Leibniz Universität Hannover

www.martinwerner.de

Martin Werner

Deep Learning

with applications to point clouds

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung

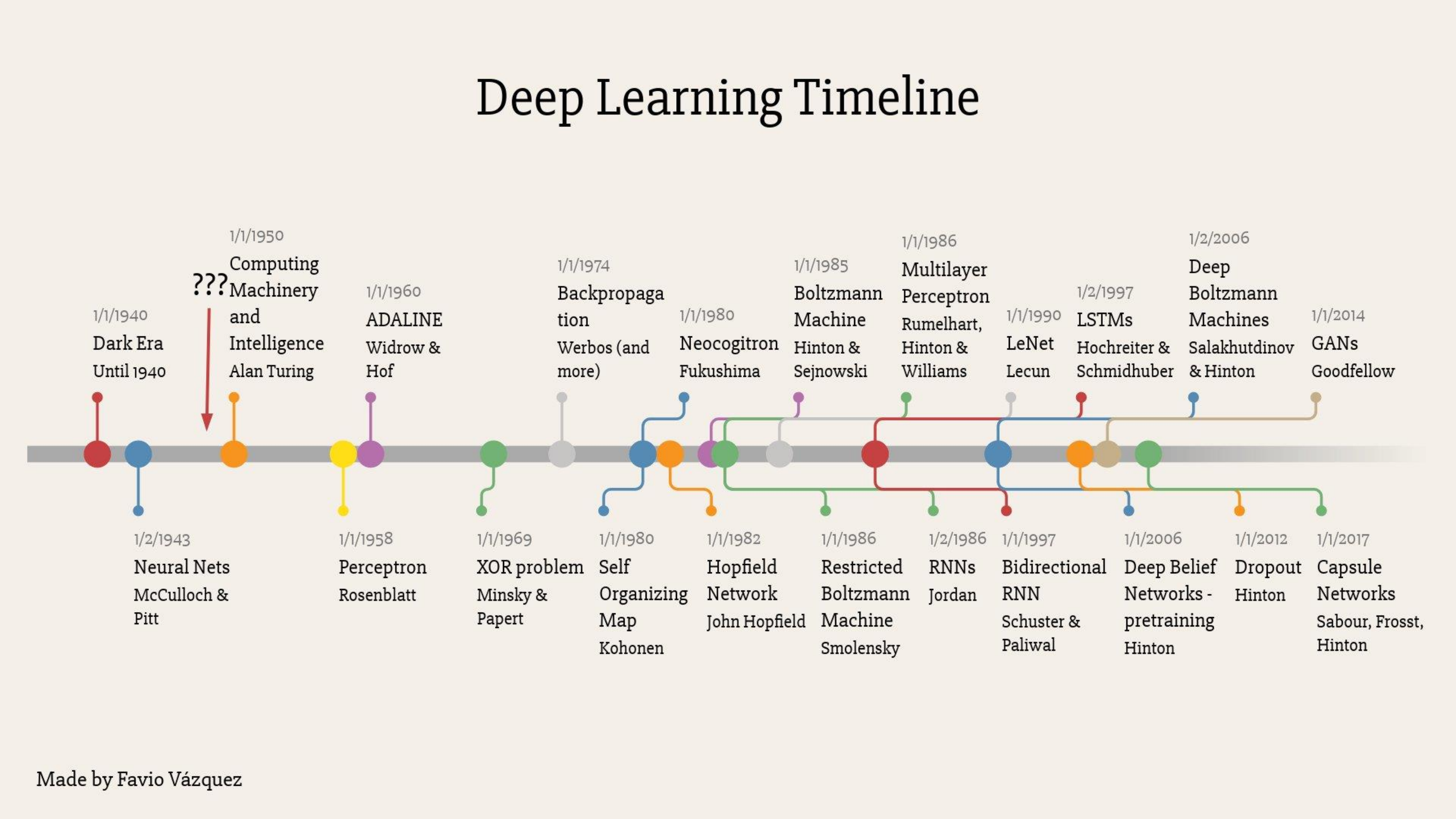
Outline

- A Short History of Deep Learning
- Deep Learning Elements
  - Neurons
  - Neural Networks
  - Back Propagation and Gradient Descent
- Some Basic Deep Learning Architectures
- Dealing with Point Clouds
- And now? How would I?

Martin Werner

A Short History of Deep Learning

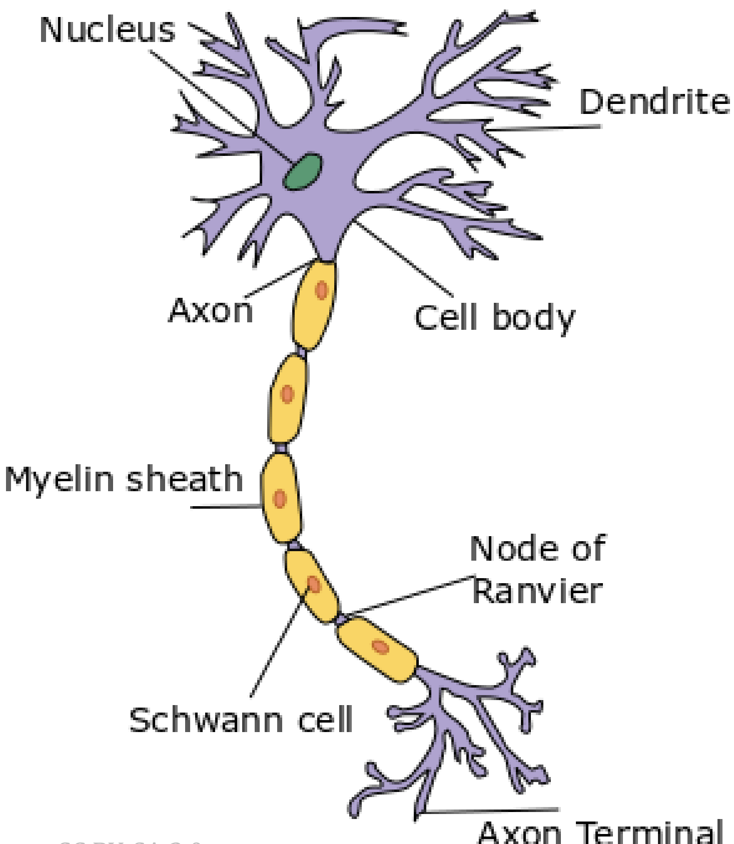
History of Deep Learning

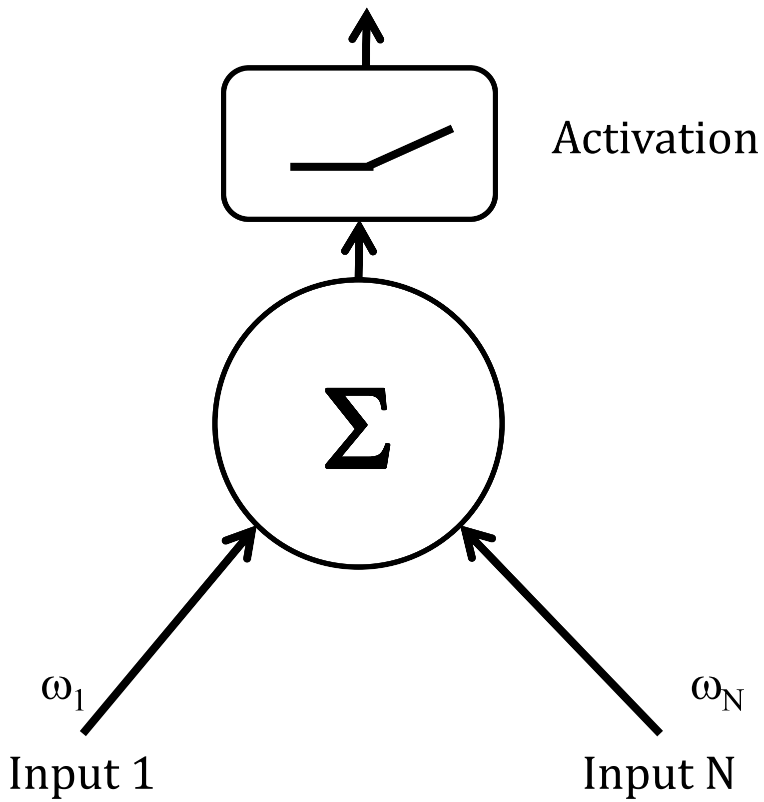


- Nothing of this is really new. It is an old and established discipline.
- The current hype comes from **several factors**
  - Advances in computational performances (GPUs, TPUs)
  - Creation of Huge Datasets
  - (Smaller) Advances in Stochastic Gradient Decent
  - Novel Ideas about Regularization
  - Novel Ideas for Capacity (Weight) Reduction
    - Convolutional Neural Networks
- But, Deep Learning is **not very powerful per se**:
  - Energy Consumption
  - Dataset Creation Cost
  - Performance of the Deployed System
  - Understandability and Certification of Systems

Deep Learning Elements

Neurons and Neural Networks


Nucleus
Dendrite
Axon
Cell body
Myelin sheath
Node of Ranvier
Schwann cell
Axon Terminal


Activation
Σ
ω1
ωN
Input 1
Input N

The simplest Neuron is a linear one.

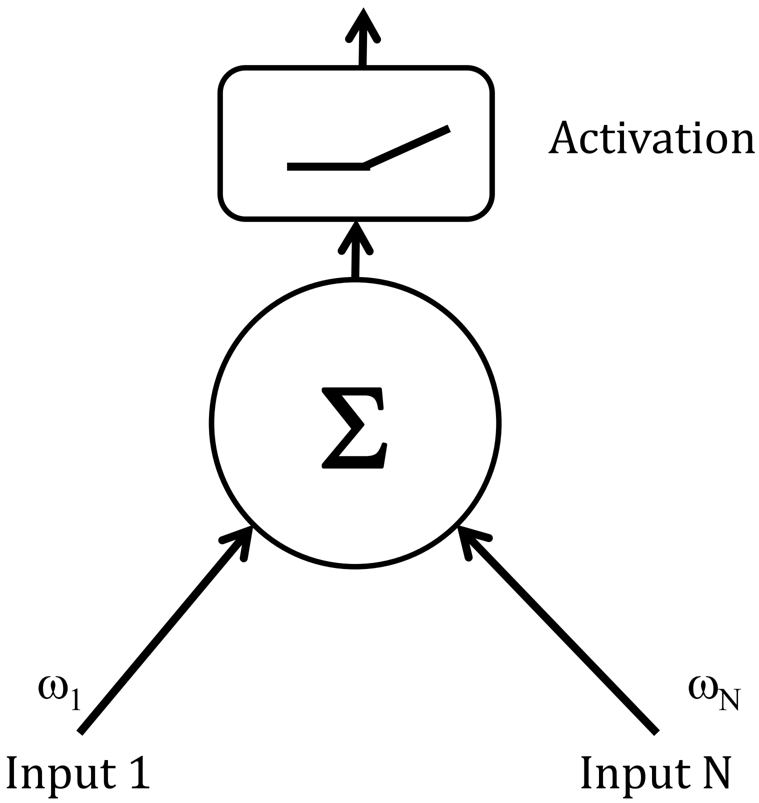
This means

- Activation Function is linear
- A bias term is added
- Then, we can write the output as

$$\sum_{i=1..N} \omega_i x_i + b$$

- For simplicity, the bias is often made an artifical input to the system such that it reads even simpler (w_0 = 1, x_0 = b)

$$\sum_{i=0..N} \omega_i x_i$$



- Lets assume two inputs to the neuron and f(x)=y the activation function.
- Question: What can we represent in this way:
- Answer: Lets calculate a bit (with explicit bias)

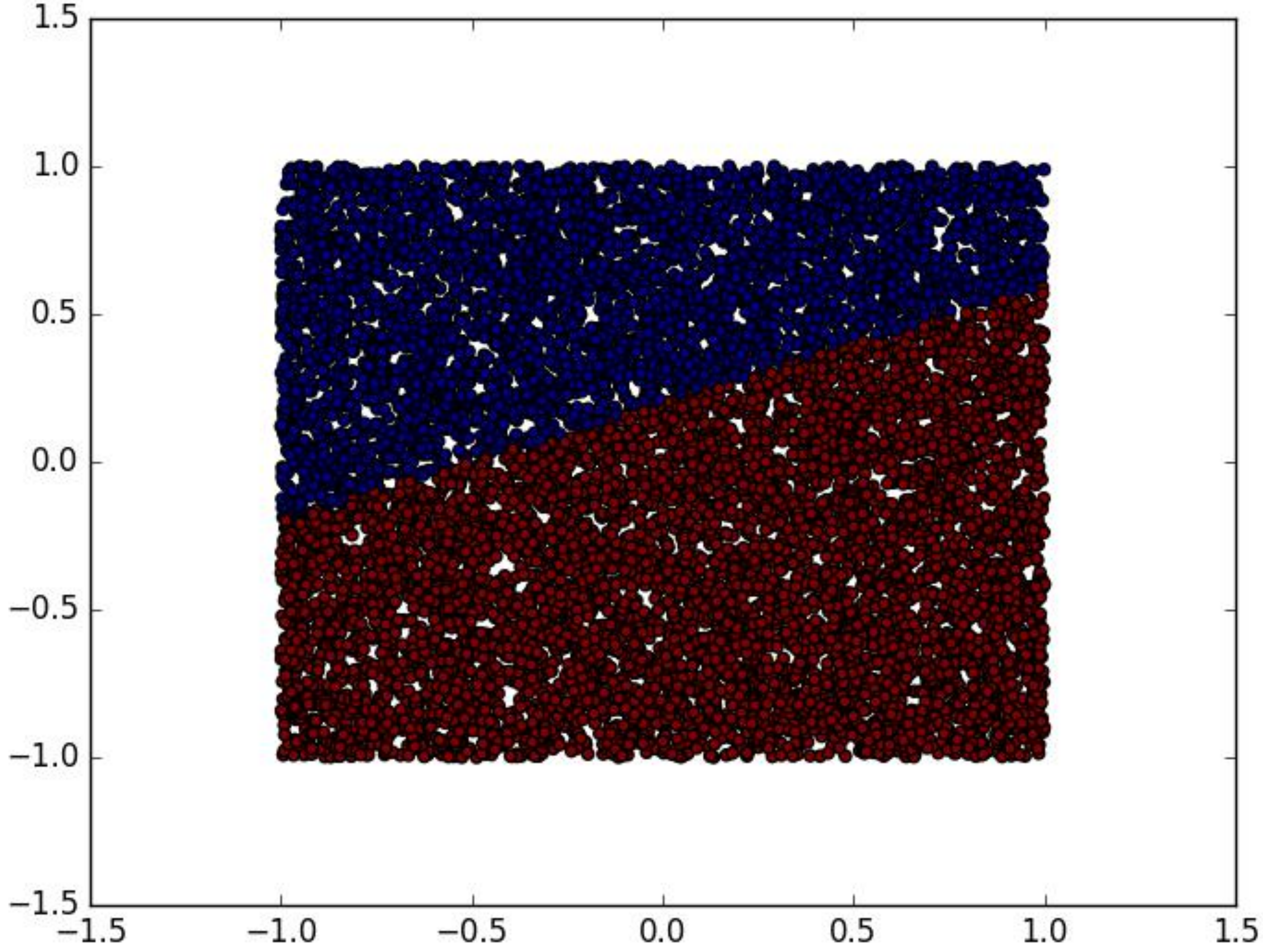
$$\sum_i \omega_i + b = \omega_1 x_1 + \omega_2 x_2 + b$$

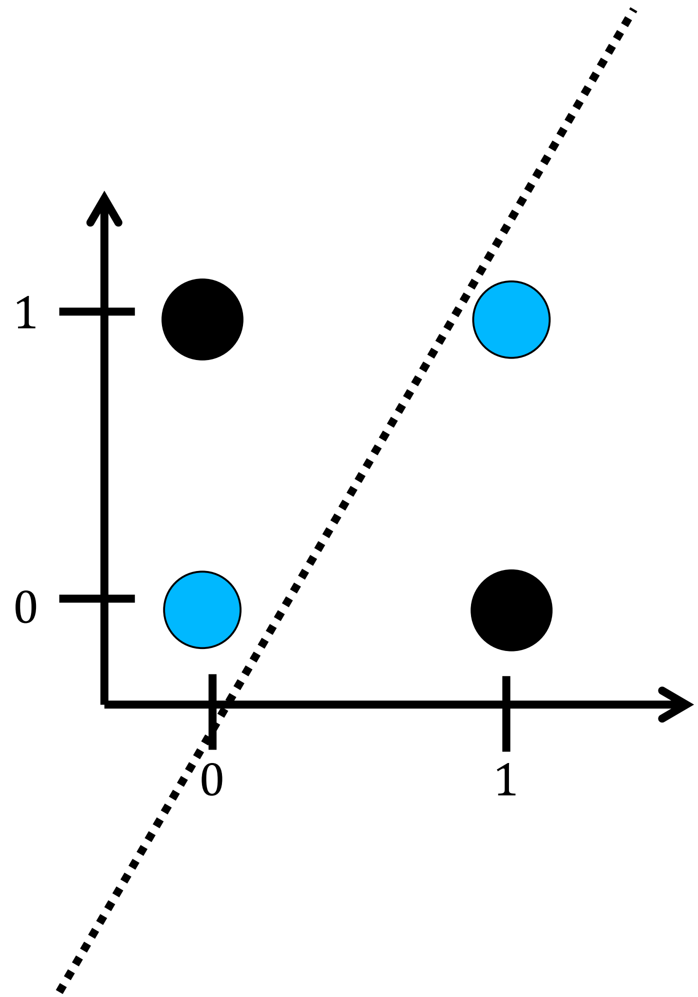
- Now, for binary classification, we need a simple decision rule. What about (output > 0)
- Then, we can learn sets that have the structure

$$\sum_i \omega_i + b = \omega_1 x_1 + \omega_2 x_2 + b \geq 0$$

- This is easily seen to be a split along a line in space. Lets do this

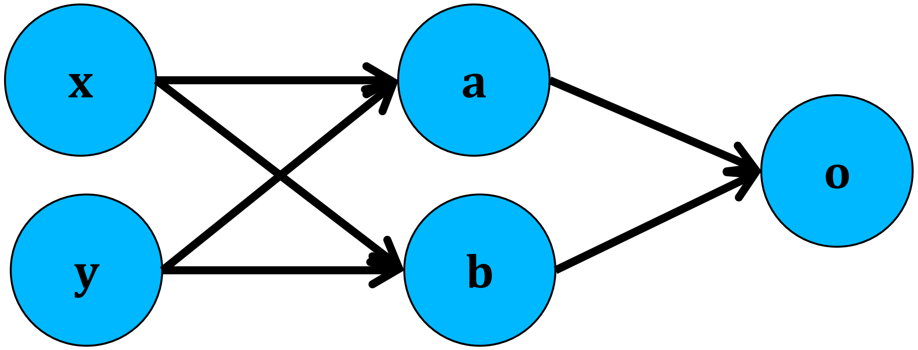
This is a typical linear neuron decision





There is no line that separates the two colors!

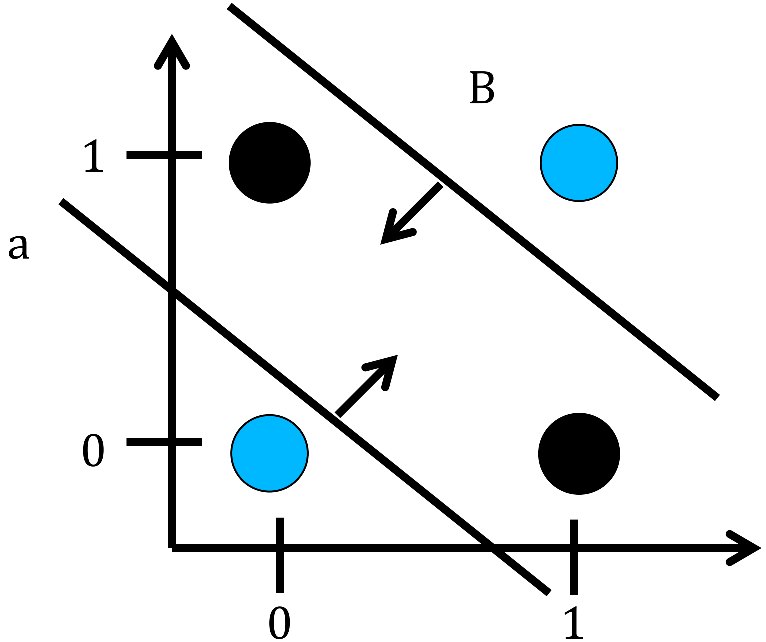
This architecture has an bias term for all hidden nodes (a) and the output node which is hidden.



That is, there are nine weights!

Each of the early neurons decides

a) Above the line
b) Below the line



The last neuron calculates A AND B, which is easily possible !

Assignment: Find a set of weights for the network to model XOR

Now, for a long time, no real progress was made. People got frustrated, left the field. The frustration points were:

- Finding optimal weights is NP-complete – exponential runtime
- While solving XOR is possible with a MLP, it is impossible to train, because the expected output of the inner connections is unknown.
- Many people turned away from this part of machine learning
- Dates are difficult to assign as related machine learning techinques are still evolving:
  - Starts about the time that the implications of the unsolvability of XOR for general intelligence become clear
    - **Challenge Problem has been identified: train MLP**
  - Ends about the time where multilayer perceptrons are successfully trained
    - **Challenge Problem has been fully solved without avoiding it.**

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight

- Where do the weights come from?
  - Finding the optimal weights is **NP-complete** (that is, as hard as the TSP; Blum and Rivest, 1992)
  - Fortunately, we can find **a sufficient set of weights** through back propagation (e.g., Rumelhart et al. (1985))
- First, we compare the output of a forward pass with the expected value.
- Then, we slightly adjust each of the weights backwards in the network by a very small amount.
- We do this over and over again (training)
- We do so, because the error function we chose is differentiable and sufficiently smooth such that the local direction of error reduction is sensible globally (which need not be the case)

- **Forward Pass**
  - All units within a layer have their values set in parallel
  - Next layer only after first layer has completely been computed
- **Layer Function needs to**
  - Have bounded derivative only
  - However, linear aggregation of the input before applying **one** non-linear function simplifies learning procedure
- **Total Error Function**
  - $E = \frac{1}{2} \sum_c \sum_j \left(y_{j,c} - d_{j,c}\right)^2$
- **Idea: Use Gradient Decent** of this with partial derivatives with respect to each and every weight.

Let us fix a single case c. Then

- $\frac{\partial E}{\partial y_j} = y_j - d_j$

Now, let x_j denote the activity of a unit in the forward pass. Then use the chain rule

- $\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j}$

Now, with an activity function of $y_j = \frac{1}{1+e^{-x_j}}$ we can calculate and substitude the second factor:

- $\frac{\partial E}{\partial x_j} == \frac{\partial E}{\partial y_j} \cdot y_j(1 - y_j)$
- This means, we know how the total input of node x_j changes the total error for this case. But as the total input is a linear sum of the inputs, we can compute

- $\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \cdot y_i$
- Und analog dazu können wir auch diese Ableitung für y ausrechnen:
- $\frac{\partial E}{\partial y_i} = \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial y_i} = \frac{\partial E}{\partial x_j} \cdot \mathrm{w_{ji}}$
- Now, we have seen how to calculate $\frac{\partial E}{\partial y}$ for any unit in the penultimate layer when given information $\frac{\partial E}{\partial y}$ from the last layer
- This can be iterated backwards such that the derivatives $\frac{\partial E}{\partial w_{ji}}$ become known along the way.
- These are used for (stochastic) gradient descent!

- It is a very good idea to spell out this for the XOR problem. You can follow the following article (using different names than here)
  - https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c

- One way of thinking about back-propagation is that it is a major factorization of the derivative into things that we can calculate as numbers!

$$\frac{dE}{dw} = \frac{dE}{dy} \cdot \frac{dy}{dx} \cdot \frac{dx}{dw}$$

- **Classical Networks**
  - Input, a few hidden layers, an output
  - Difficulty: expressivity (number of layers) vs. trainability (number of parameters)
- **Convolutional Neural Networks and Pooling**
  - Input an image, Layers are now calculating some local convolution of the image and dimensionality is reduced by pooling, that is taking only a subset of the data points.
  - Less Weights (only once for the convolution kernel which is swiped over the image, not for every pixel)
- **Recurrent Networks**
  - They can have loops. That is the output of a layer serves as the input of a previous layer. Sequences are typical examples, the network can remember (learn to remember)

- Now, Backpropagation can train deep networks and, therefore, XOR,but
  - Not enough processing power (no GPUs, for example)
  - Lack of Datasets (big and annotated datasets, because in real-world scenarios you would need those)
  - Overfitting (mainly, because you need to choose a sufficiently expressive architecture but don‘t have enough data to train)
  - Vanishing Gradient Problem
    - During learning, you multiply a lot of very small numbers which eventually get too small for sensible learning on finite accuracy machines
- People turned away, because practical examples of deep networks were not brought to significant success, especially as other techniques became very powerful including support vector machines

- Training tricks
- ImageNet Dataset (2009, 16 million annotated images)
- Visibility through ILSVRC (1 million images, 1,000 classes)

2013: AlexNet trained on ImageNet using two GPUs

- Dropout
- Rectified Linear Units (ReLU) instead of sigmoid or tanh activations
- Data Augmentation

In computer vision

www.martinwerner.de

- Errors drop significantly year by year
- Architectures get deeper and deeper
  - Trainable with tricks
- Some results from the golden years of CNNs follow

ILSVC over the early years

CNN based, non-CNN based

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

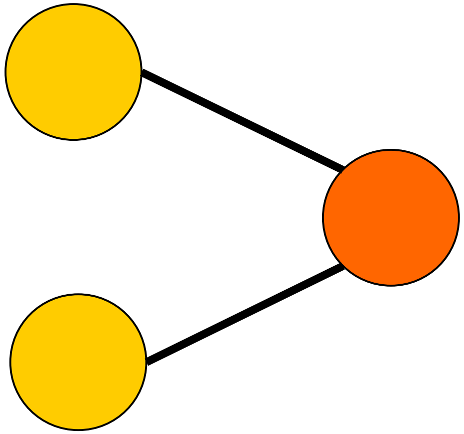
2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

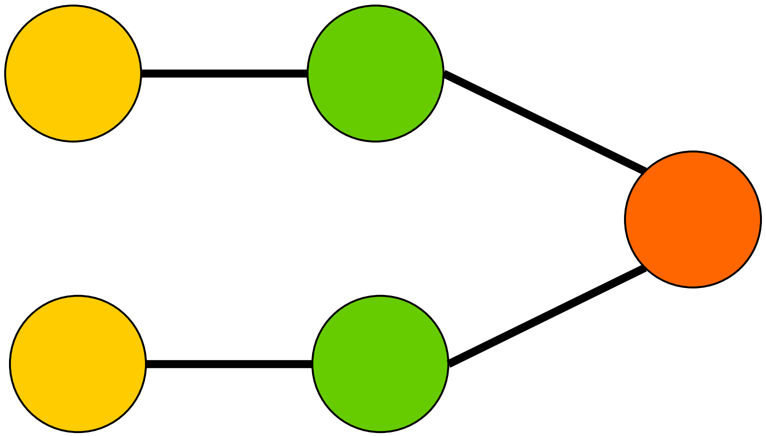
- In 2015, Microsoft Research Asia won with a 150 layer network
  - Almost superhuman performance (3.5 % error, later even improved)
- GoogLeNet 2014 had 22 layers
- Is the next AI winter just around the corner?
  - We have been successful in image regognition, speech, and translation.
  - But we rely on excessive datasets that we cannot generate
  - By abuse of language (AI vs. ML) also termed **„narrow AI“**

Some Basic Deep Learning Architectures

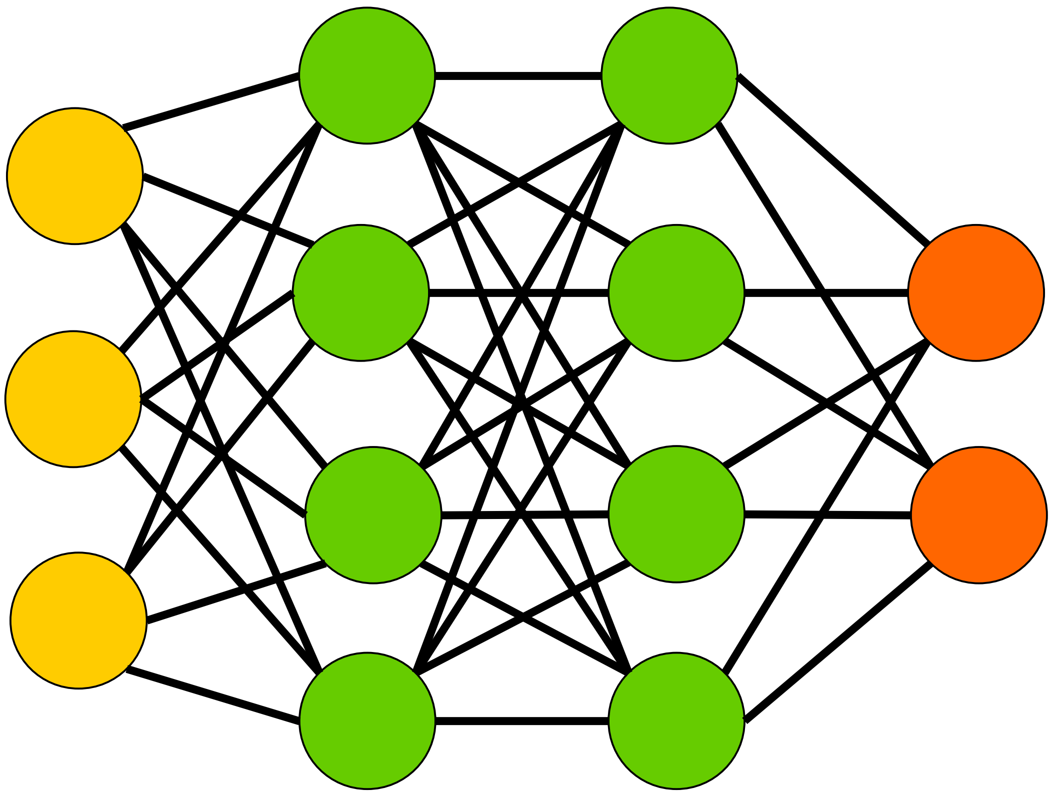
Perceptron (P)



Feed Forward (FF)

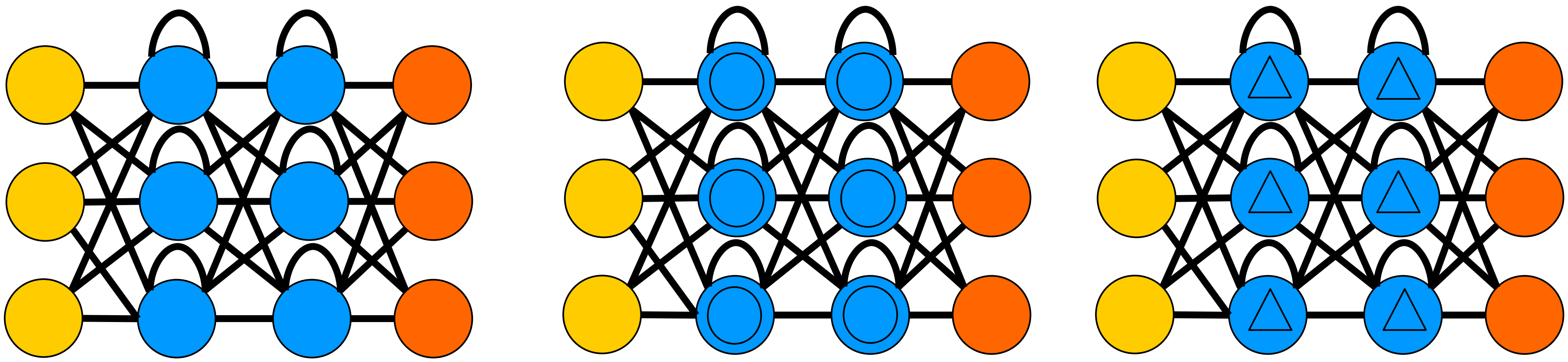


Deep Feed Forward (DFF)

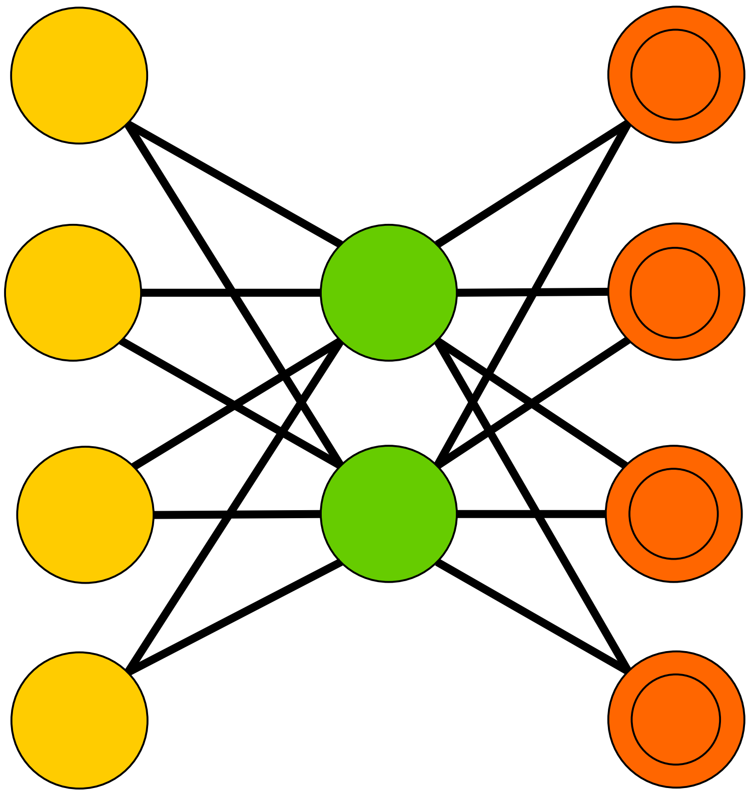


Architectures

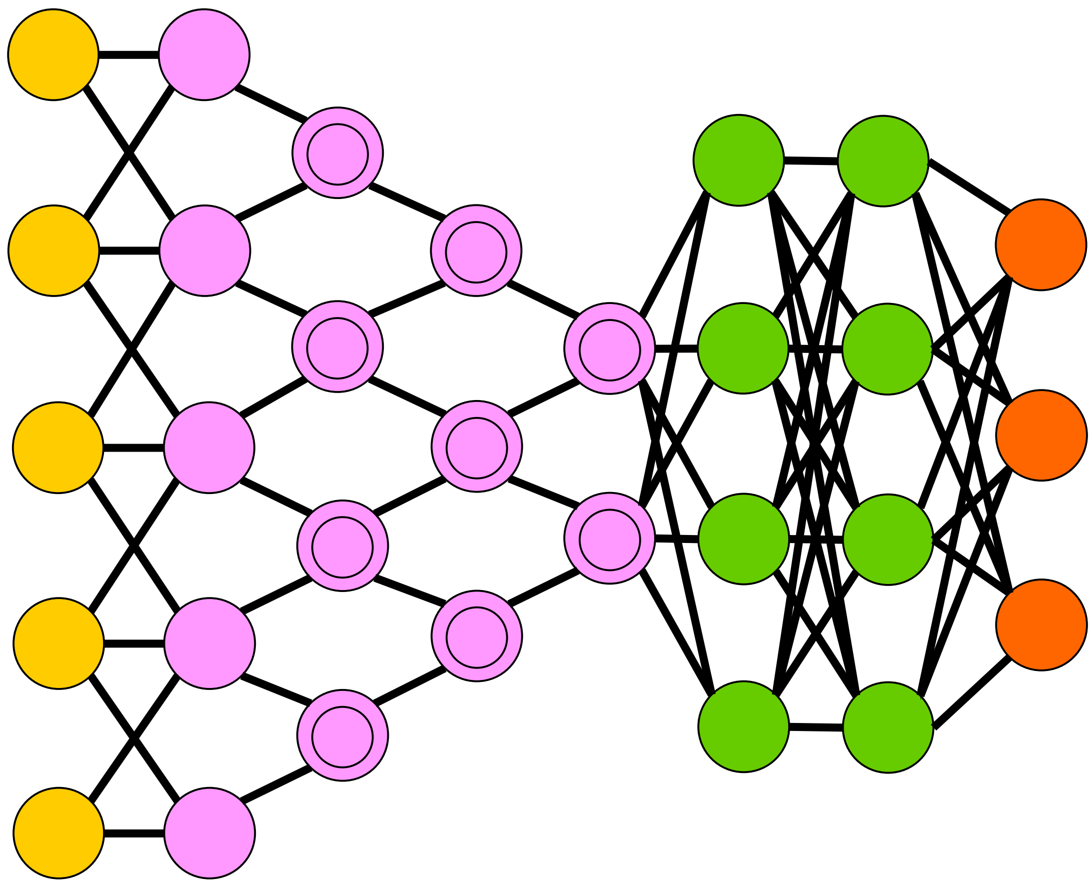
Recurrent Neural Network (RNN)	Long / Short Term Memory (LSTM)	Gated Recurrent Unit (GRU)



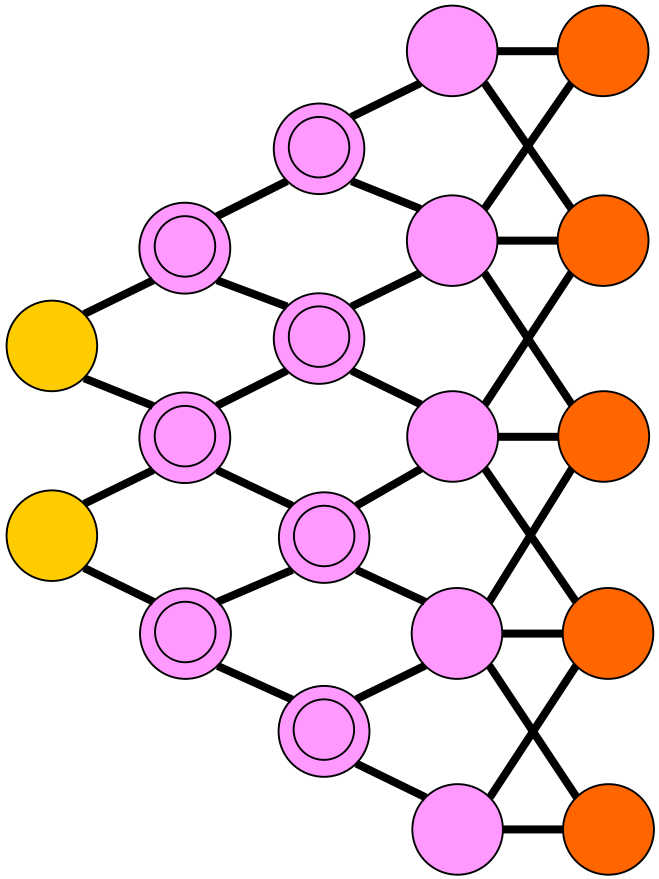
Auto Encoder (AE)



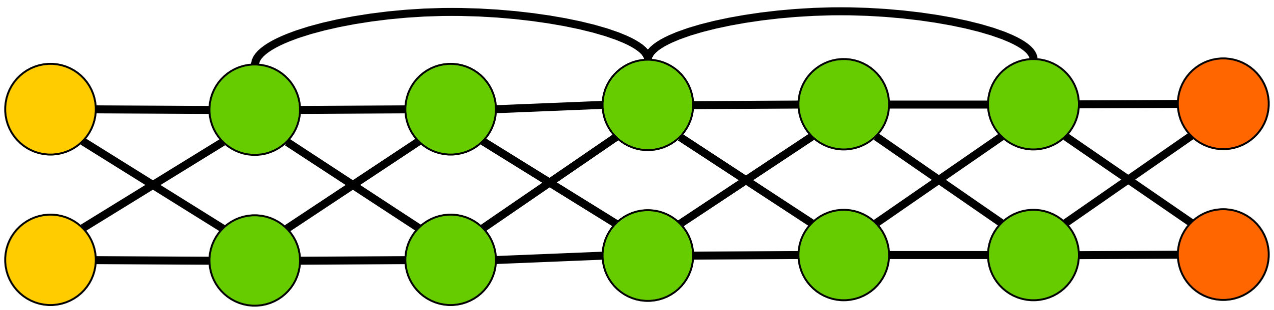
Deep Convolutional Network (CNN)



Deconvolutional Network (DN)



Deep Residual Network (DRN)

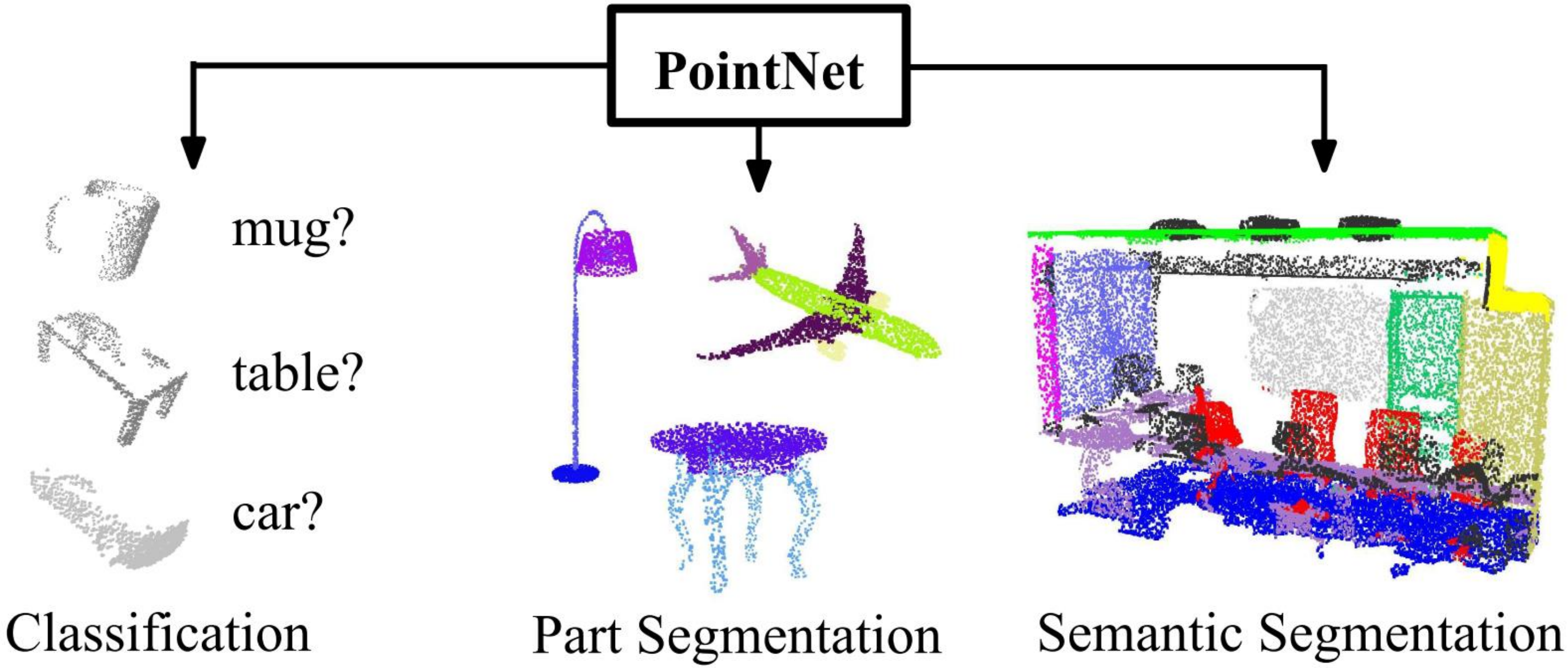


Dealing with Point Clouds

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

Charles R. Qi* Hao Su* Kaichun Mo Leonidas J. Guibas
Stanford University

Why?


PointNet
mug?
table?
car?
Classification
Part Segmentation
Semantic Segmentation

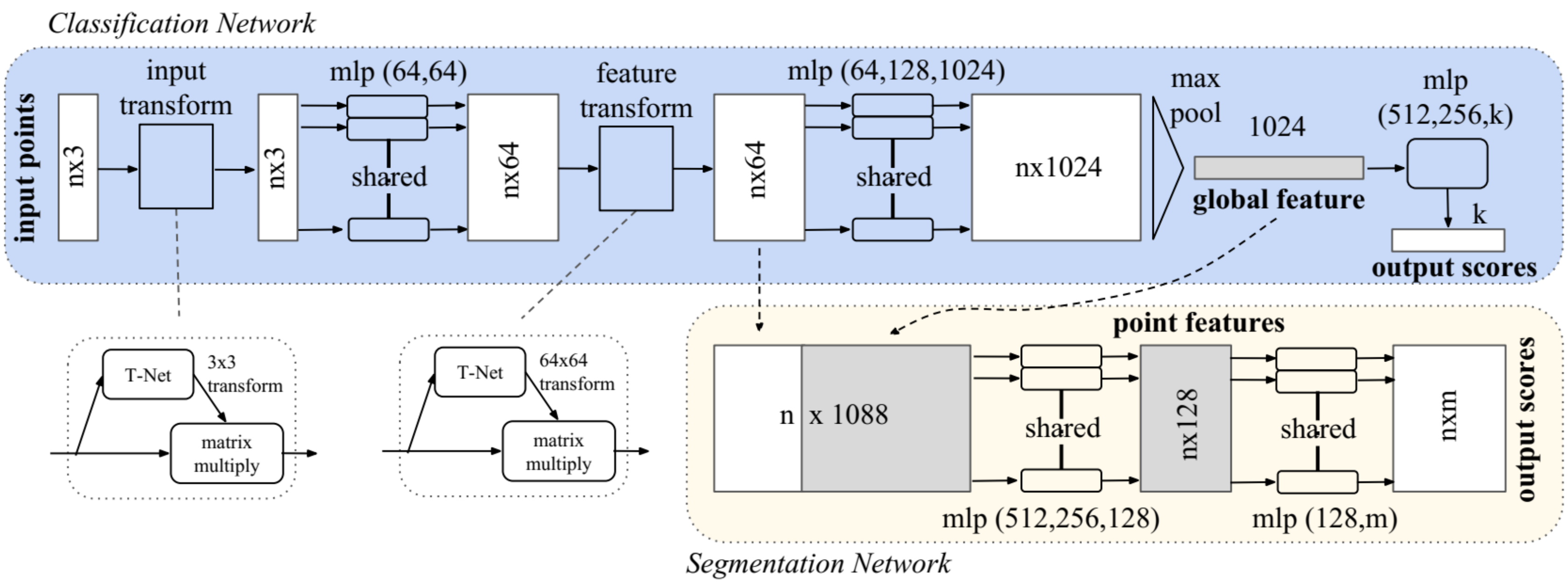
- **Extract hand-crafted features** (e.g., structure tensor + friends)
  - Should be invariant for certain transformations
  - Can be global or local
  - Usually need a context definition (for pure 3D points)
  - Including Deep Feed-Forward Architectures!
- **Volumetric CNNs**
  - Step towards a voxelgrid and use (learned) 3D convolutions
- **Multiview CNNs**
  - Render several perspective views of the point clouds and feed them to a CNN
  - Limited to aspects represented by 2D aspects (e.g., classification, but not completion)

- Point Clouds are **Unordered Collections of Points**
  - and there is no sensible ordering function

Model Functionalities Needed

- Classification outputs a score for each candidate class
- For Scene Understanding / Segmentation, the model outputs scores for each point and each candidate class

A first glance at the architecture



- Based on three main properties, assertions and their consequences
  - **The order of the points shall not matter**
  - **Nearby things shall be able to interact with each other**
  - The system should become **invariant under rigid transformation** including rotation, translation, and flip

- To make a model invariant under the order of input points can be done basically in three ways:
  - **Sort input into a canonical order**,
    - However, no order exists that preserves data locality completely
  - **Treat the input as a sequence** and train with *all permutations* of the input
    - However, it has been shown that order matters still.
    - Excessive training times (There are n! permutations)
  - Use a **simple, symmetric function** to aggregate information from each point
    - Okay, lets go for it…

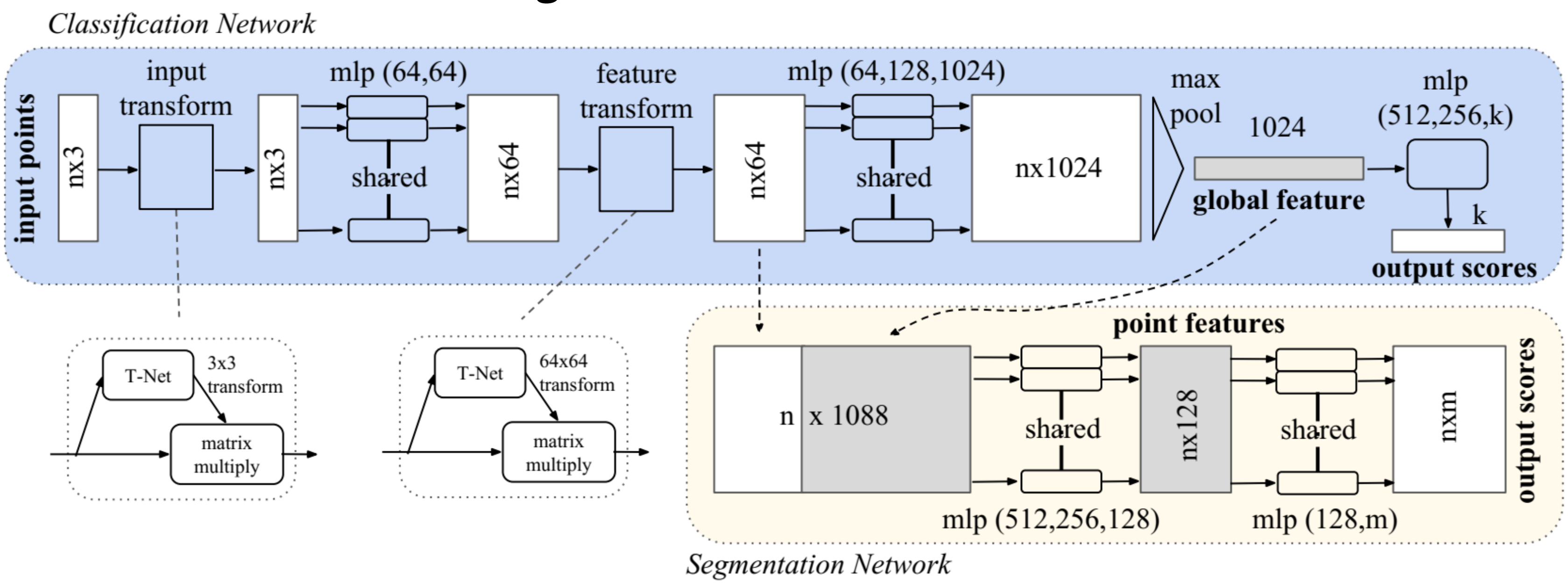
- It would be easy to use addition or multiplication as they are perfectly commutative. But more flexibility is needed and a trainable (learnable) function is preferred.

$$f(\{x_1, \dots, x_n\}) \approx g(h(x_1), \dots, h(x_n)), \qquad (1)$$

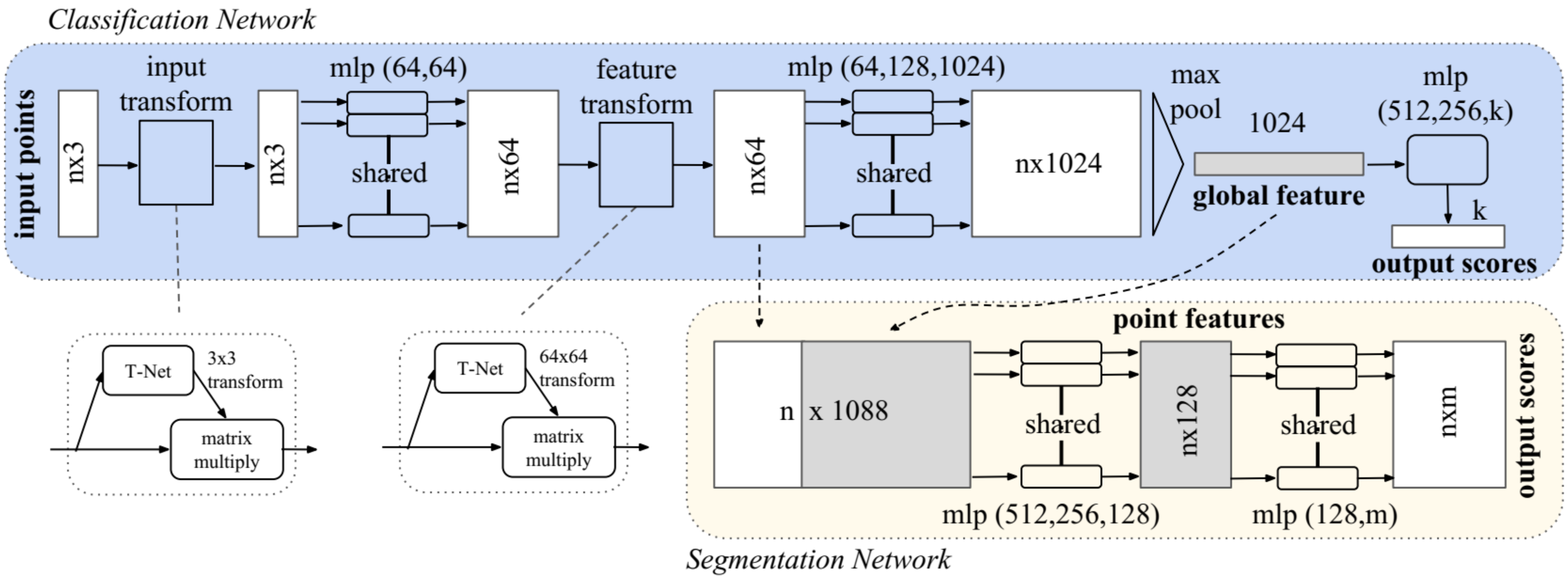
where $f : 2^{\mathbb{R}^N} \rightarrow \mathbb{R}$, $h : \mathbb{R}^N \rightarrow \mathbb{R}^K$ and $g : \underbrace{\mathbb{R}^K \times \dots \times \mathbb{R}^K}_{n} \rightarrow \mathbb{R}$ is a symmetric function.

