. The convolutional layers successively reduce the channel dimension from the initial 768 down to 512, 256, 128, and finally 64. Concurrently, the spatial dimensions of the feature map are reduced from 32×32 to 16×16, 8×8, 4×4, and ultimately 2×2.

Following this, an adaptive average pooling layer compresses the spatial dimensions to a single pixel while preserving the 64-channel depth. The resulting $1 \times 1 \times 64$ tensor is passed through a fully connected linear layer to map it to a scalar output. Finally, a sigmoid activation function is applied to produce the final prediction in the $[0, 1]$ range.