- Therefore, f is a function mapping the point cloud to a single real number (e.g., a point feature)
- But it is being factorized into a function **g** representing **max-pooling** and **h** representing **multilayer perceptron networks.**
- **Several functions h lead to several features now independent from the point set ordering**

- For now, we just transformed the **whole point cloud** into a **single** feature vector $f_1 \dots f_k$
  - We can now just train any machine learning system like a SVM or a MLP on this very result
  - However, this can only rely on **global information**
- But, we will need a combination of local and global information
- This is done in the ***Segmentation Network***



The Segmentation Network



- It concatenates 64 per point features with 1024 global features for a matrix of nx1088 of features
  - Thus, it can use local and global informamtion
  - Experimentally shown that, for example, normals can be predicted from this stage

Invariance w.r.t. rigid transformations

- The remaining piece is how to achieve invariance under rotation, translation etc.
- Idea: Predict an affine transformation matrix (T-Net) and apply this transformation to the input points
  - These mini-networks have the same structure as the global network: point independent feature extraction, max pooling, and fully connected layers
- This can as well be applied again to the feature space.
  - But beware, it is a large matrix and difficult to optimize
  - Therefore, a constraint makes it almost orthogonal by adding to the loss

$$L_{reg} = \|I - AA^T\|_F^2,$$

Why PointNet? Because it looks nice and works in practice

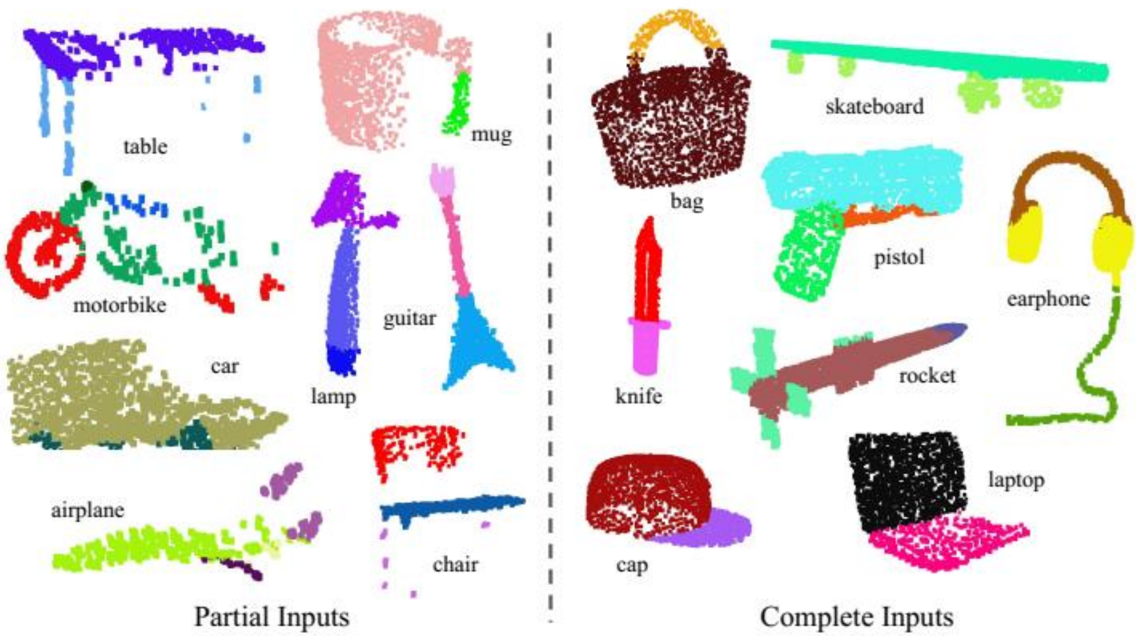
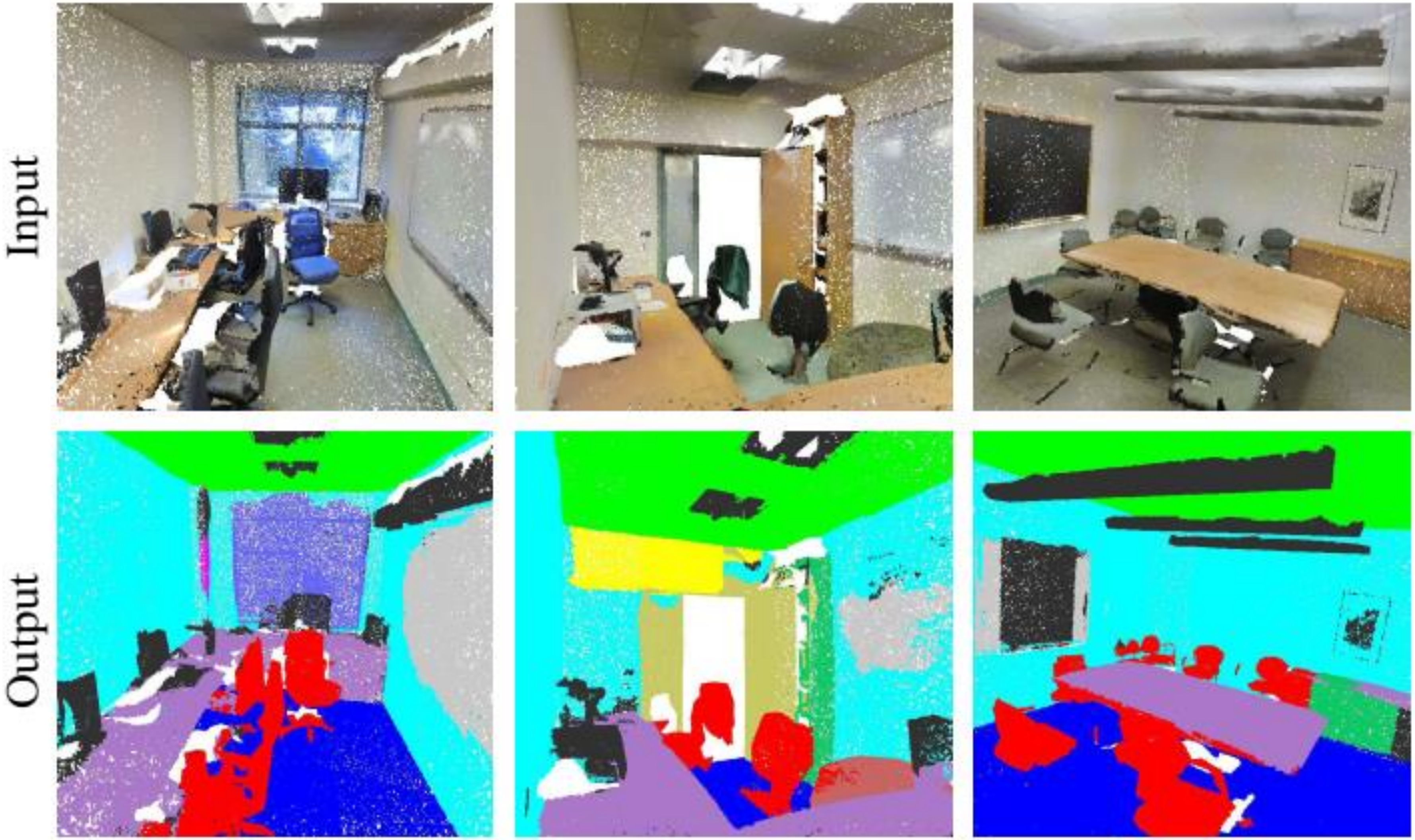


Figure 3. **Qualitative results for part segmentation.** We visualize the CAD part segmentation results across all 16 object categories. We show both results for partial simulated Kinect scans (left block) and complete ShapeNet CAD models (right block).

Yes, it works…



Theorem 1 *Suppose* $f : \mathcal{X} \rightarrow \mathbb{R}$ *is a continuous set function w.r.t Hausdorff distance* $d_H(\cdot, \cdot)$. $\forall \epsilon > 0$, $\exists$ *a continuous function* h *and a symmetric function* $g(x_1, \ldots, x_n) = \gamma \circ MAX$, *such that for any* $S \in \mathcal{X}$,

$$\left| f(S) - \gamma \left(\underset{x_i \in S}{MAX} \{h(x_i)\} \right) \right| < \epsilon$$

where $x_1, \ldots, x_n$ *is the full list of elements in* S *ordered arbitrarily,* γ *is a continuous function, and MAX is a vector max operator that takes* n *vectors as input and returns a new vector of the element-wise maximum.*

Funktionen h und g existieren also tatsächlich für jede Fehlerschranke.
Allerdings ist das kein Ergebnis zur Trainierbarkeit. Nur die Existenz…

PointNet++

- PointNet uses a single Max-Pooling layer, which means that **all features are single-scale**
  - Point Clouds have varying sampling density, especially with fixed sensors
- PointNet++ is based on a **hierarchical grouping** analyzing larger and larger extracts of the point cloud
  - Implemented as Compression: At each and every step, a point set is abstracted to a point set with fewer points
  - Three „layers“:
    - **Sampling,** selects a set of points as centroids
    - **Grouping,** assigns points to centroids
    - PointNet++ uses a **„mini“-PointNet** to extract features

- **Sampling Layer**
  - **Iterative Farthest Point Sampling (FPS)**
    Iteratively add the farthest point from the input to the current set
- **Grouping Layer**
  - Assign some neighboring points using a
    - ball query
      - Pro: same scale, Con: different number of elements
    - kNN
      - Pro: same number of elements, Con: different scale
    - Ball query preferred as PointNet can deal with varying inputs
- **Many additional tricks**
  - See https://arxiv.org/pdf/1706.02413.pdf

And now? How would I?

- **Run a computer / container with tensorflow**
  - I am runnning NVIDIA‘s optimized tensorflow container (need an account at NVIDIA container registry)
    - Optimized by NVIDIA for DGX-1 Familiy
  - On 8 interconnected V100 GPUs (256 GB total memory)
  - Trains about **2 hours to 88.8 % accuracy** on point cloud classification for ModelNet 40 dataset
- Inside the container (or in the Dockerfile)
  - **apt-get install libhdf5-dev (for HDF5 file support)**
  - **pip install h5py**
  - **git clone https://github.com/charlesq34/pointnet**
  - **python train.py**
    - Automatically downloads dataset
    - Runs a few epoochs and outputs results

```
root@ede2a32eccac:/workspace/pointnet# python train.py
--2019-01-17 06:41:18--
https://shapenet.cs.stanford.edu/media/modelnet40_ply_hdf5_
2048.zip
Resolving shapenet.cs.stanford.edu
(shapenet.cs.stanford.edu)...
171.67.77.19
Connecting to shapenet.cs.stanford.edu
(shapenet.cs.stanford.edu)|171.67.77.19|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 435212151 (415M) [application/zip] Saving to:
'modelnet40_ply_hdf5_2048.zip'

modelnet40_ply_hdf5 100%[===================>] 415.05M
310KB/s    in 22m 30s
```

Results look like

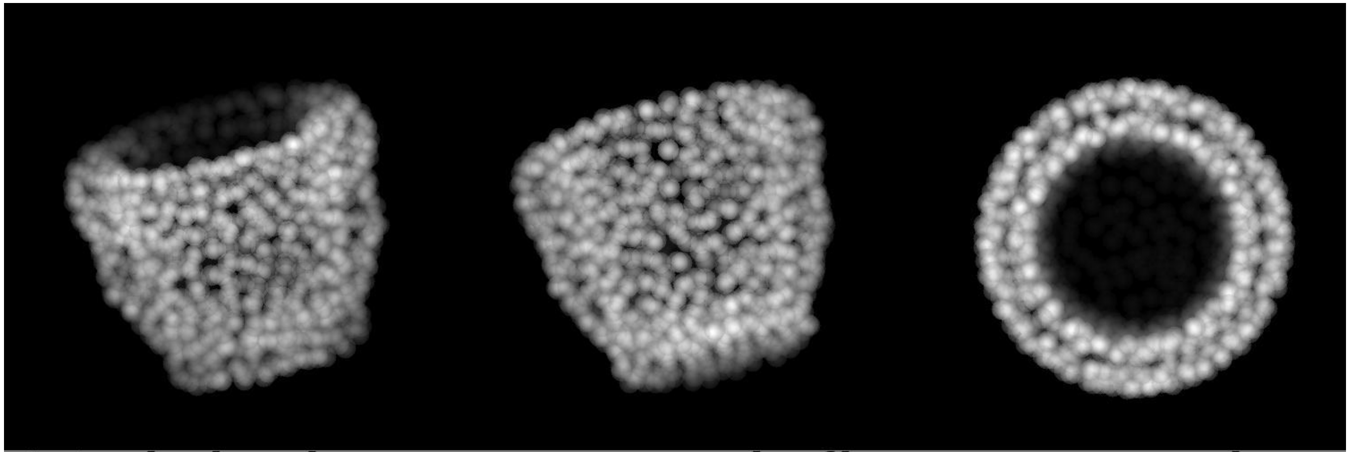

```
[…]
eval mean loss: 0.549058
eval accuracy: 0.886769
eval avg class acc: 0.860618
**** EPOCH 249 ****
[…]
----1-----
eval mean loss: 0.546670
eval accuracy: 0.888393
eval avg class acc: 0.858817

(after 2 hours including data download on a single DGX-1)
```

Per-class performance and visualization of errors

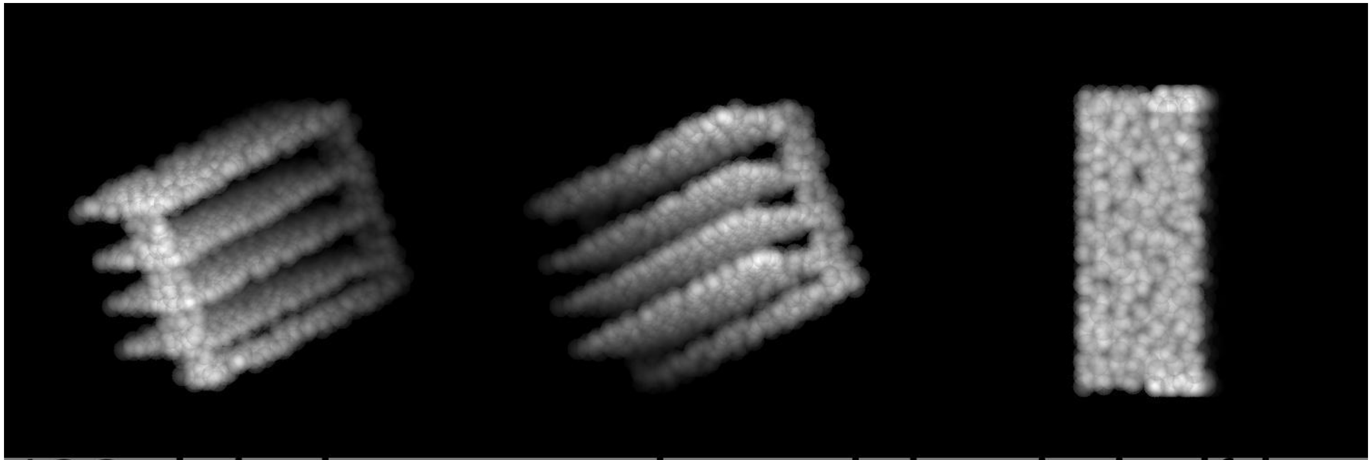

```
$> pip install scipy
$> pip install image # for PIL
$> pip install matplotlib # for visualizations
$> python evaluate.py -visu
```

This now creates output of erroneous classifications in the dump folder and gives per class performance results. Looks like

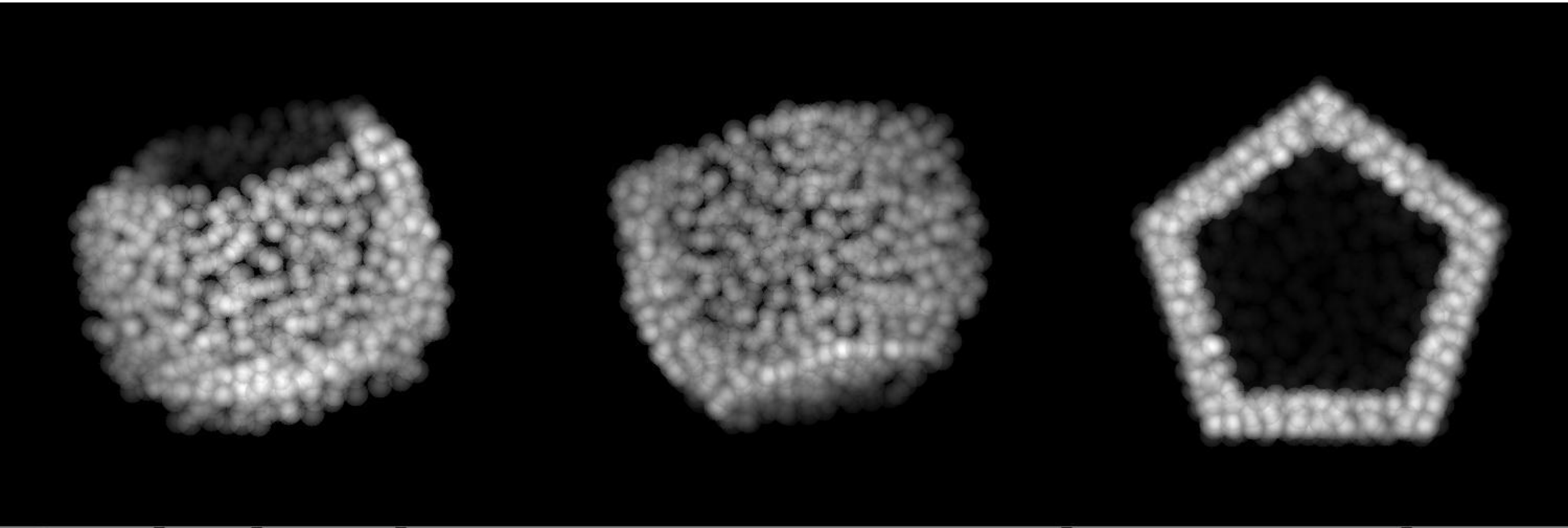
```
   airplane:    1.000
    bathtub:    0.860
        bed:    0.980
      bench:    0.700
  bookshelf:    0.900
     bottle:    0.940
       bowl:    0.950
        car:    0.990
      chair:    0.980
       cone:    0.950
        cup:    0.550
```



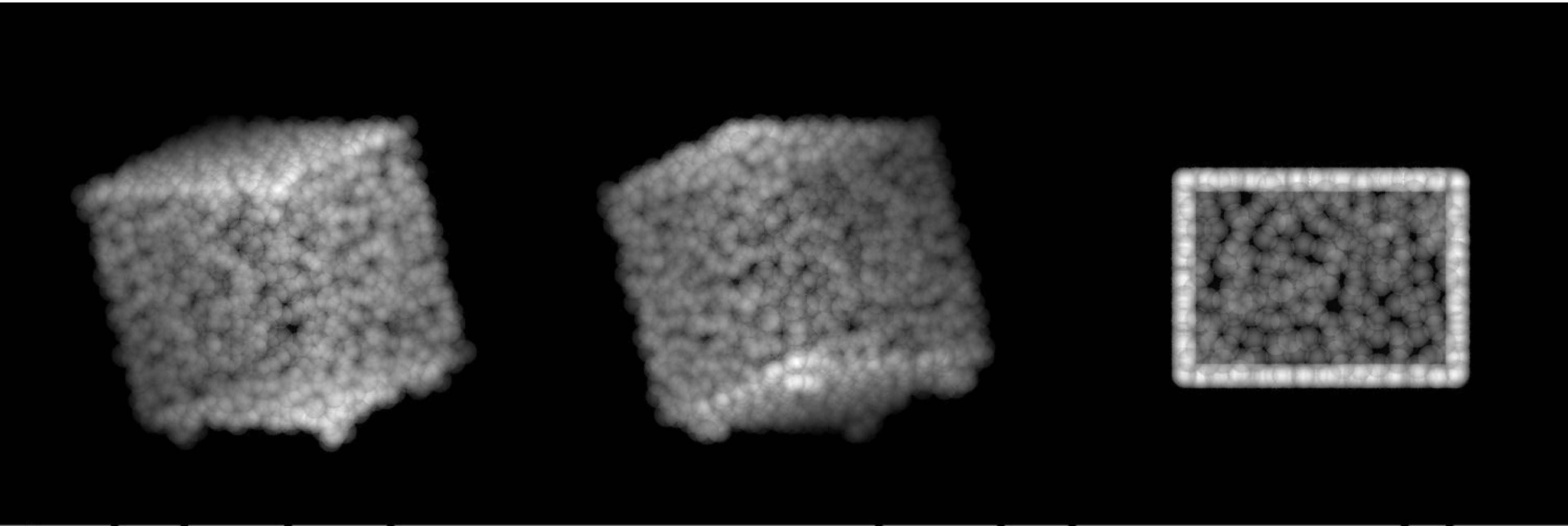
14_label_vase_pred_flower_pot.jpg



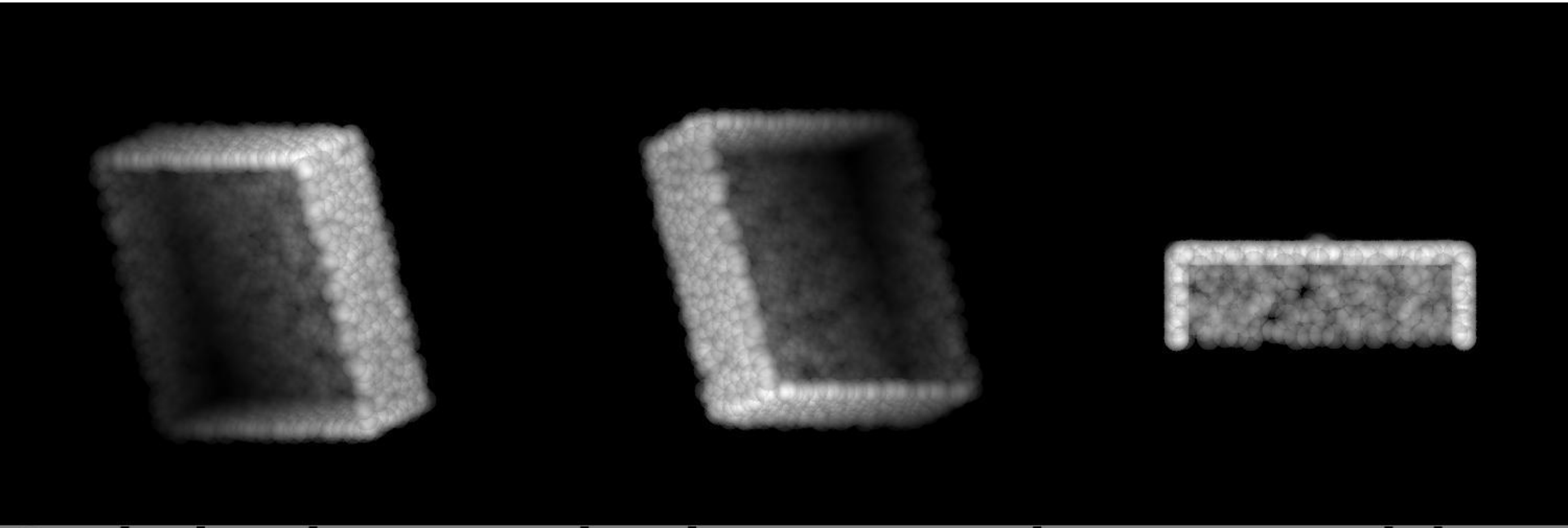
133_label_tv_stand_pred_bookshelf.jpg



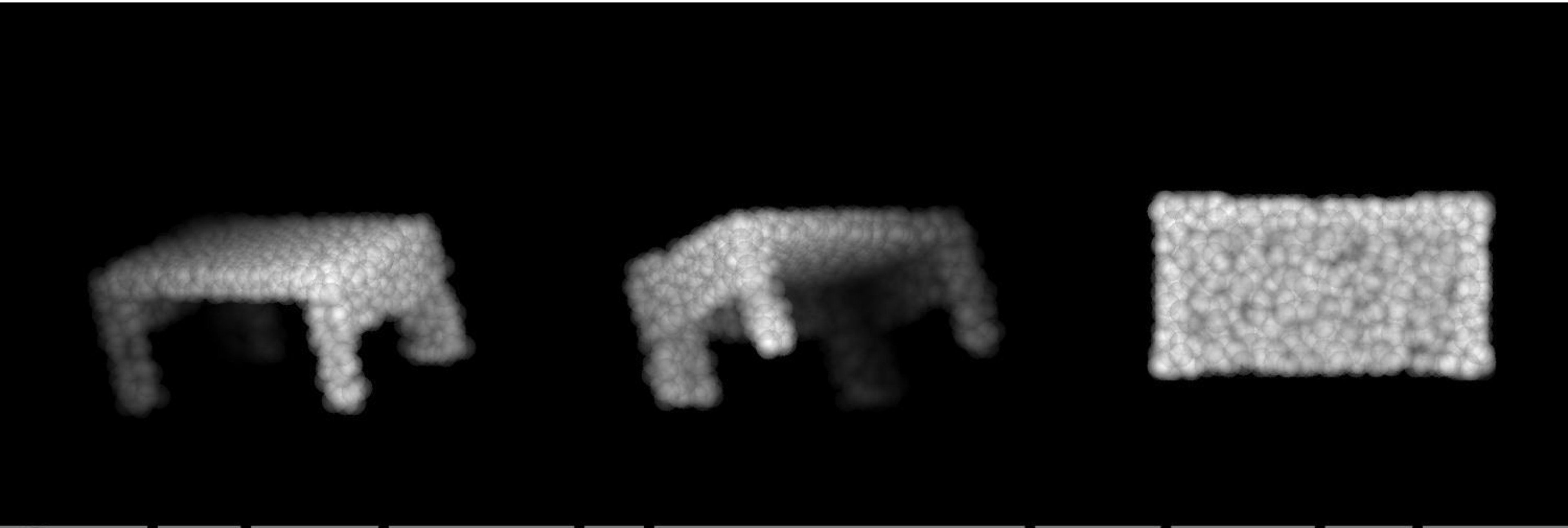
0_label_vase_pred_cup.jpg

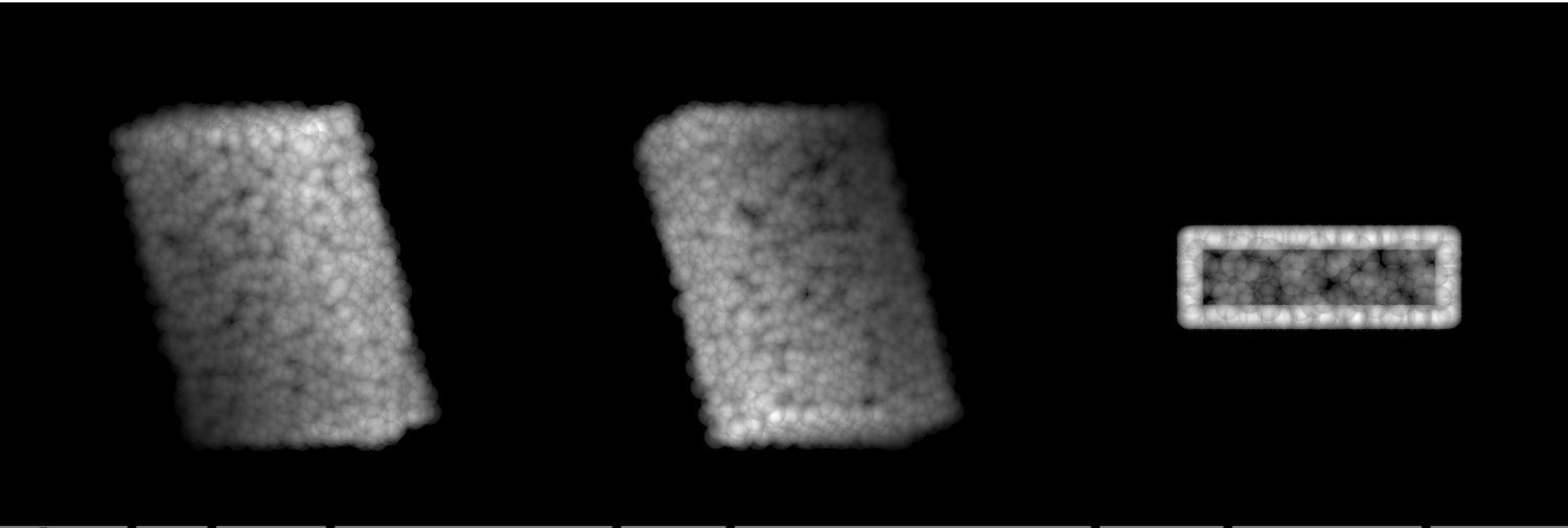


1_label_dresser_pred_night_stand.jpg

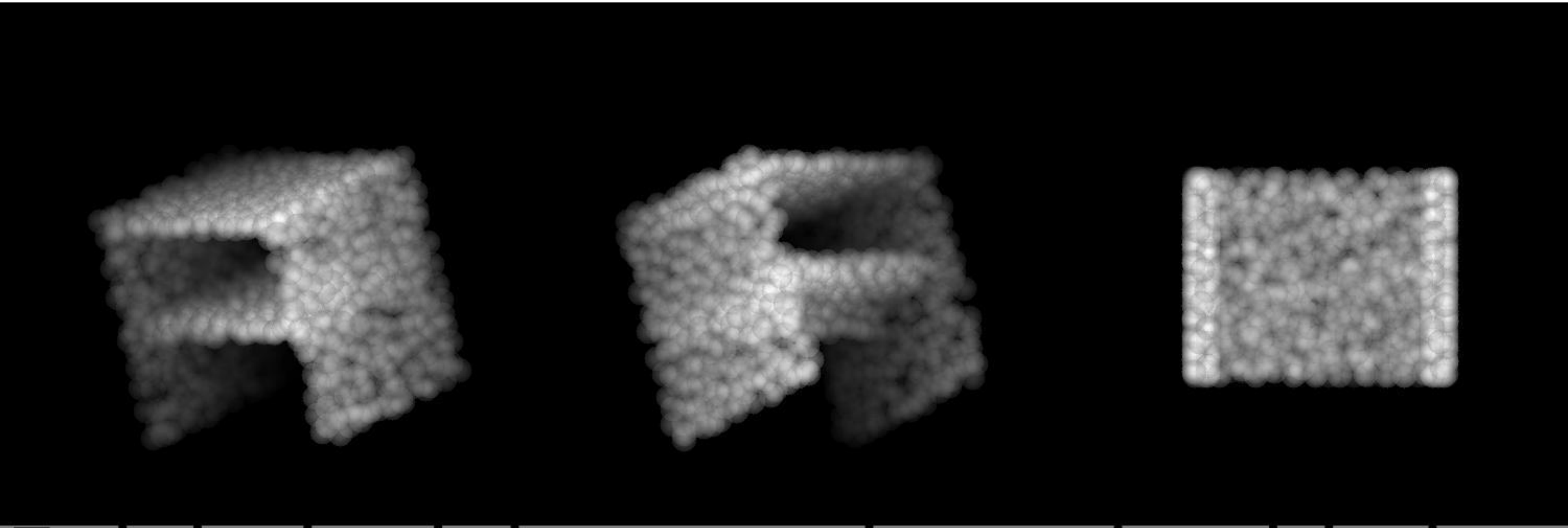


2_label_wardrobe_pred_mantel.jpg

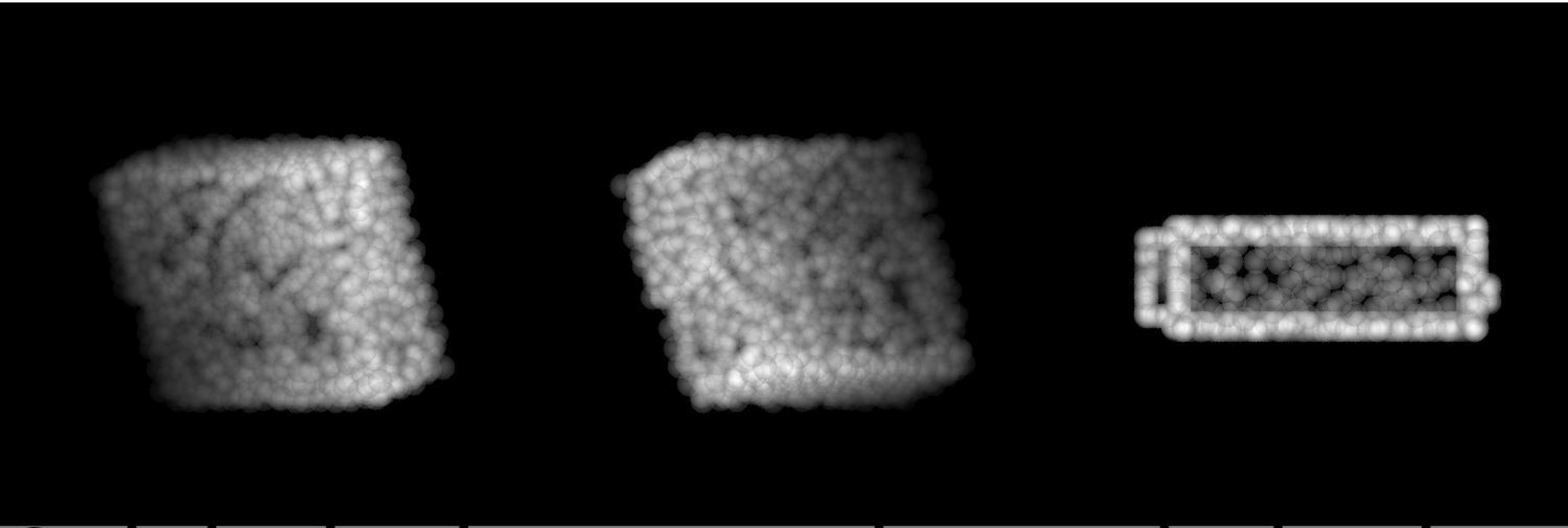

3_label_table_pred_desk.jpg

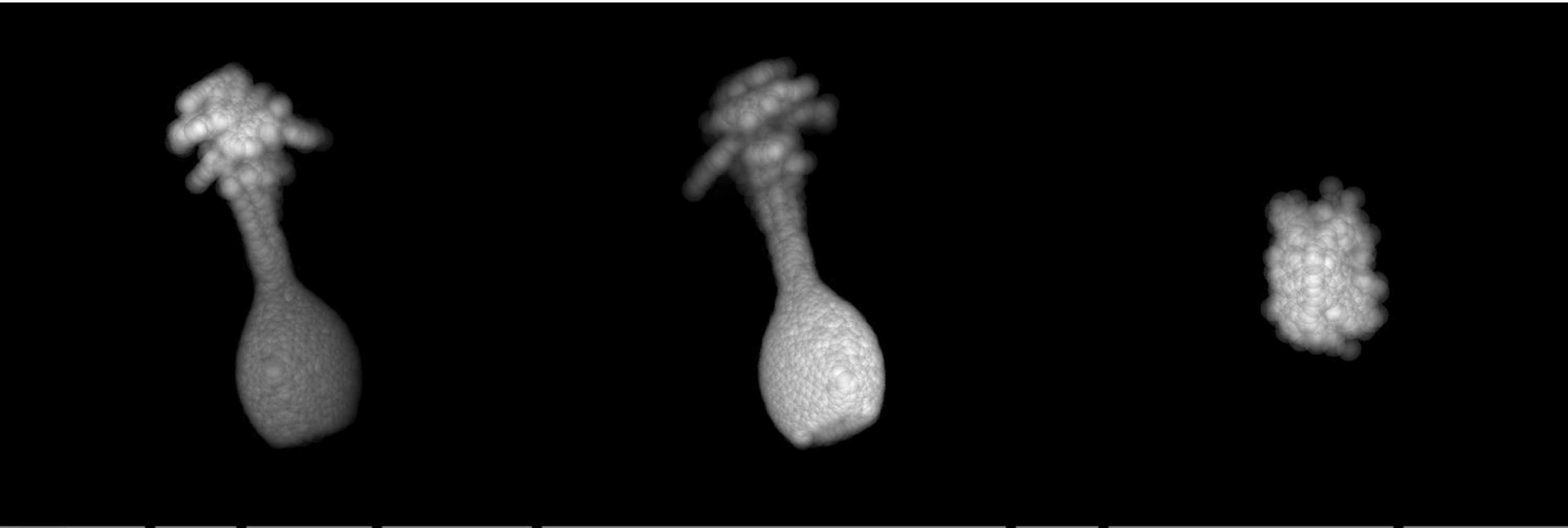


4_label_wardrobe_pred_xbox.jpg

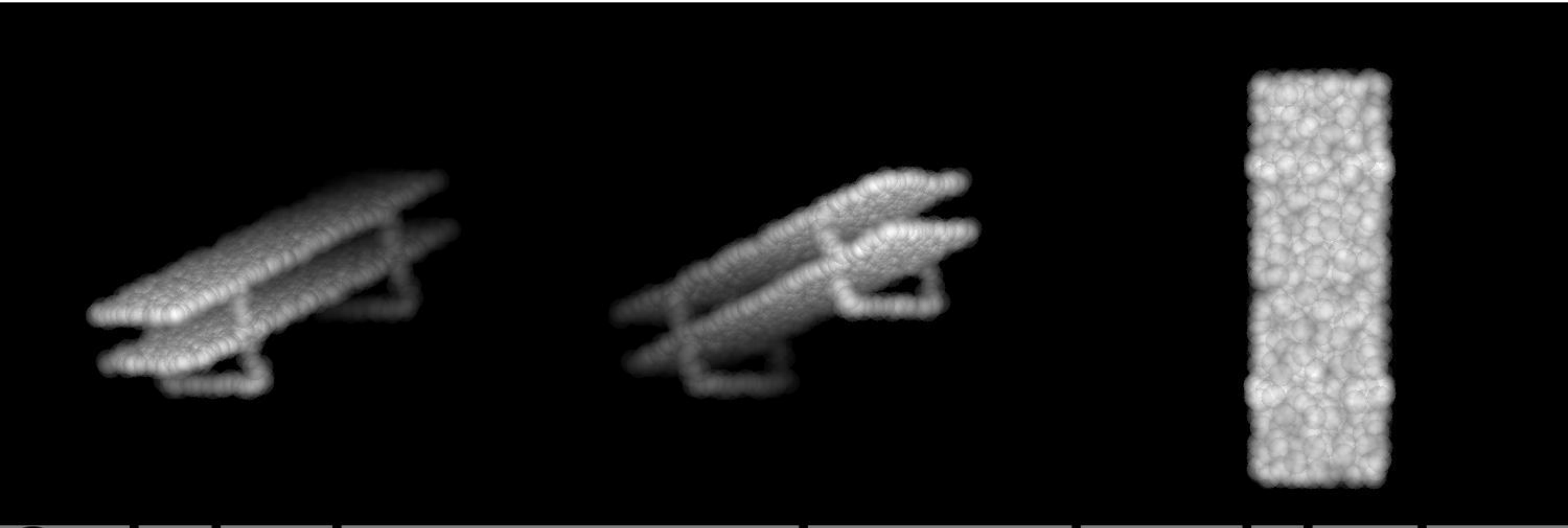


5_label_night_stand_pred_table.jpg

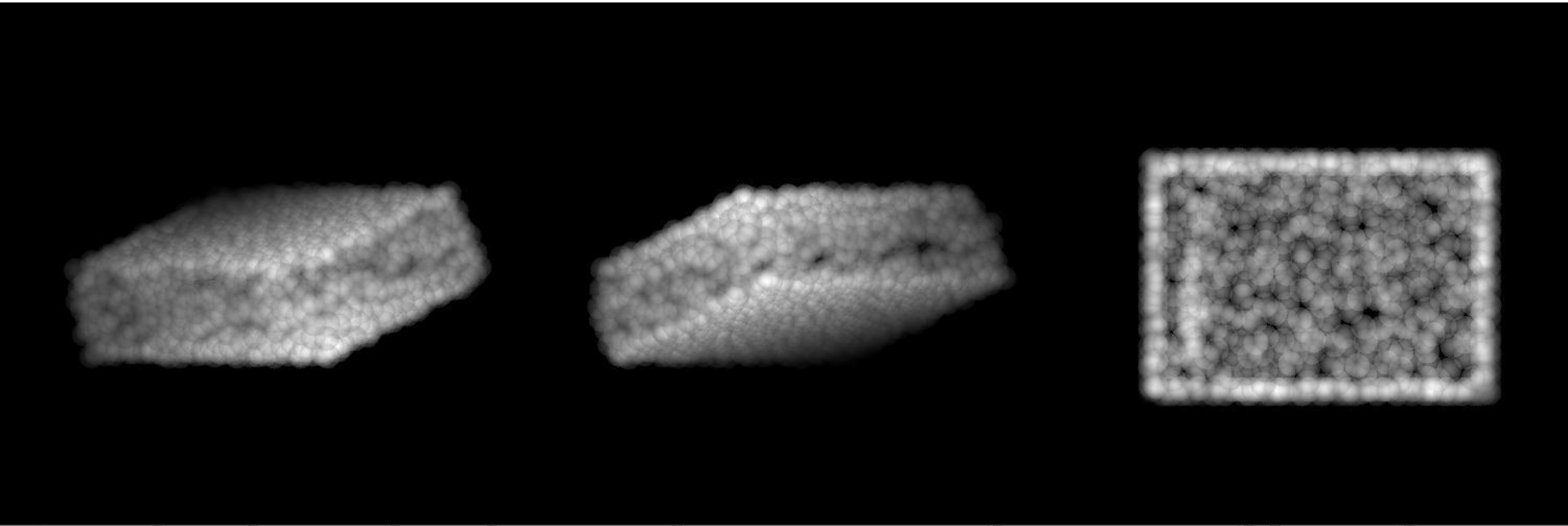

6_label_xbox_pred_wardrobe.jpg



7_label_plant_pred_lamp.jpg

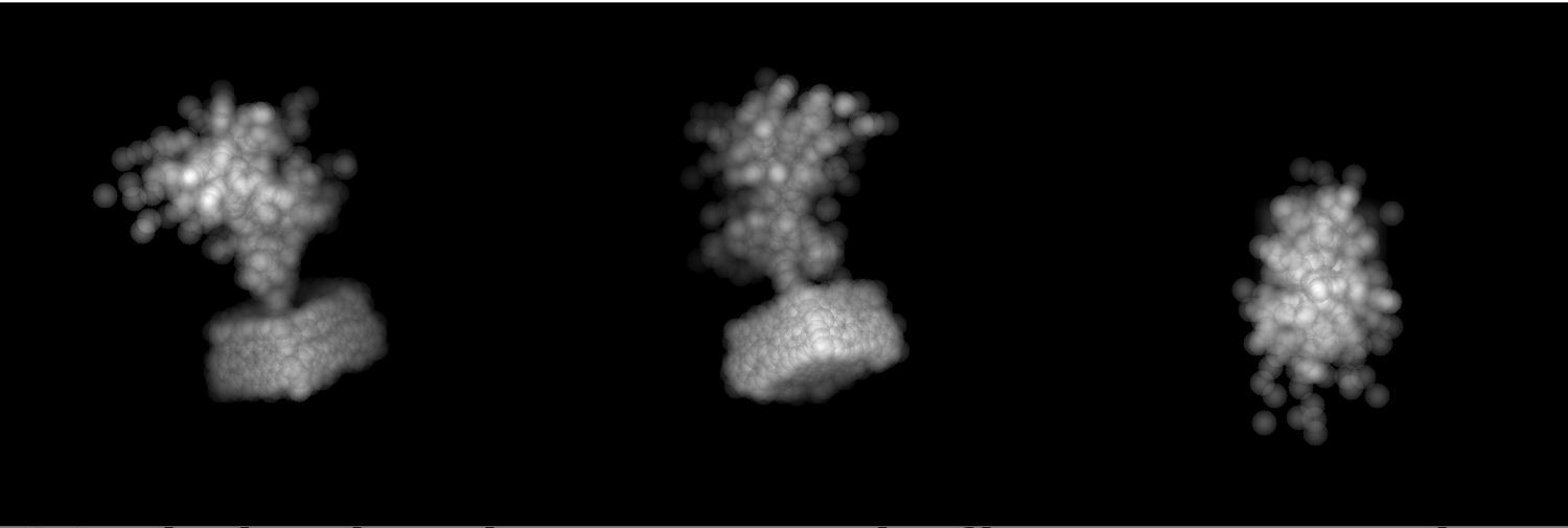


8_label_tv_stand_pred_table.jpg

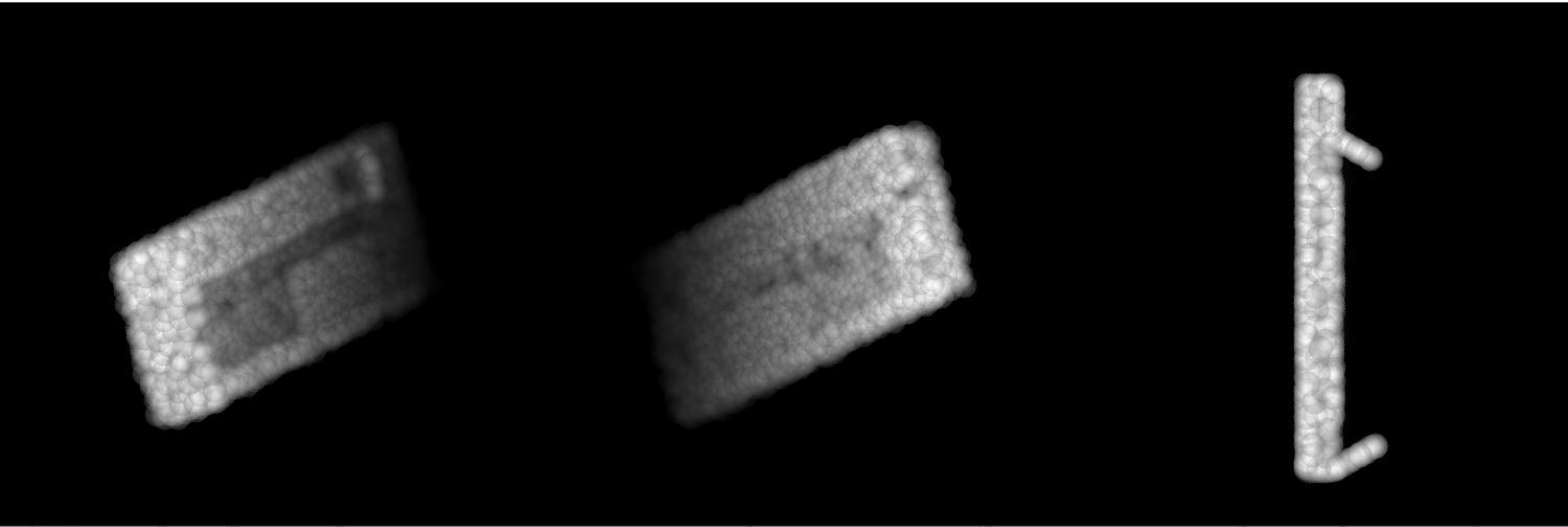


9_label_bed_pred_radio.jpg

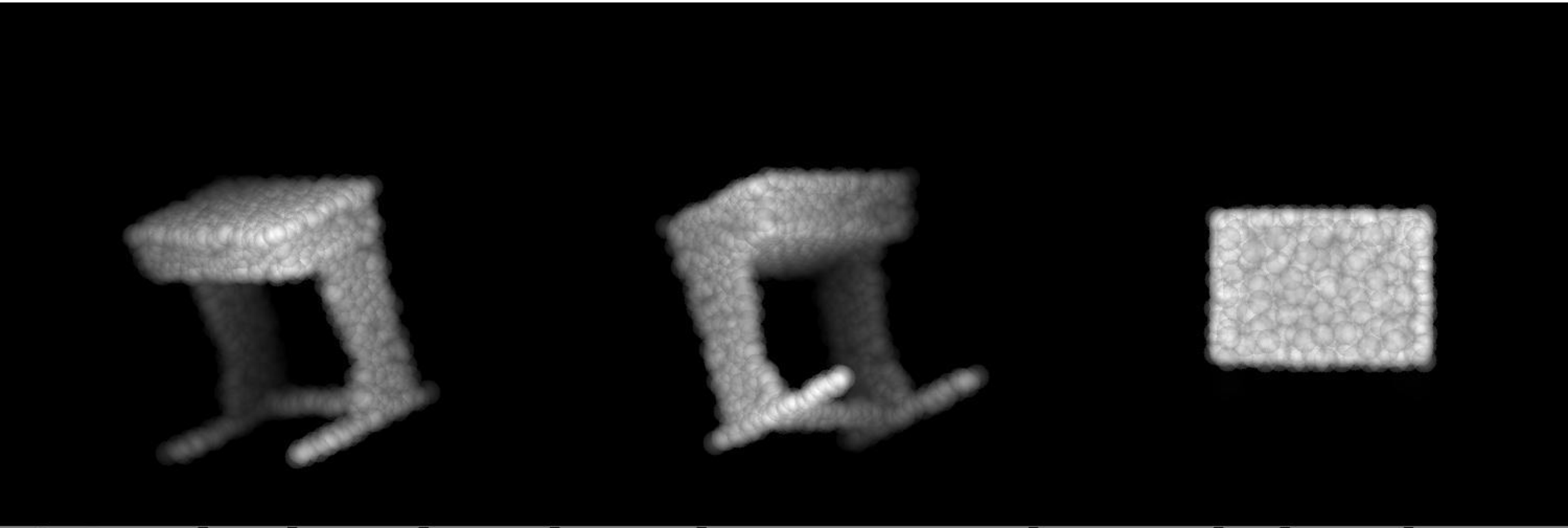
10_label_flower_pot_pred_plant.jpg



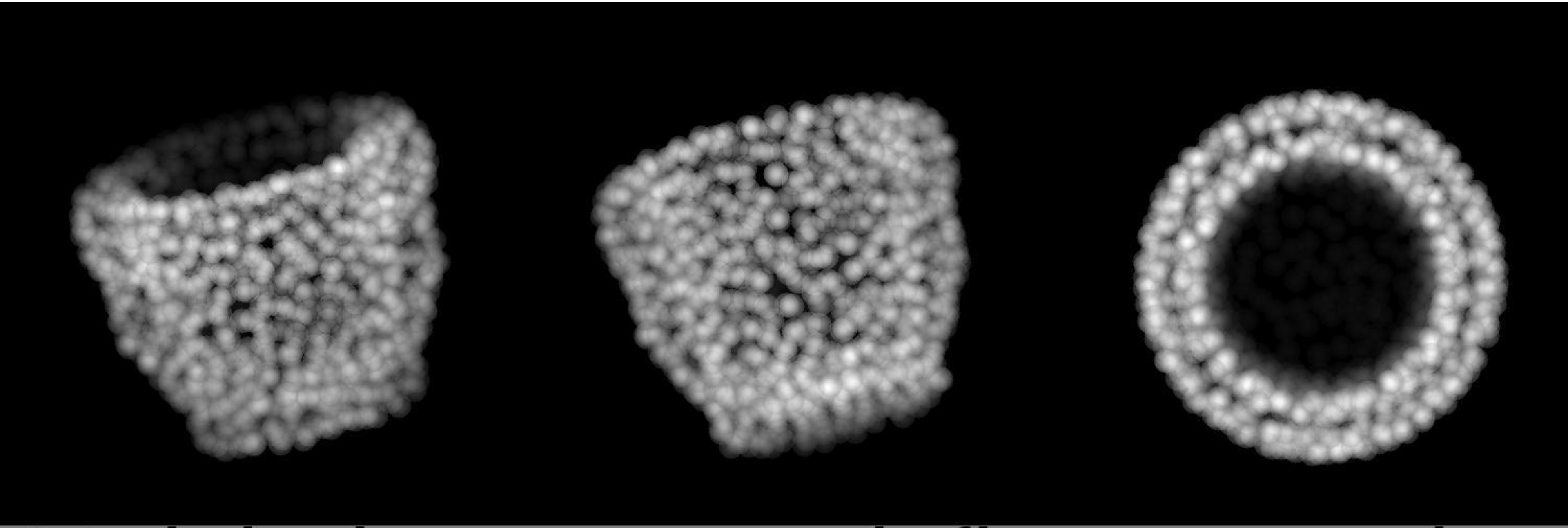
11_label_plant_pred_flower_pot.jpg



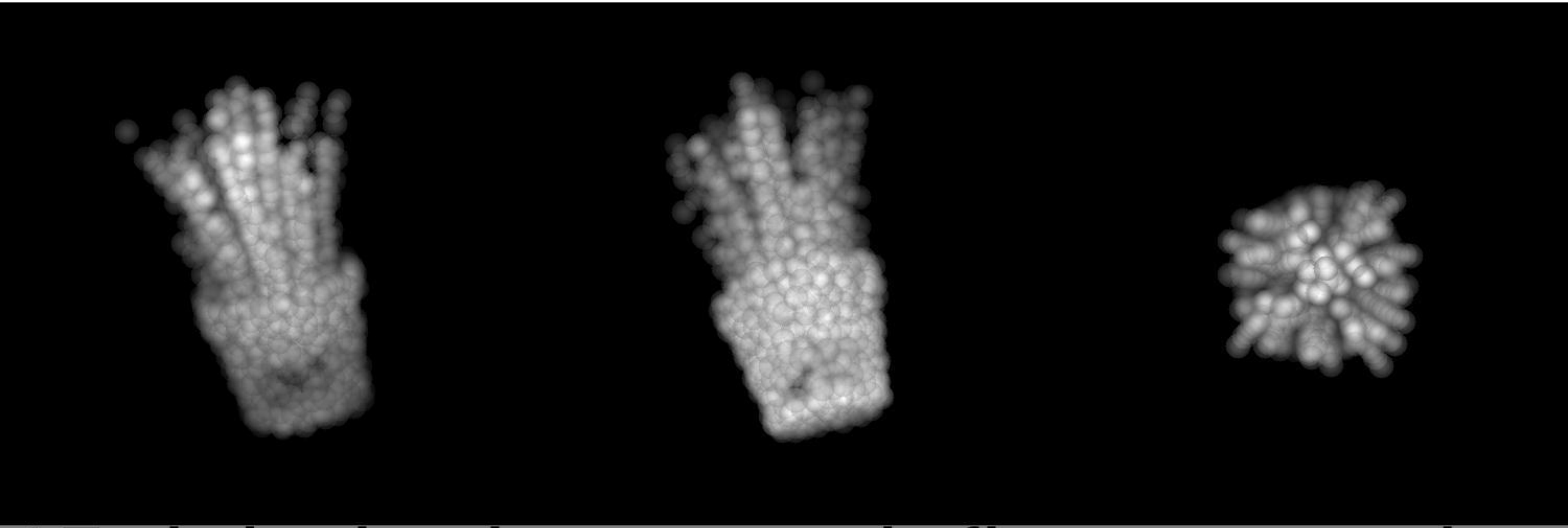
12_label_tv_stand_pred_wardrobe.jpg



13_label_desk_pred_table.jpg

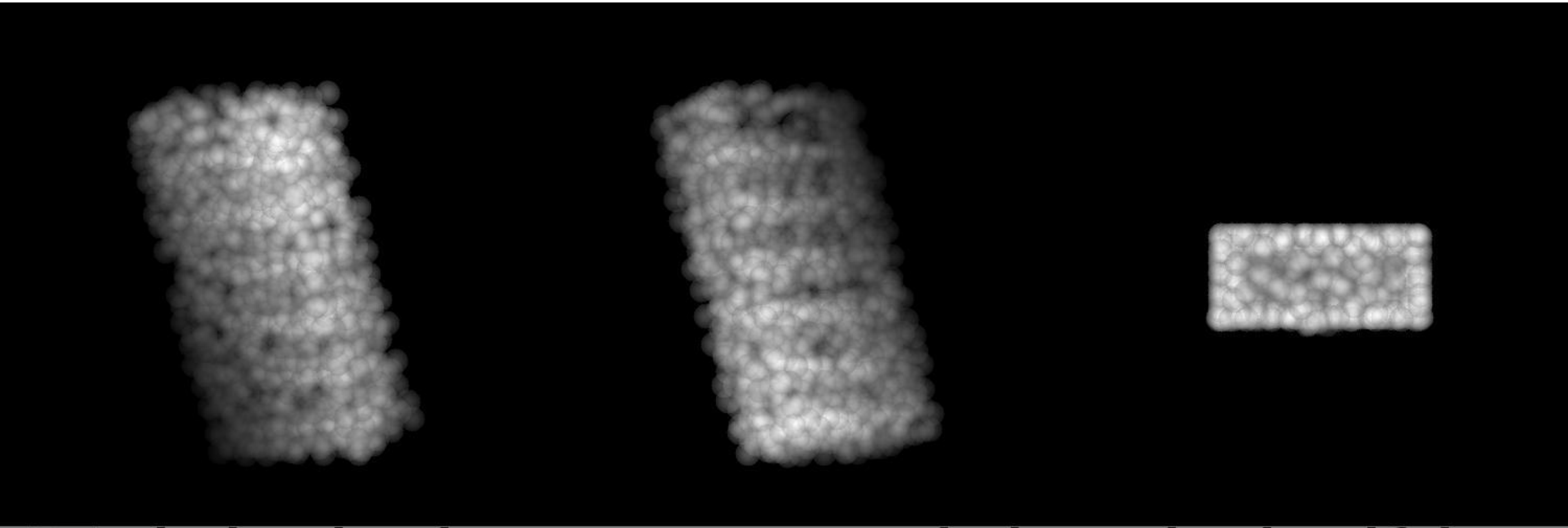


14_label_vase_pred_flower_pot.jpg



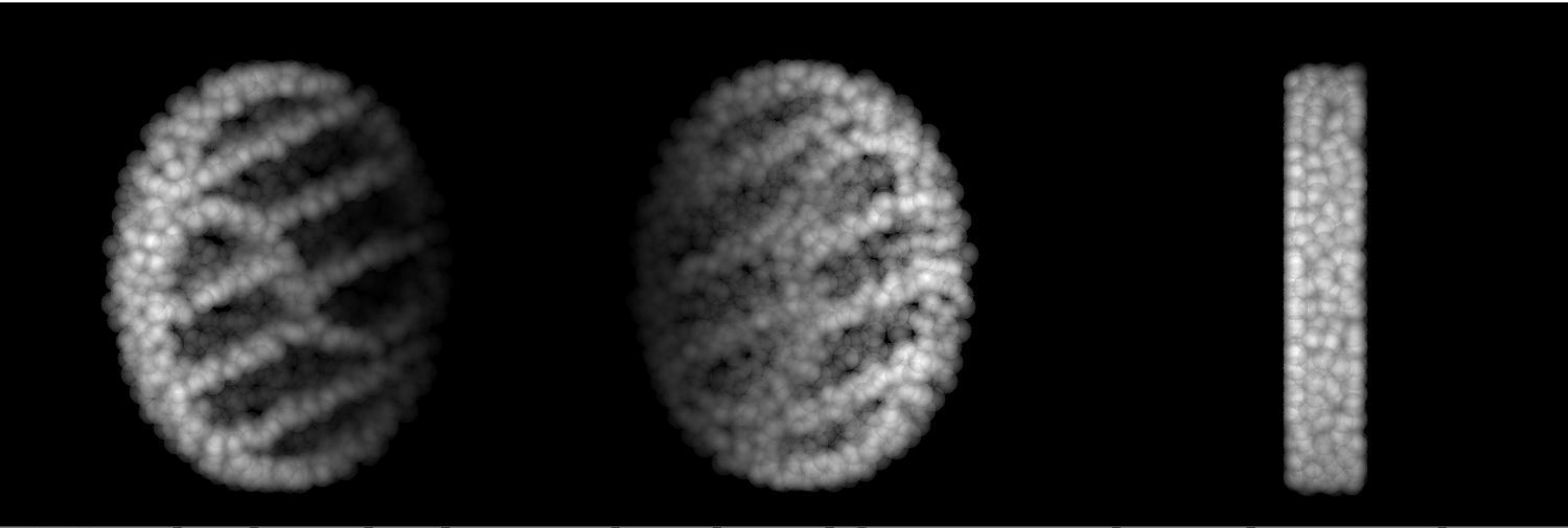
15_label_plant_pred_flower_pot.jpg

16_label_flower_pot_pred_plant.jpg

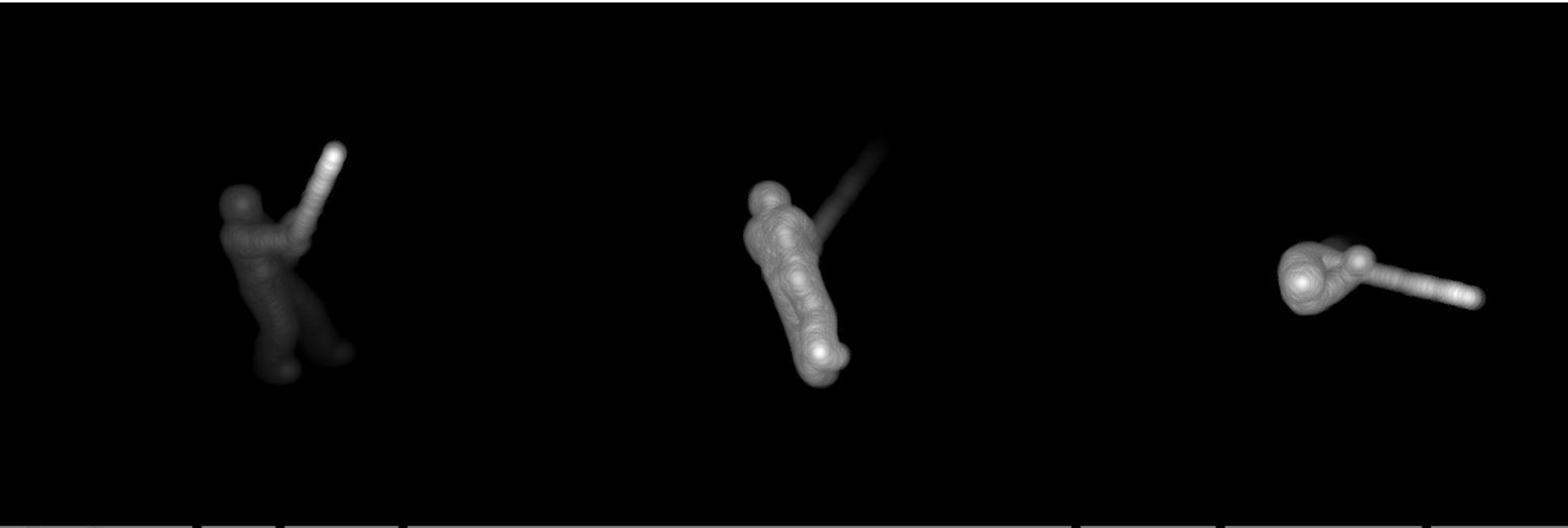


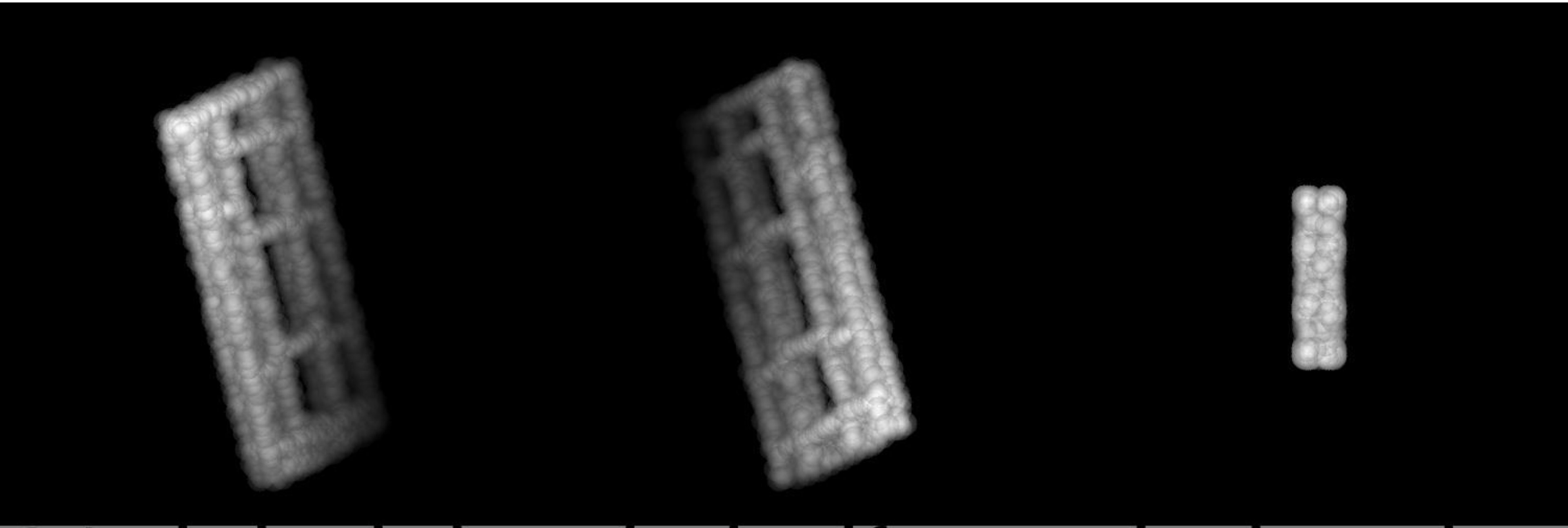
17_label_dresser_pred_bookshelf.jpg


18_label_bathtub_pred_table.jpg

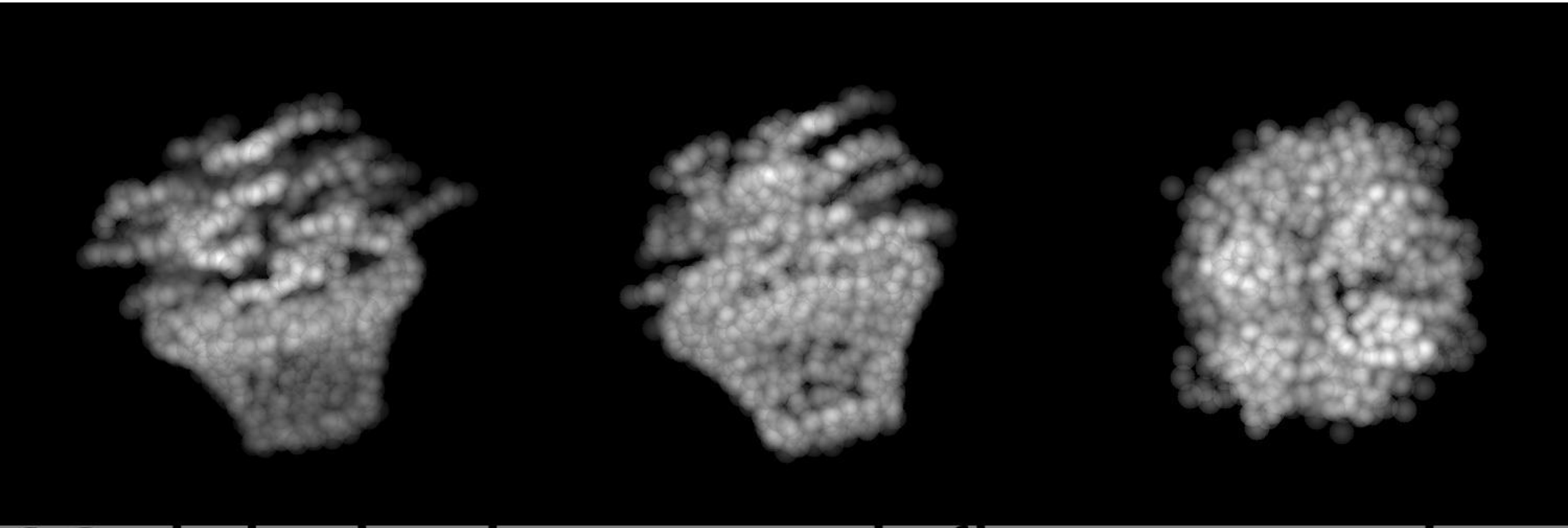


19_label_bookshelf_pred_plant.jpg

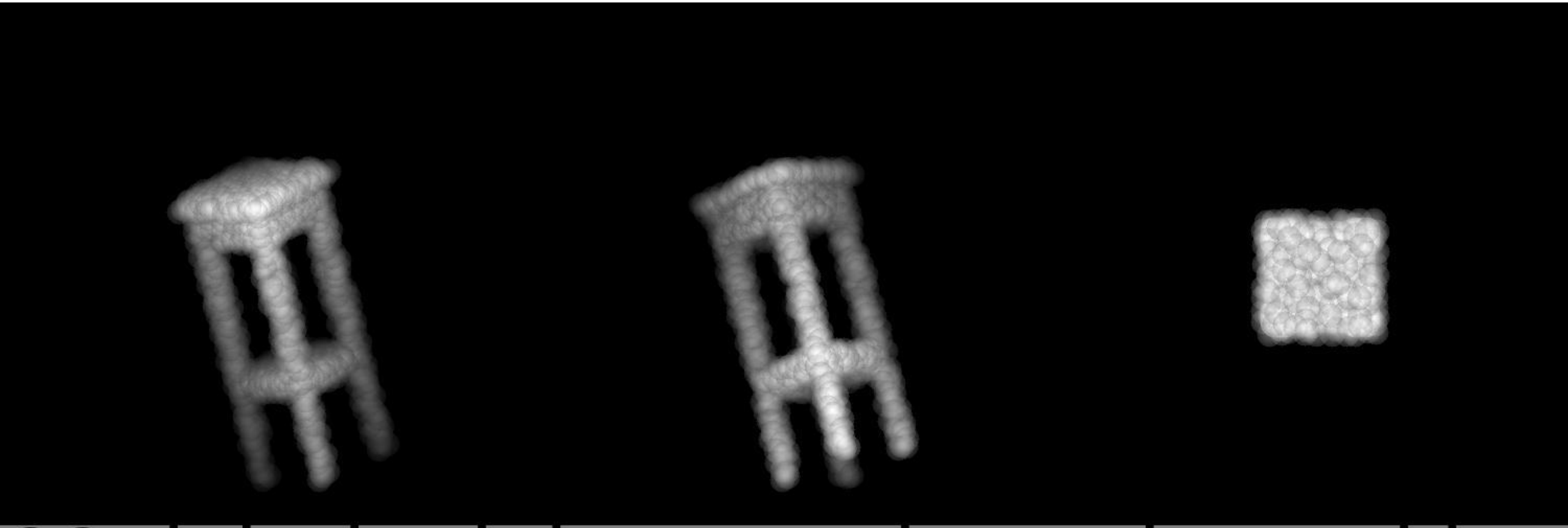

20_label_person_pred_plant.jpg



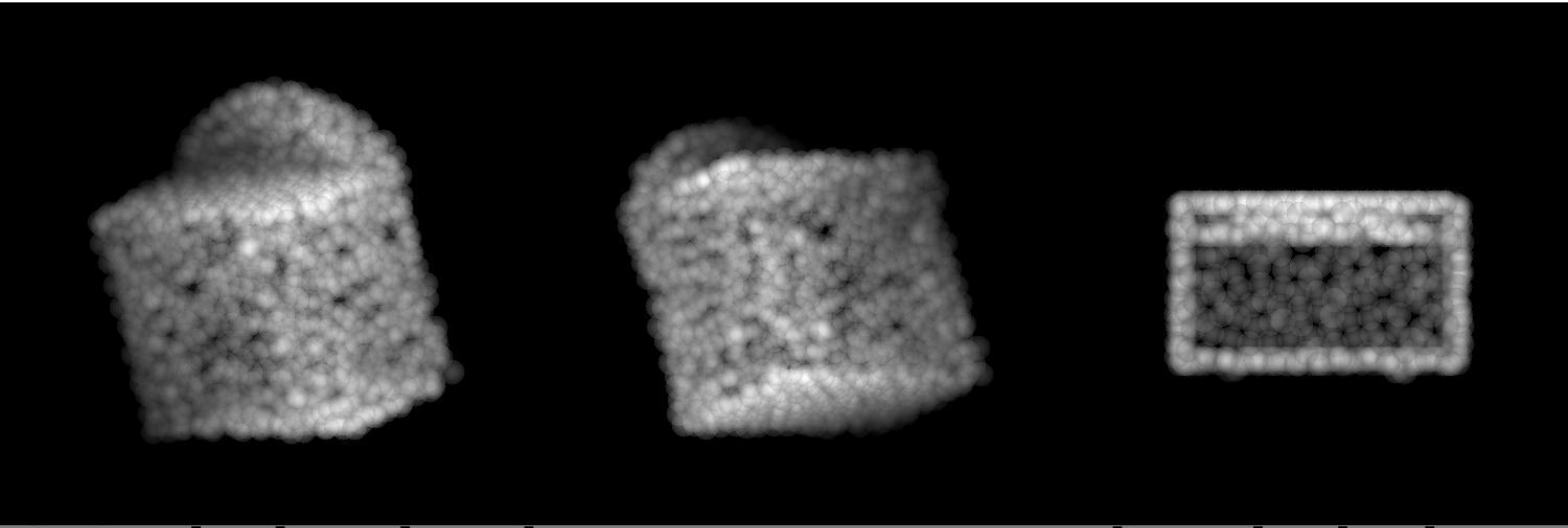
21_label_bookshelf_pred_door.jpg



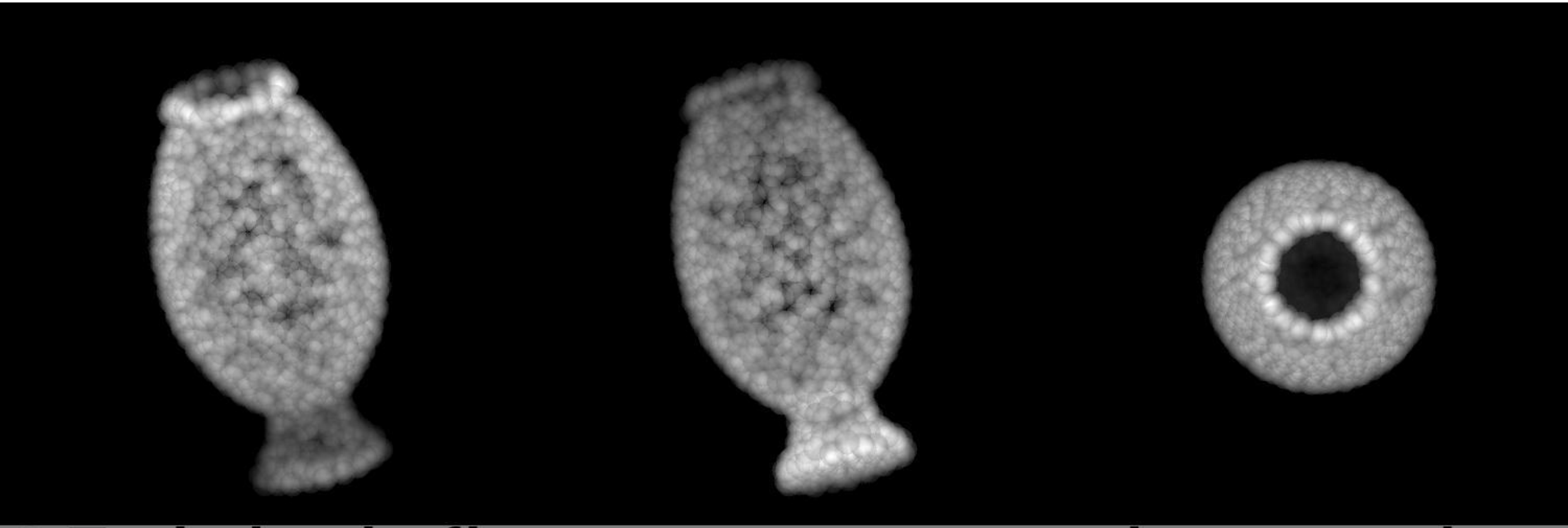
22_label_plant_pred_flower_pot.jpg

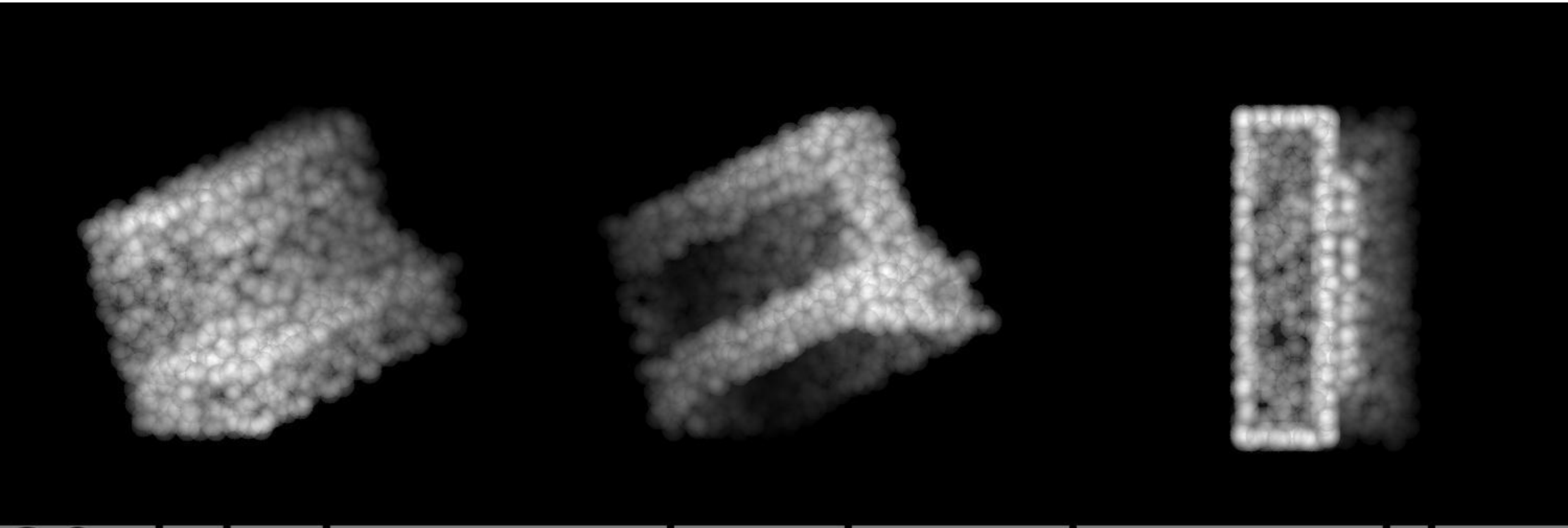


23_label_night_stand_pred_stool.jpg

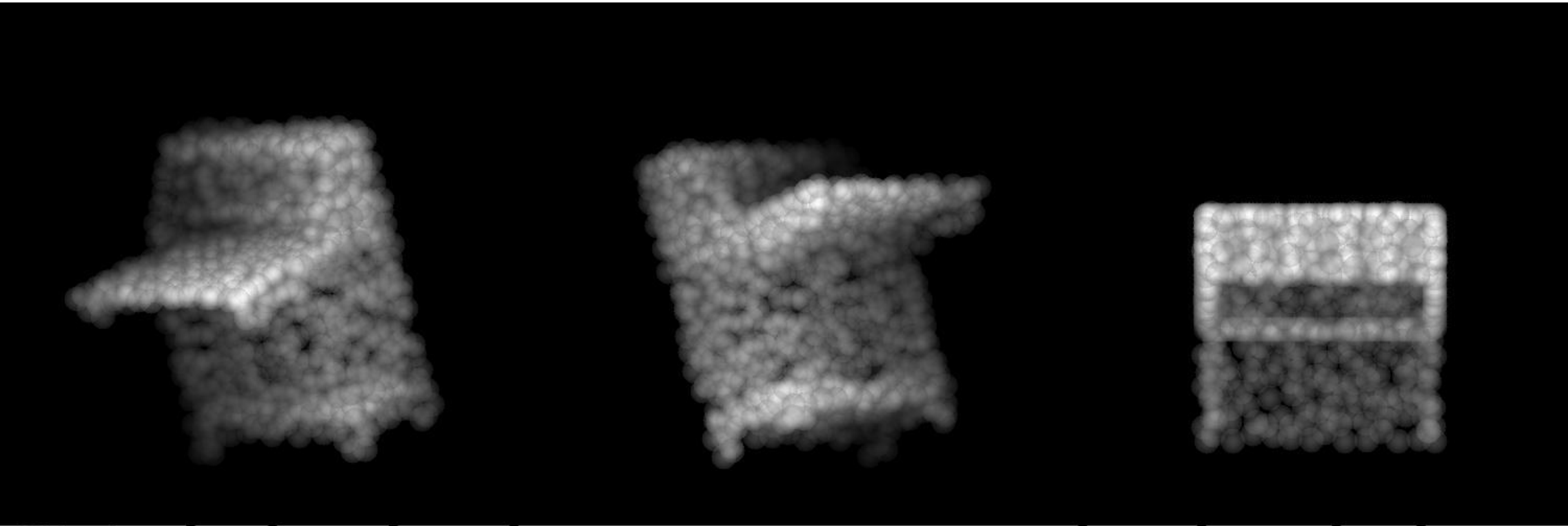


24_label_dresser_pred_sink.jpg

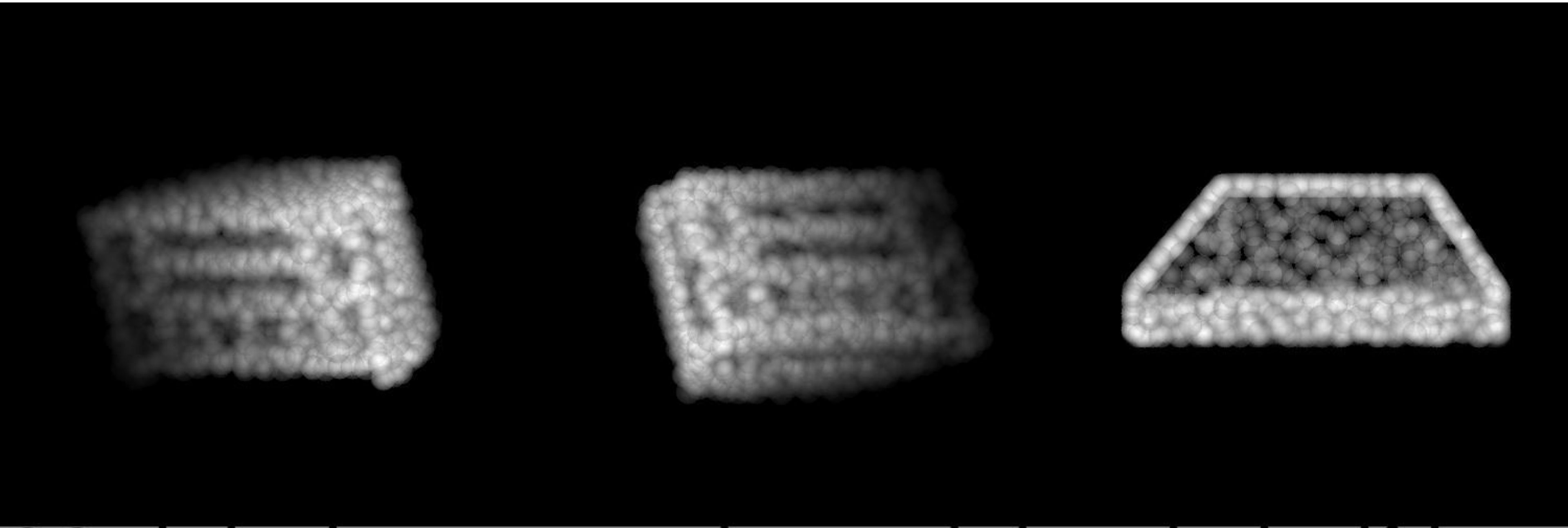

25_label_flower_pot_pred_vase.jpg



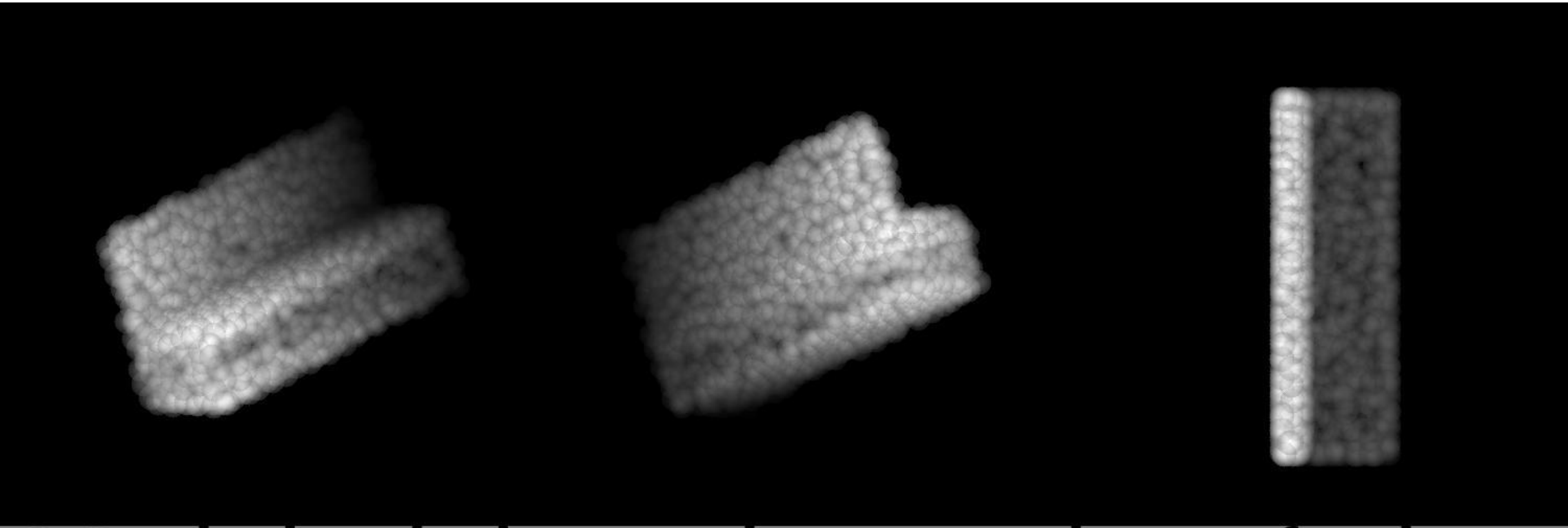
26_label_range_hood_pred_mantel.jpg



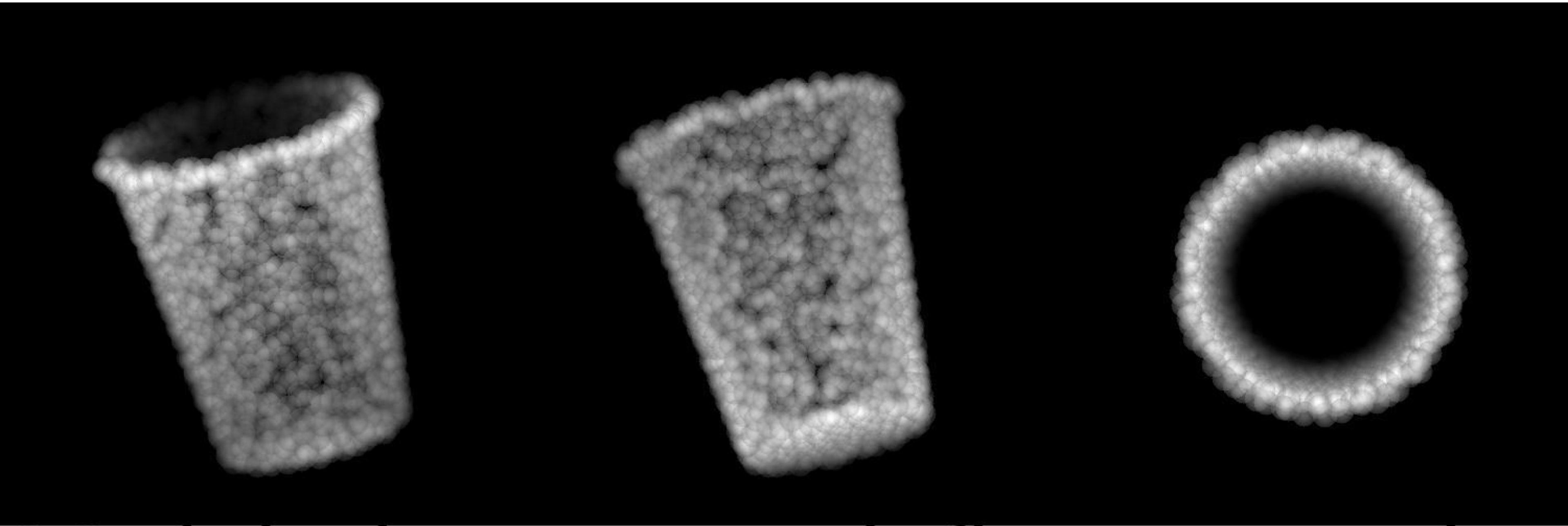
27_label_dresser_pred_desk.jpg



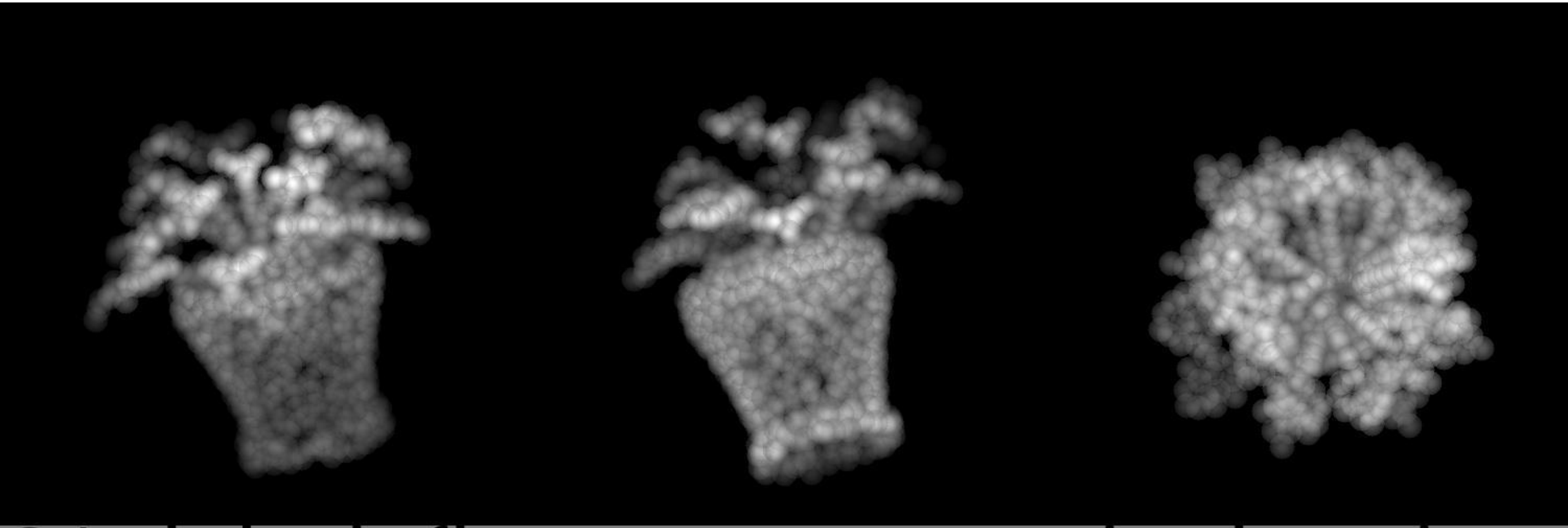
28_label_tv_stand_pred_bookshelf.jpg



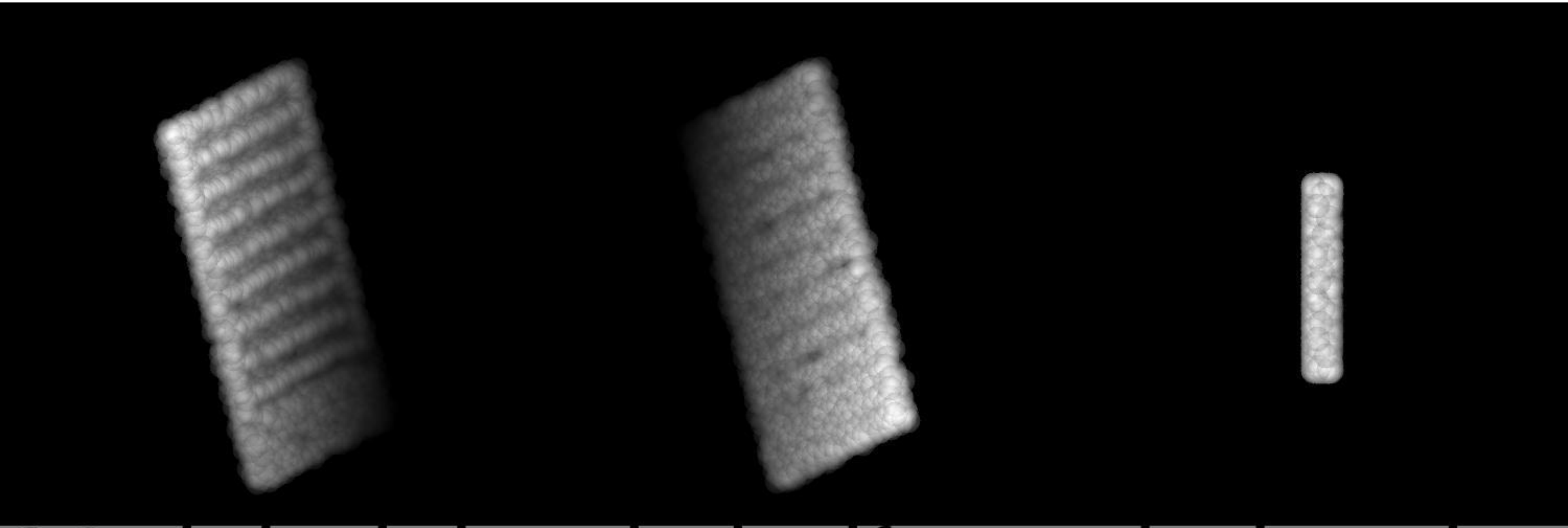
29_label_bench_pred_sofa.jpg



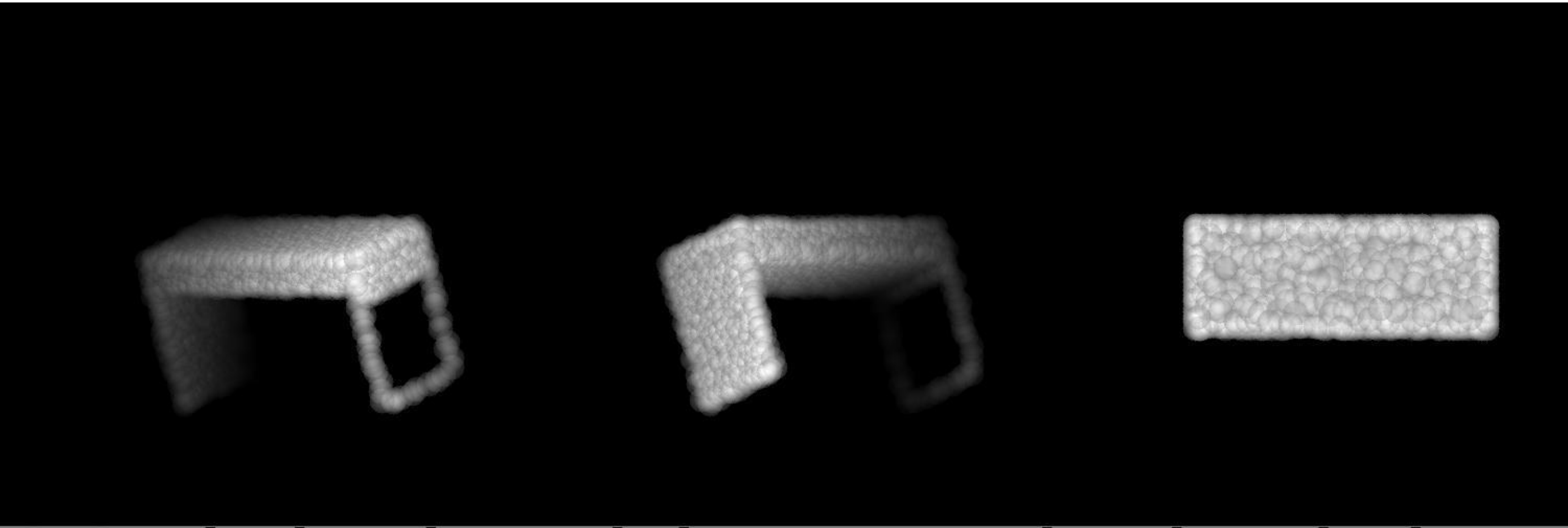
30_label_cup_pred_flower_pot.jpg



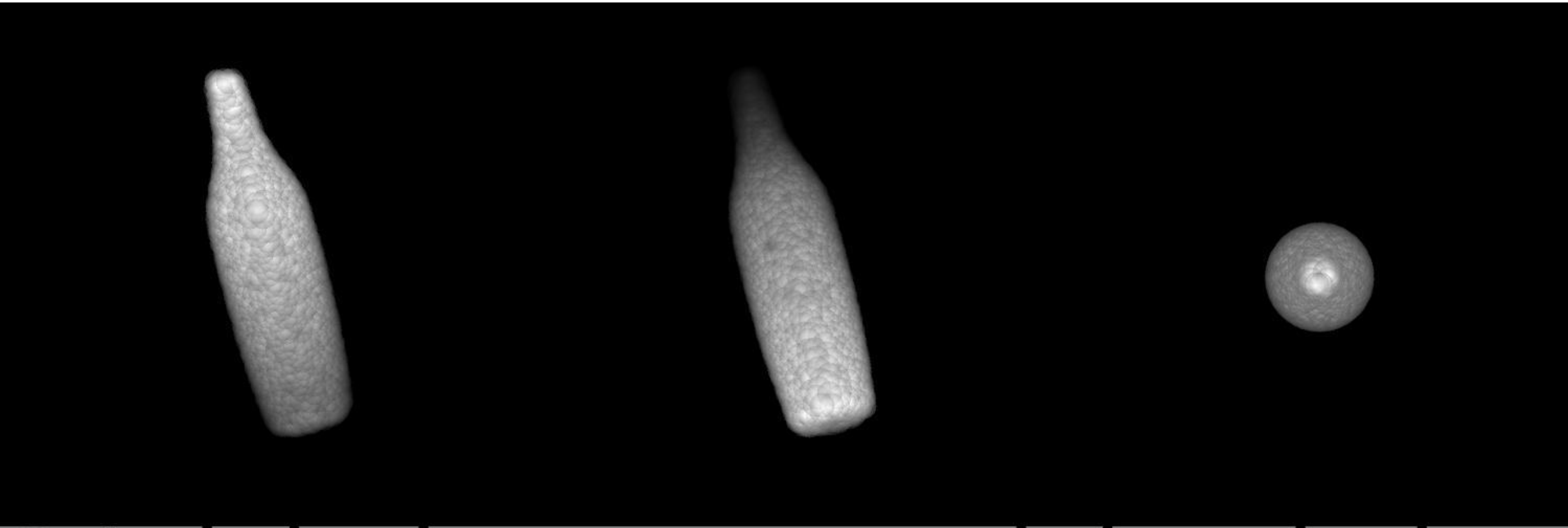
31_label_flower_pot_pred_plant.jpg



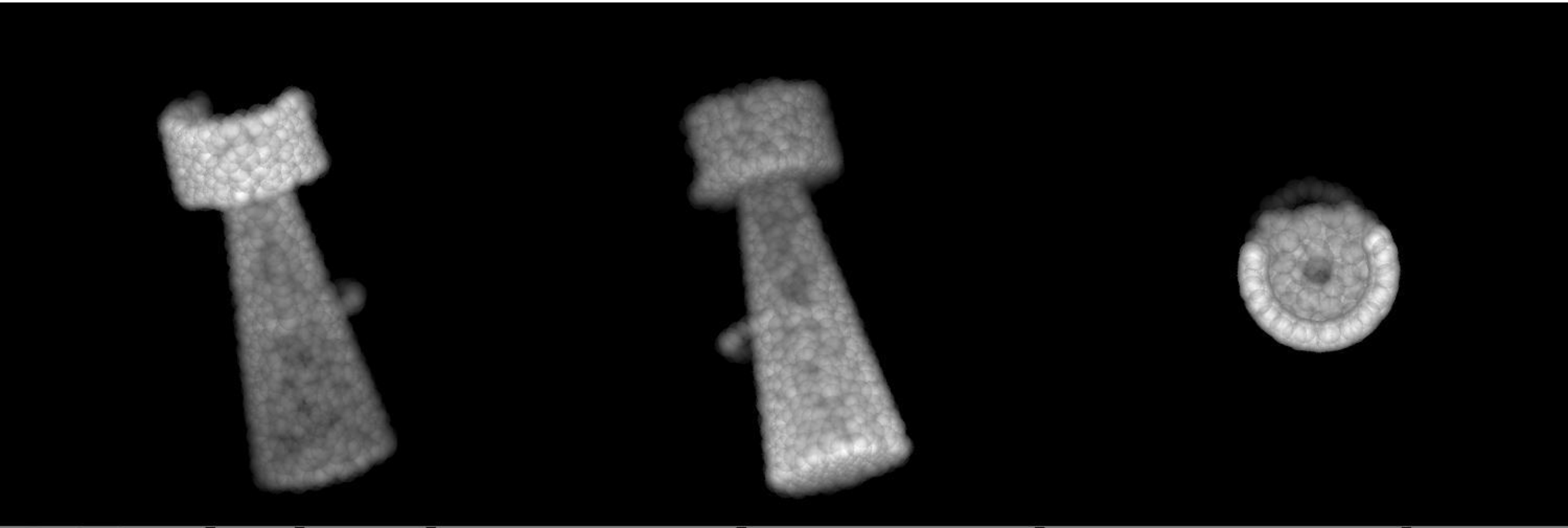
32_label_bookshelf_pred_door.jpg



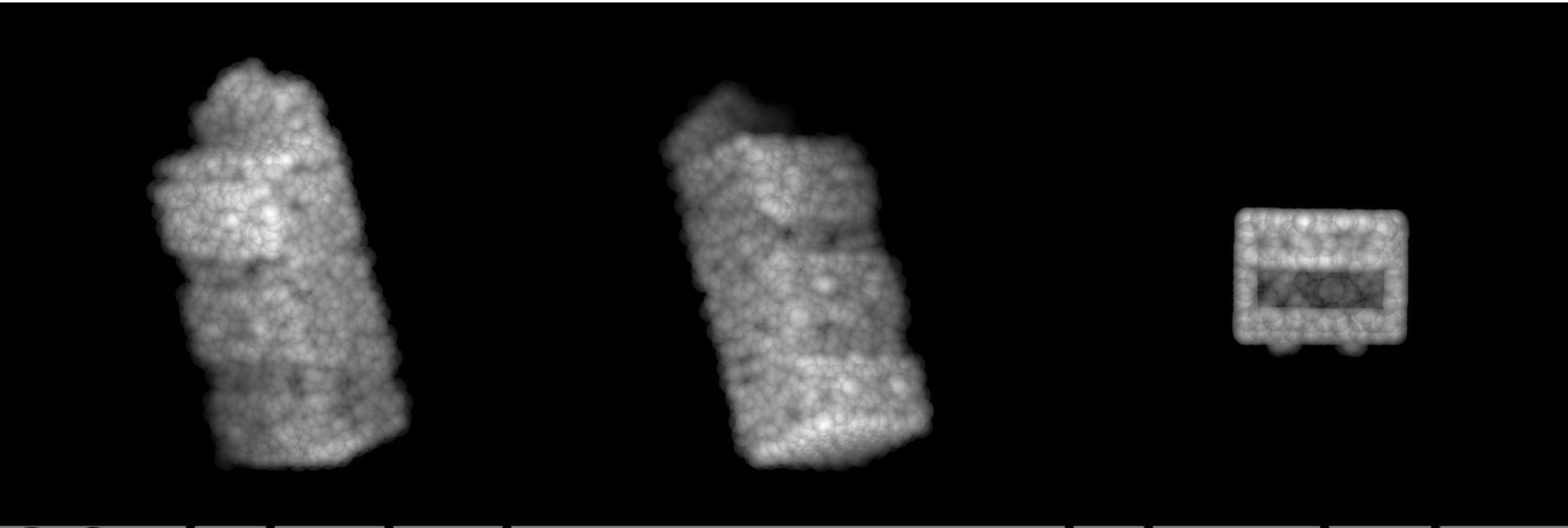
33_label_table_pred_desk.jpg



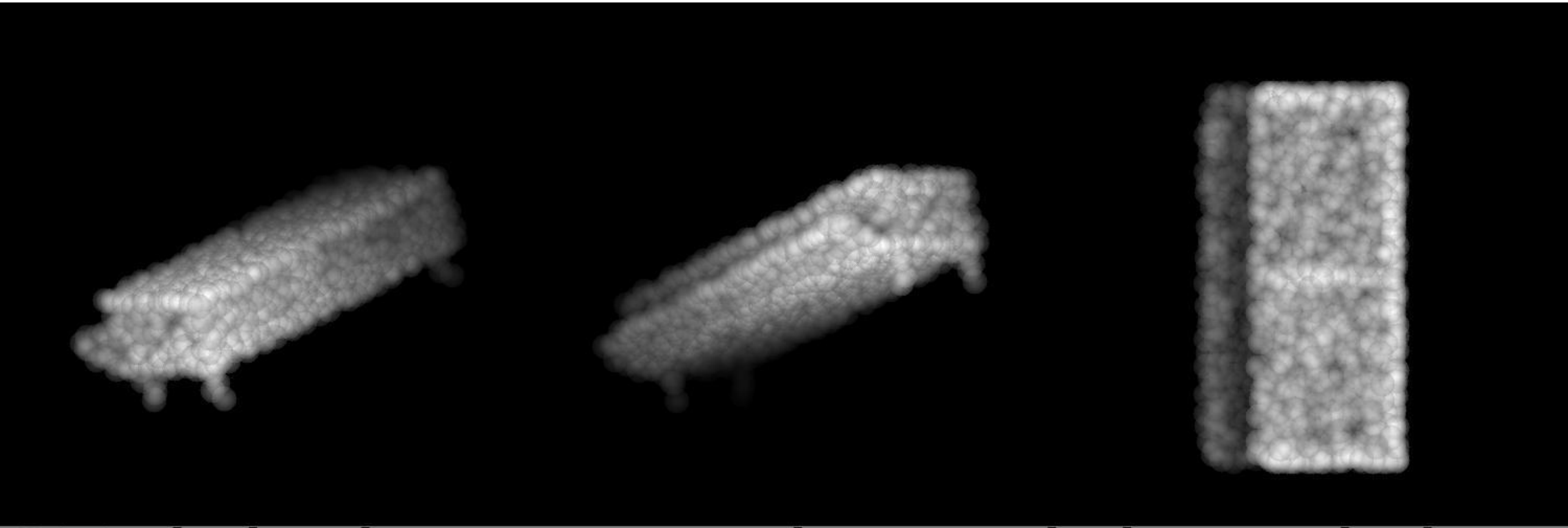
34_label_vase_pred_bottle.jpg



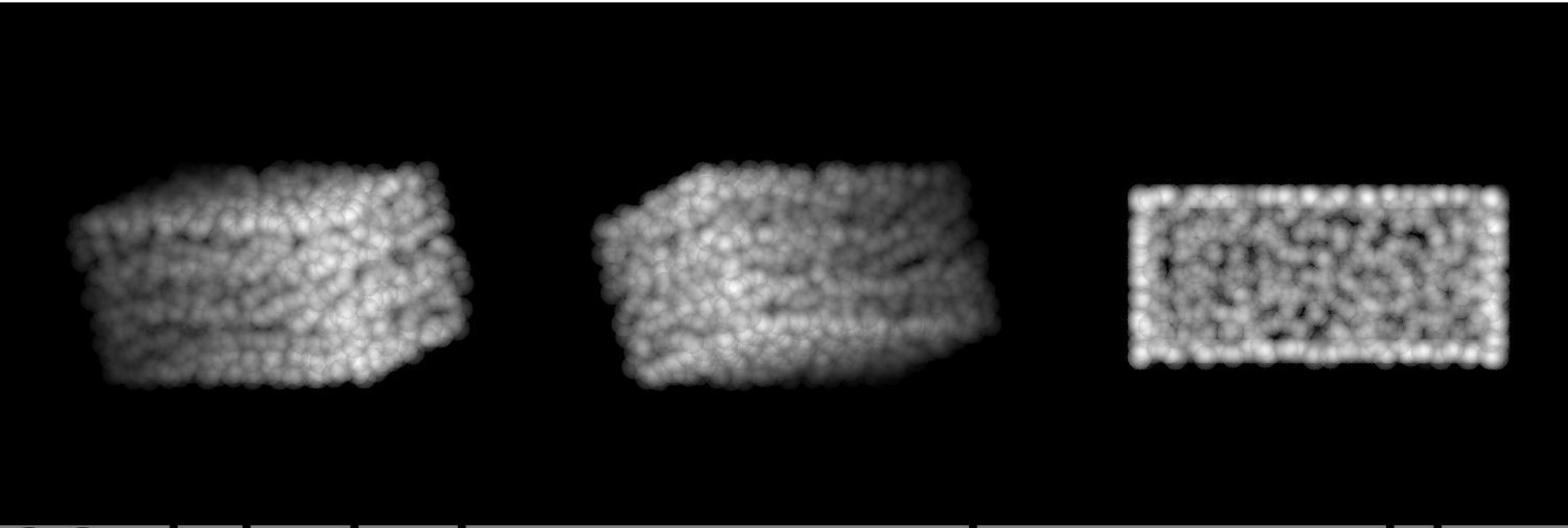
35_label_stool_pred_vase.jpg



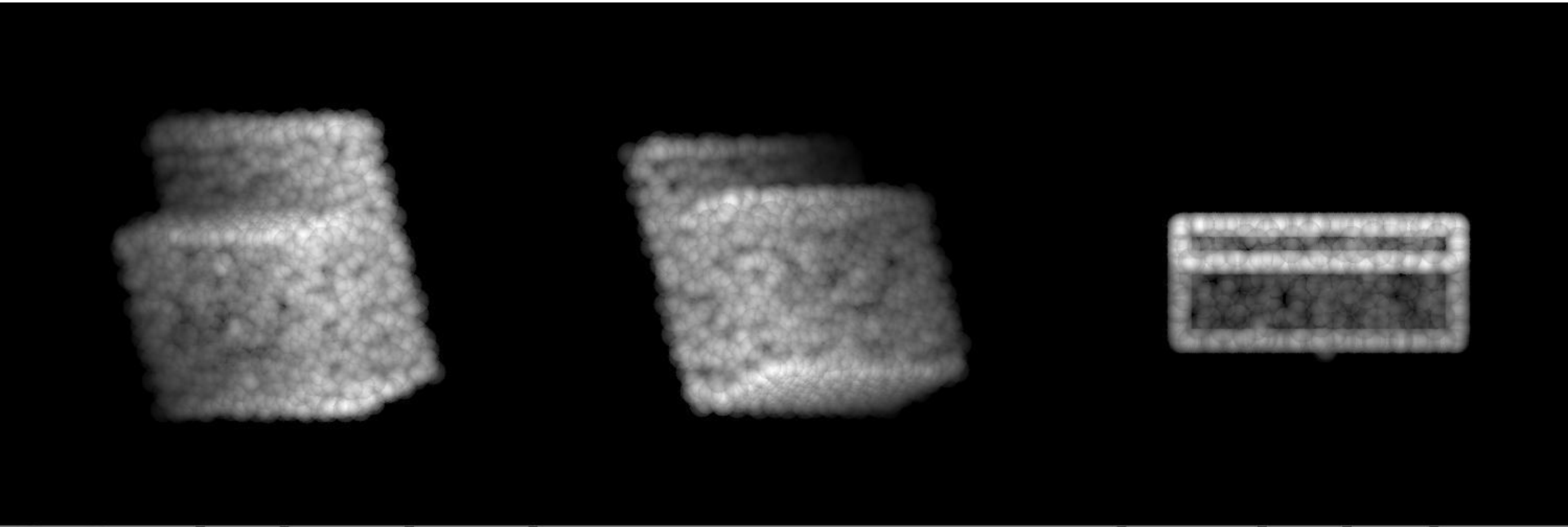
36_label_dresser_pred_bottle.jpg



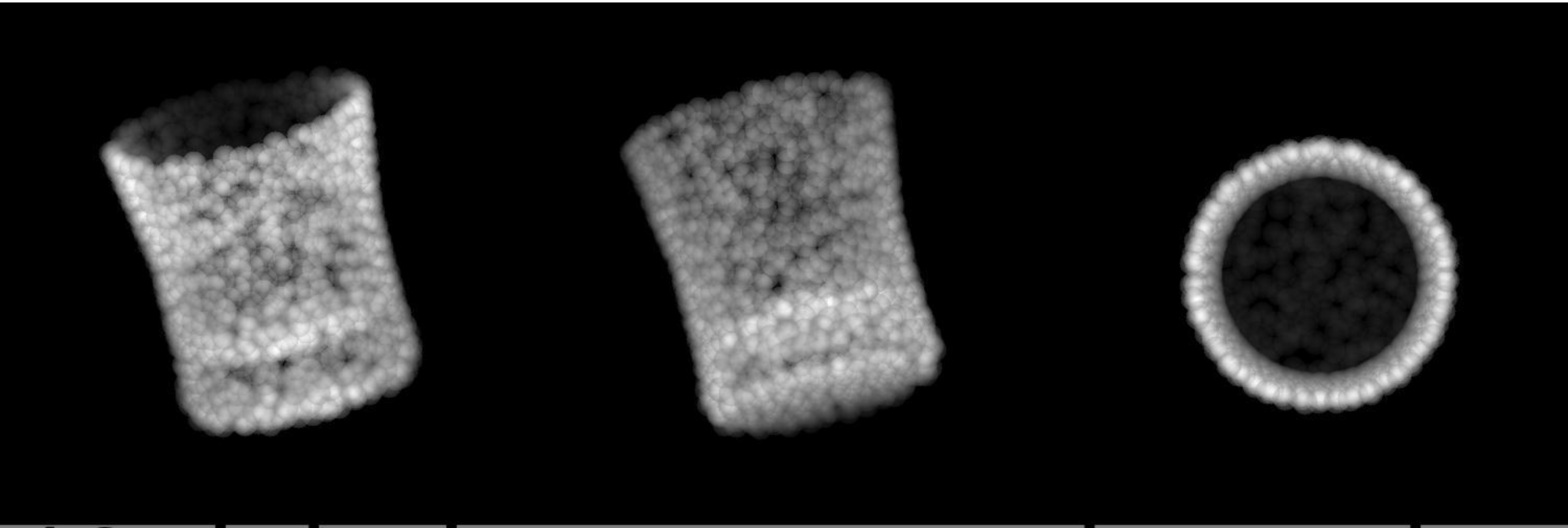
37_label_tv_stand_pred_bench.jpg



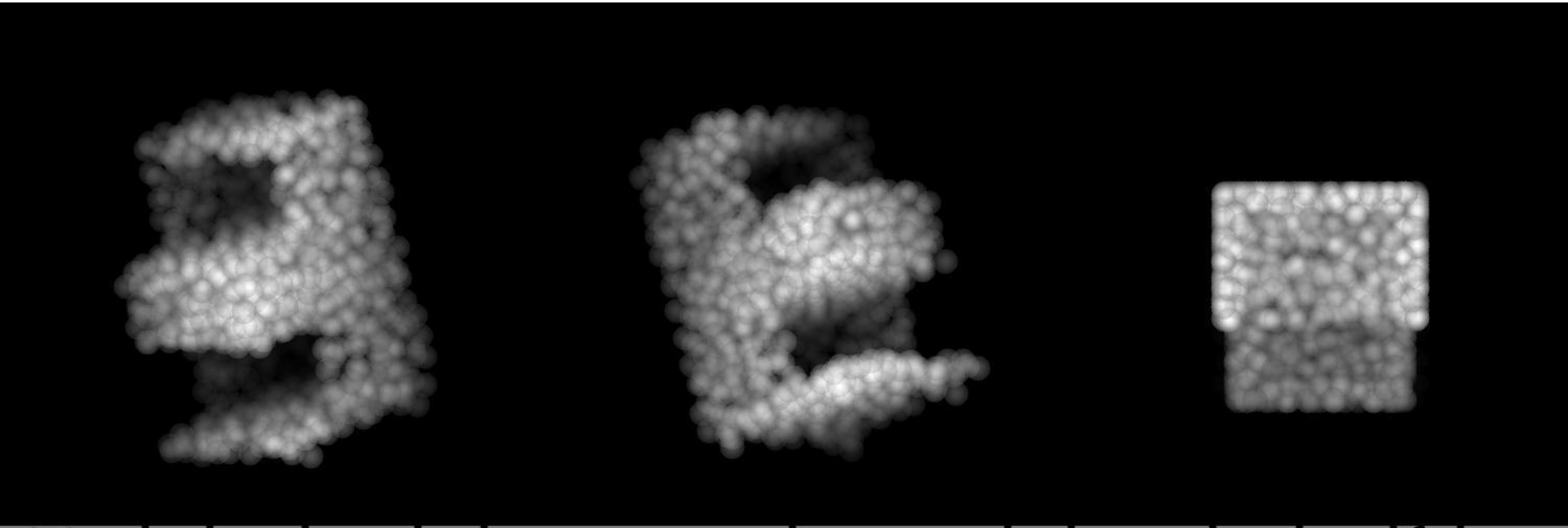
38_label_dresser_pred_tv_stand.jpg



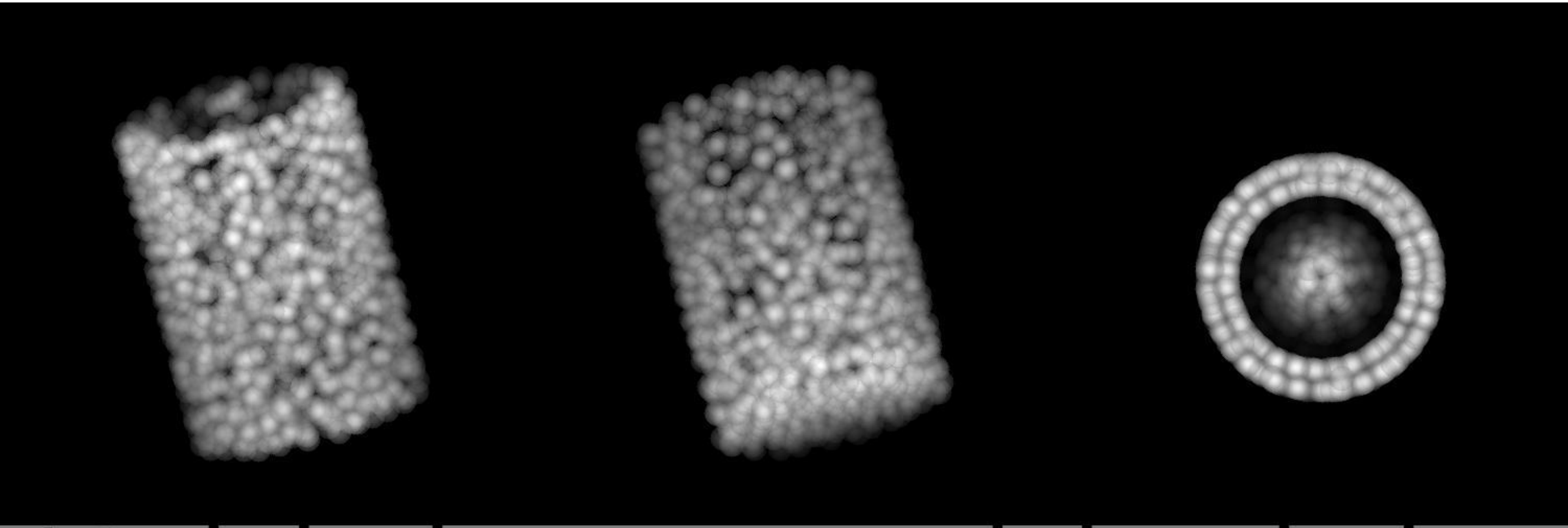
39_label_dresser_pred_sink.jpg



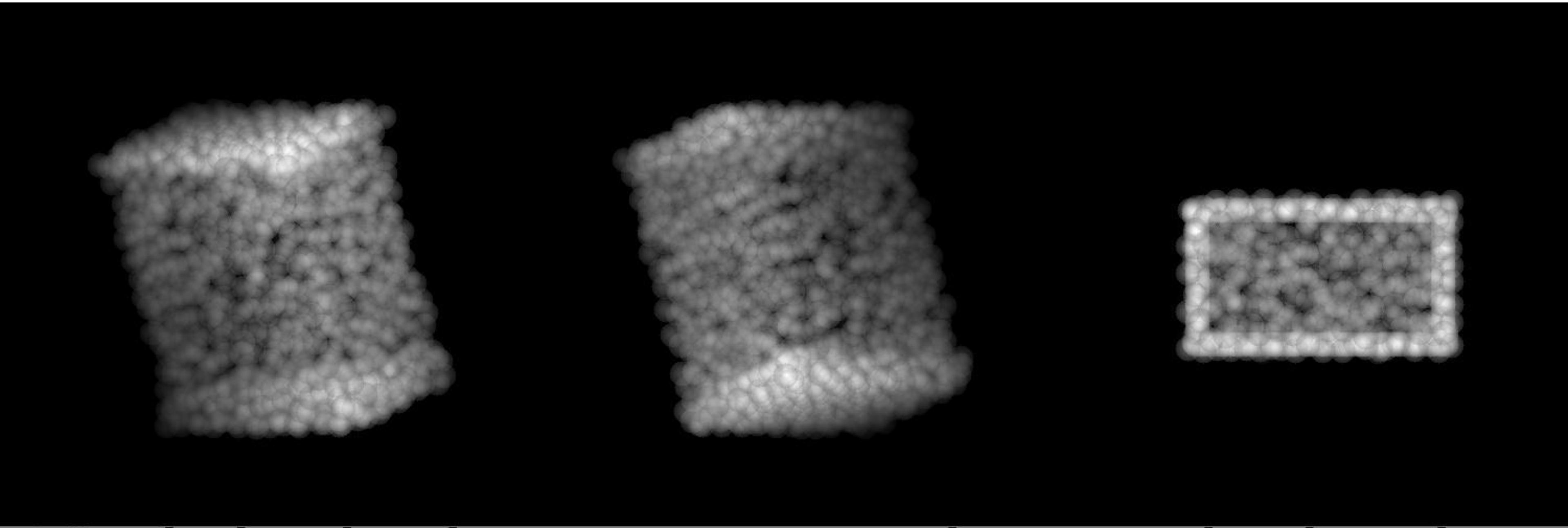
40_label_vase_pred_cup.jpg



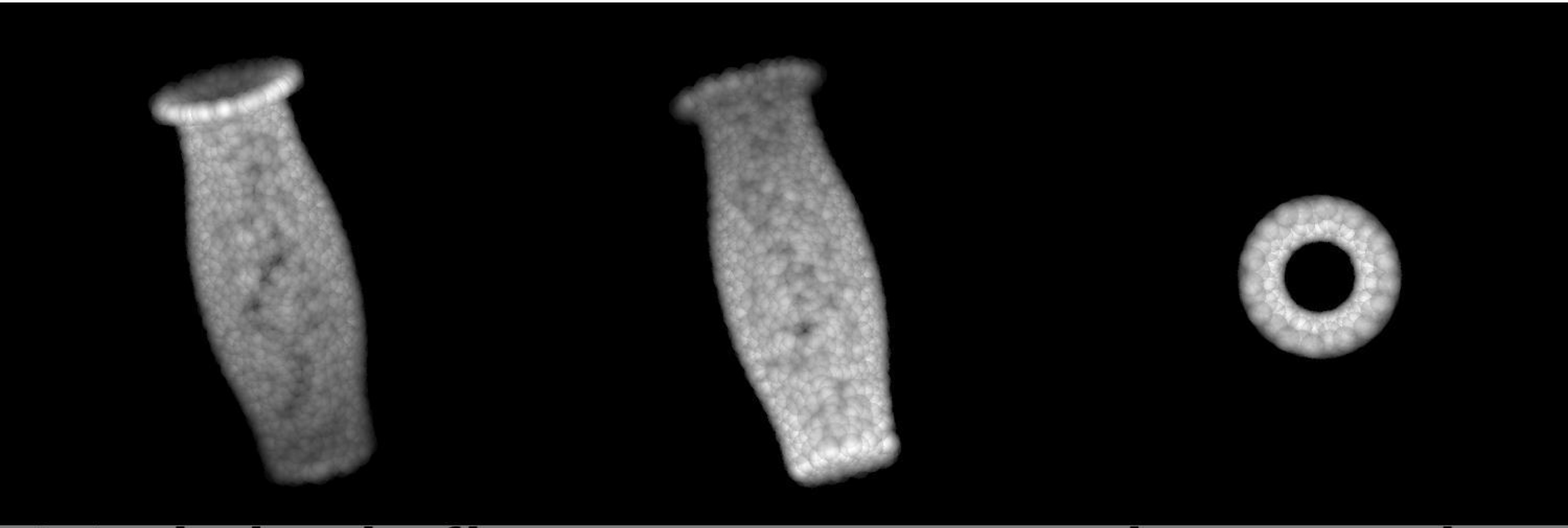
41_label_night_stand_pred_bookshelf.jpg



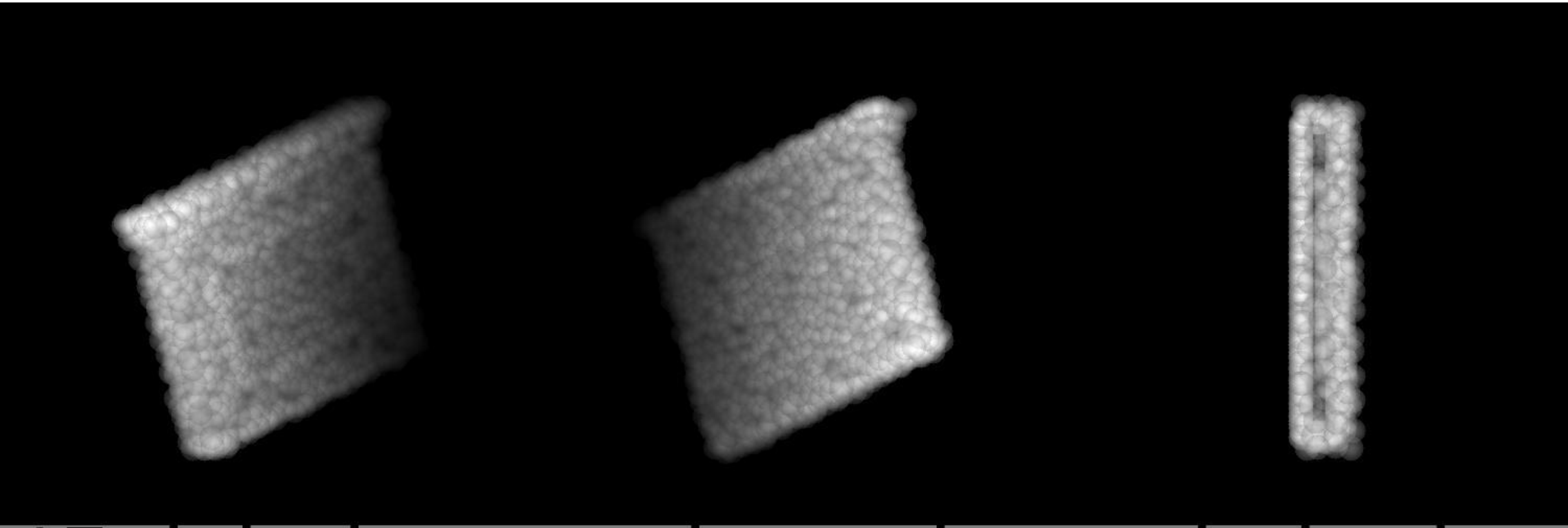
42_label_cup_pred_bottle.jpg



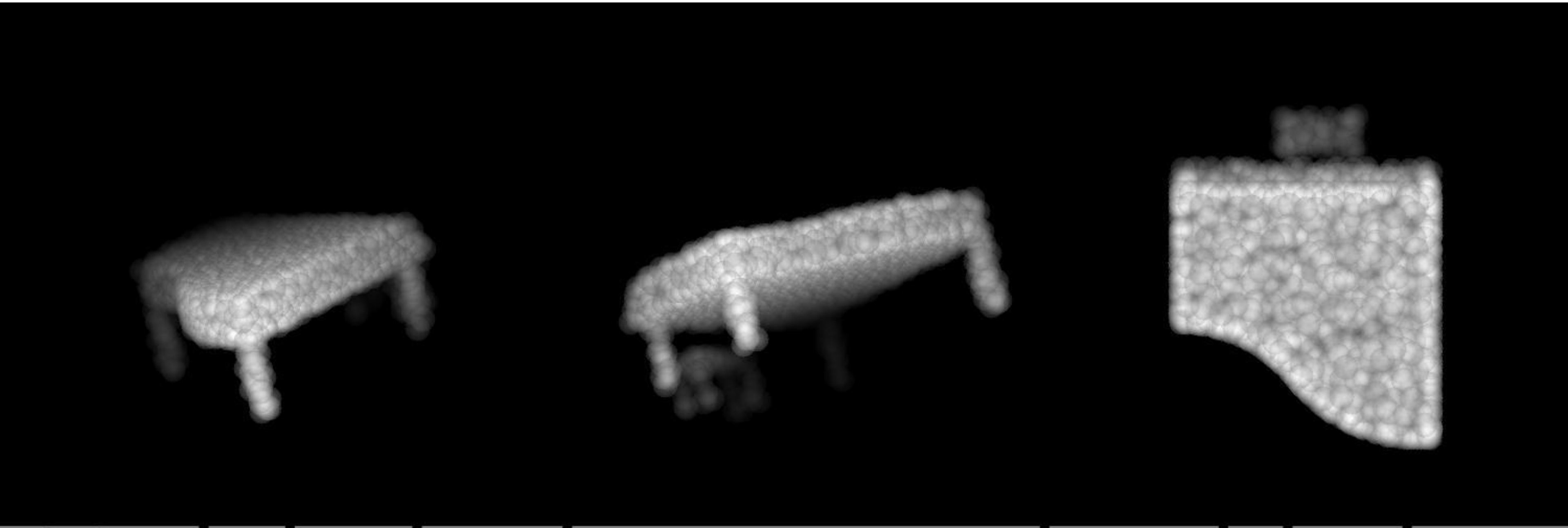
43_label_dresser_pred_wardrobe.jpg

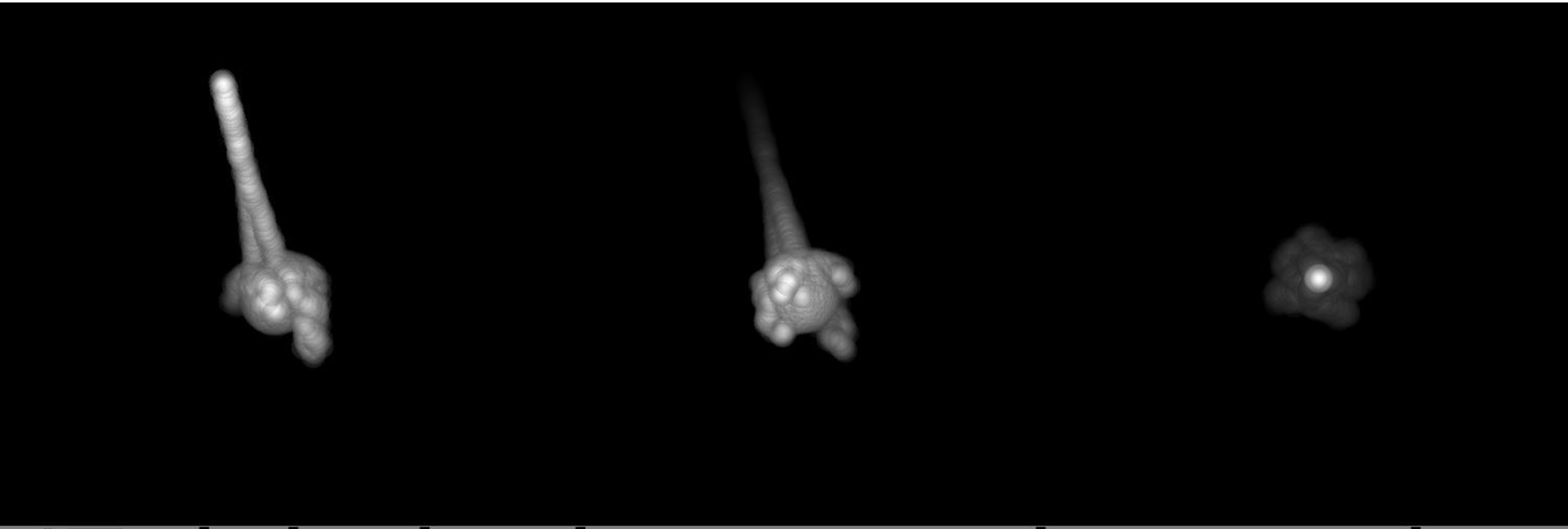


44_label_flower_pot_pred_vase.jpg

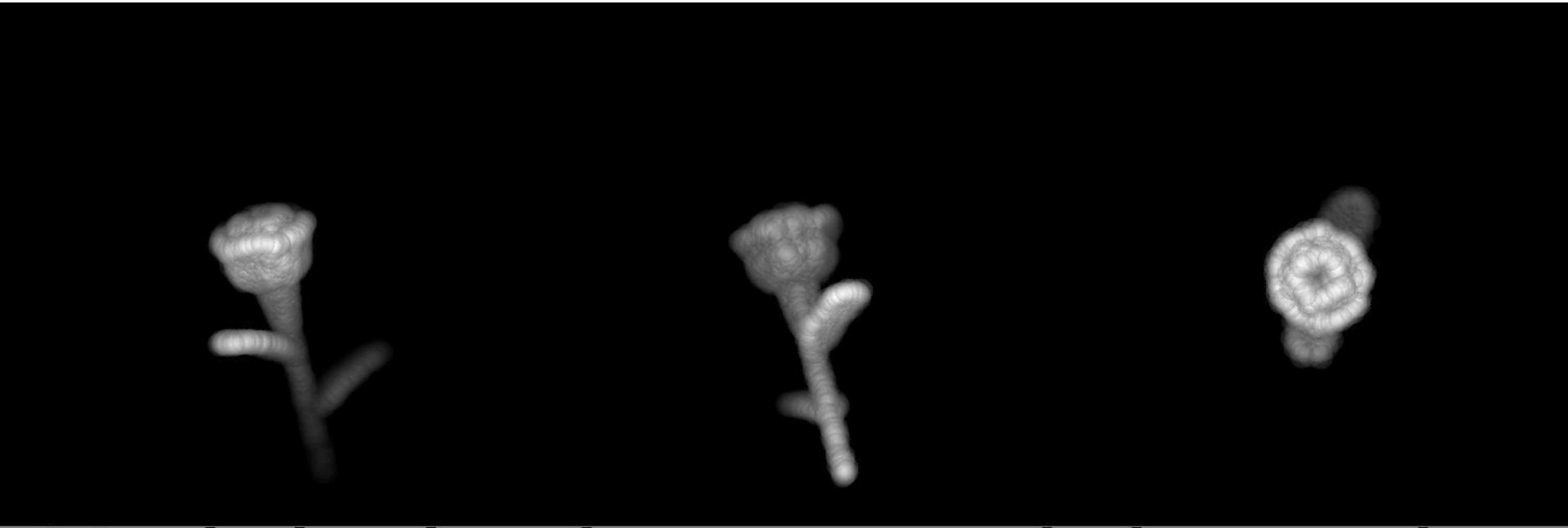


45_label_mantel_pred_wardrobe.jpg

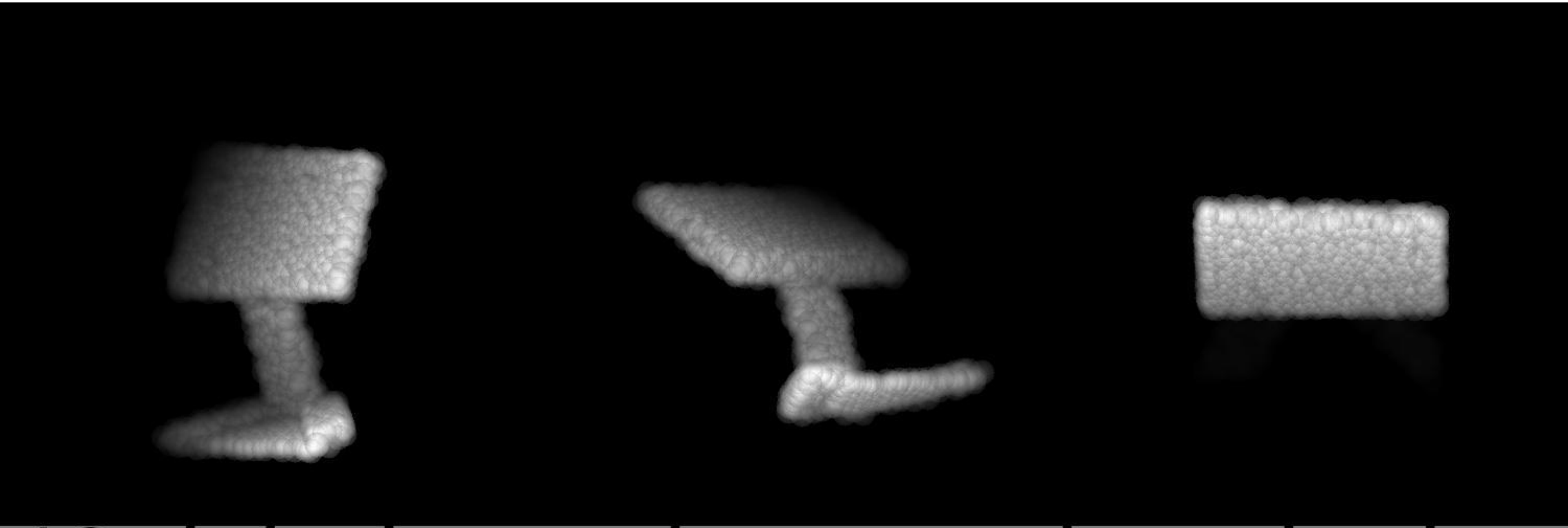

46_label_piano_pred_table.jpg



47_label_plant_pred_cone.jpg

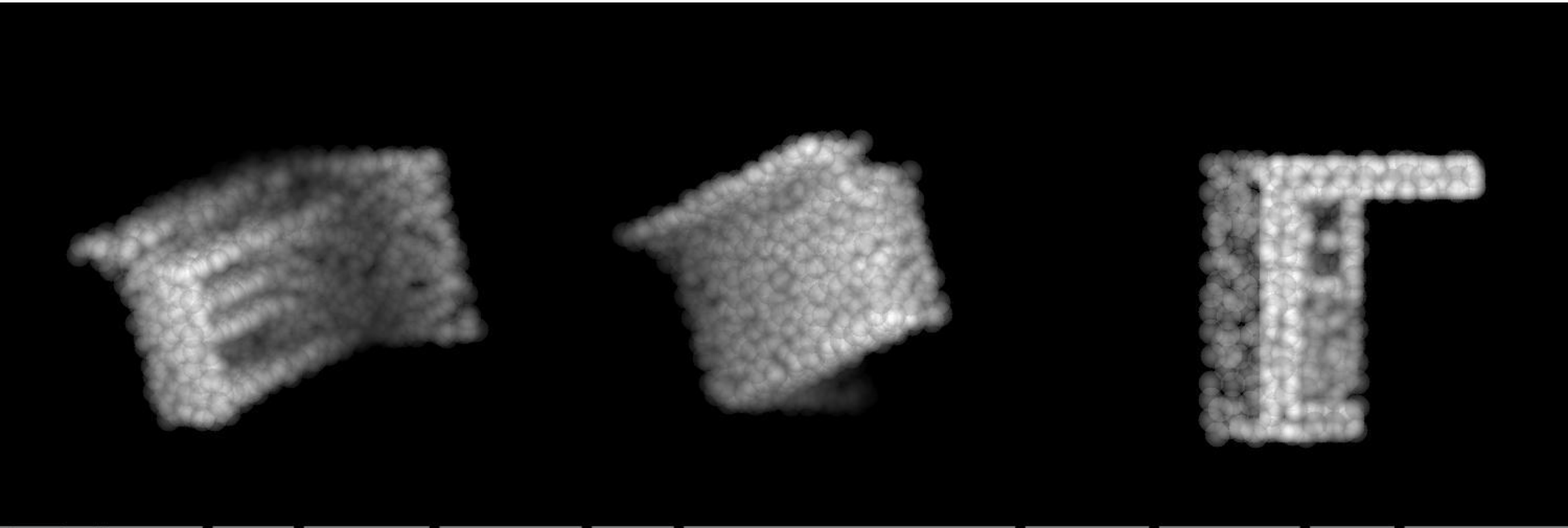


48_label_plant_pred_lamp.jpg

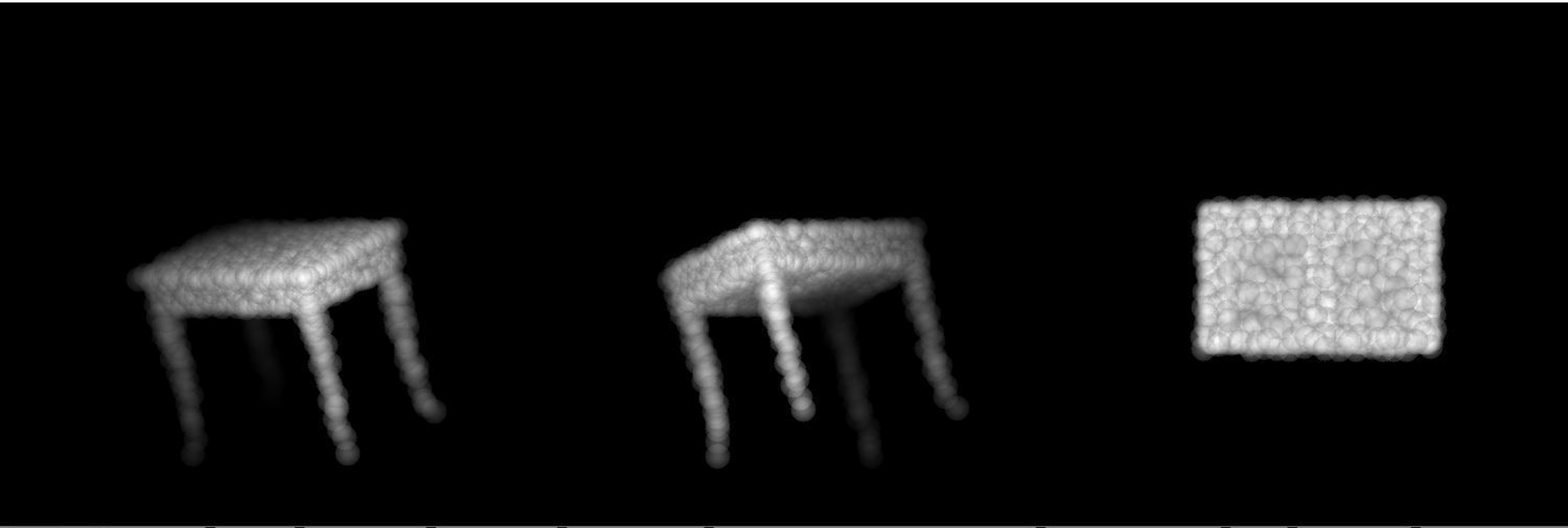


49_label_monitor_pred_stairs.jpg

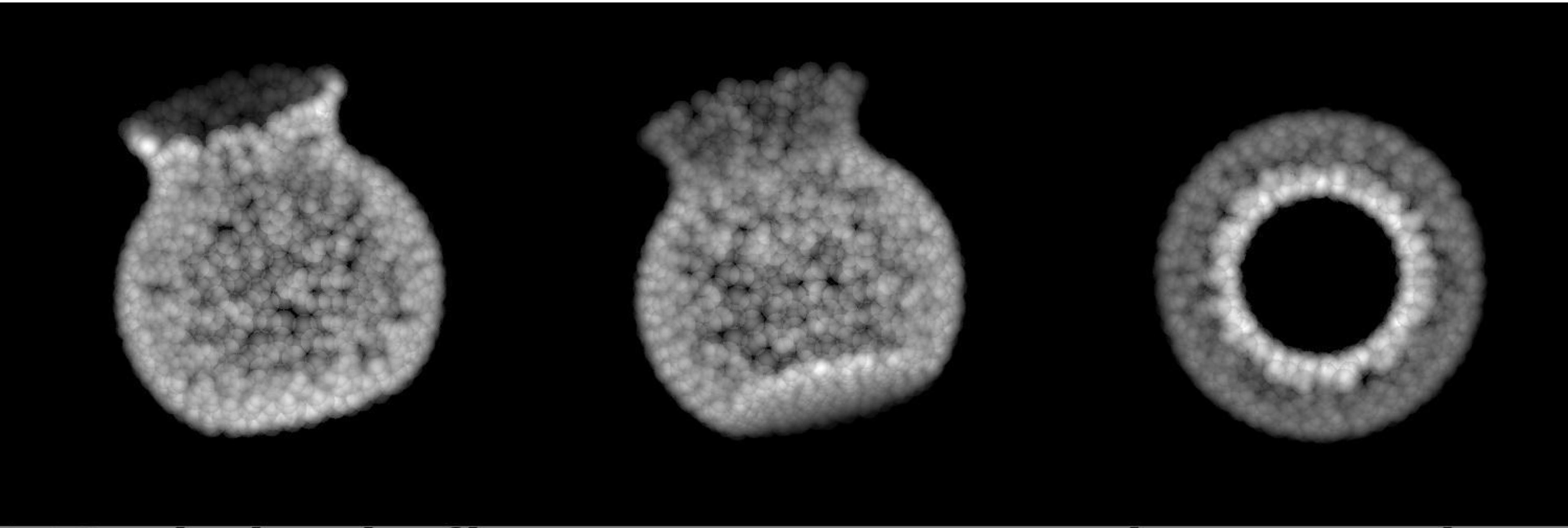
50_label_tent_pred_sofa.jpg

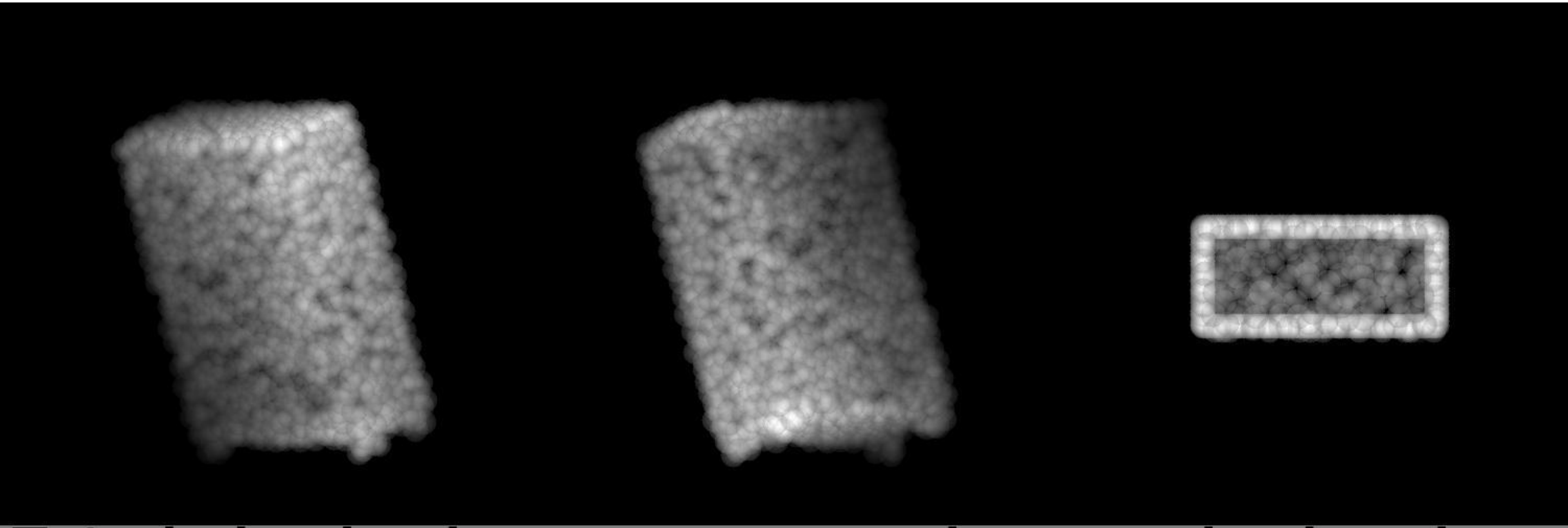


51_label_sink_pred_desk.jpg

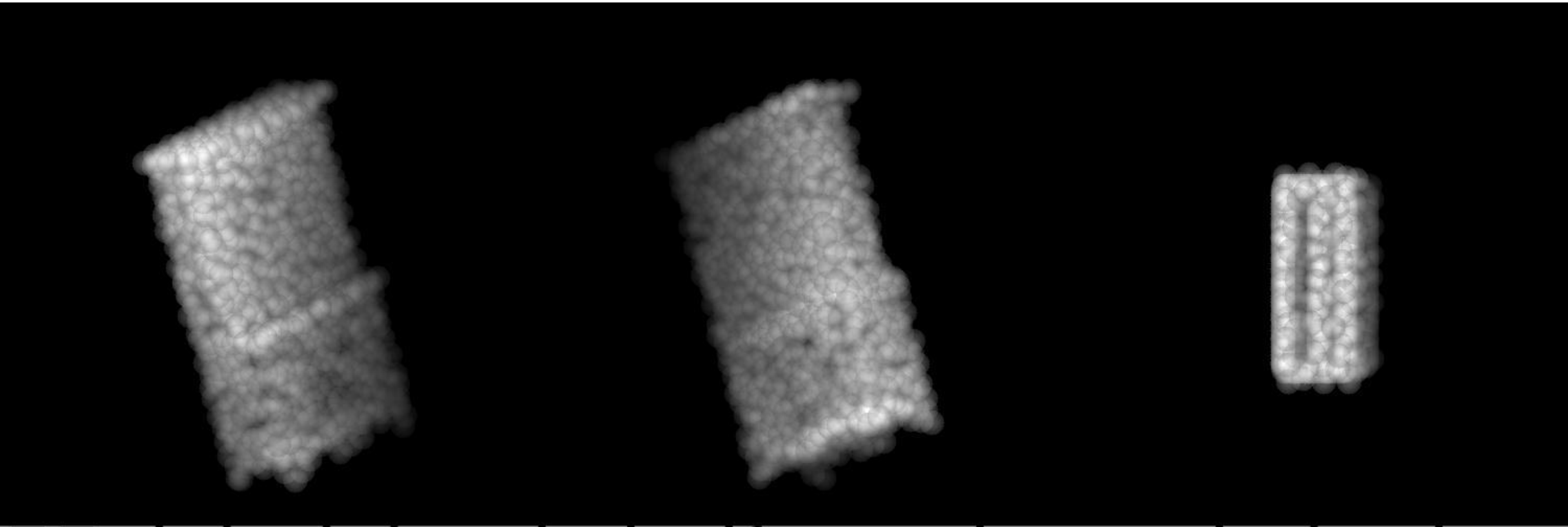


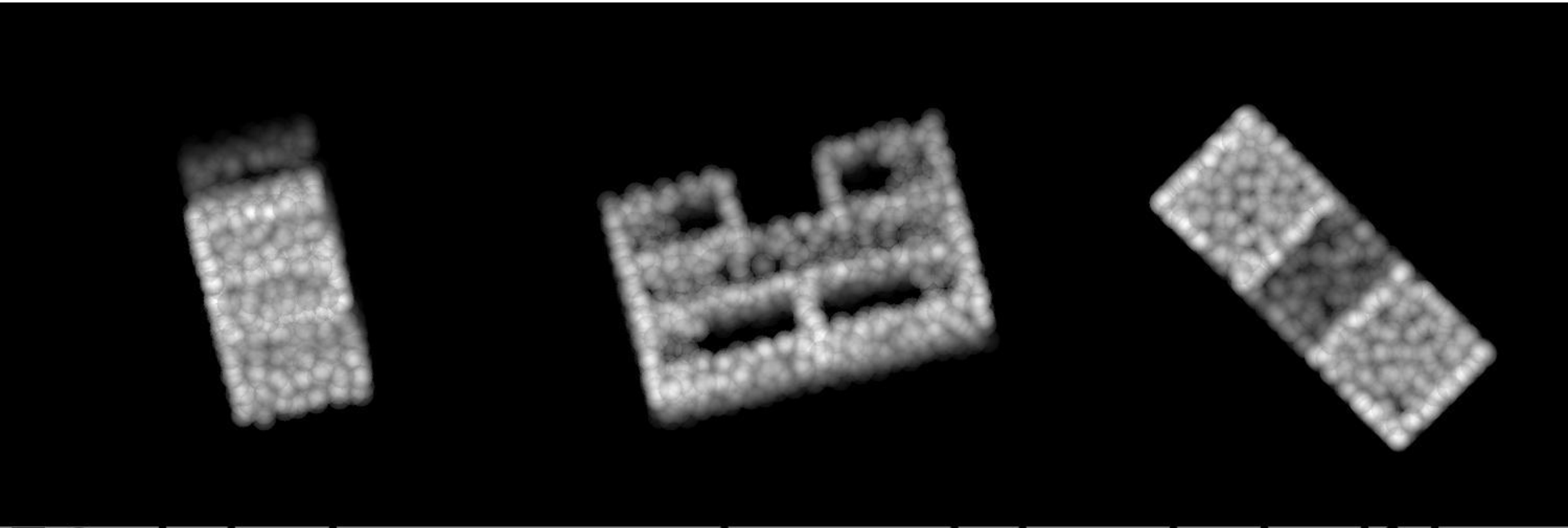
52_label_desk_pred_table.jpg


53_label_flower_pot_pred_vase.jpg

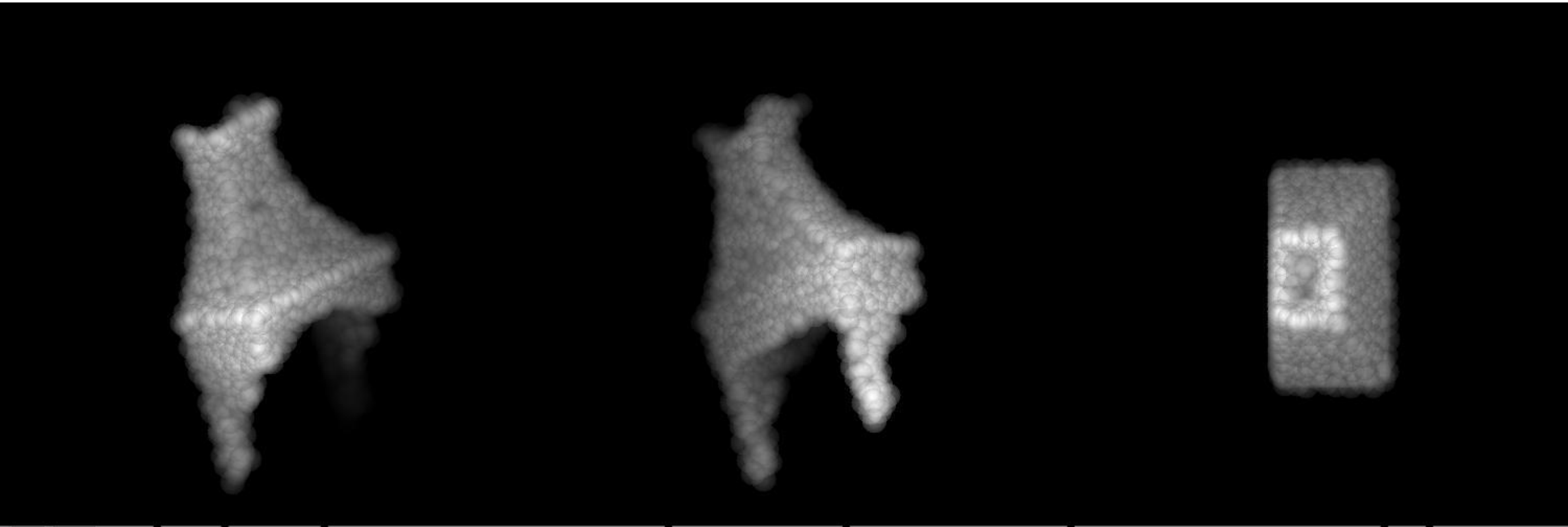


54_label_dresser_pred_wardrobe.jpg

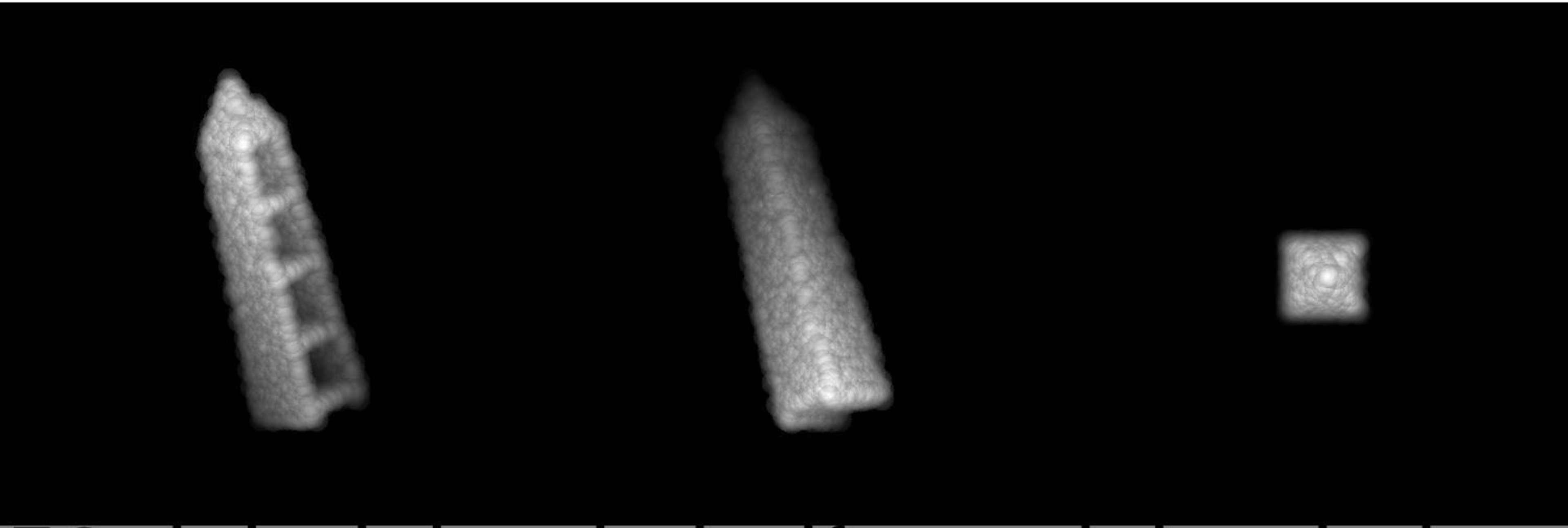

55_label_bookshelf_pred_wardrobe.jpg



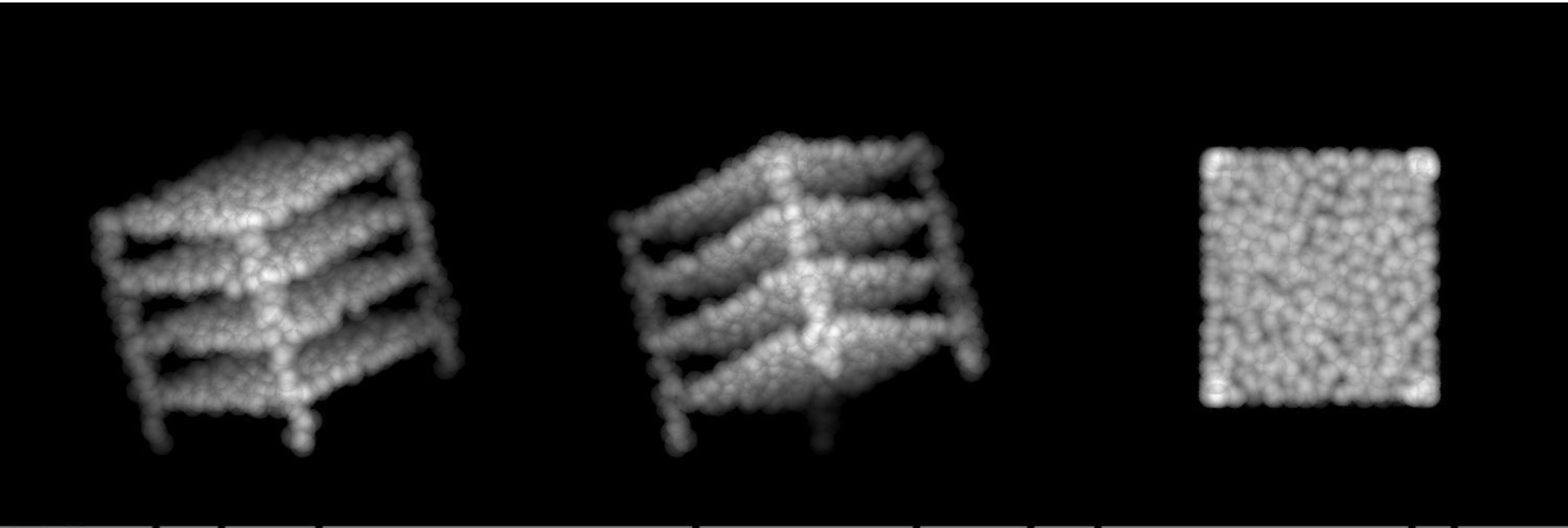
56_label_tv_stand_pred_bookshelf.jpg



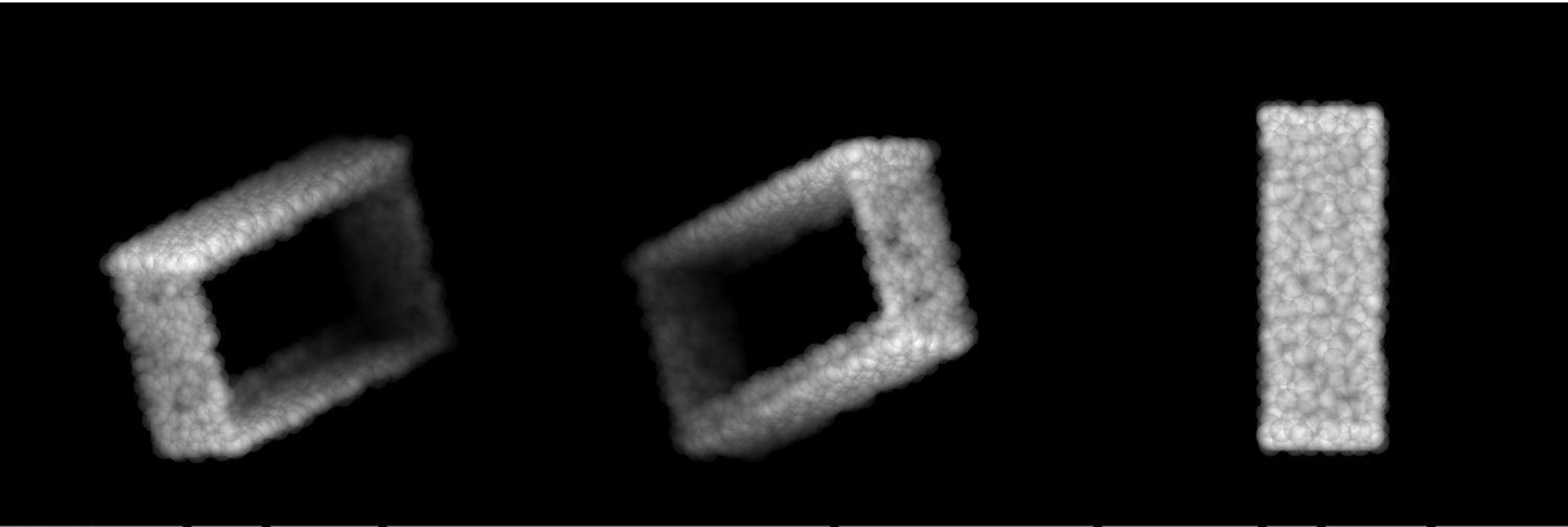
57_label_range_hood_pred_mantel.jpg

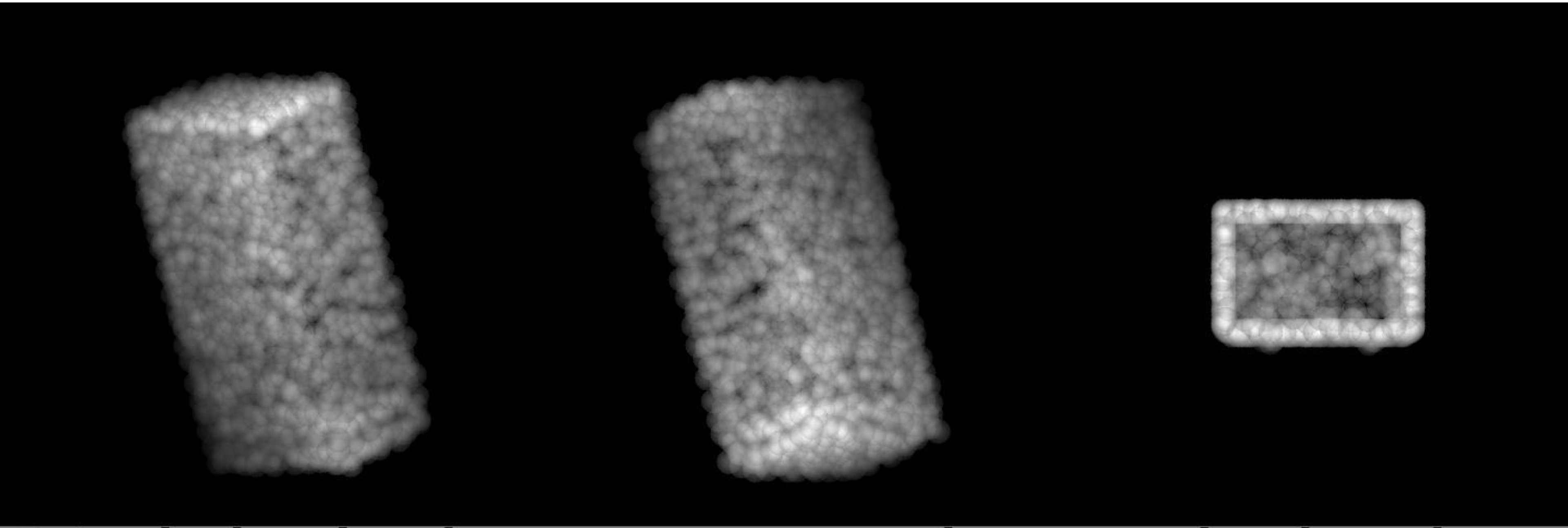


58_label_bookshelf_pred_bottle.jpg

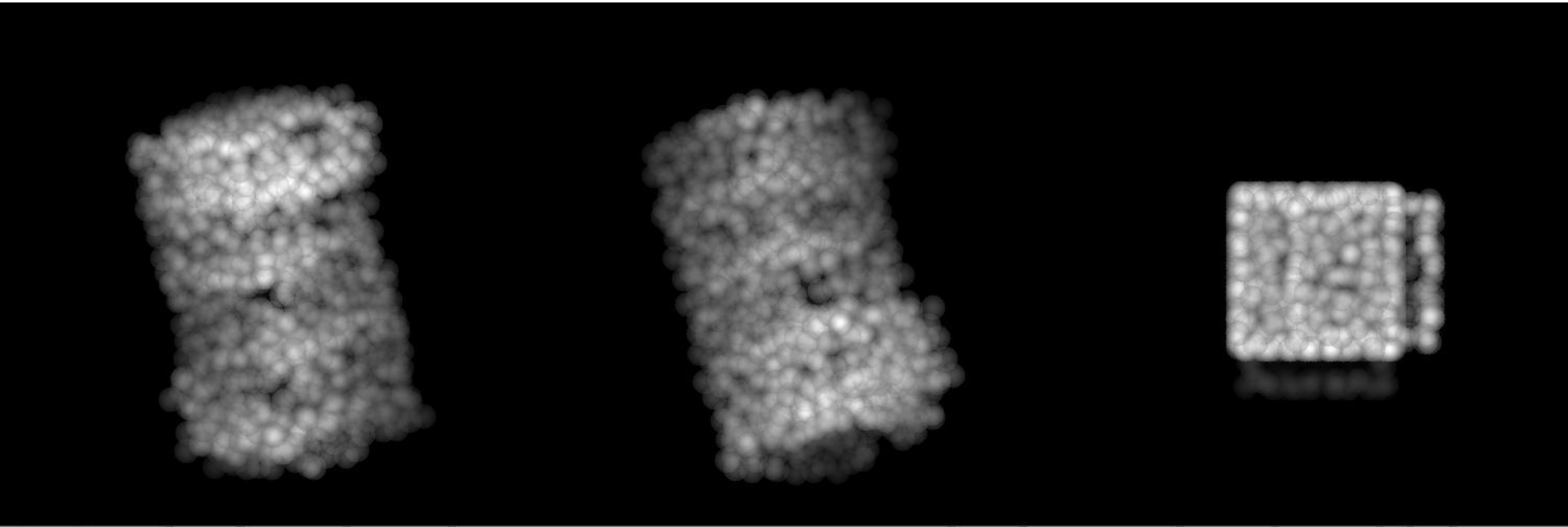


59_label_tv_stand_pred_night_stand.jpg

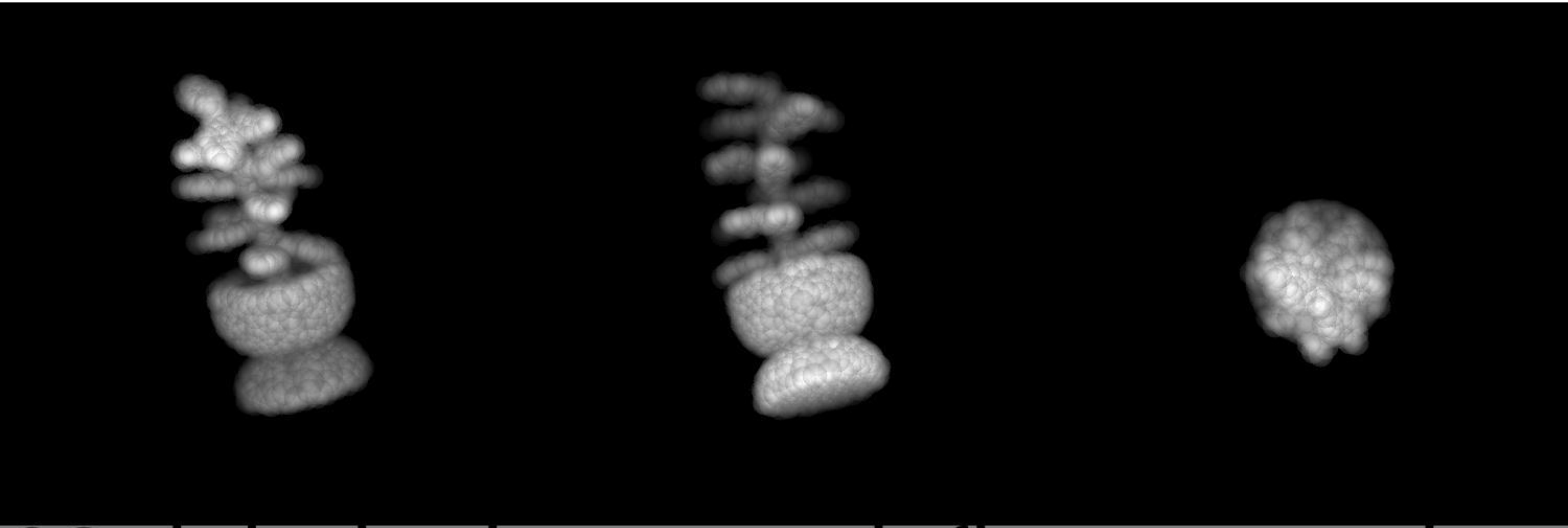

60_label_tv_stand_pred_table.jpg



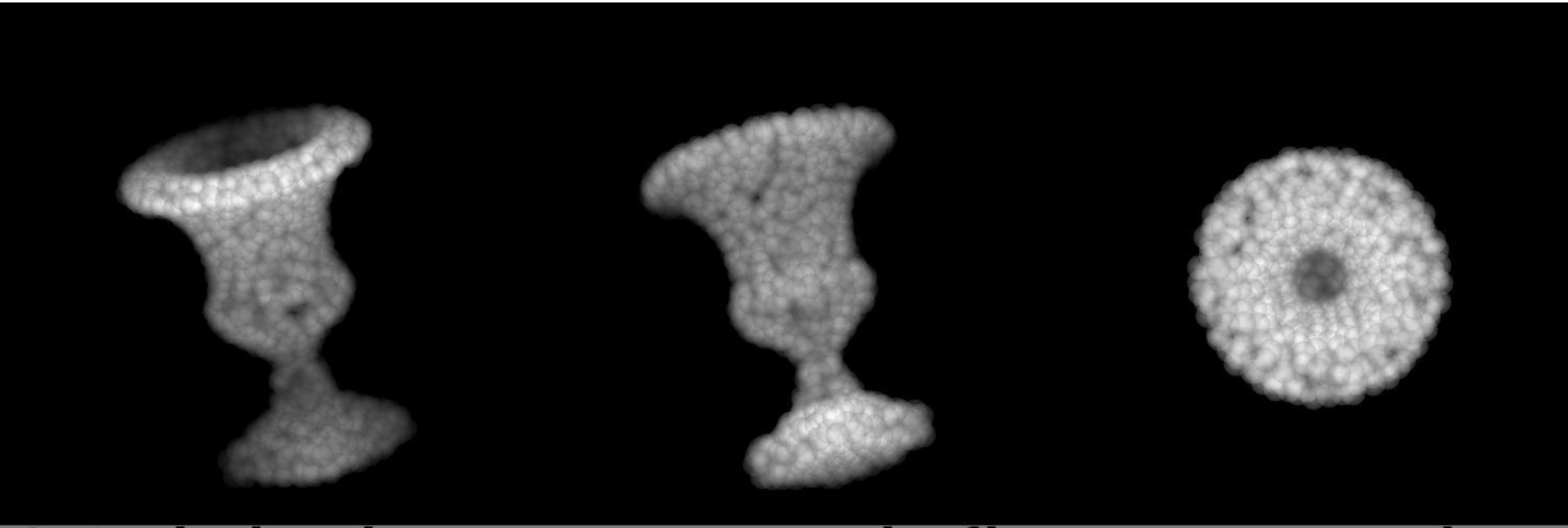
61_label_dresser_pred_wardrobe.jpg



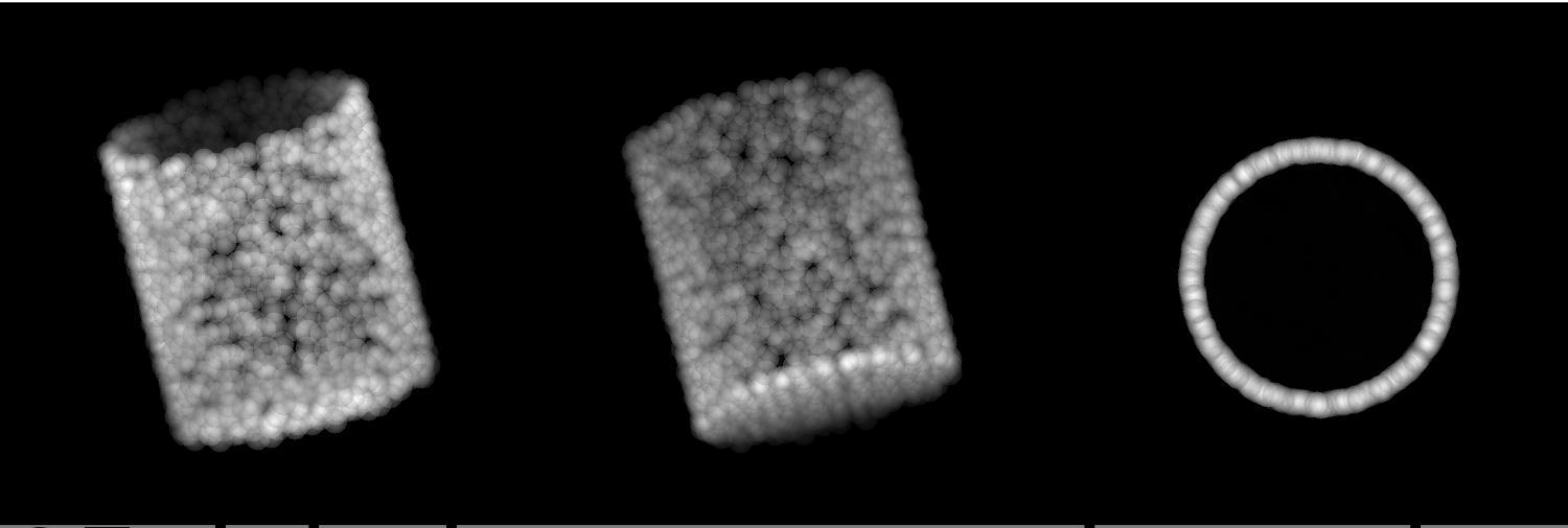
62_label_dresser_pred_bookshelf.jpg



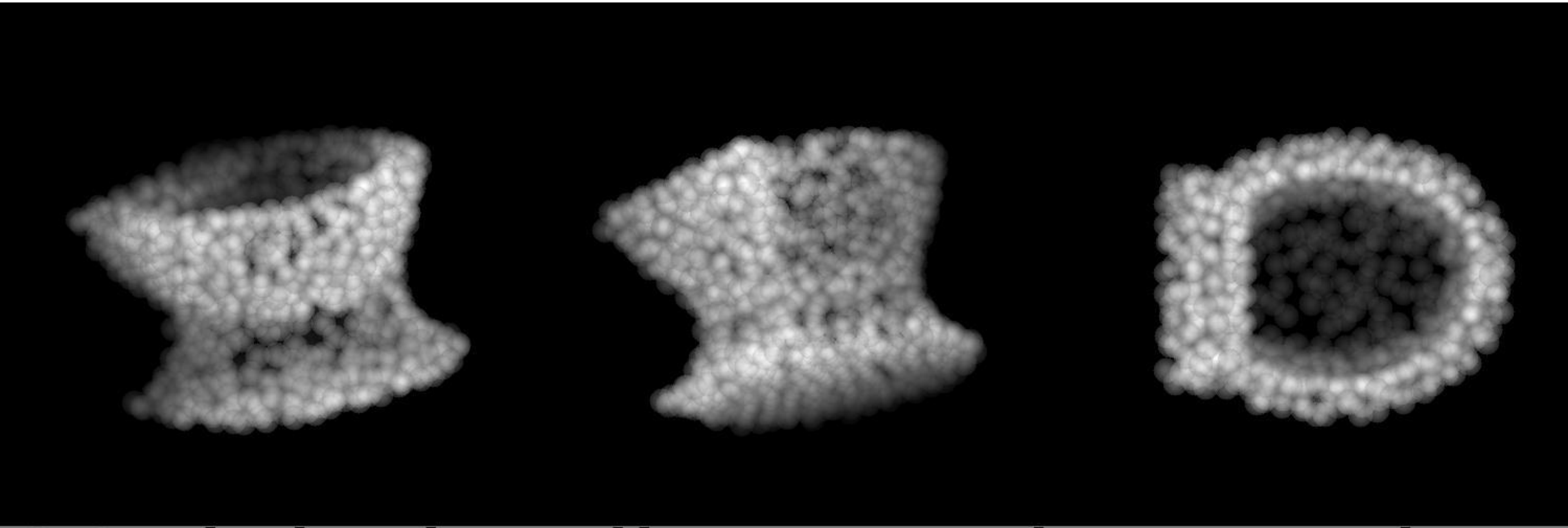
63_label_plant_pred_flower_pot.jpg

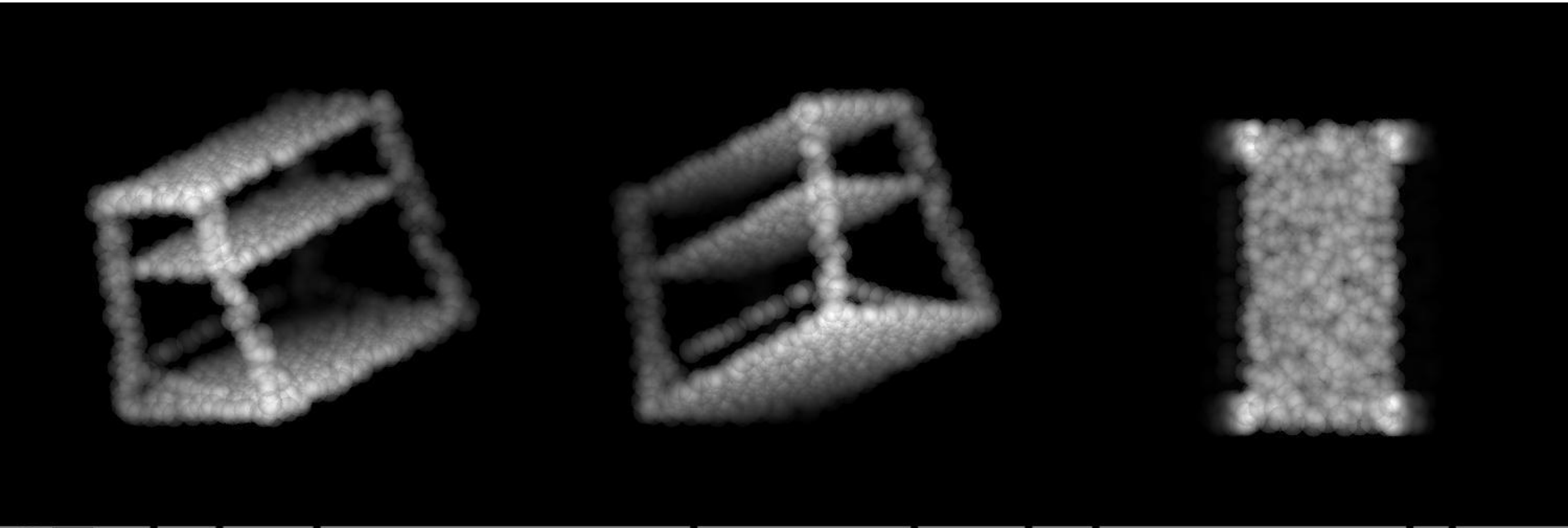


64_label_vase_pred_flower_pot.jpg

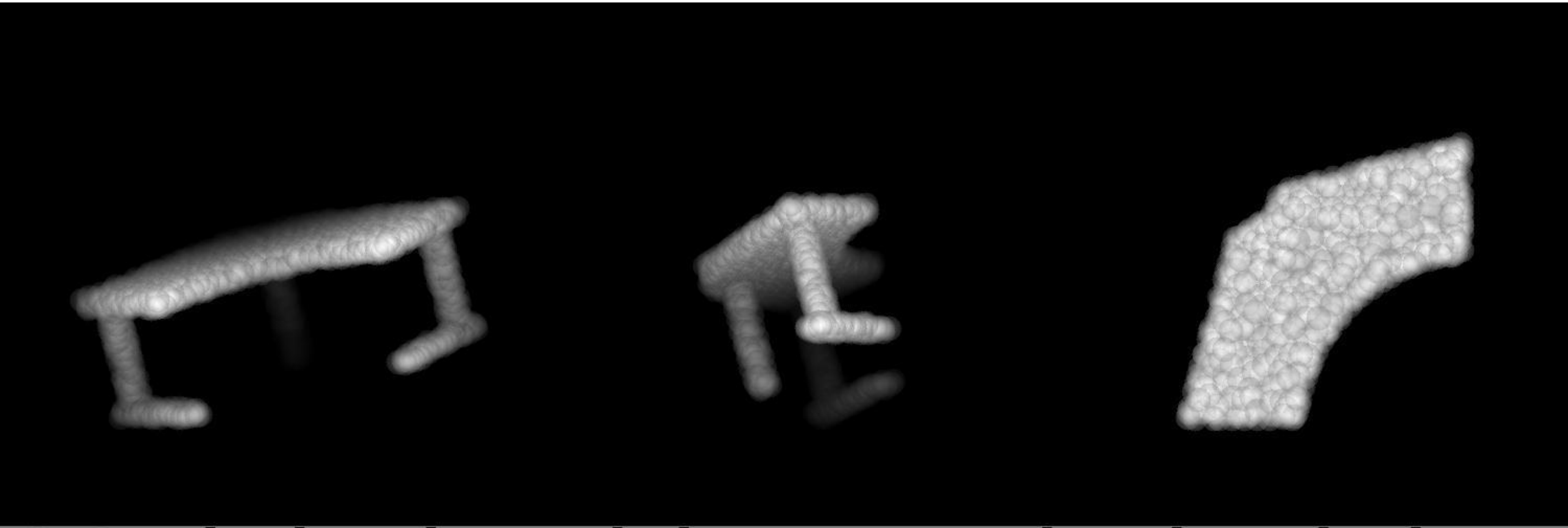


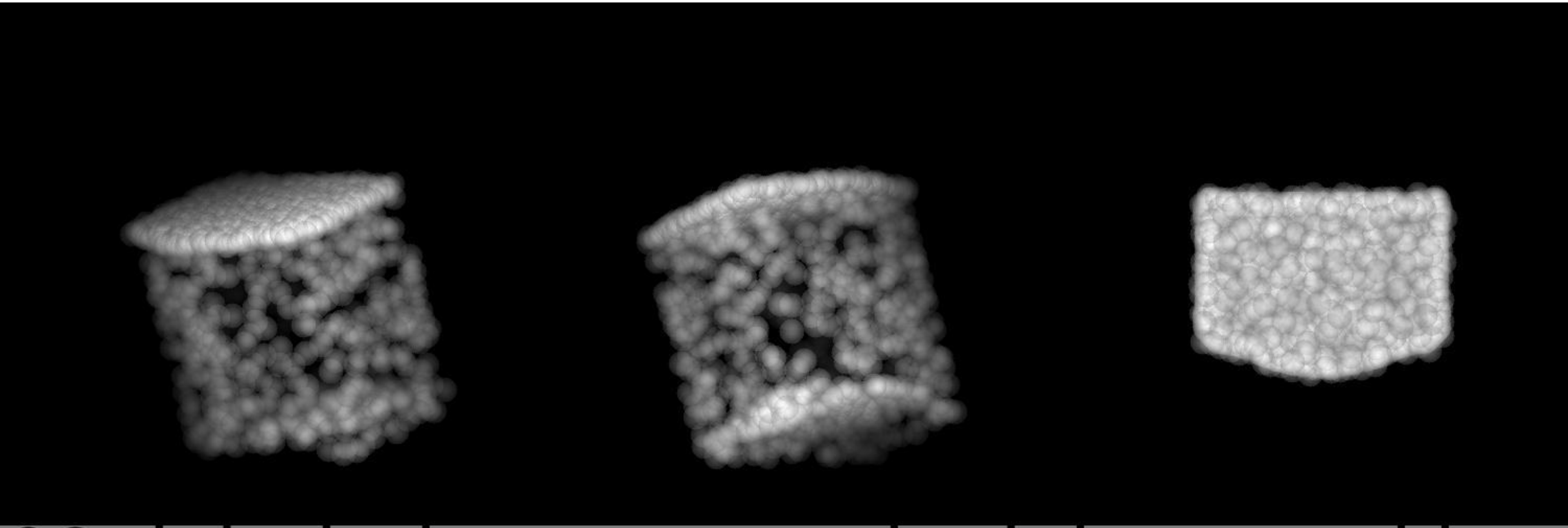
65_label_vase_pred_cup.jpg


66_label_toilet_pred_vase.jpg

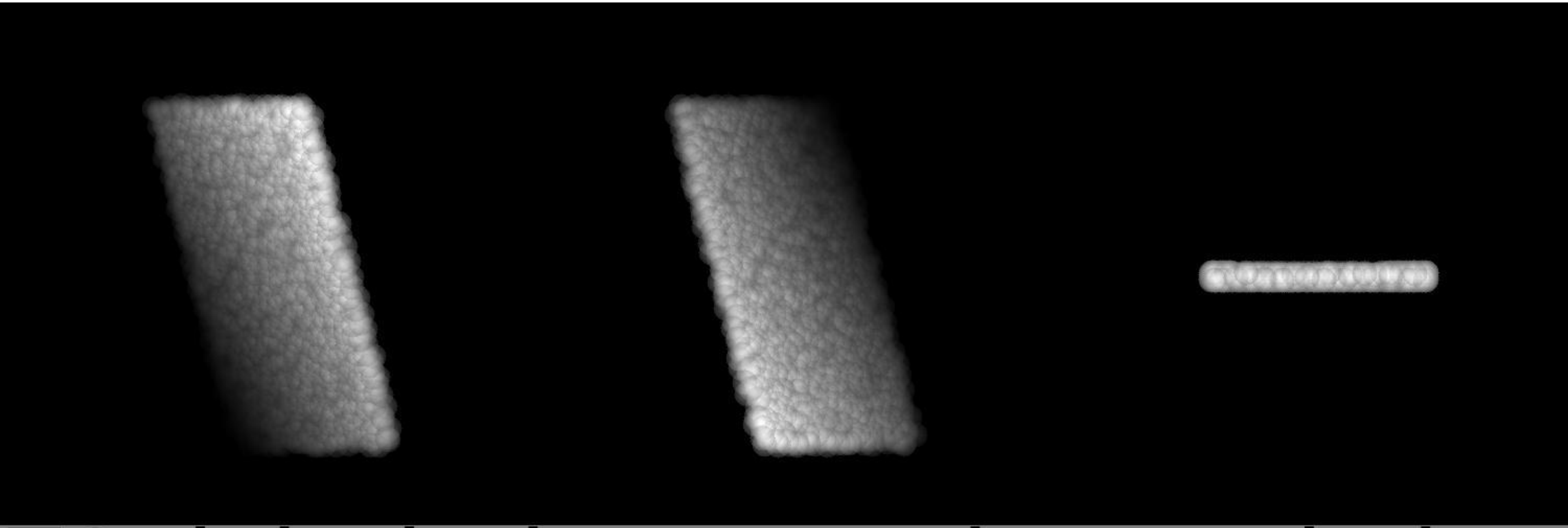


67_label_tv_stand_pred_night_stand.jpg

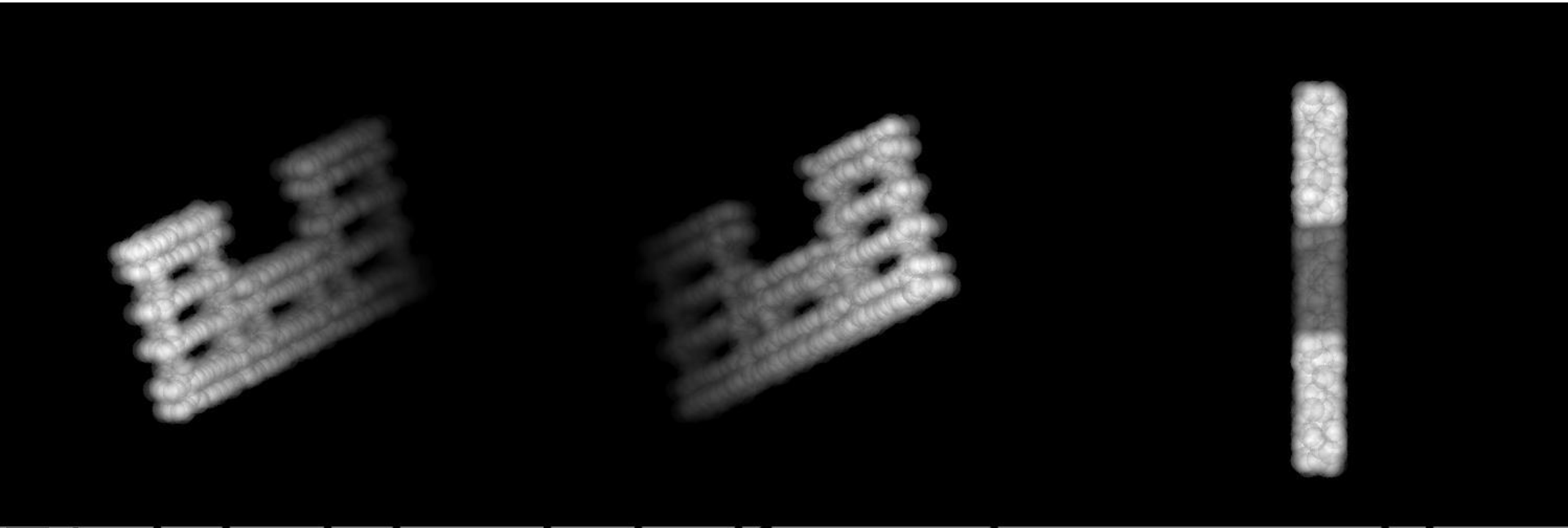

68_label_table_pred_desk.jpg



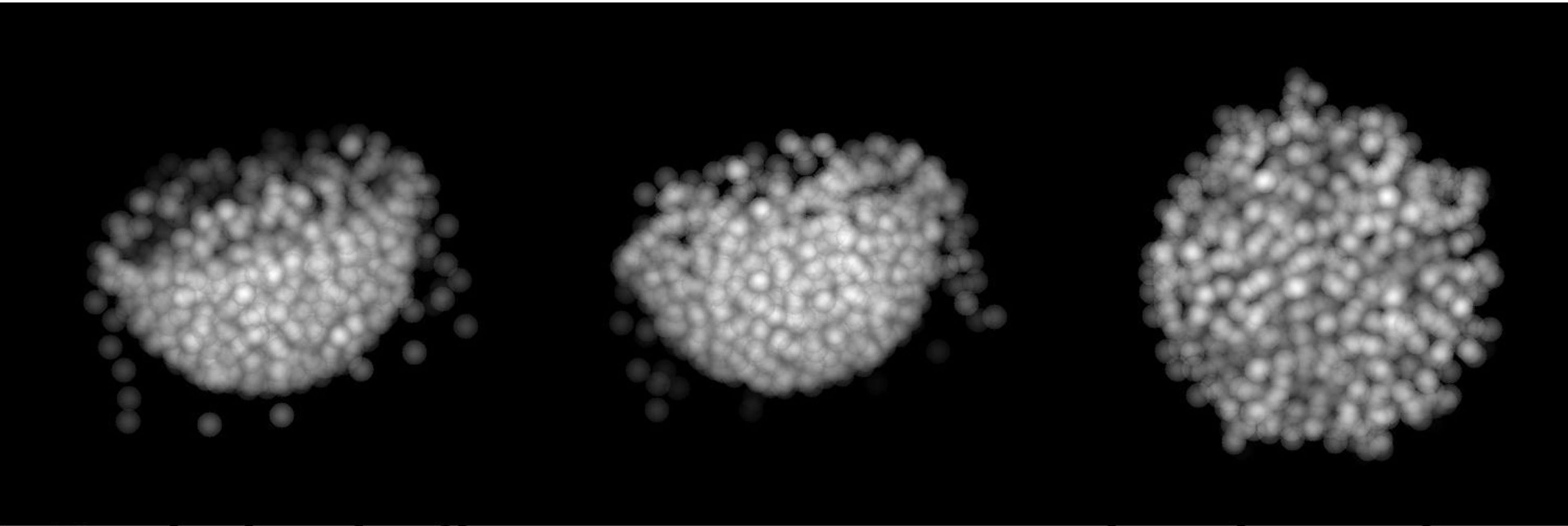
69_label_dresser_pred_night_stand.jpg



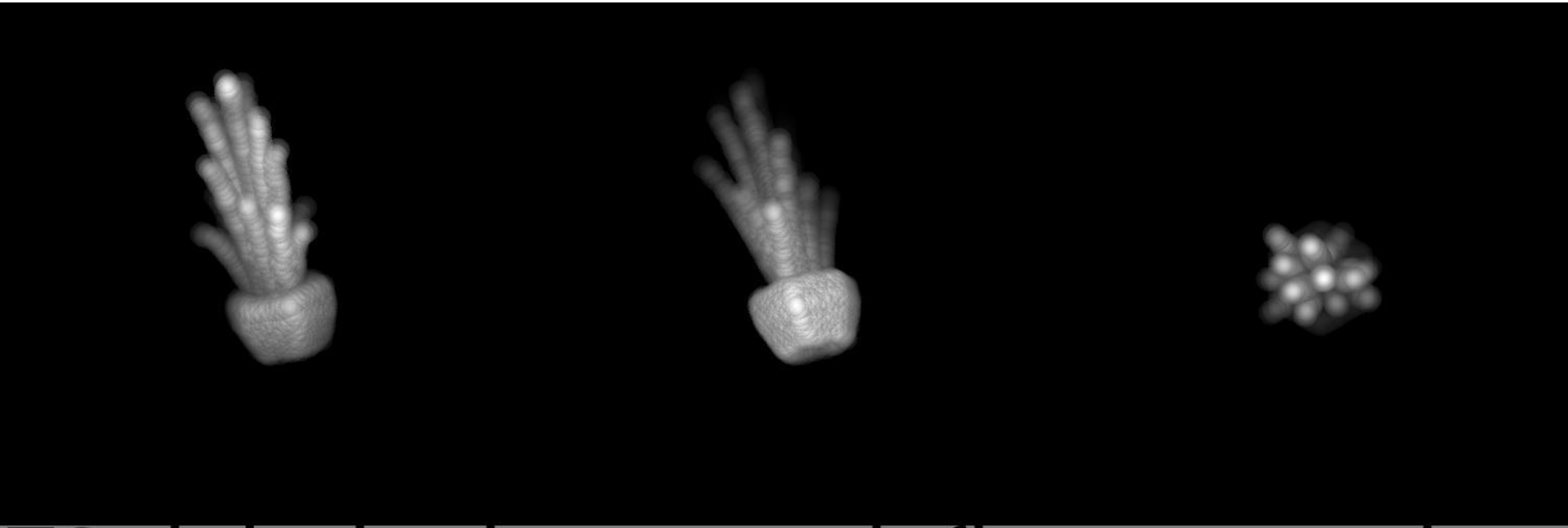
70_label_door_pred_curtain.jpg



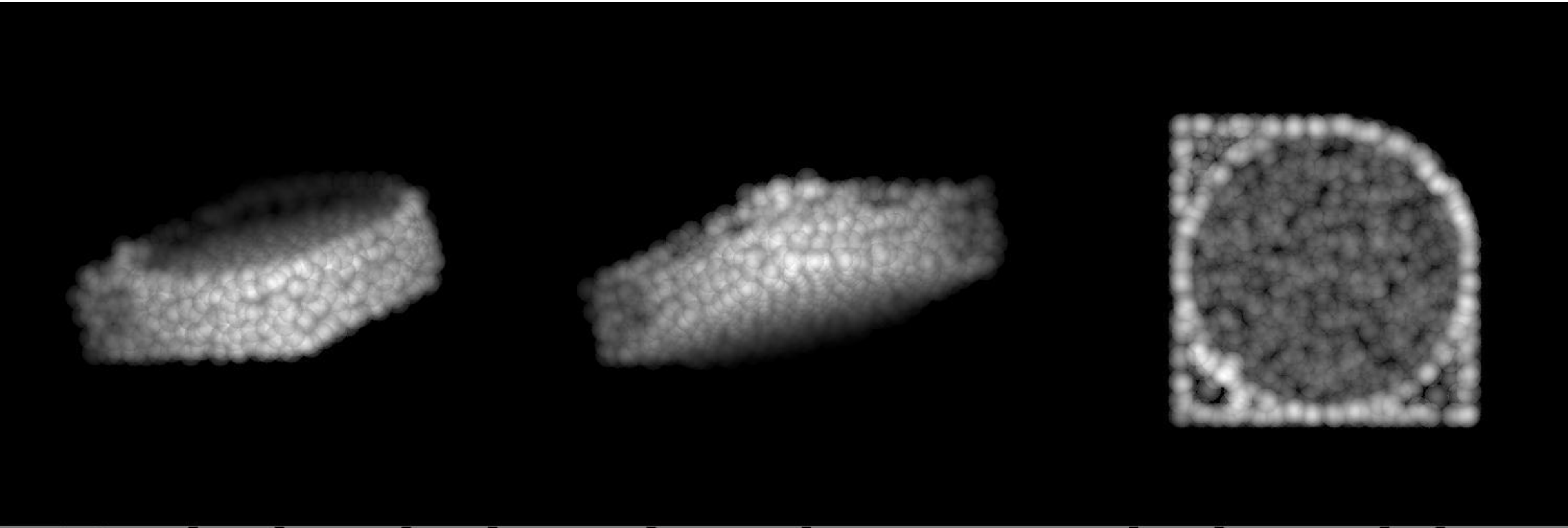
71_label_bookshelf_pred_tv_stand.jpg



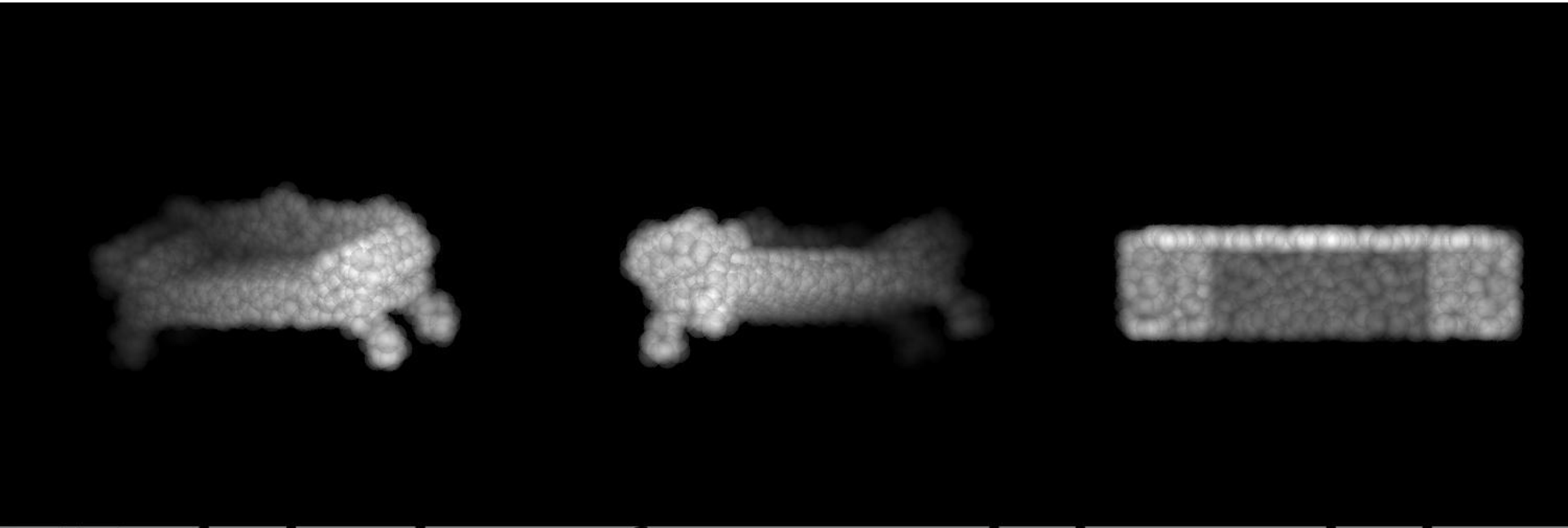
72_label_flower_pot_pred_plant.jpg



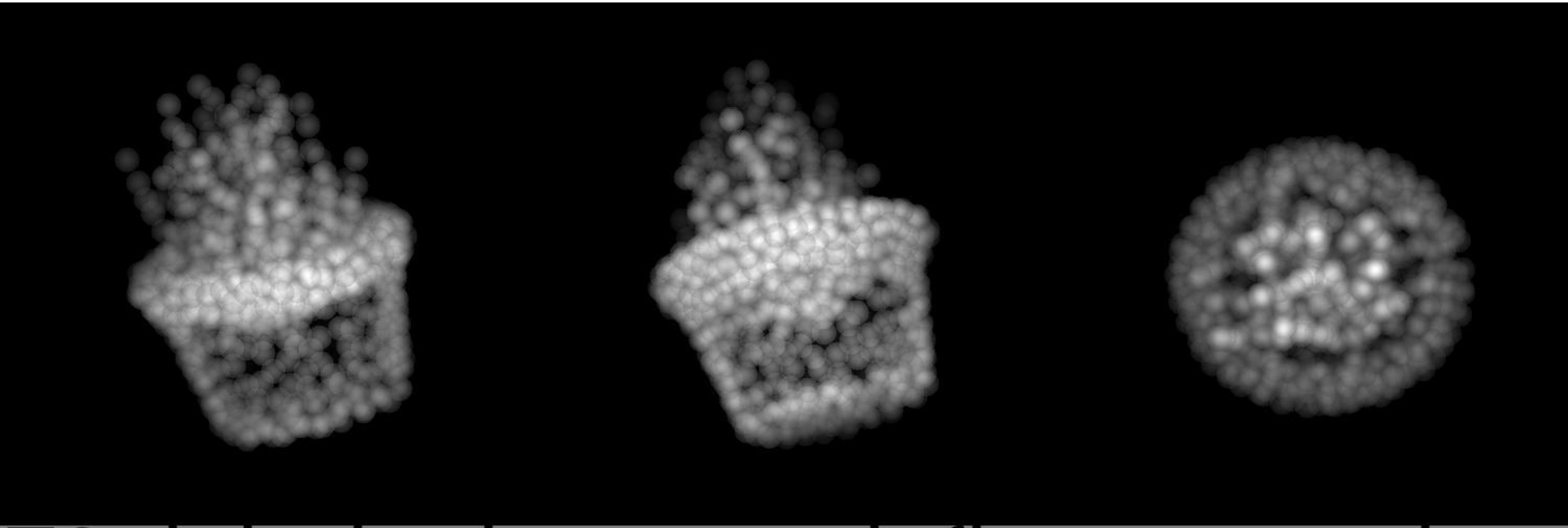
73_label_plant_pred_flower_pot.jpg



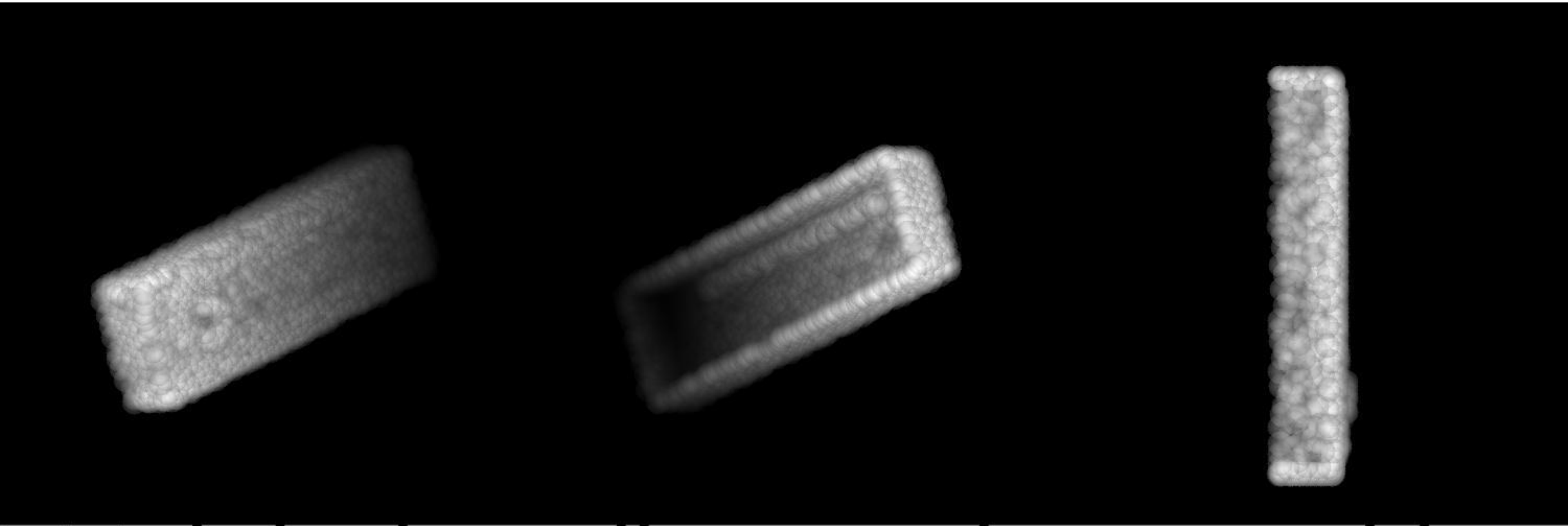
74_label_bathtub_pred_bed.jpg



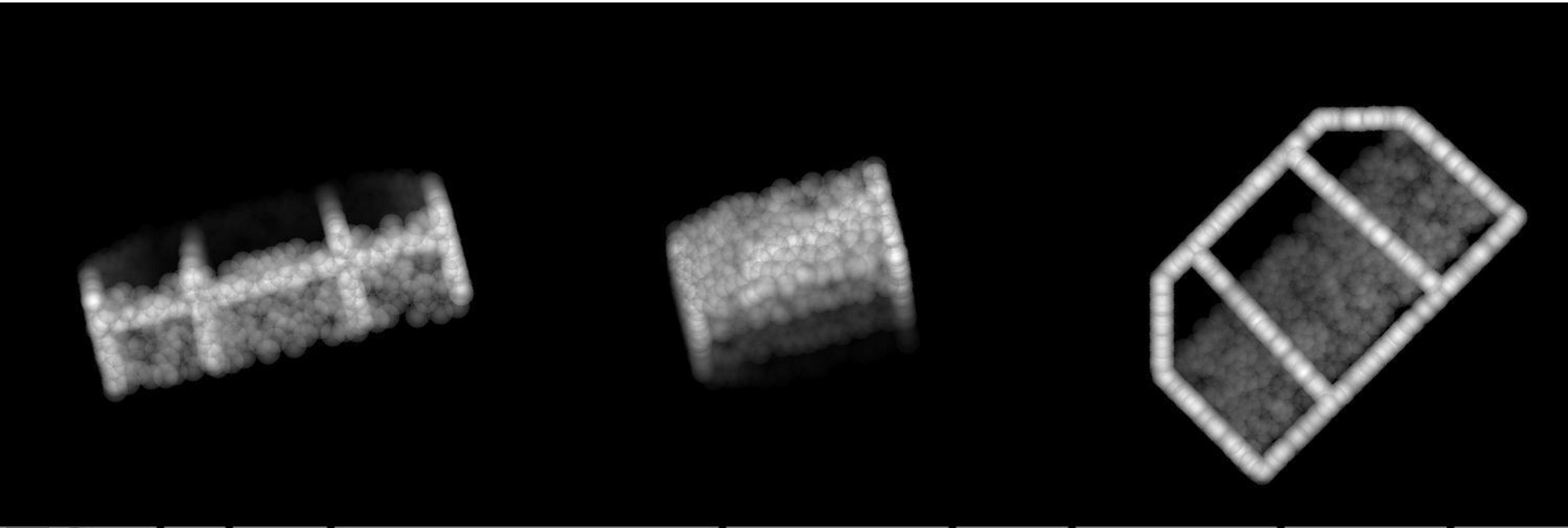
75_label_sofa_pred_bench.jpg



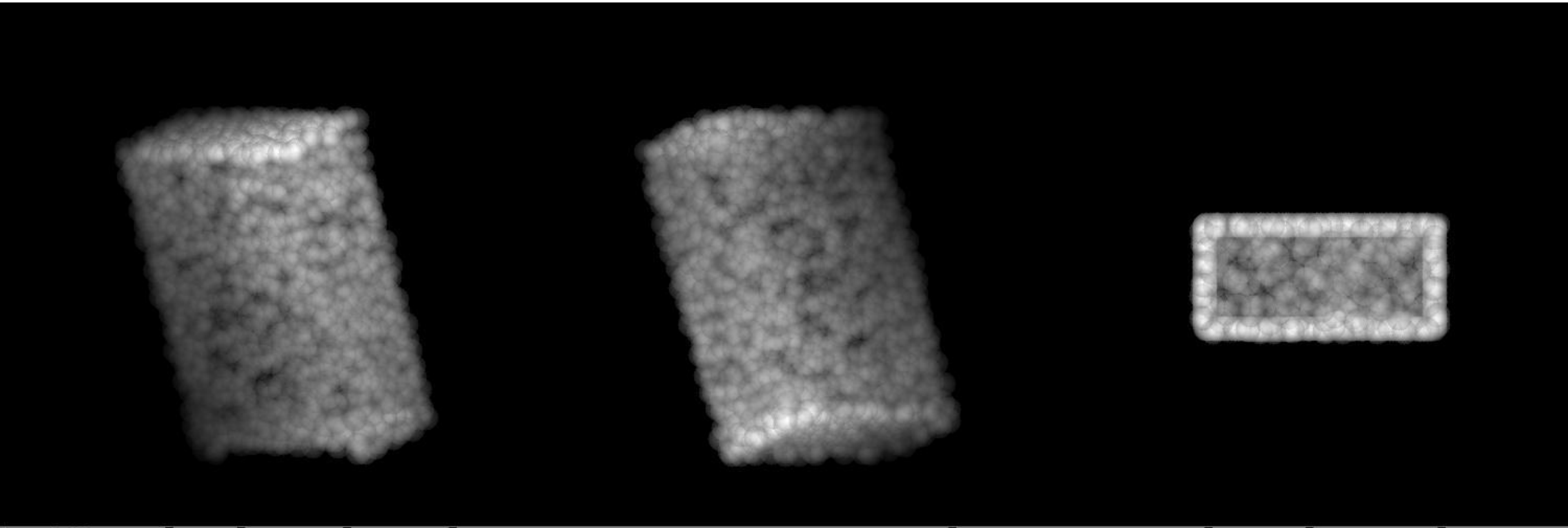
76_label_plant_pred_flower_pot.jpg



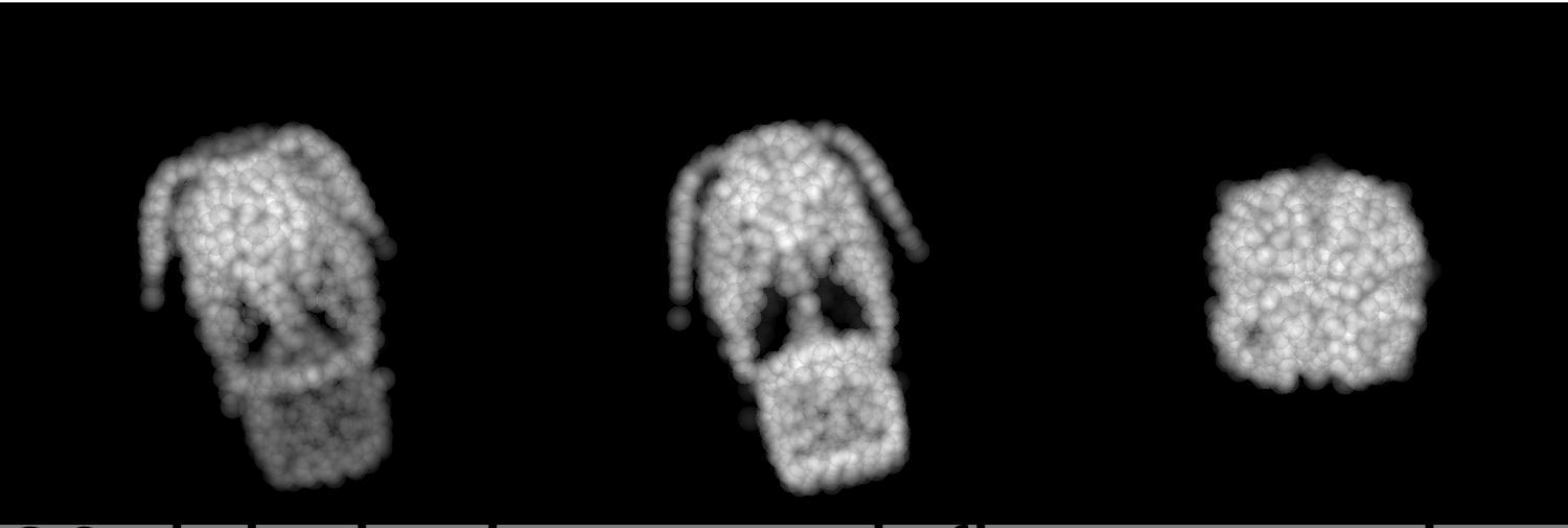
77_label_radio_pred_mantel.jpg

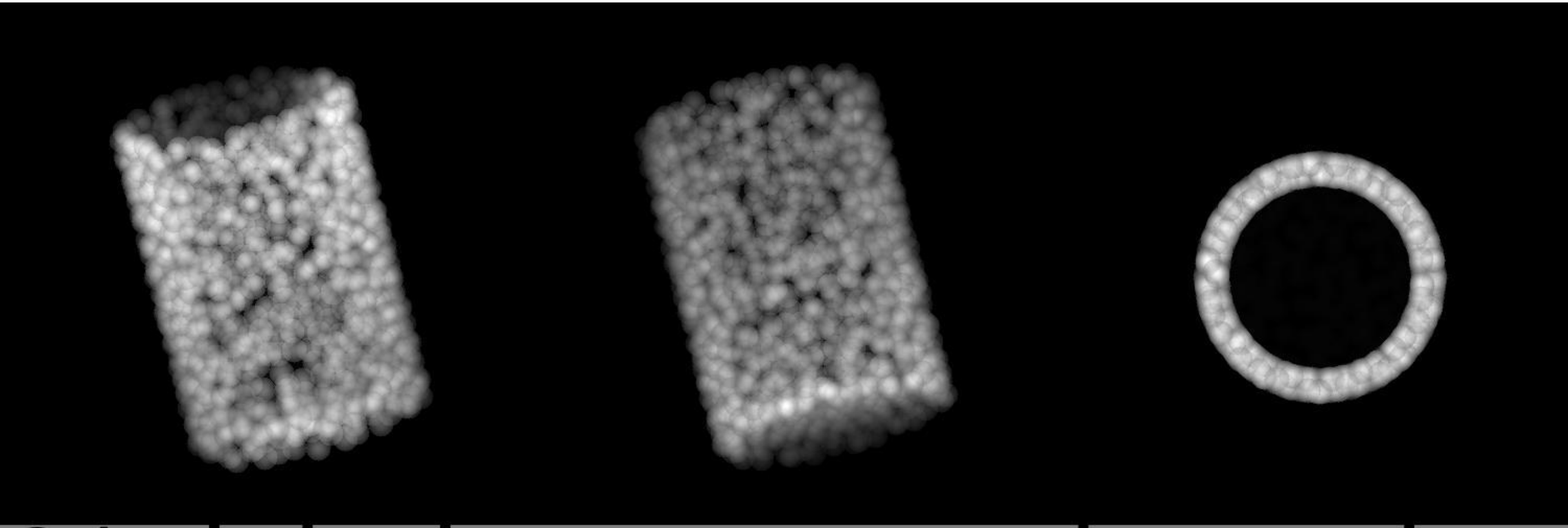


78_label_tv_stand_pred_glass_box.jpg

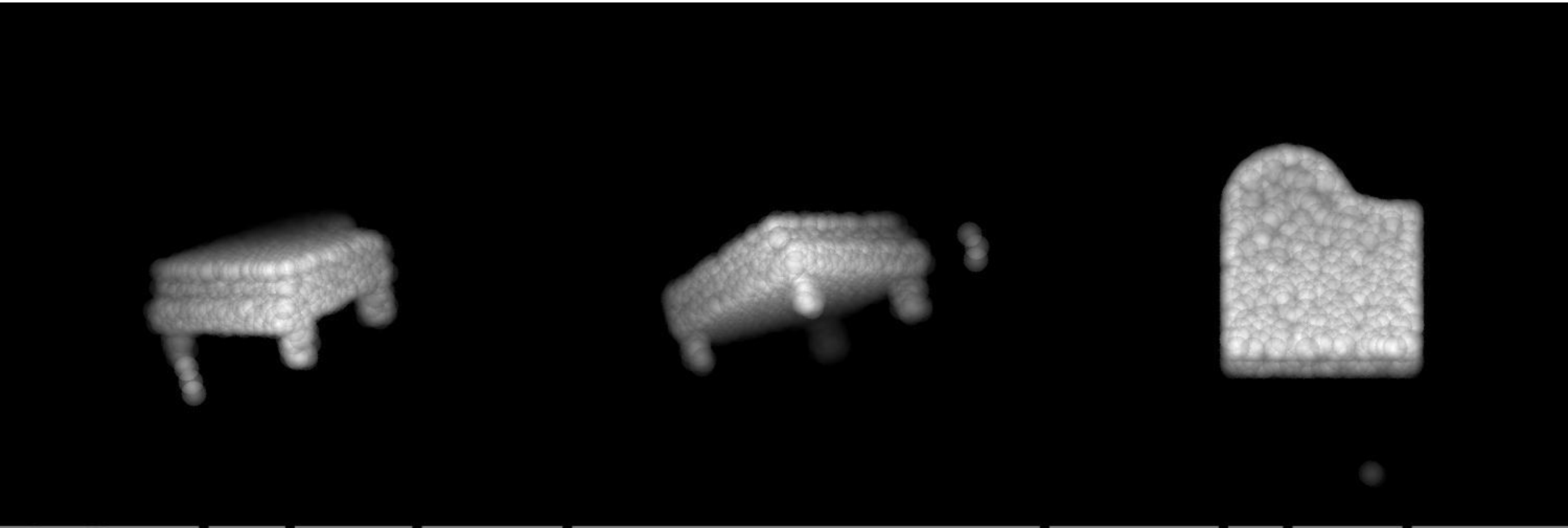


79_label_dresser_pred_wardrobe.jpg

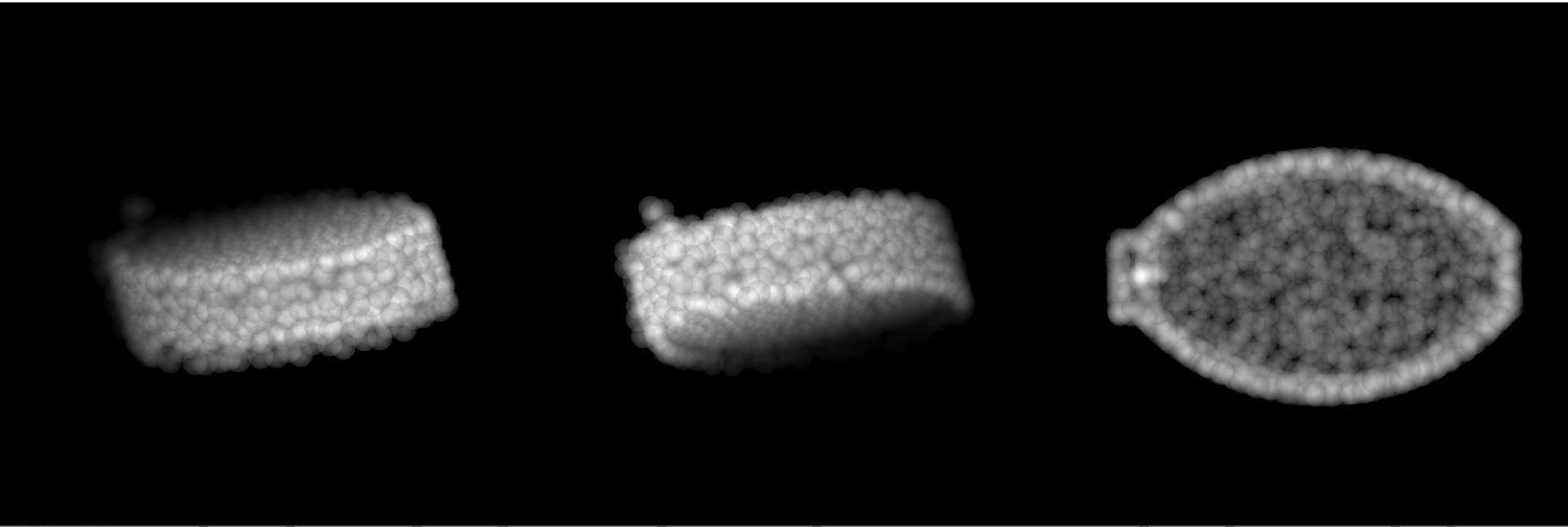

80_label_plant_pred_flower_pot.jpg



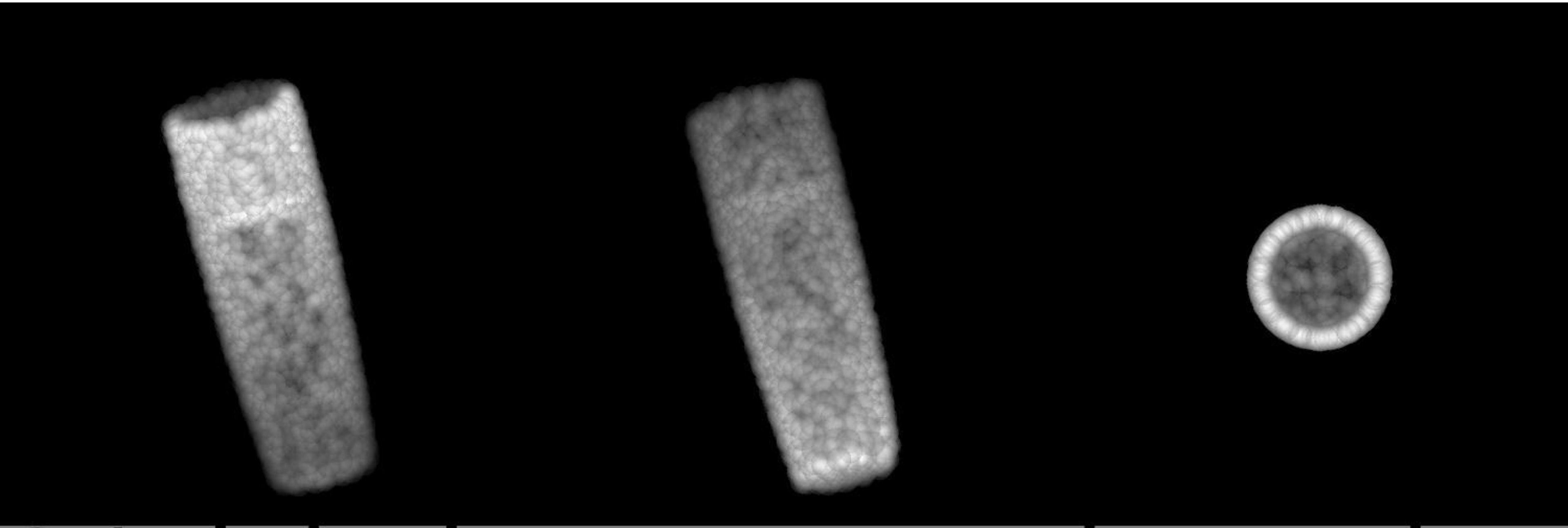
81_label_vase_pred_cup.jpg



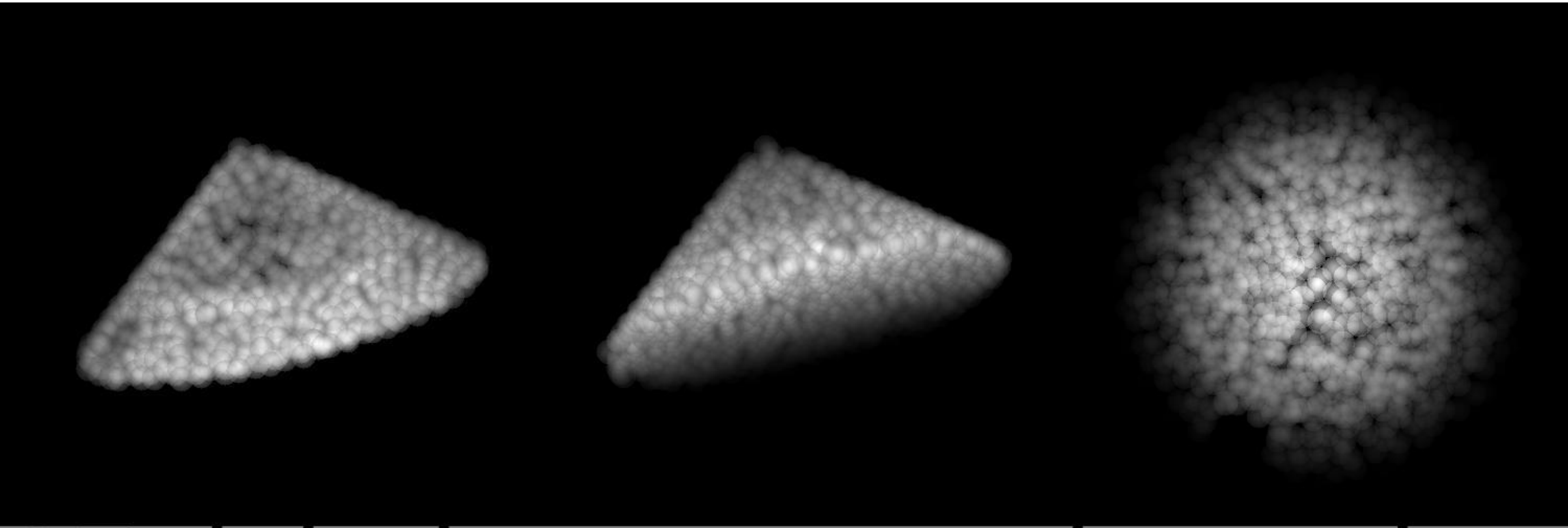
82_label_piano_pred_table.jpg



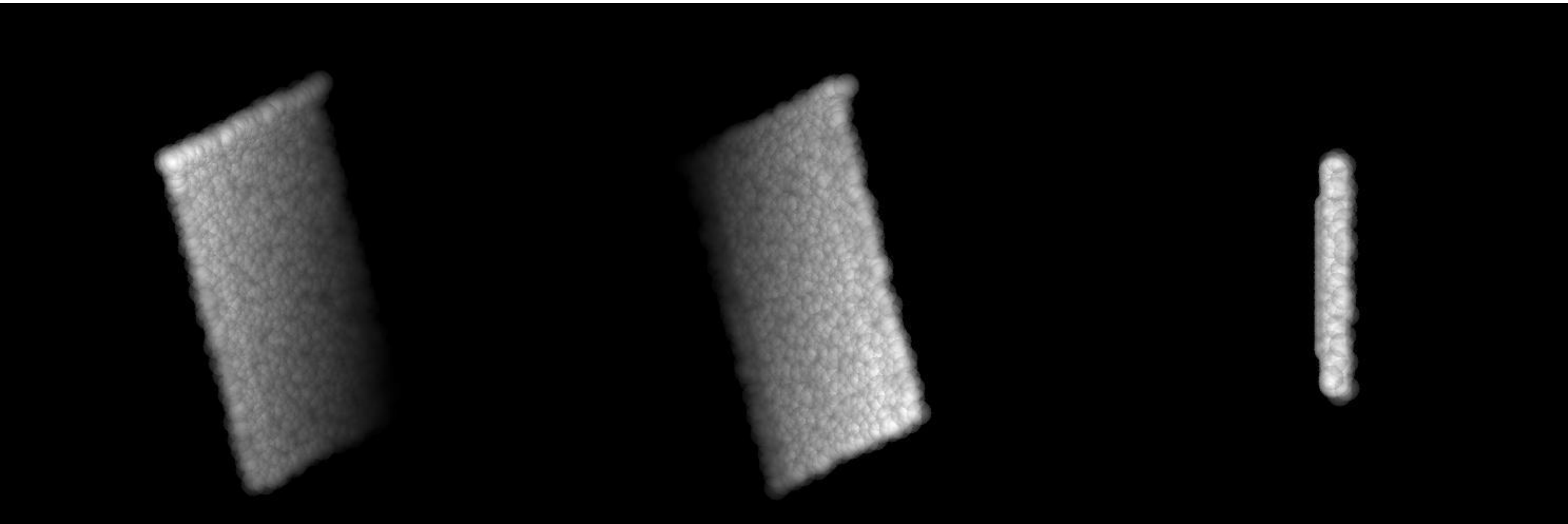
83_label_bathtub_pred_bed.jpg



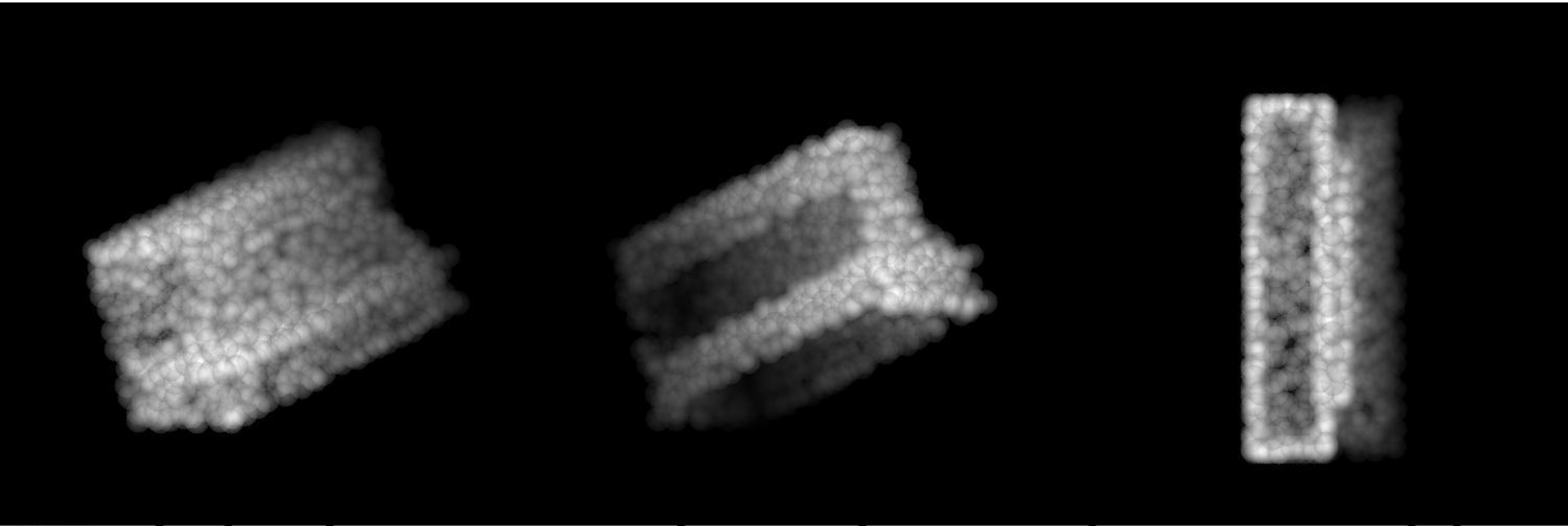
84_label_vase_pred_cup.jpg



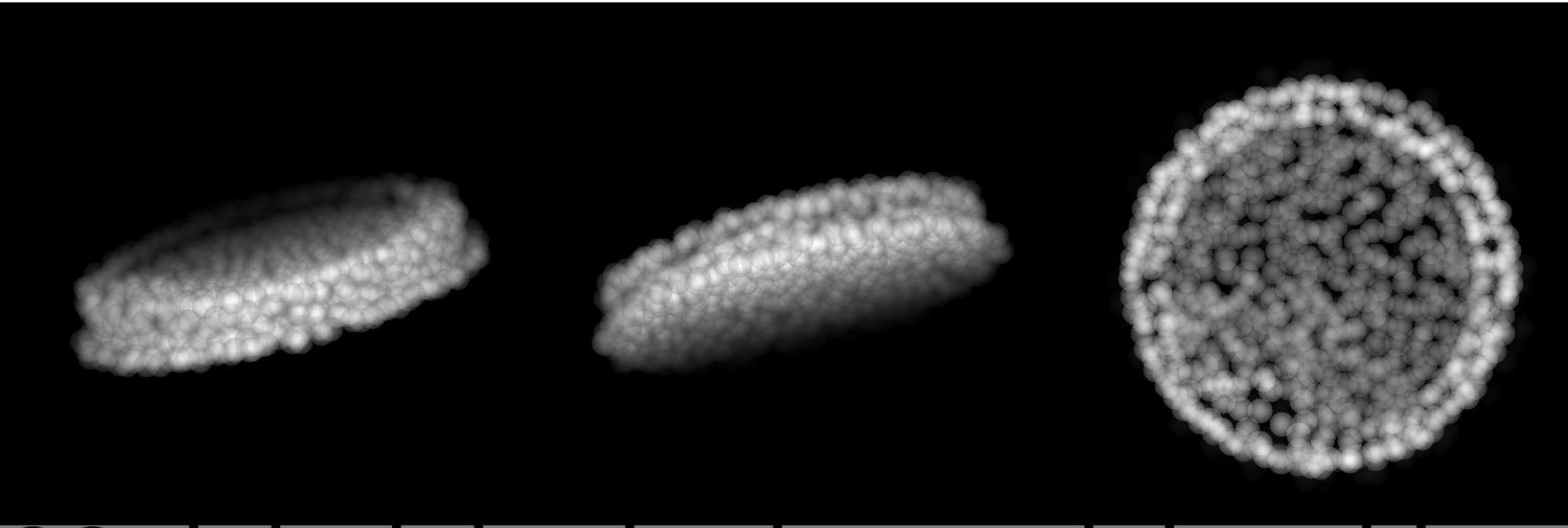
85_label_cone_pred_tent.jpg



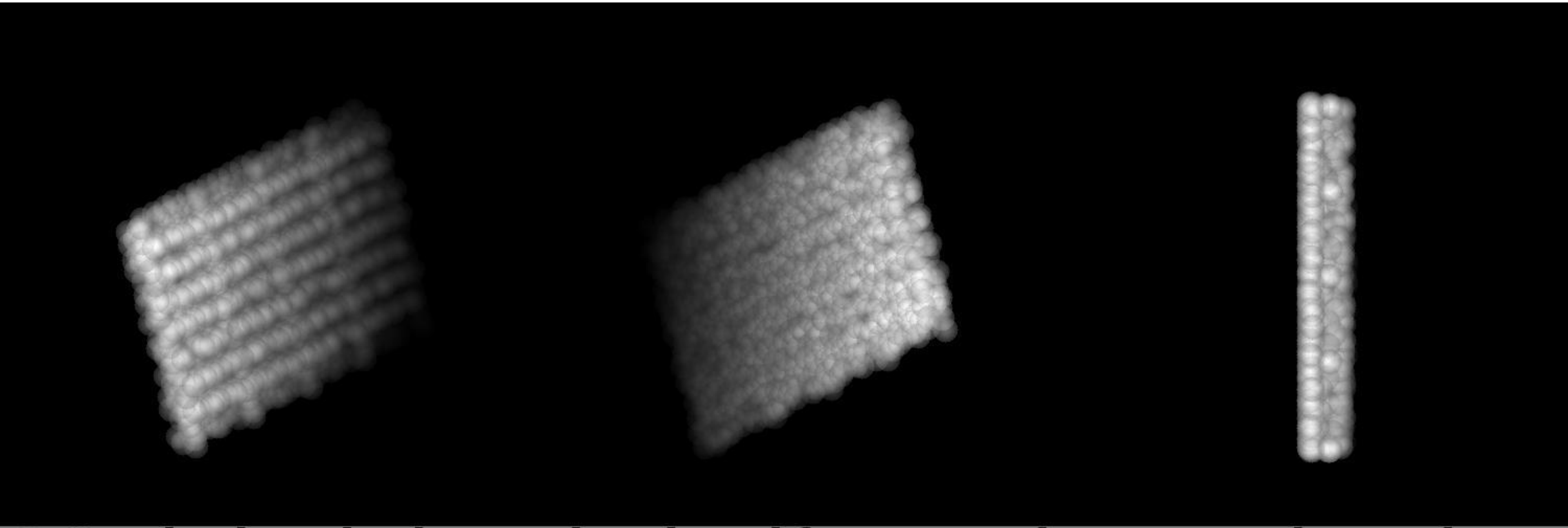
86_label_door_pred_curtain.jpg



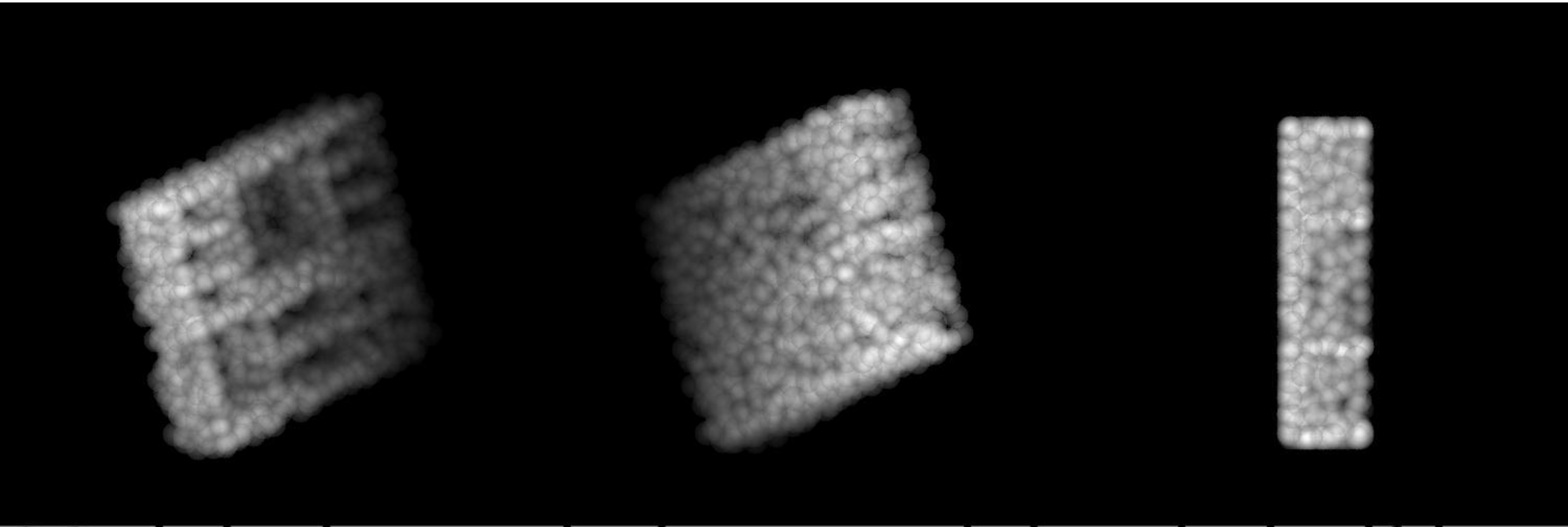
87_label_range_hood_pred_mantel.jpg



88_label_bathtub_pred_bowl.jpg



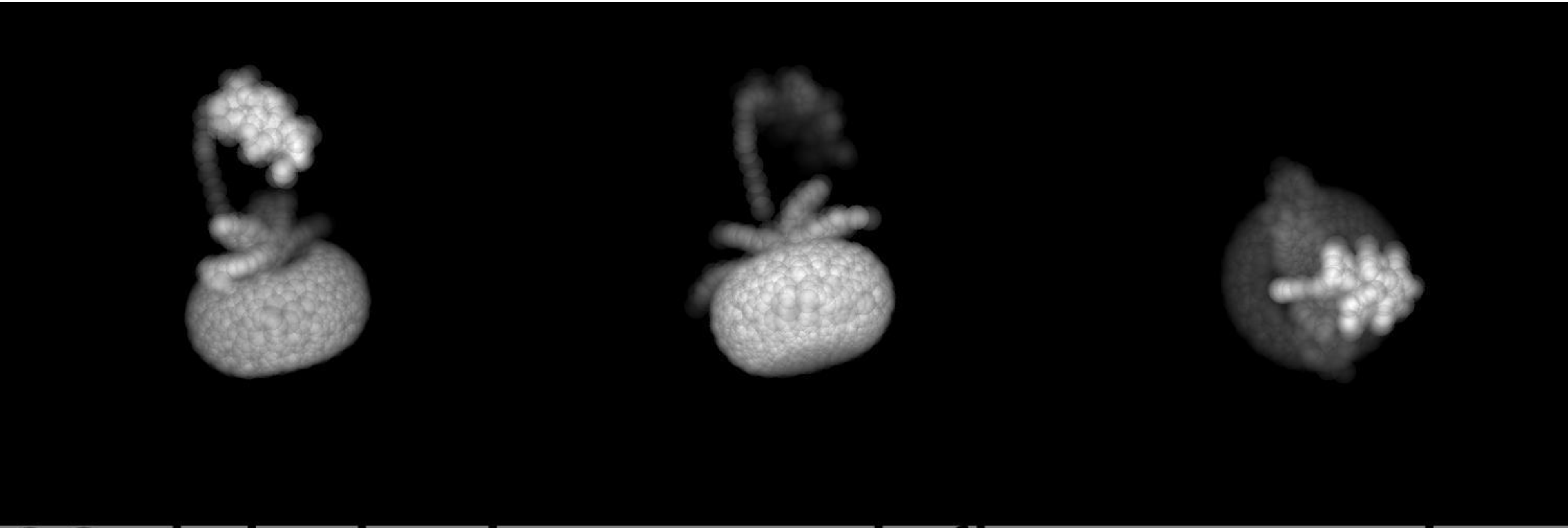
89_label_bookshelf_pred_monitor.jpg



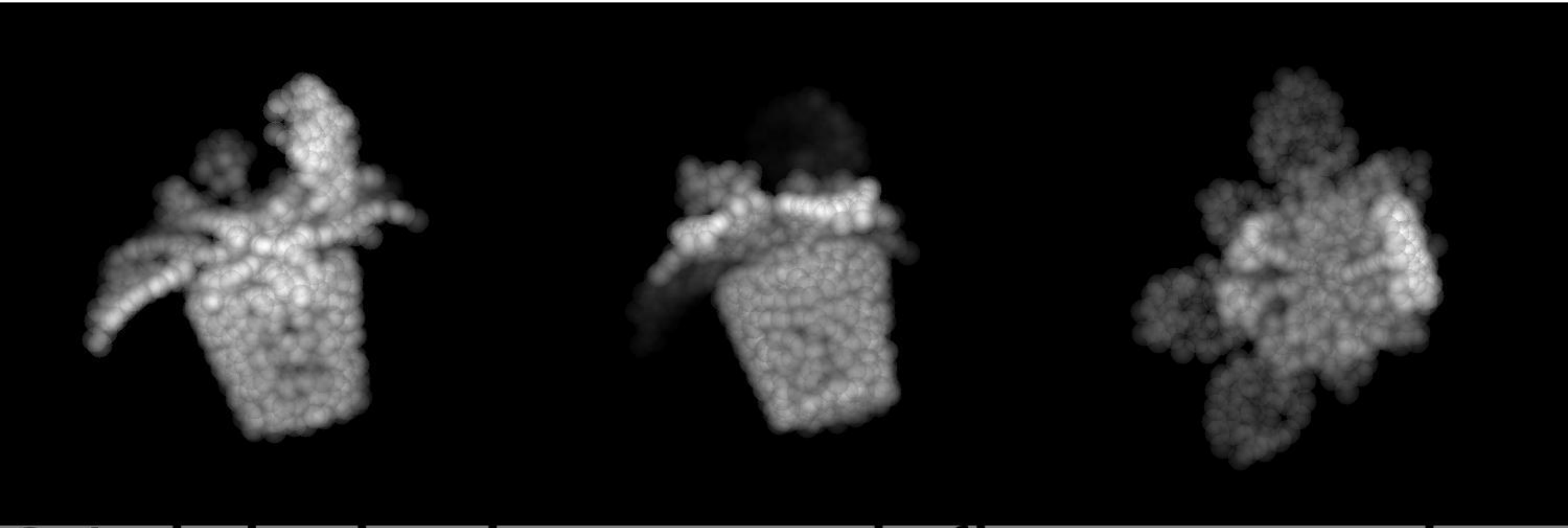
90_label_wardrobe_pred_bookshelf.jpg

91_label_plant_pred_flower_pot.jpg

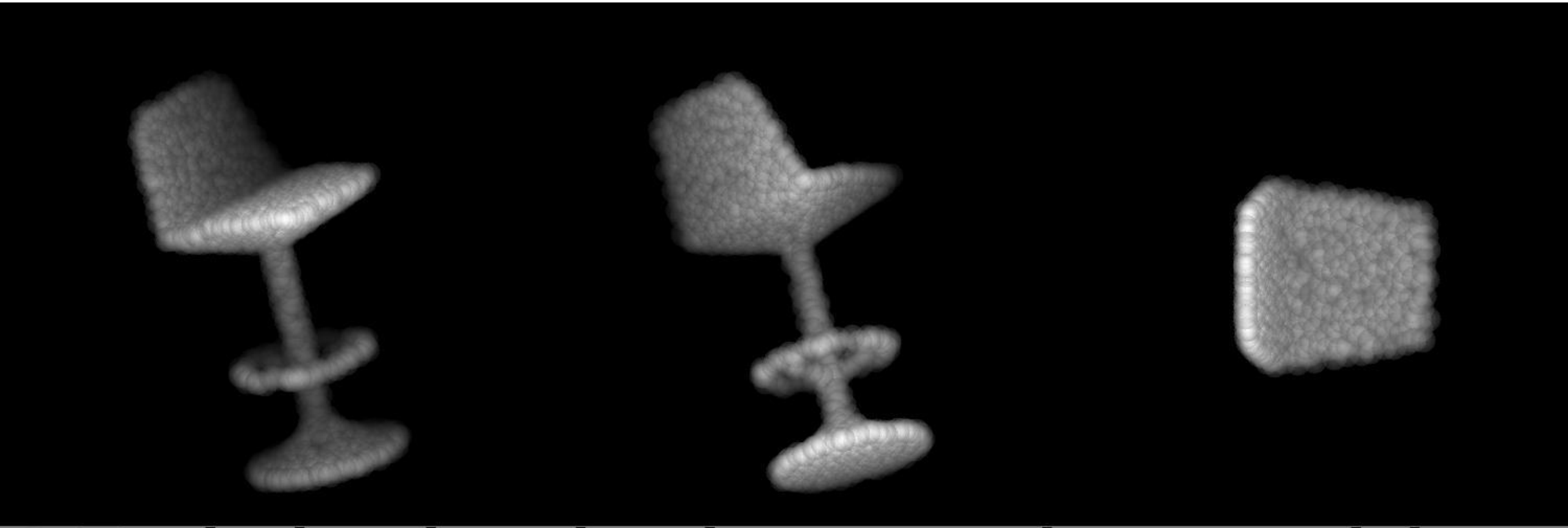
92_label_piano_pred_table.jpg



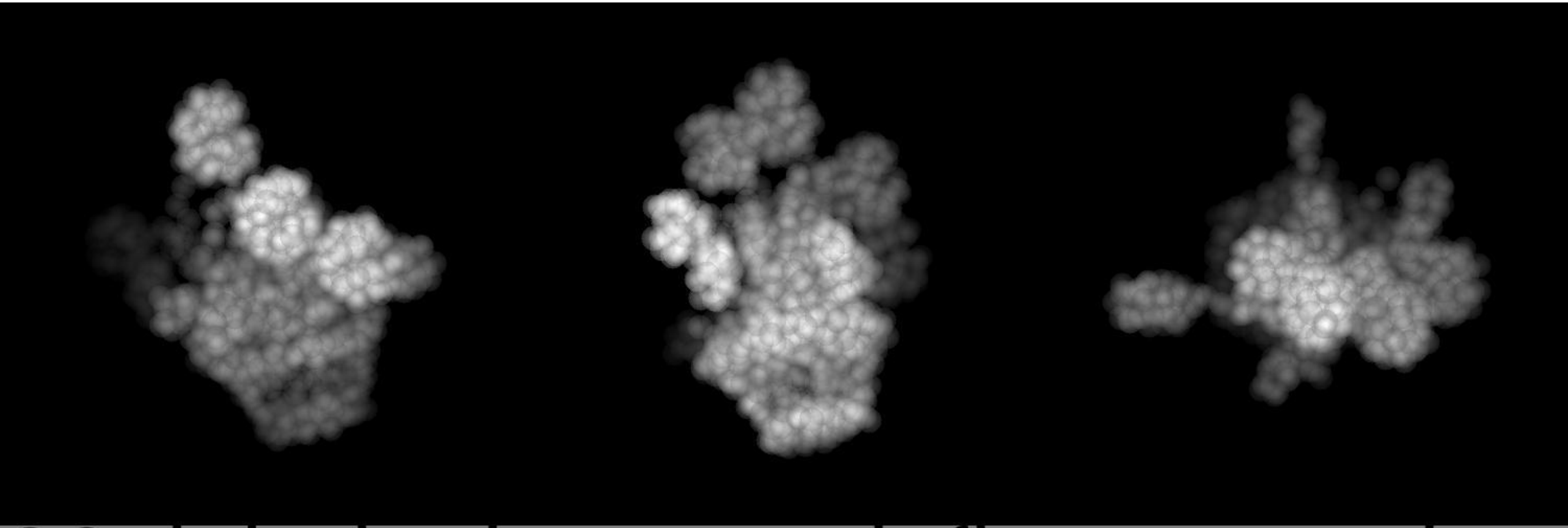
93_label_plant_pred_flower_pot.jpg



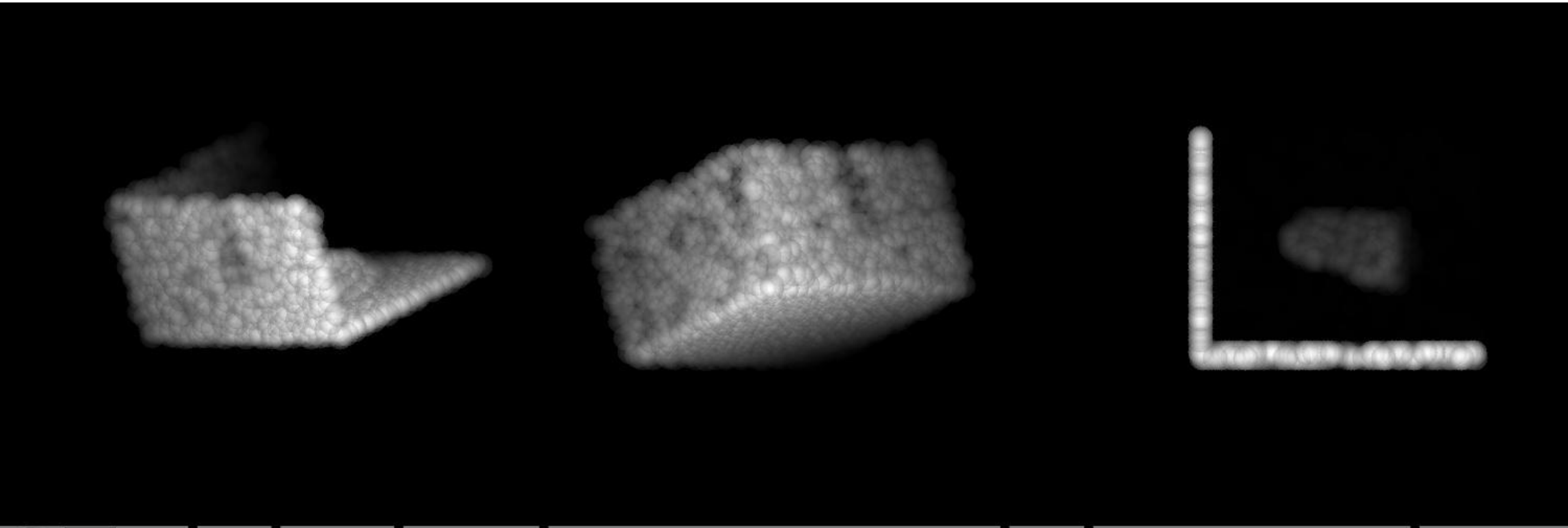
94_label_plant_pred_flower_pot.jpg



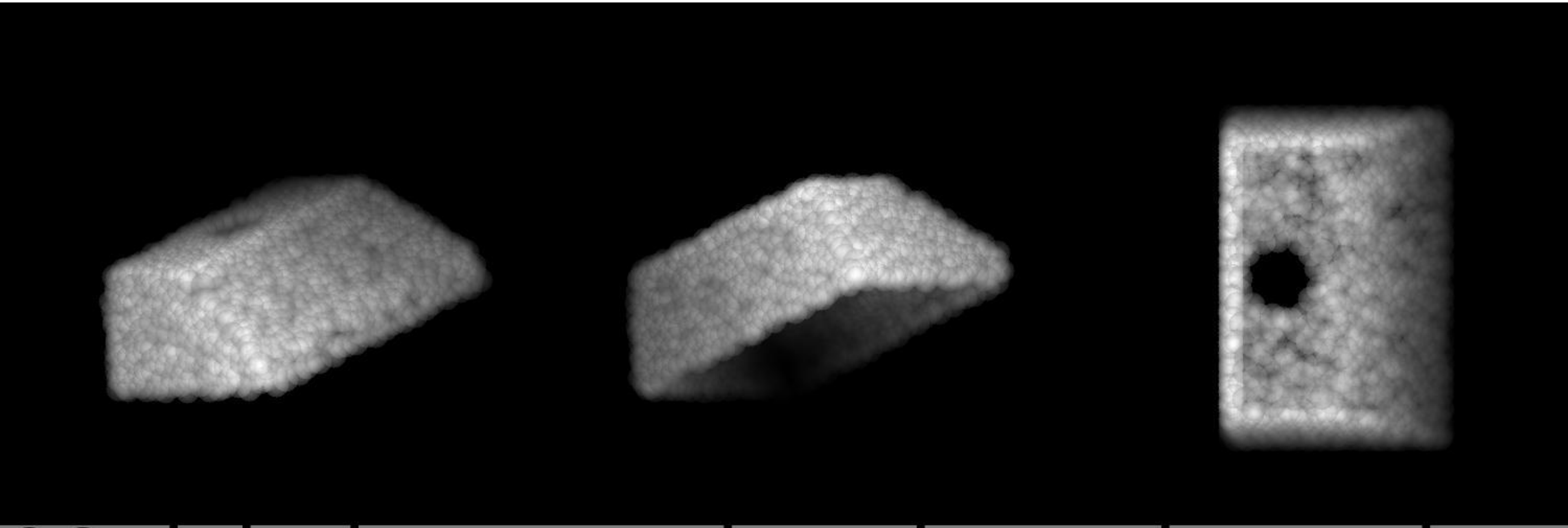
95_label_chair_pred_stool.jpg



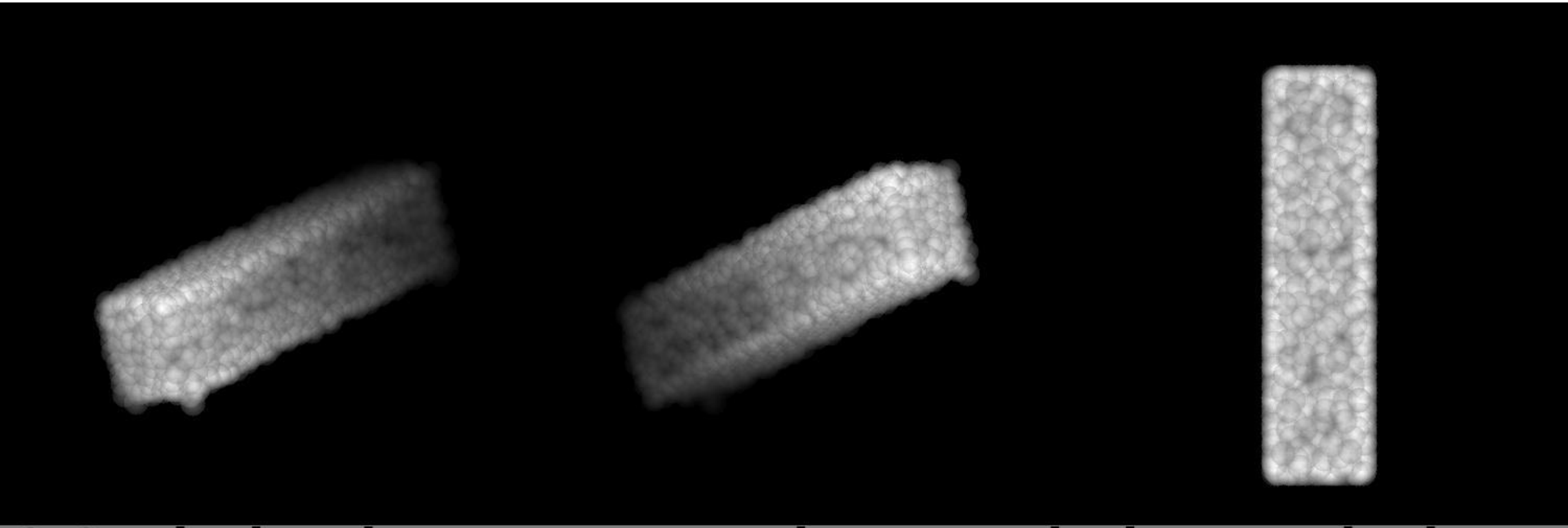
96_label_plant_pred_flower_pot.jpg



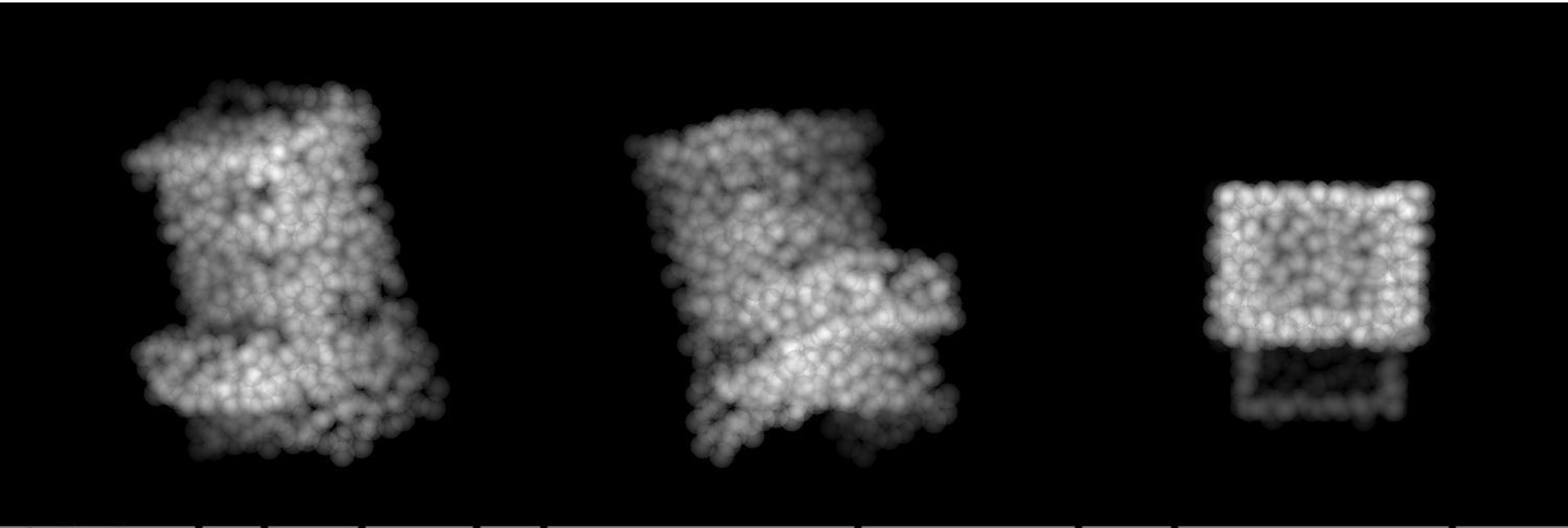
97_label_piano_pred_laptop.jpg



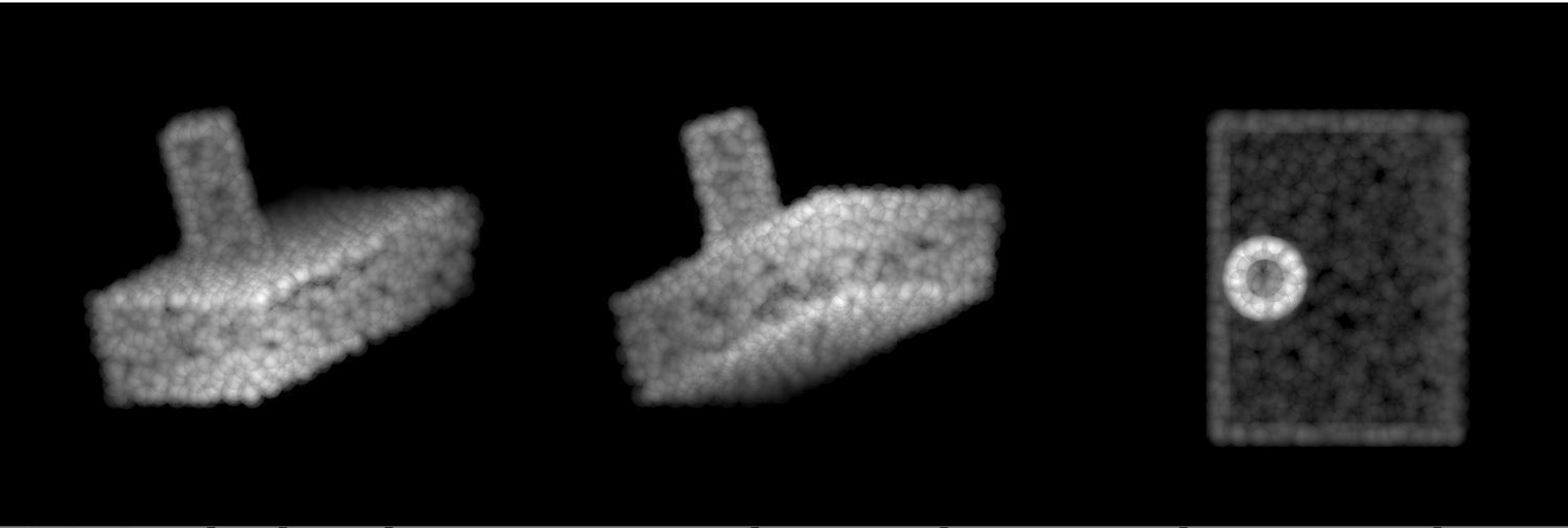
98_label_range_hood_pred_tent.jpg



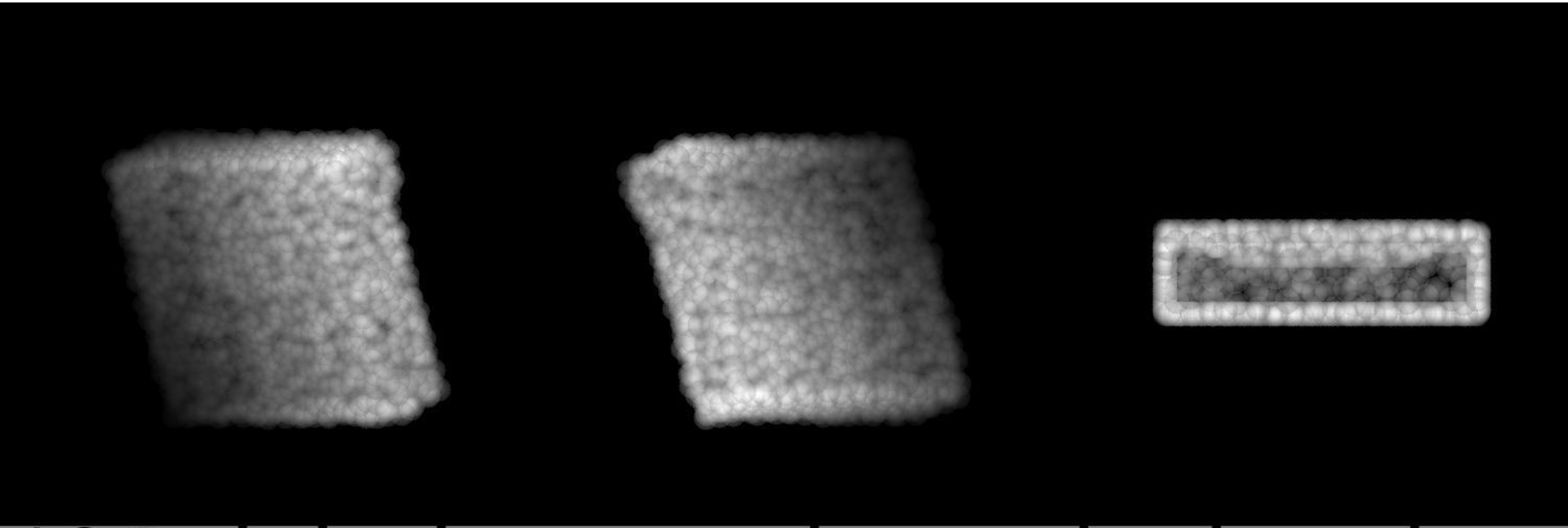
99_label_tv_stand_pred_bench.jpg



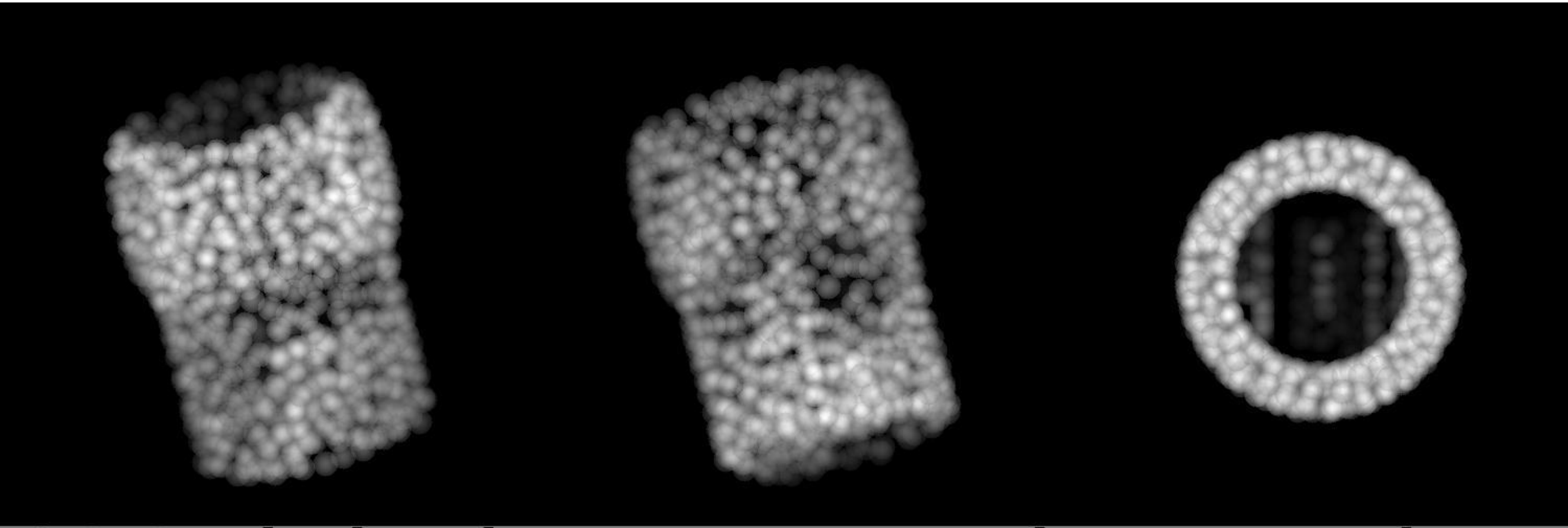
100_label_night_stand_pred_dresser.jpg



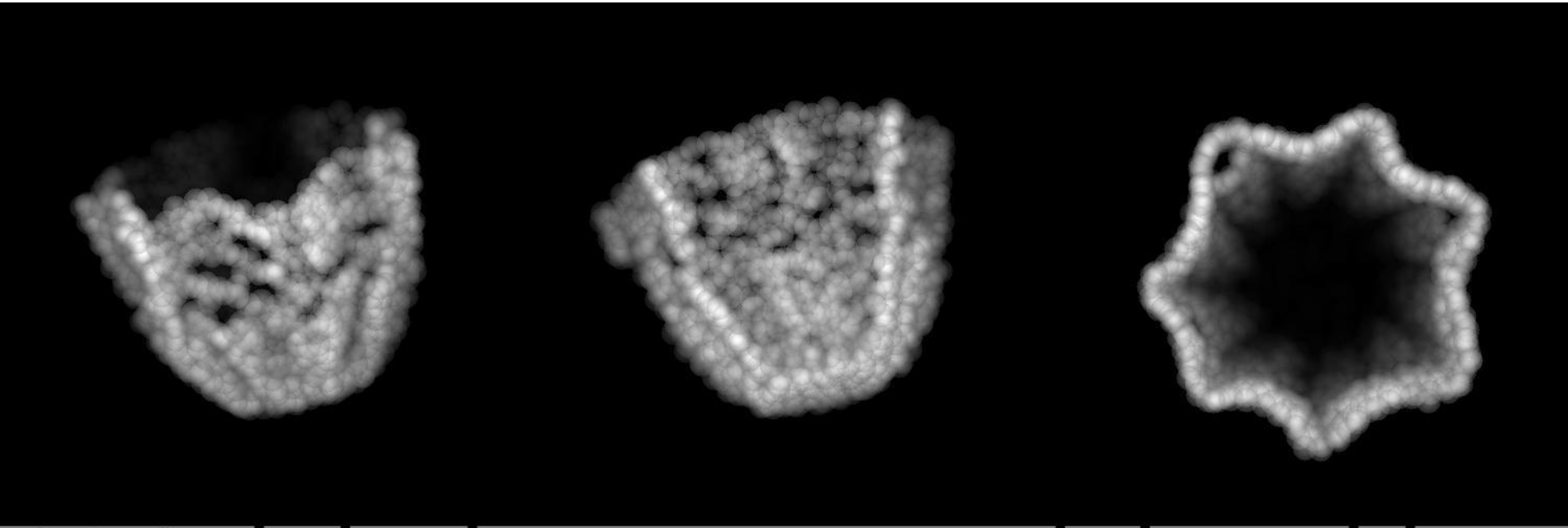
101_label_range_hood_pred_tent.jpg



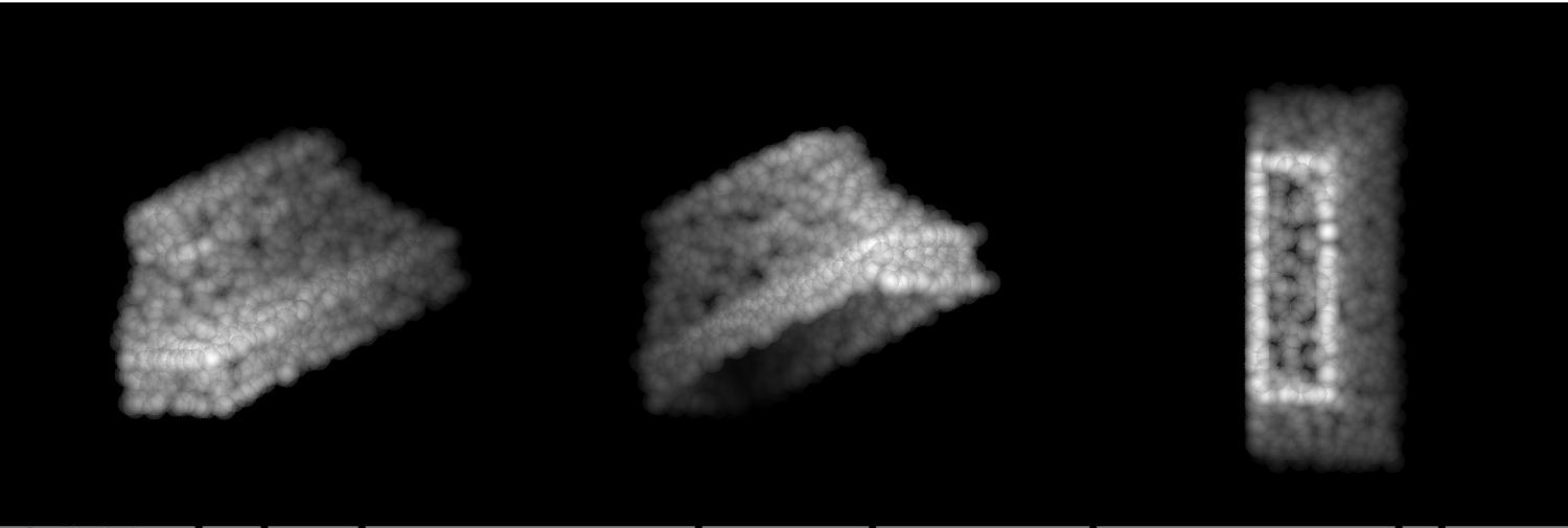
102_label_mantel_pred_xbox.jpg



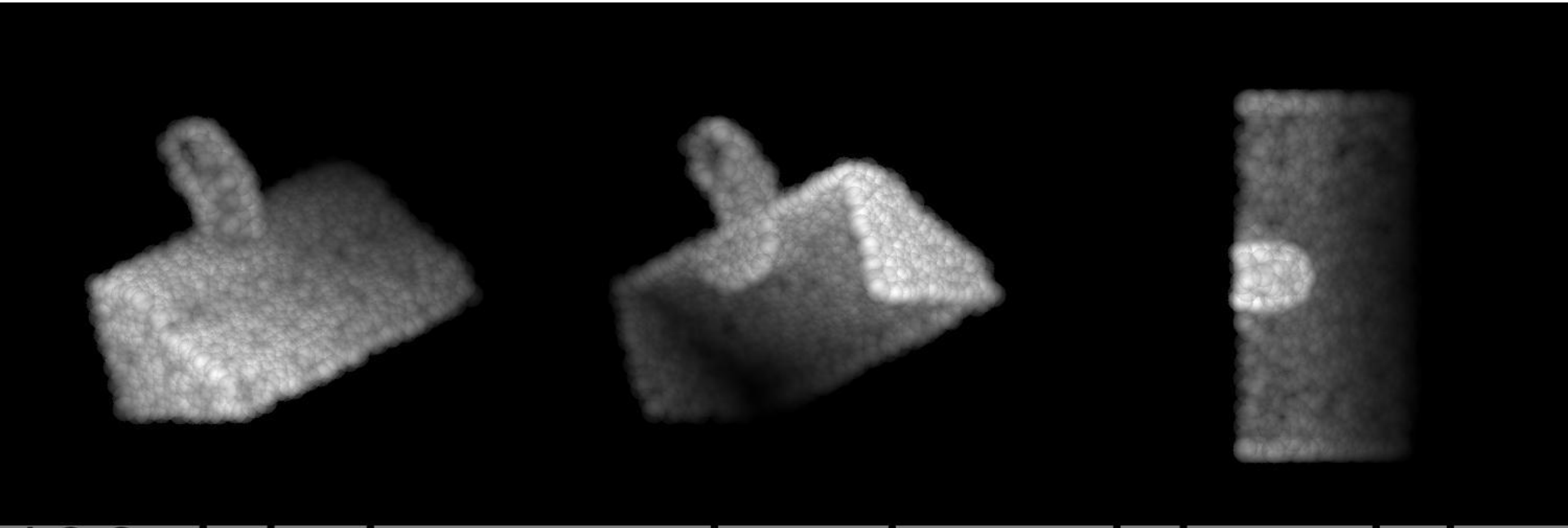
103_label_cup_pred_vase.jpg



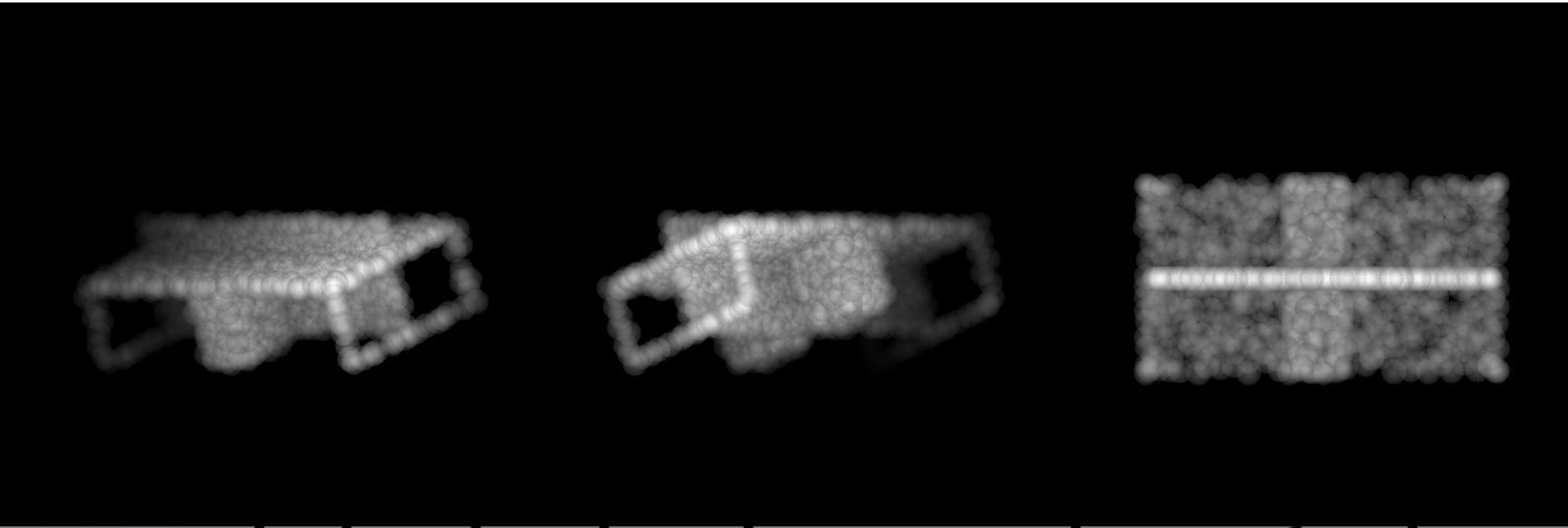
104_label_vase_pred_bowl.jpg



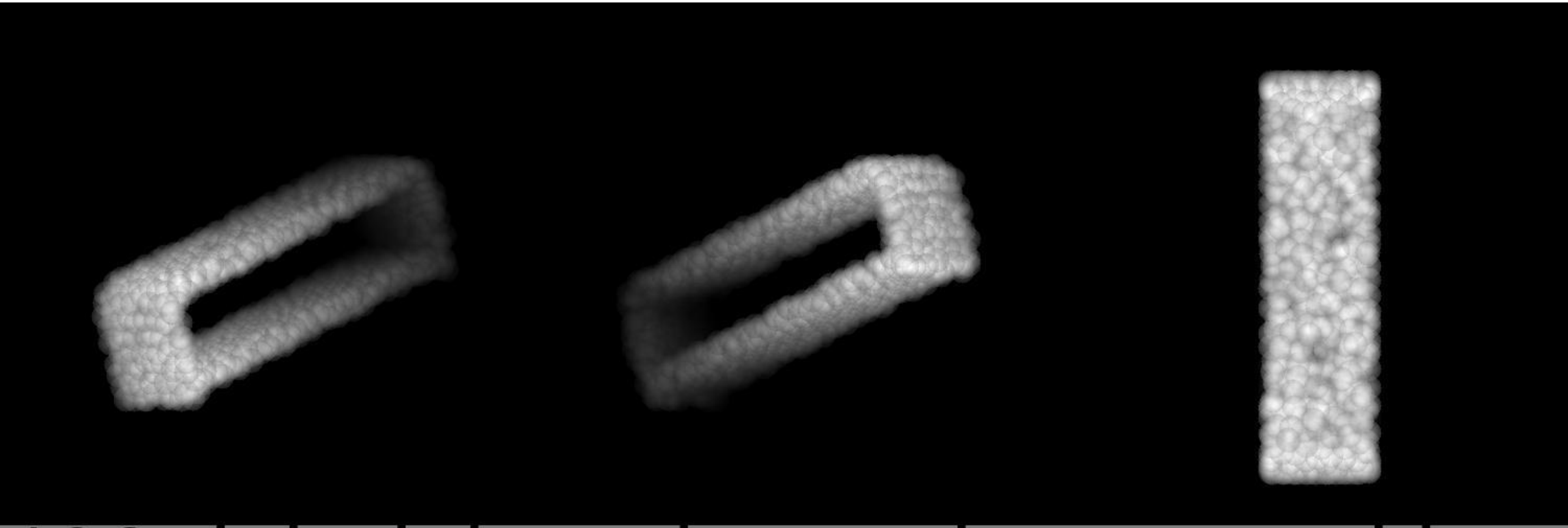
105_label_range_hood_pred_mantel.jpg



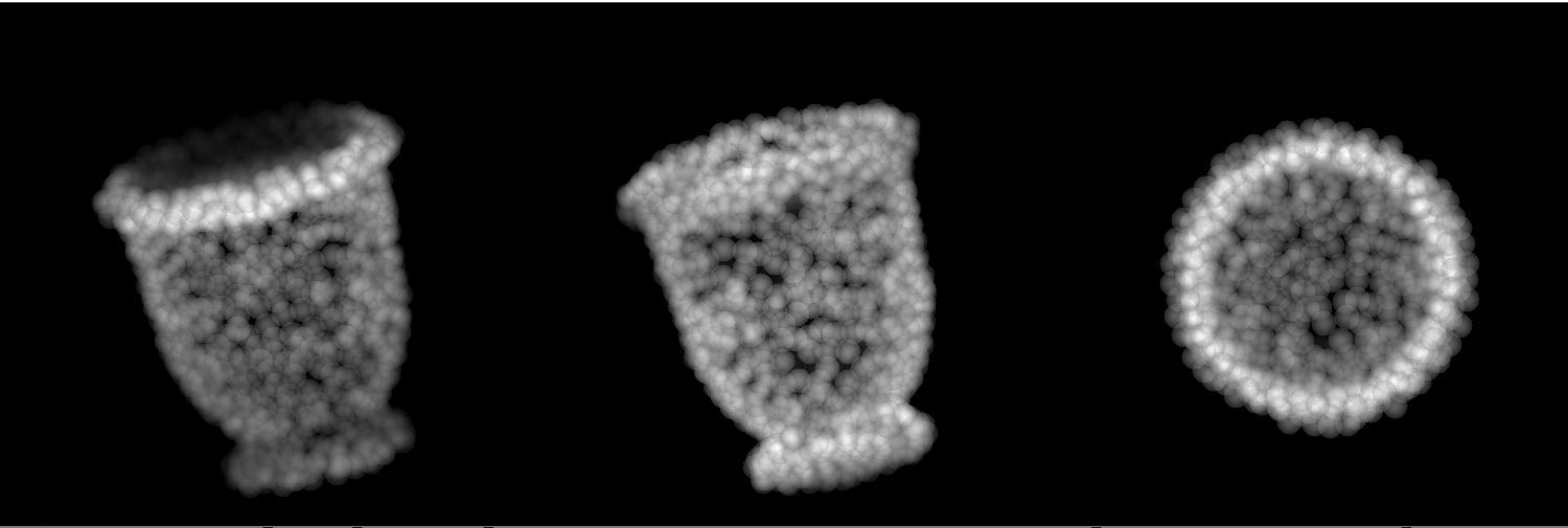
106_label_range_hood_pred_bench.jpg



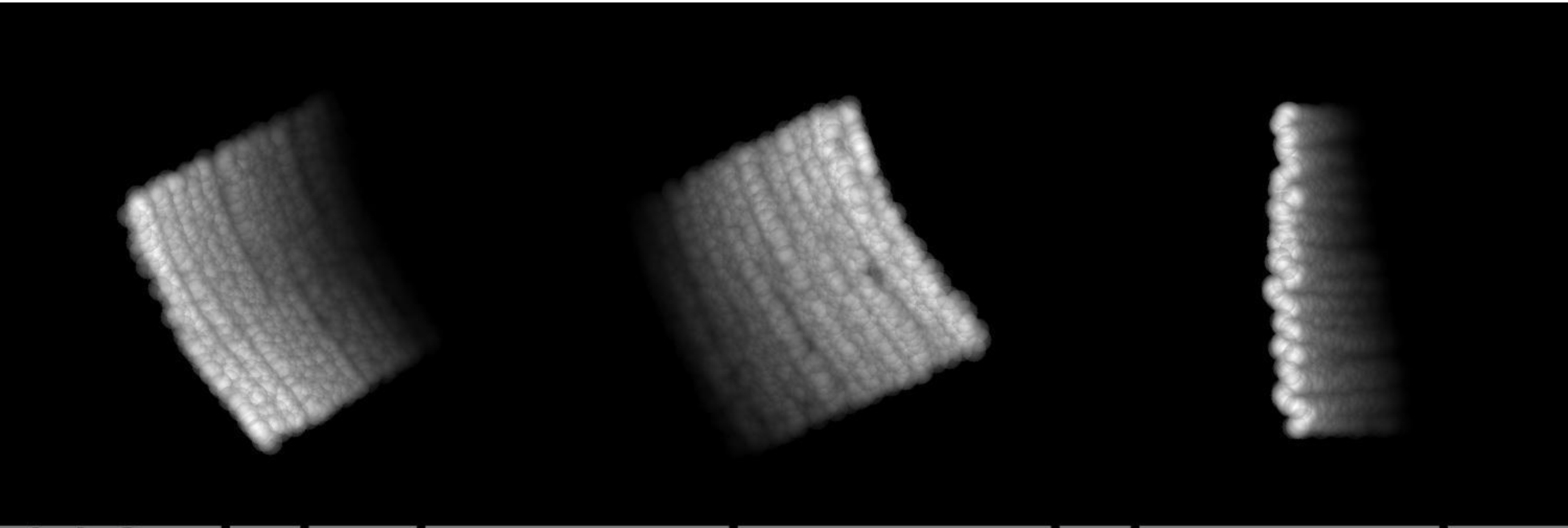
107_label_desk_pred_sofa.jpg



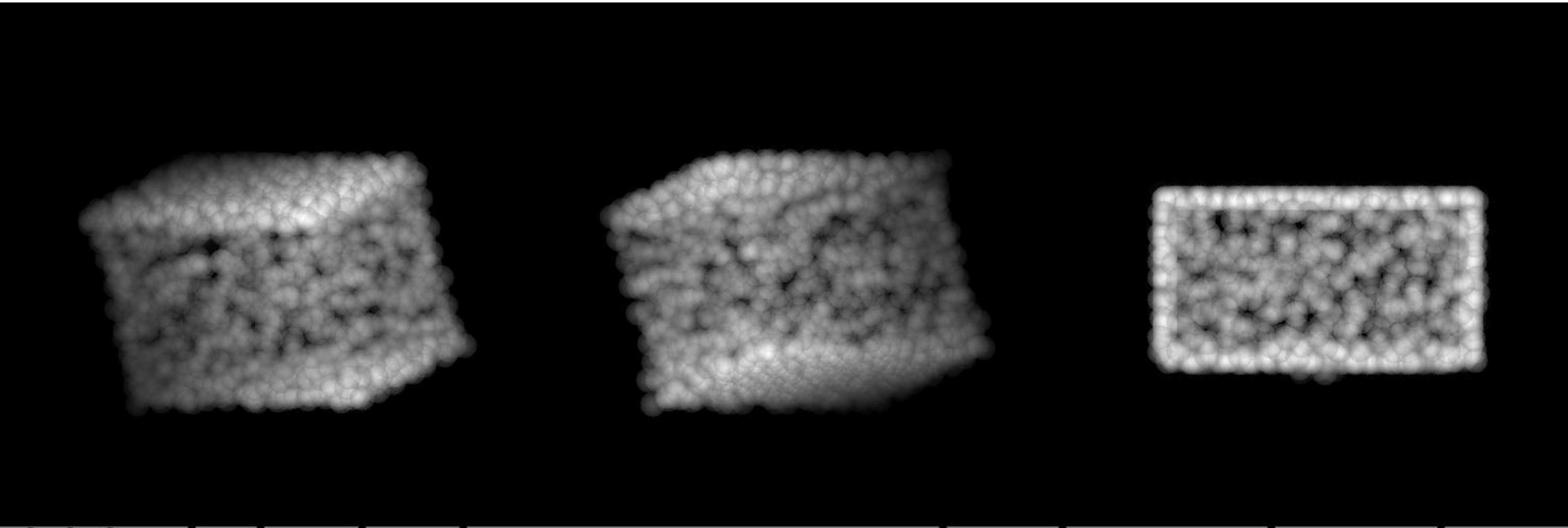
108_label_bench_pred_tv_stand.jpg



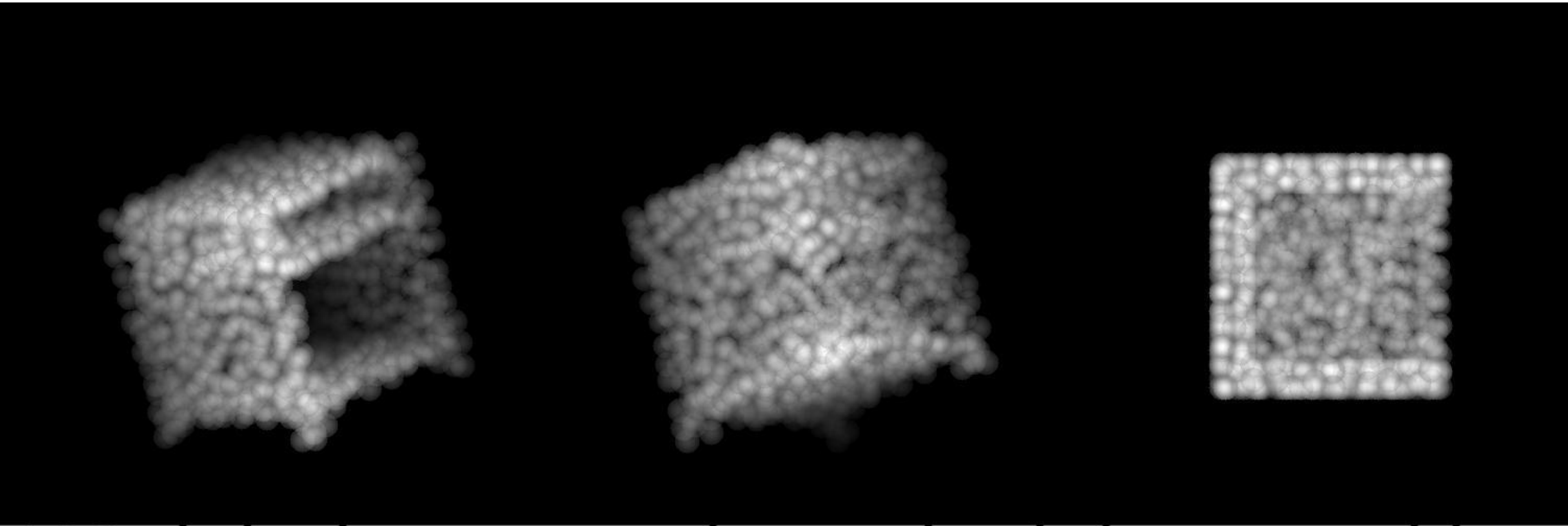
109_label_vase_pred_cup.jpg



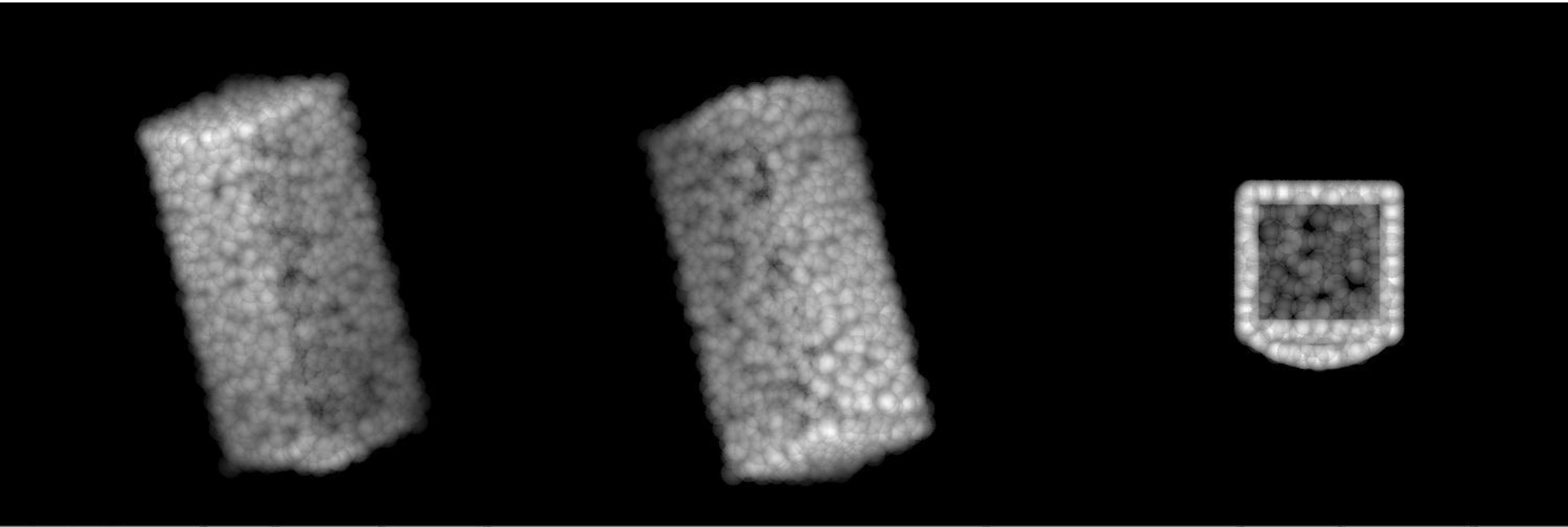
110_label_curtain_pred_laptop.jpg



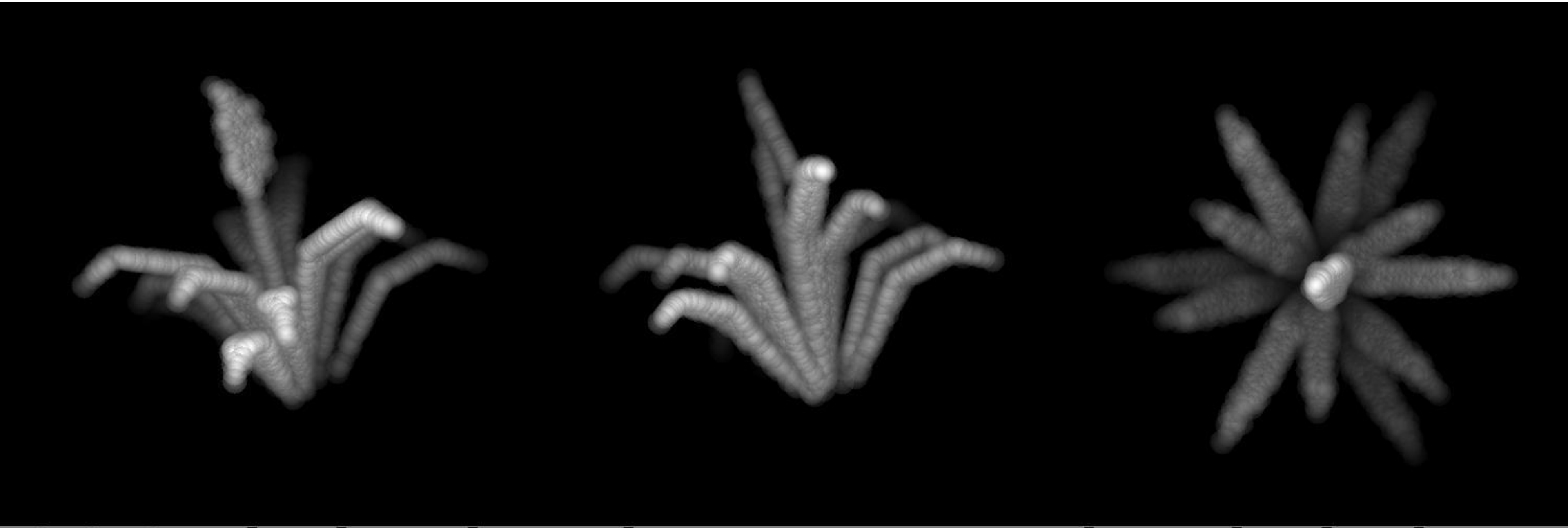
111_label_dresser_pred_glass_box.jpg

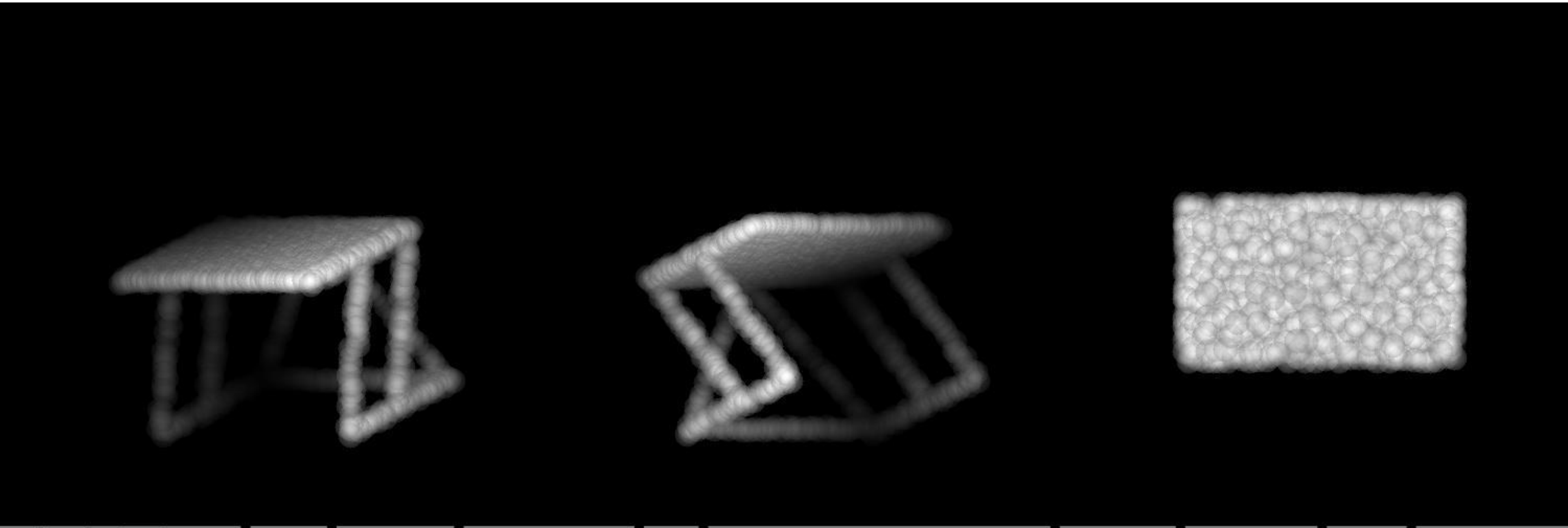


112_label_tv_stand_pred_night_stand.jpg

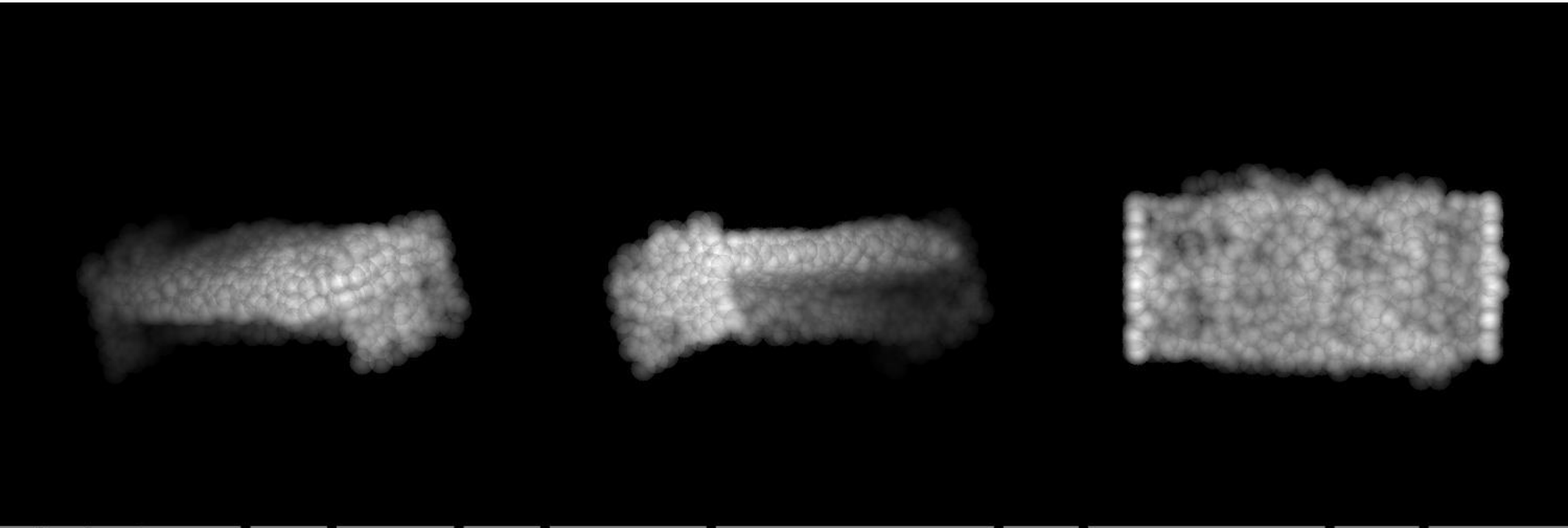


113_label_dresser_pred_wardrobe.jpg

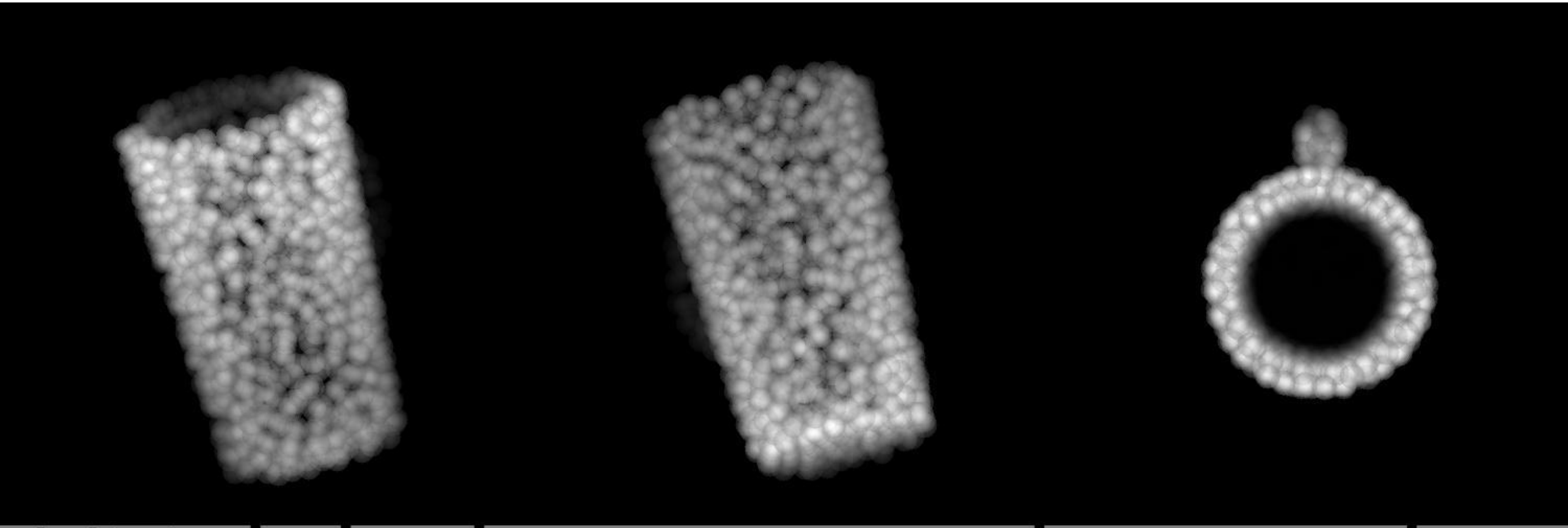

114_label_plant_pred_sink.jpg



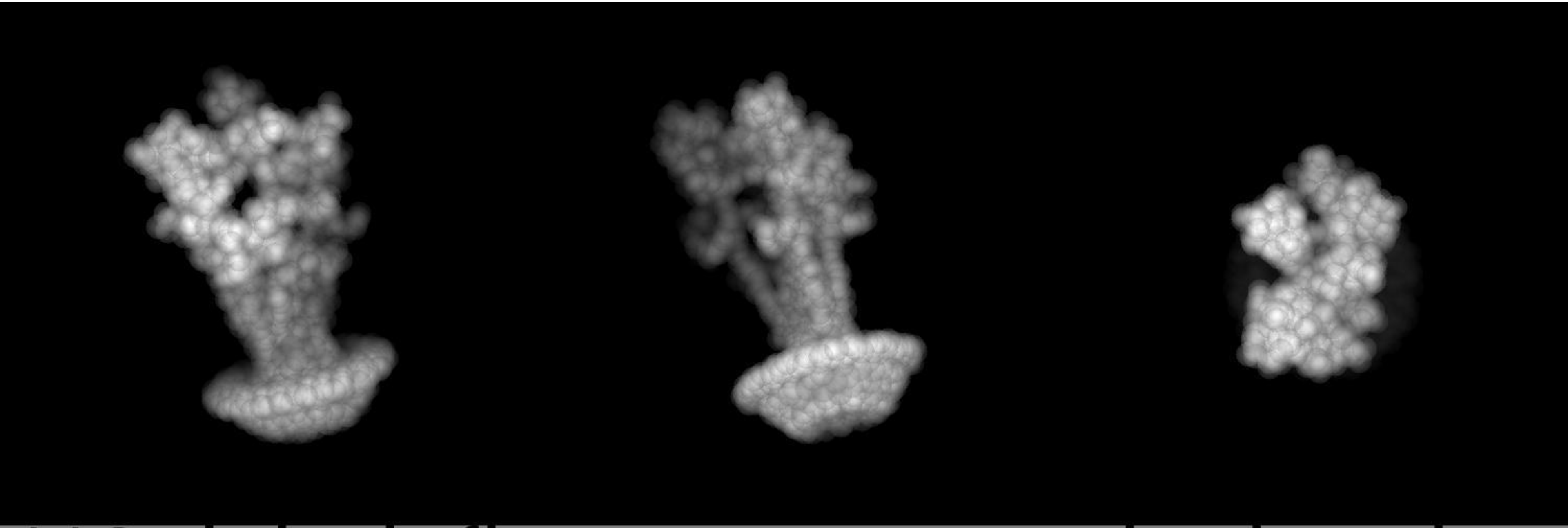
115_label_table_pred_desk.jpg



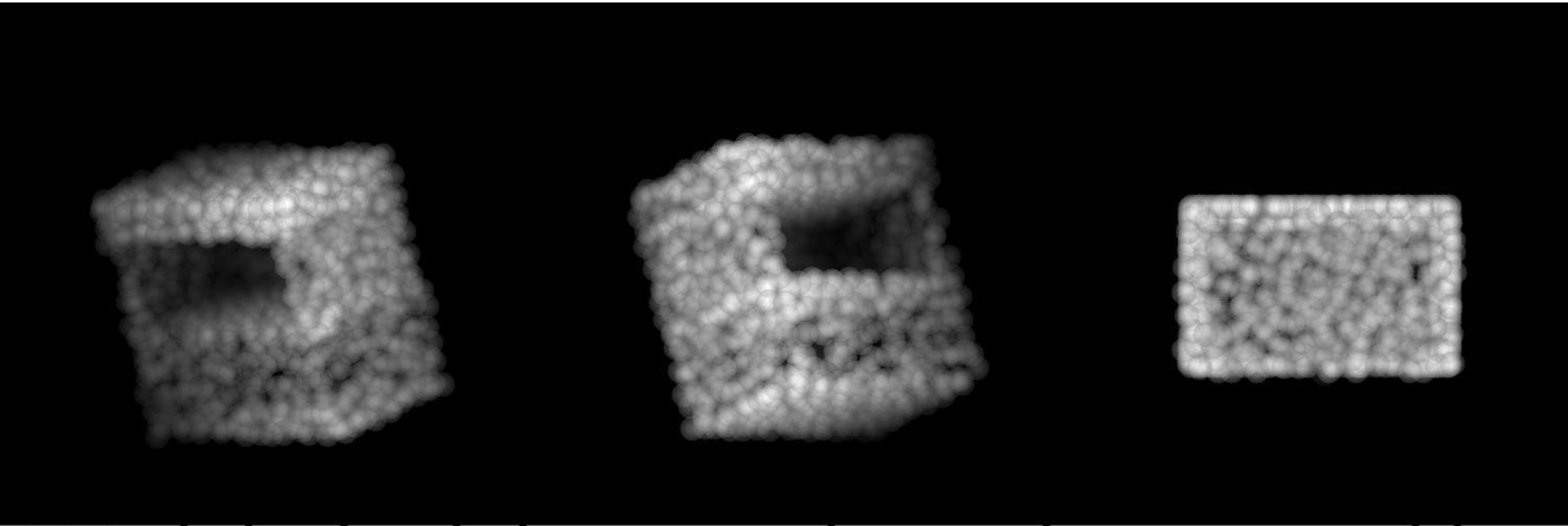
116_label_bed_pred_bench.jpg



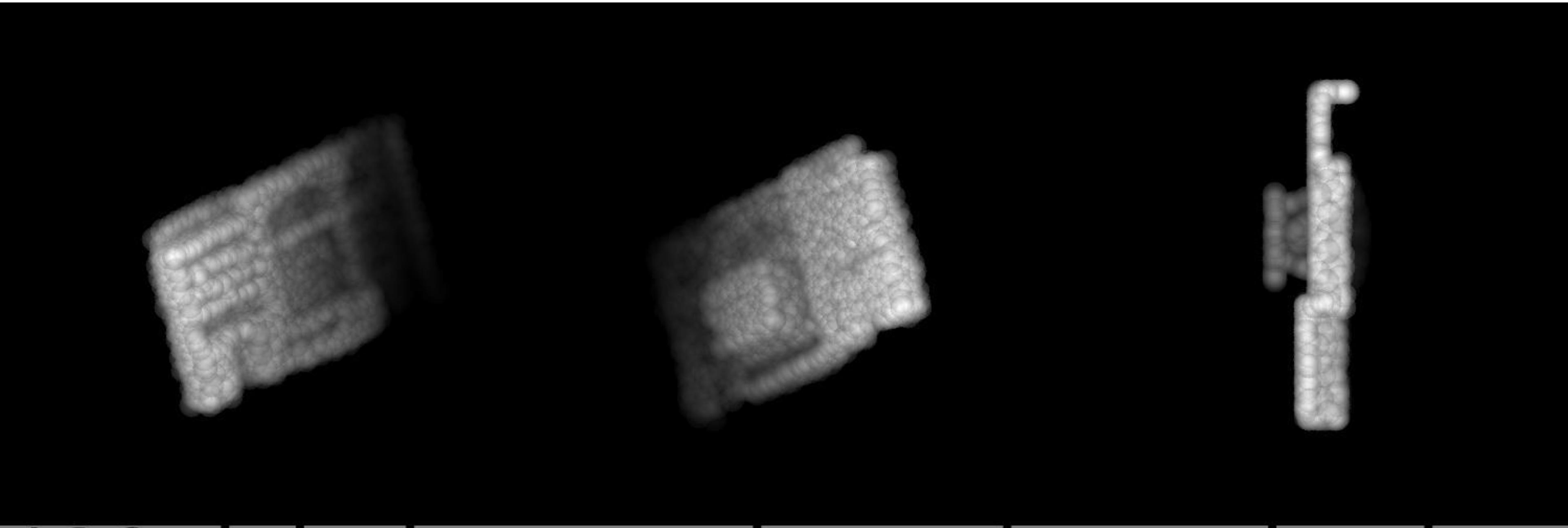
117_label_cup_pred_vase.jpg



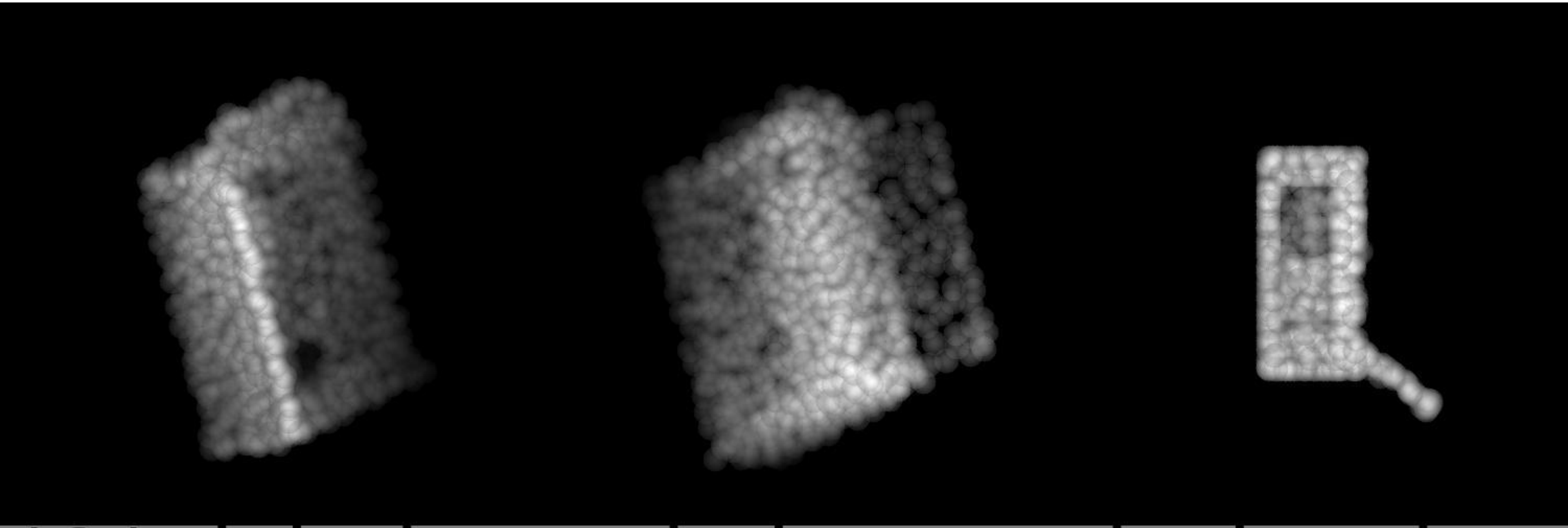
118_label_flower_pot_pred_plant.jpg



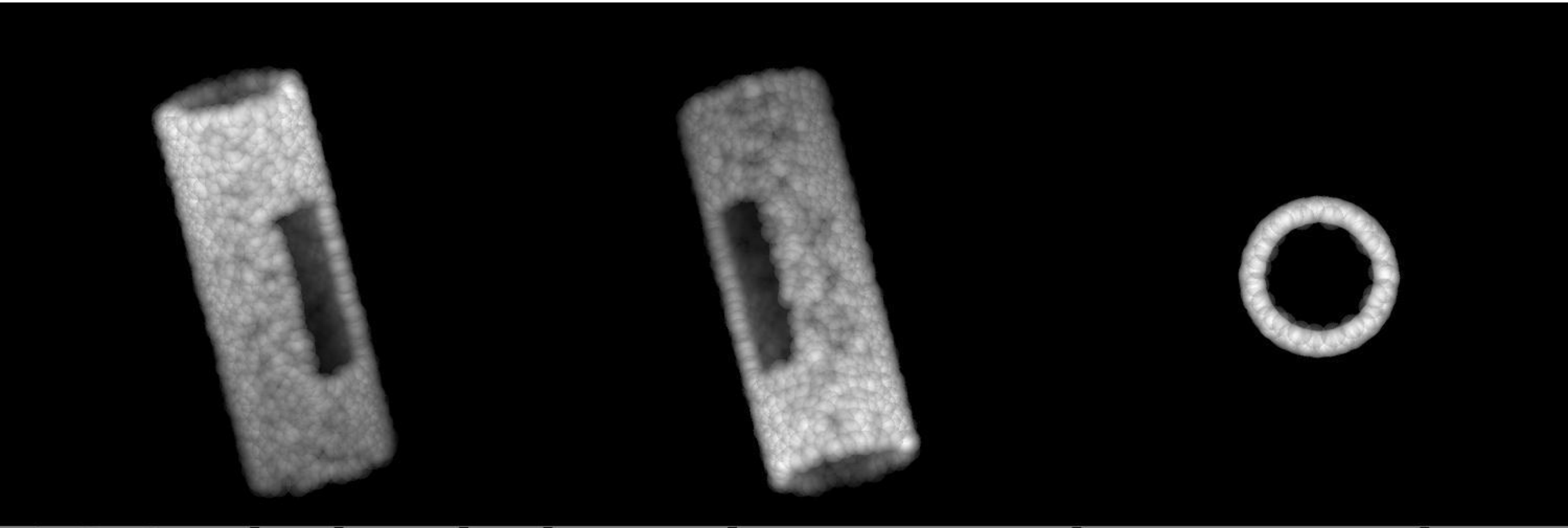
119_label_night_stand_pred_tv_stand.jpg



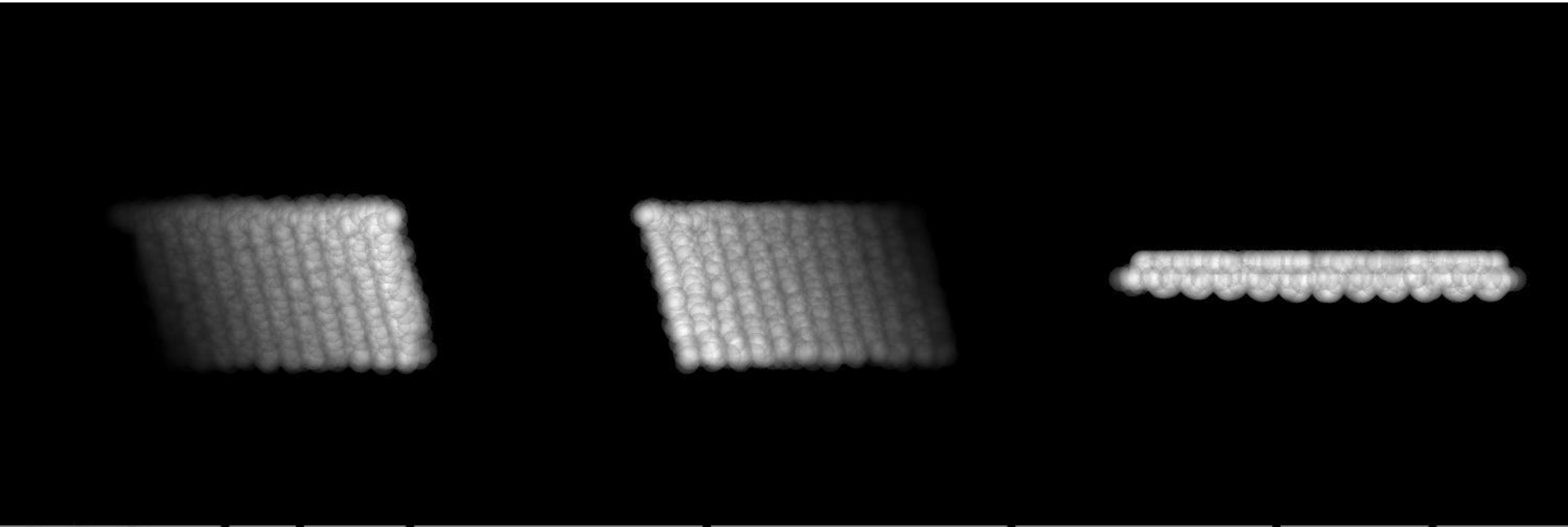
120_label_mantel_pred_monitor.jpg

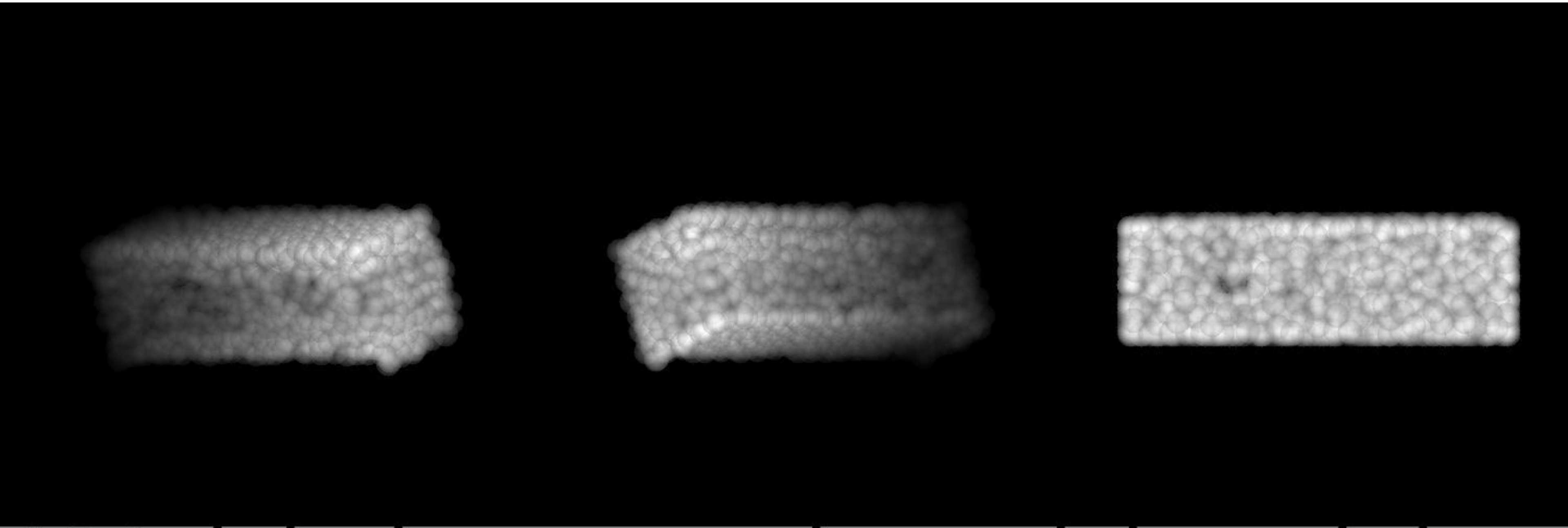


121_label_wardrobe_pred_xbox.jpg

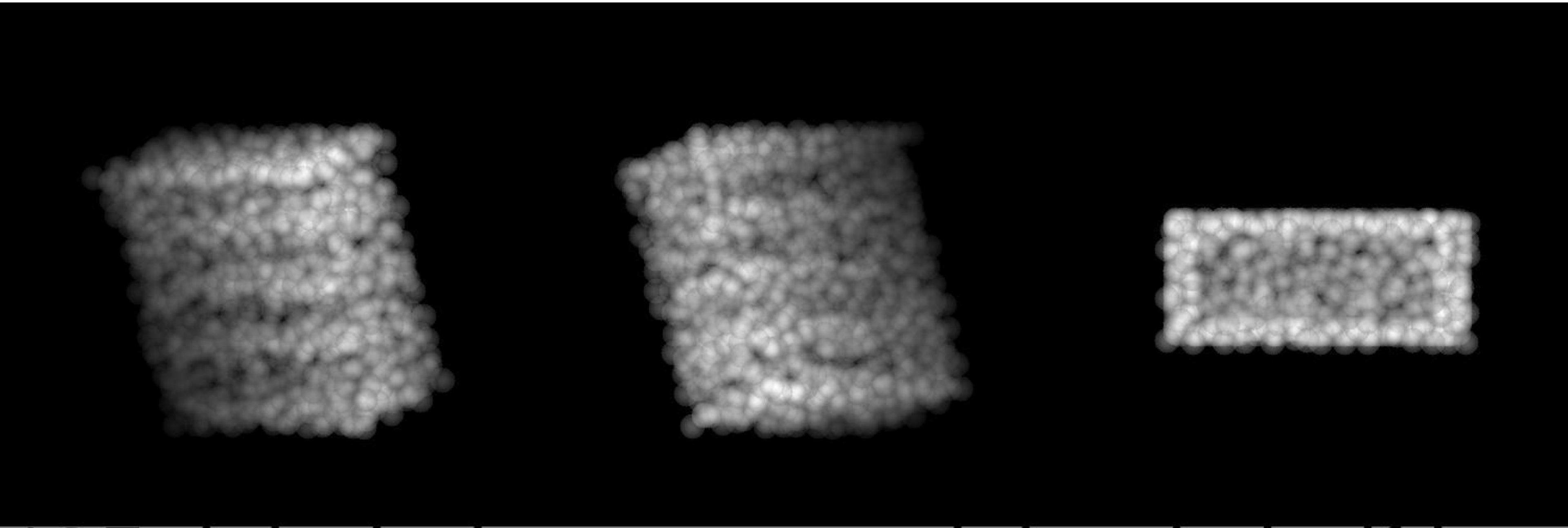


122_label_bottle_pred_vase.jpg

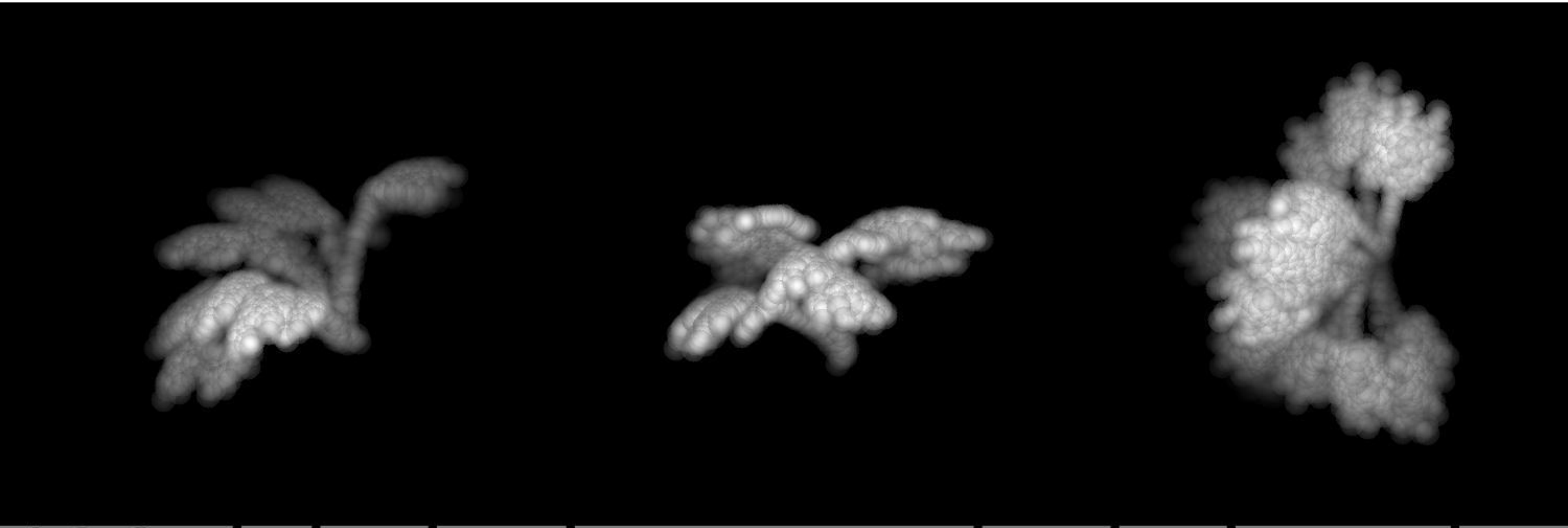

123_label_curtain_pred_monitor.jpg



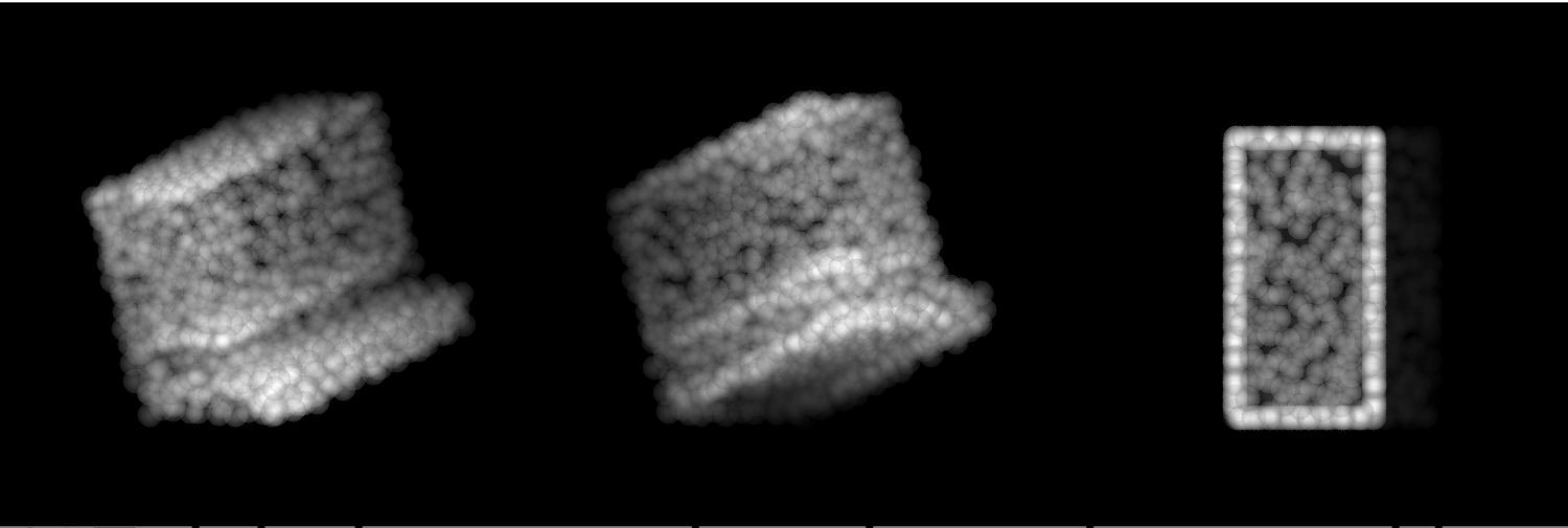
124_label_tv_stand_pred_bench.jpg



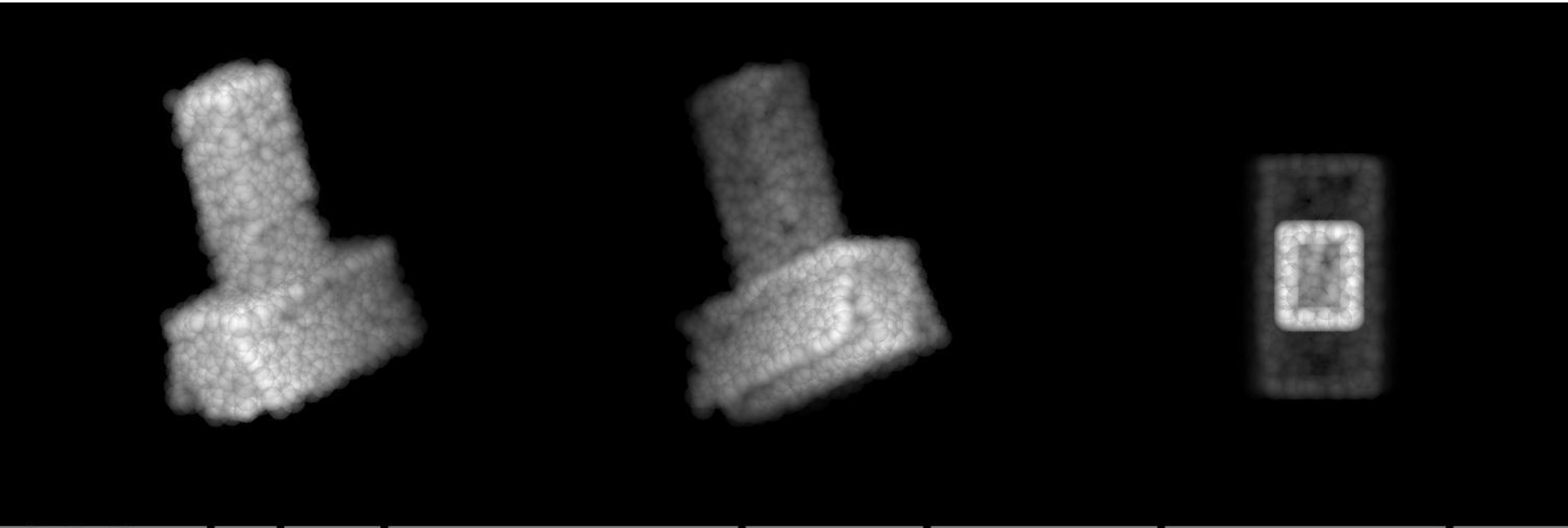
125_label_dresser_pred_bookshelf.jpg



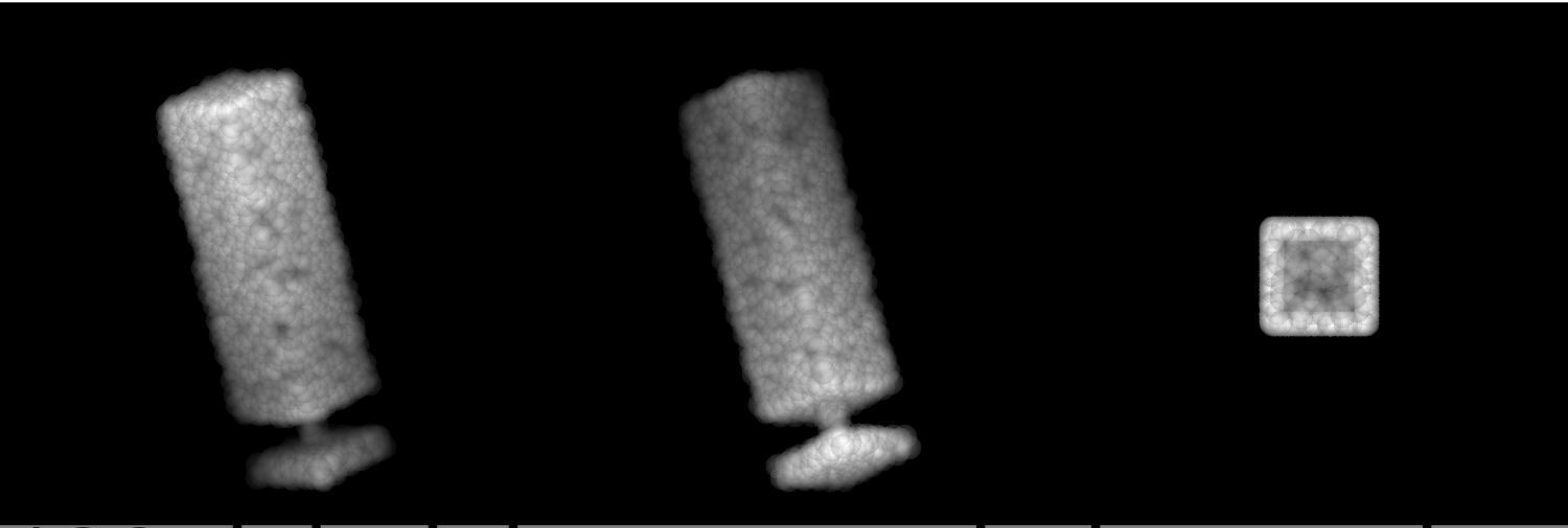
126_label_plant_pred_airplane.jpg



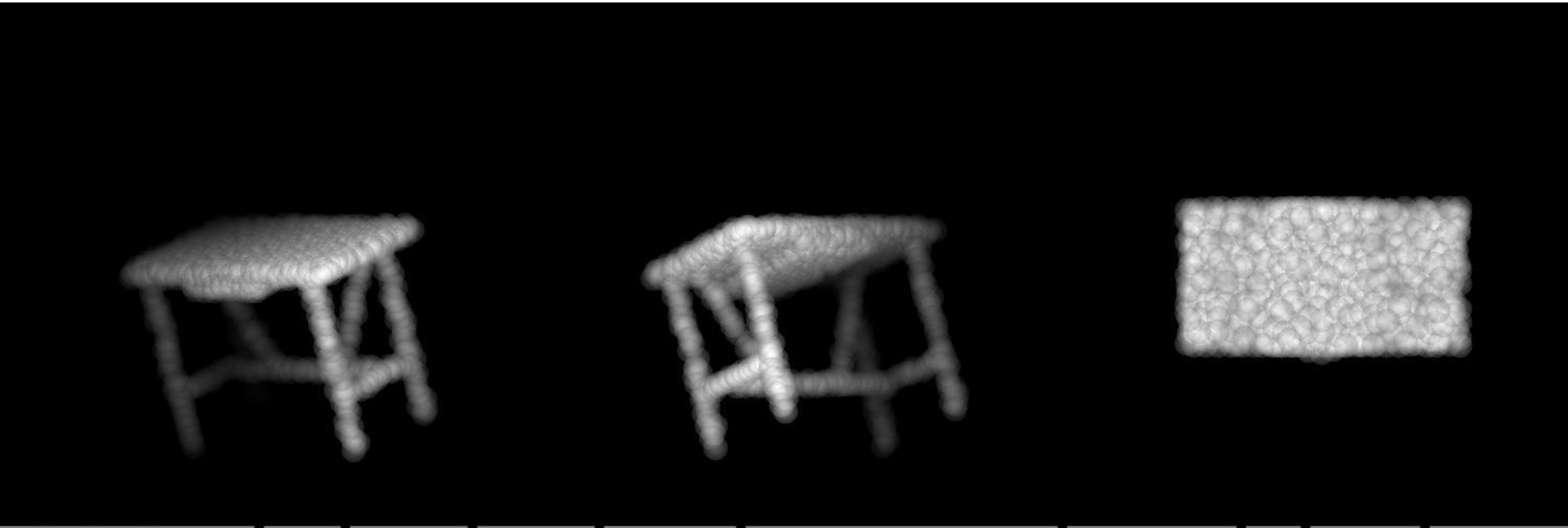
127_label_range_hood_pred_mantel.jpg



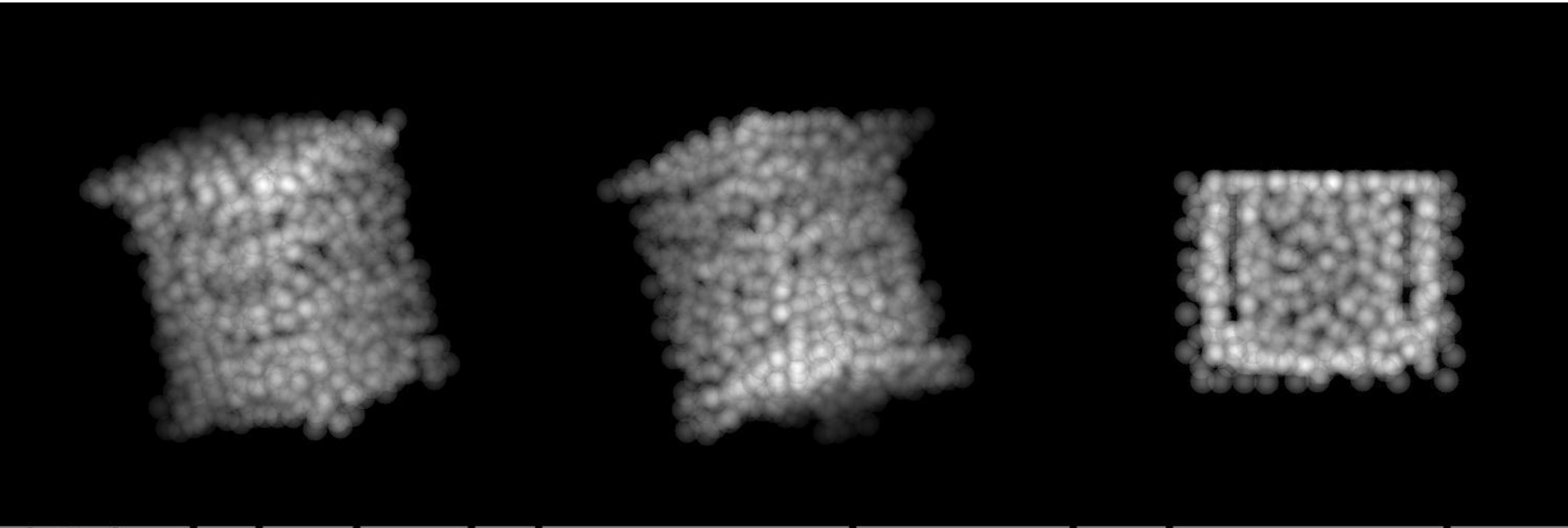
128_label_range_hood_pred_vase.jpg



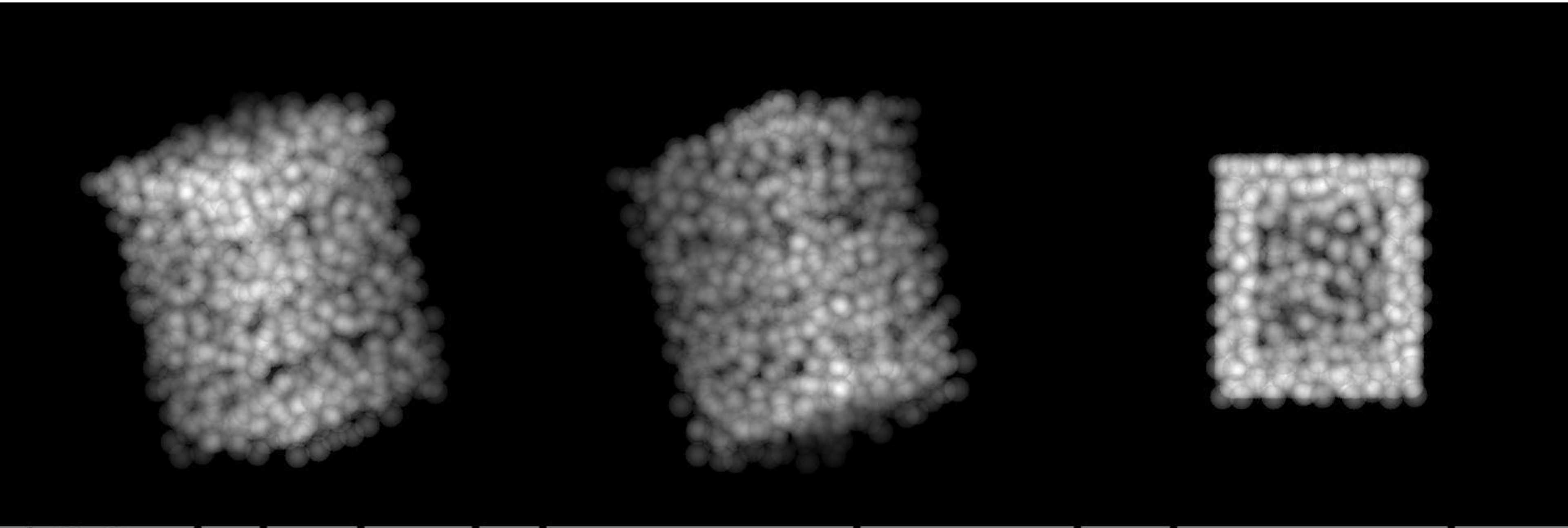
129_label_lamp_pred_dresser.jpg



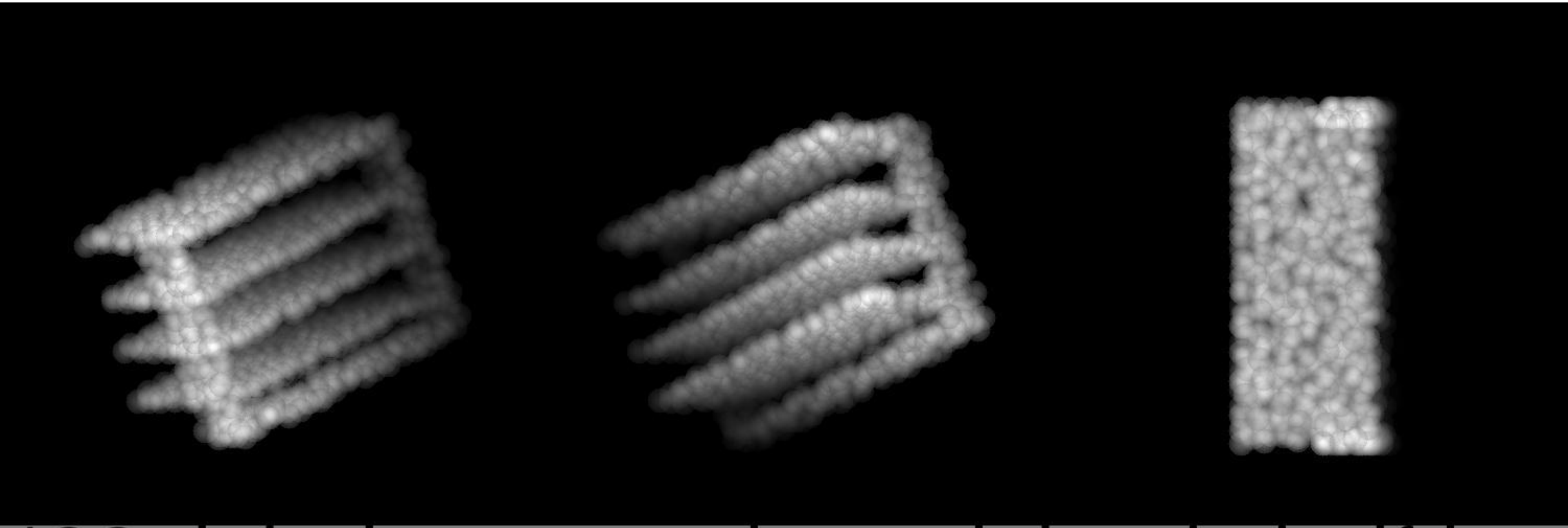
130_label_desk_pred_table.jpg



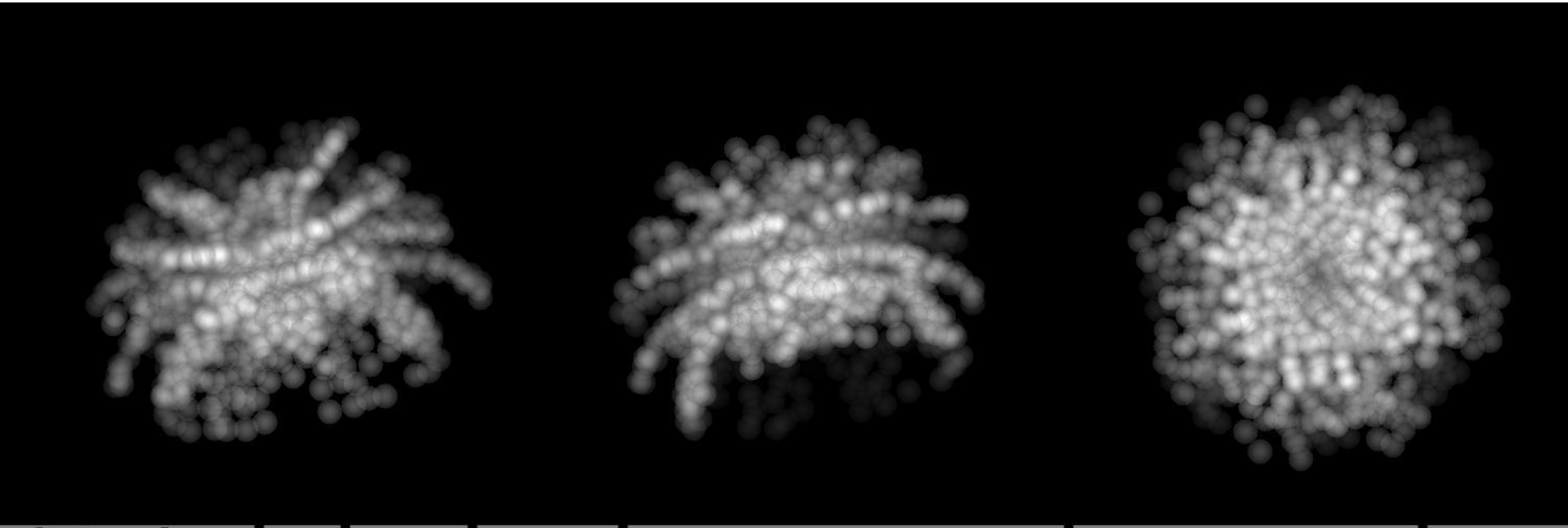
131_label_night_stand_pred_dresser.jpg



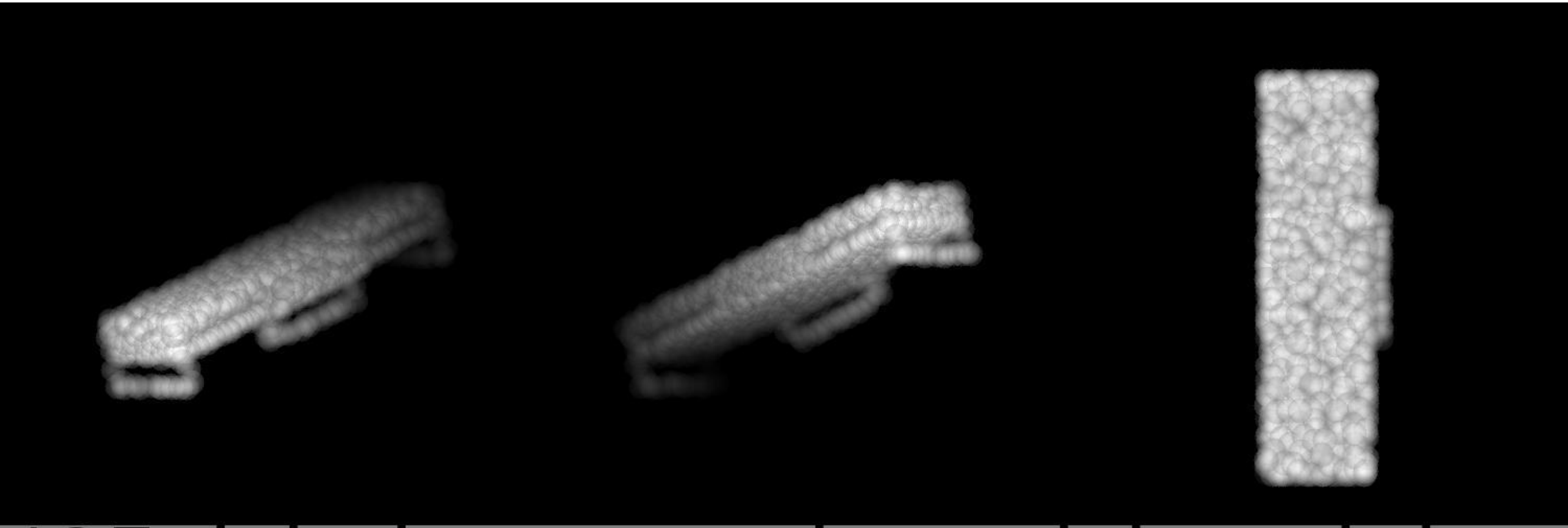
132_label_night_stand_pred_dresser.jpg



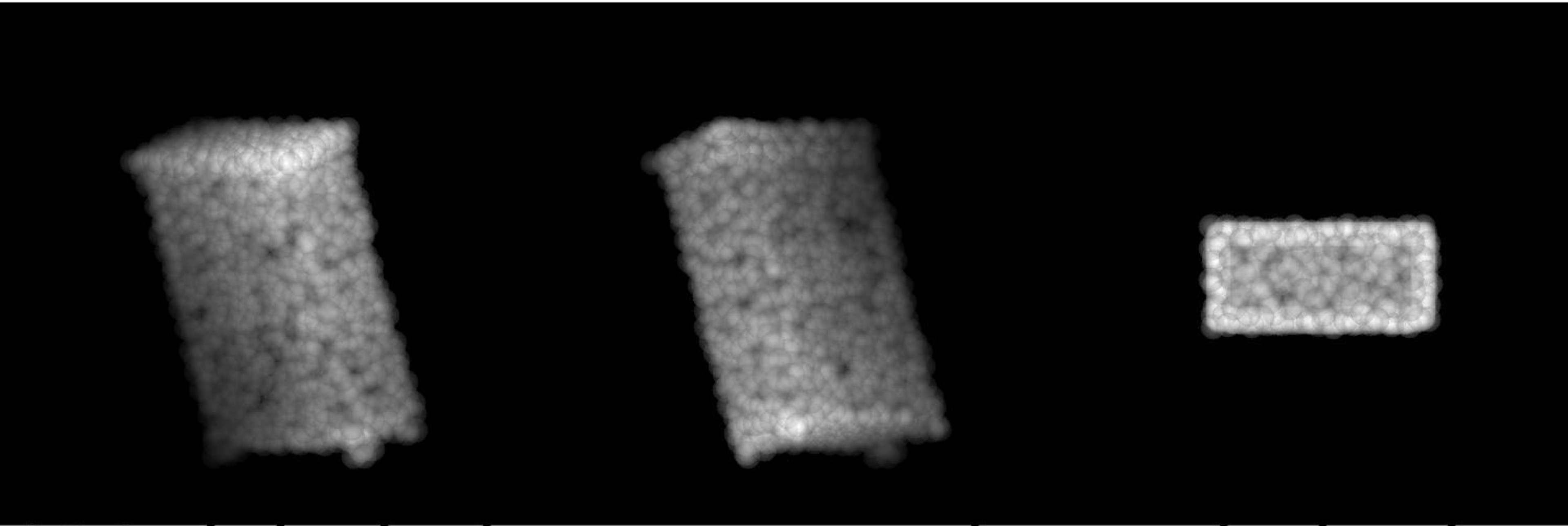
133_label_tv_stand_pred_bookshelf.jpg



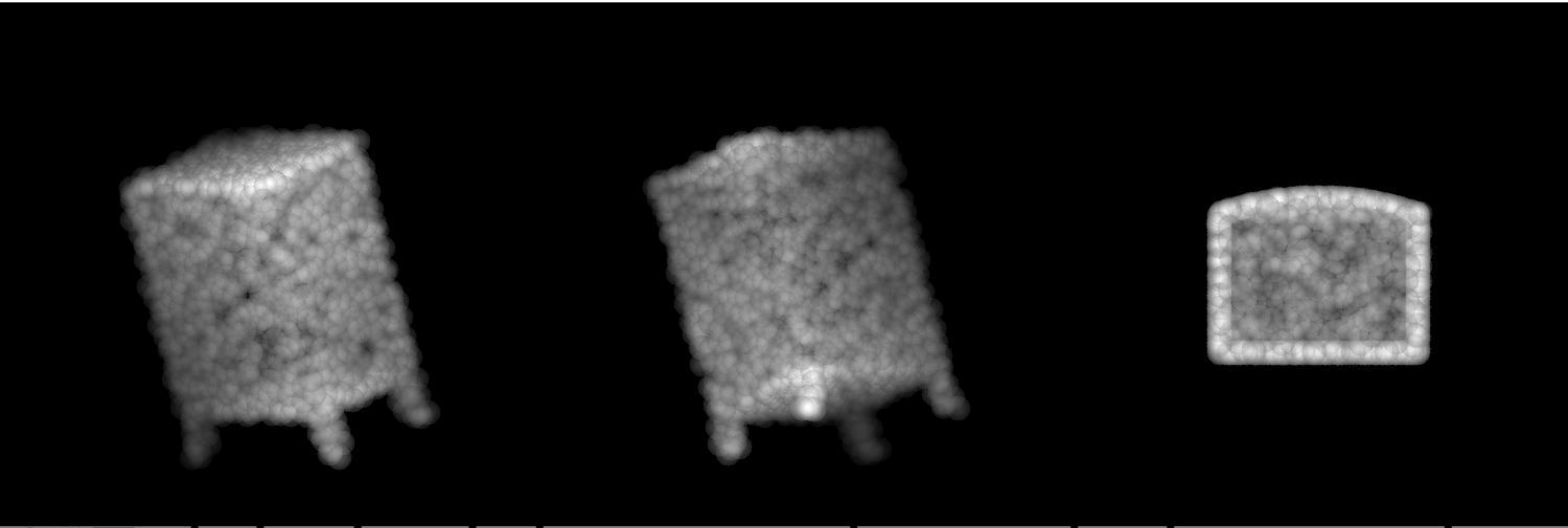
134_label_plant_pred_vase.jpg



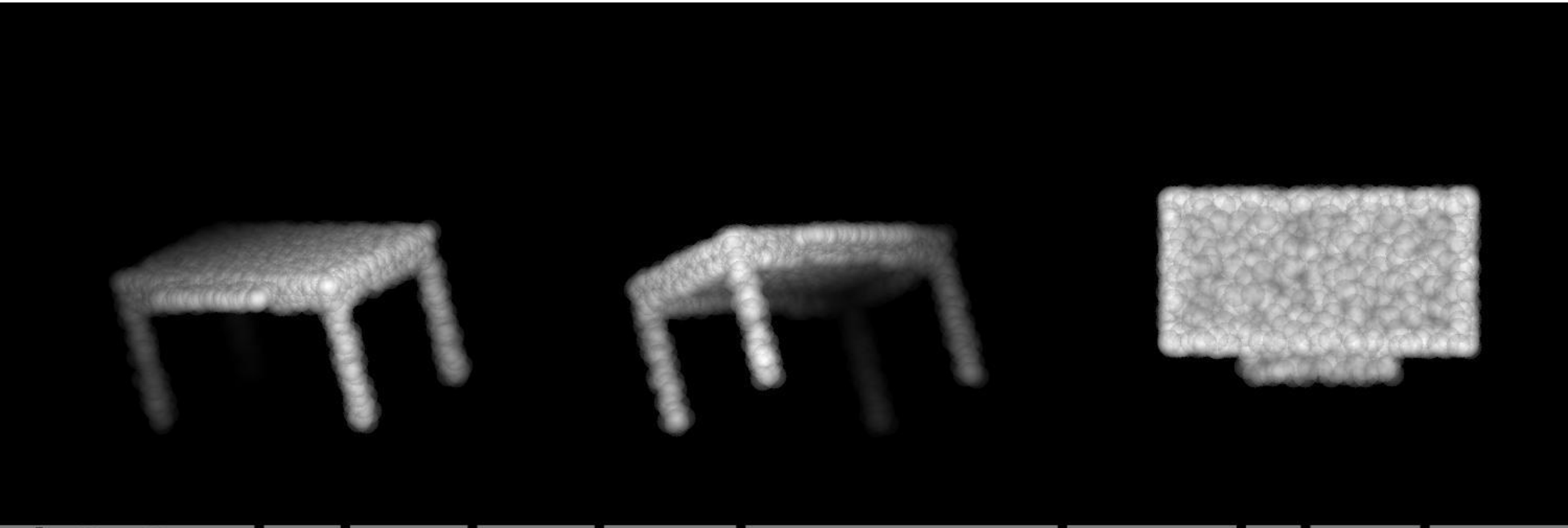
135_label_tv_stand_pred_bench.jpg



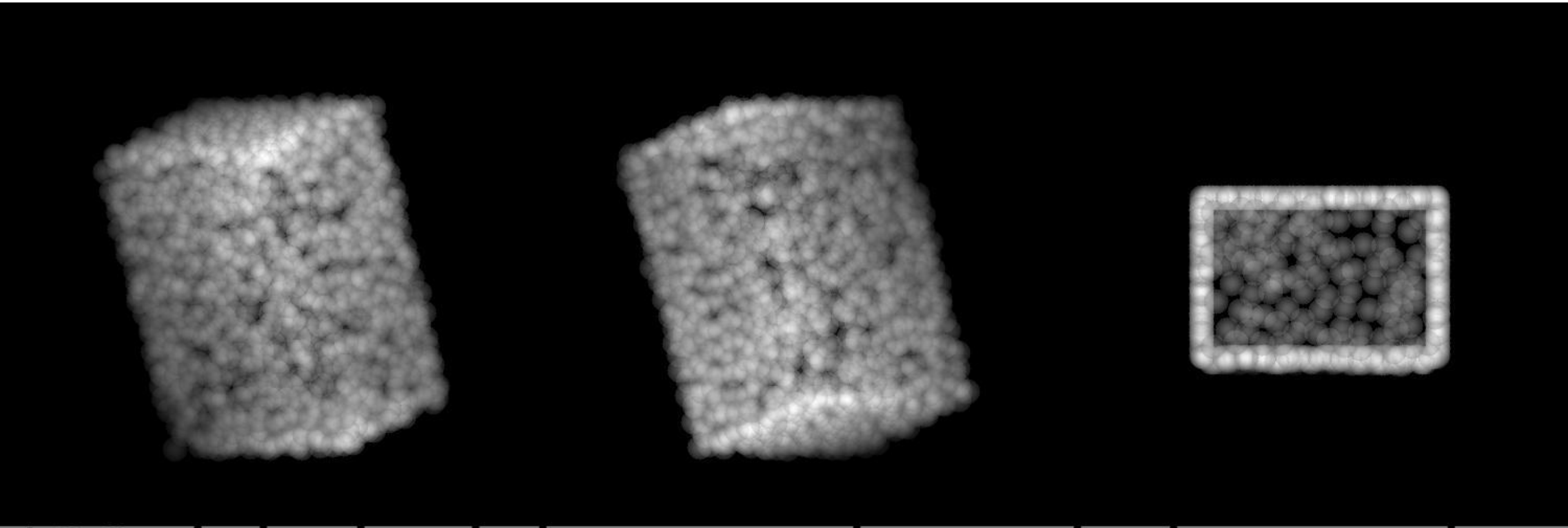
136_label_dresser_pred_wardrobe.jpg



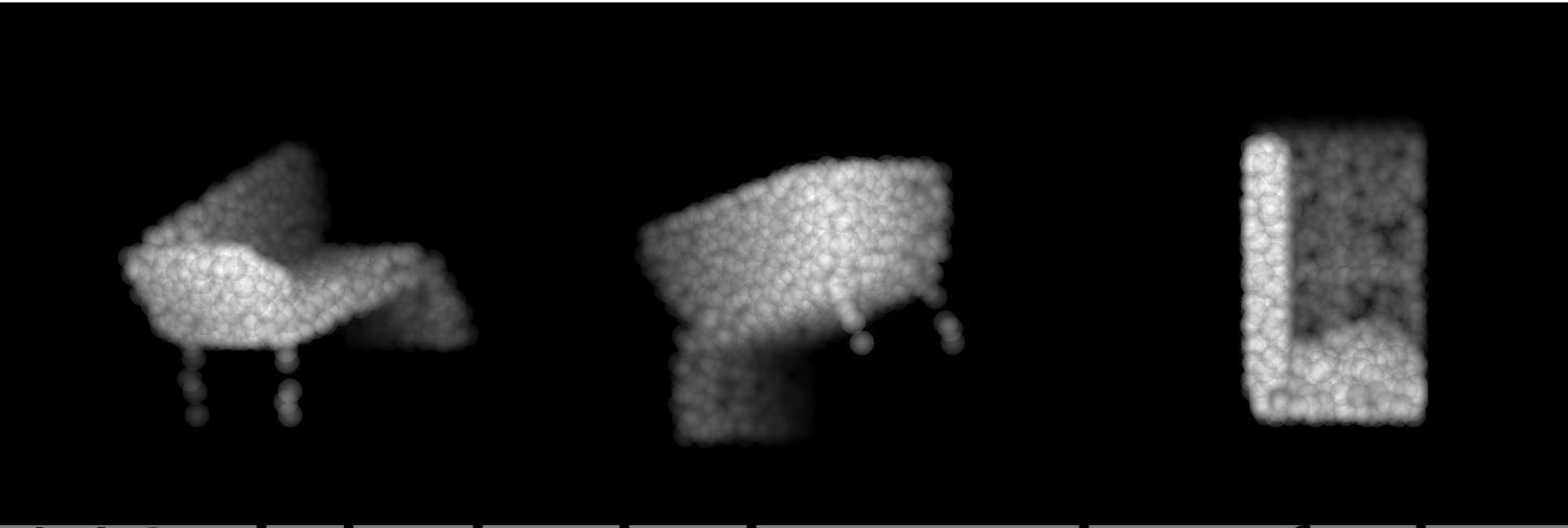
137_label_night_stand_pred_dresser.jpg



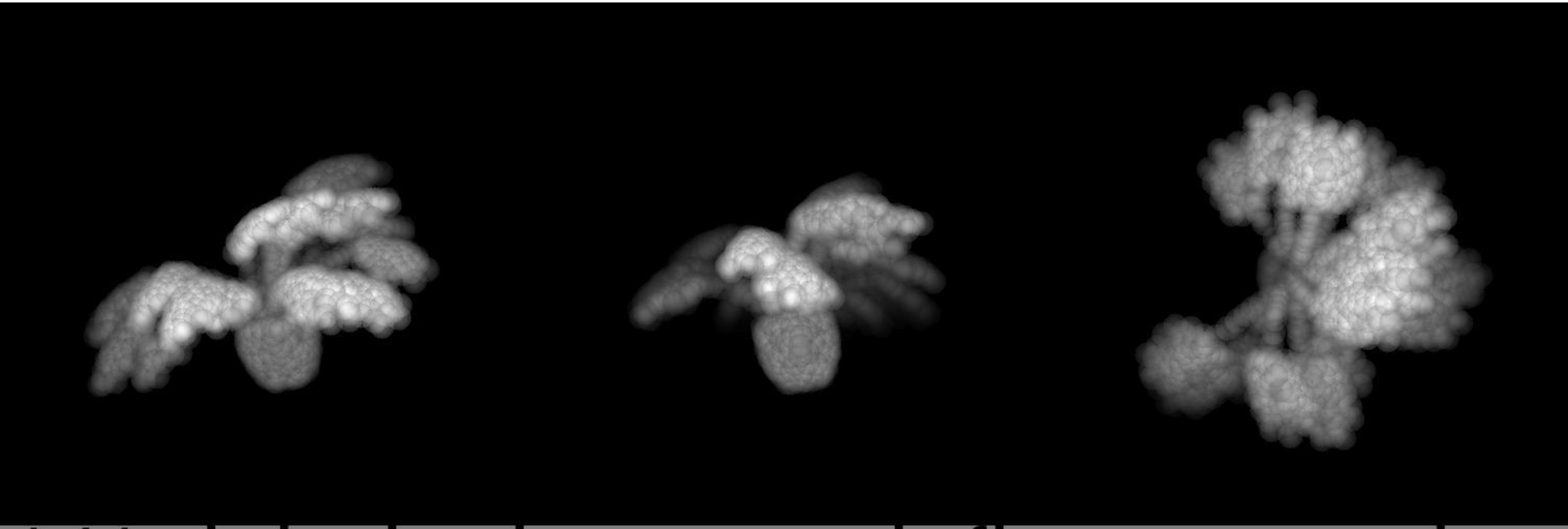
138_label_desk_pred_table.jpg



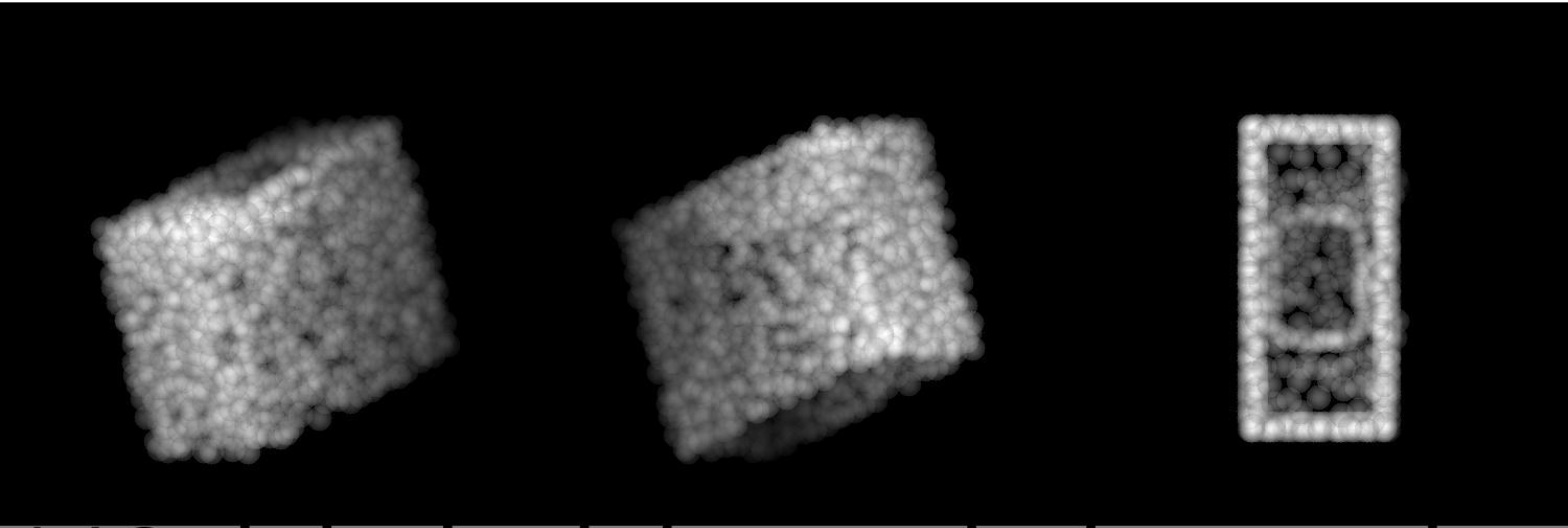
139_label_night_stand_pred_dresser.jpg



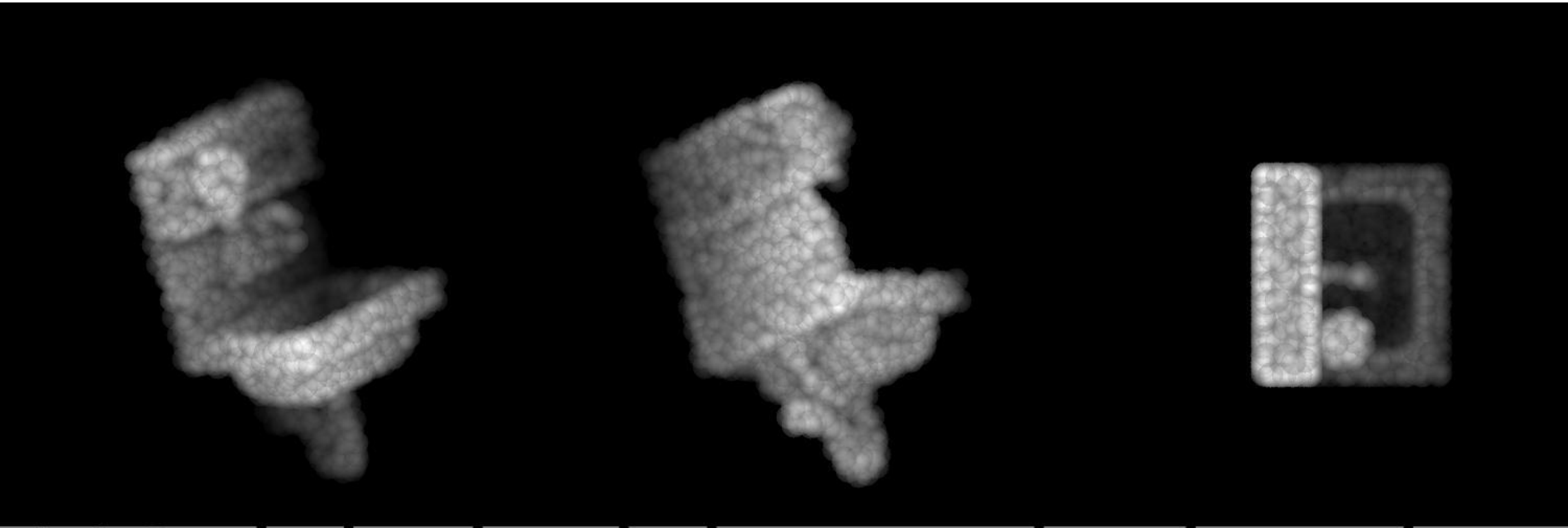
140_label_chair_pred_sofa.jpg



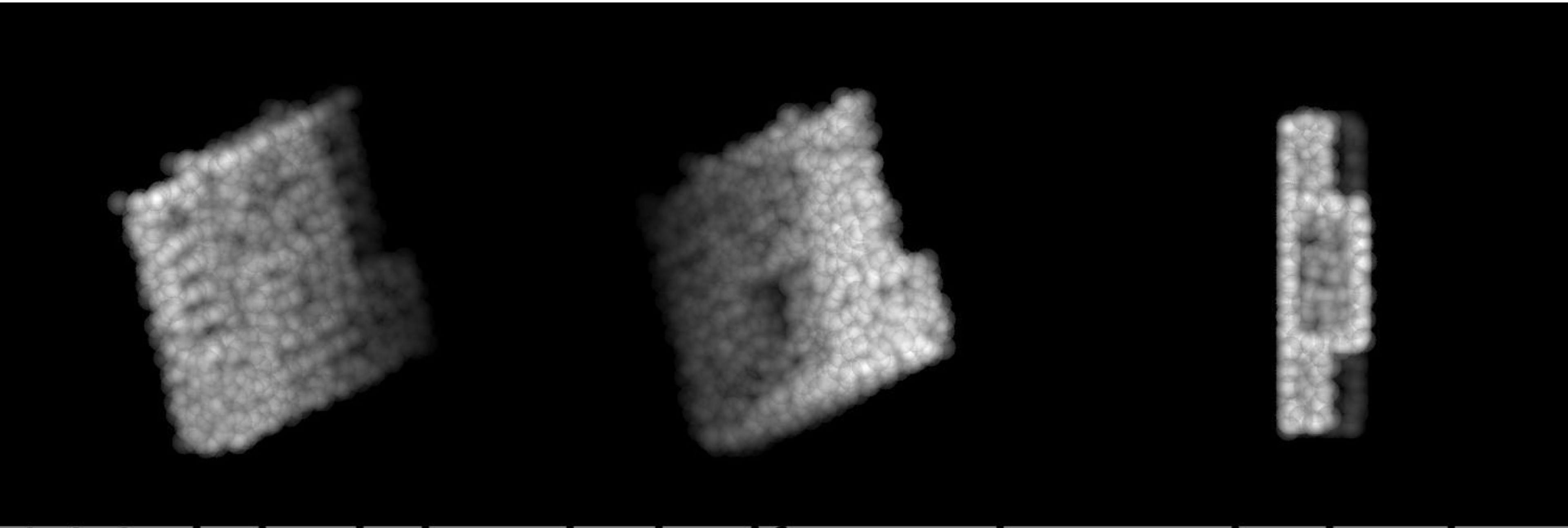
141_label_plant_pred_flower_pot.jpg



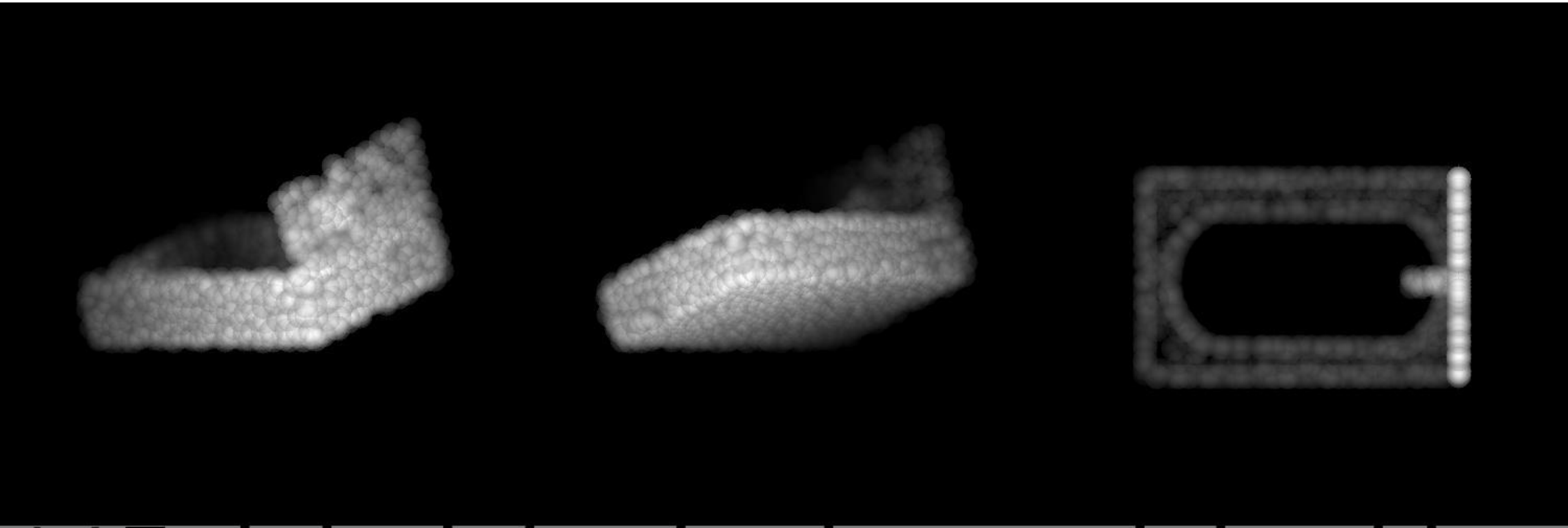
142_label_sink_pred_dresser.jpg



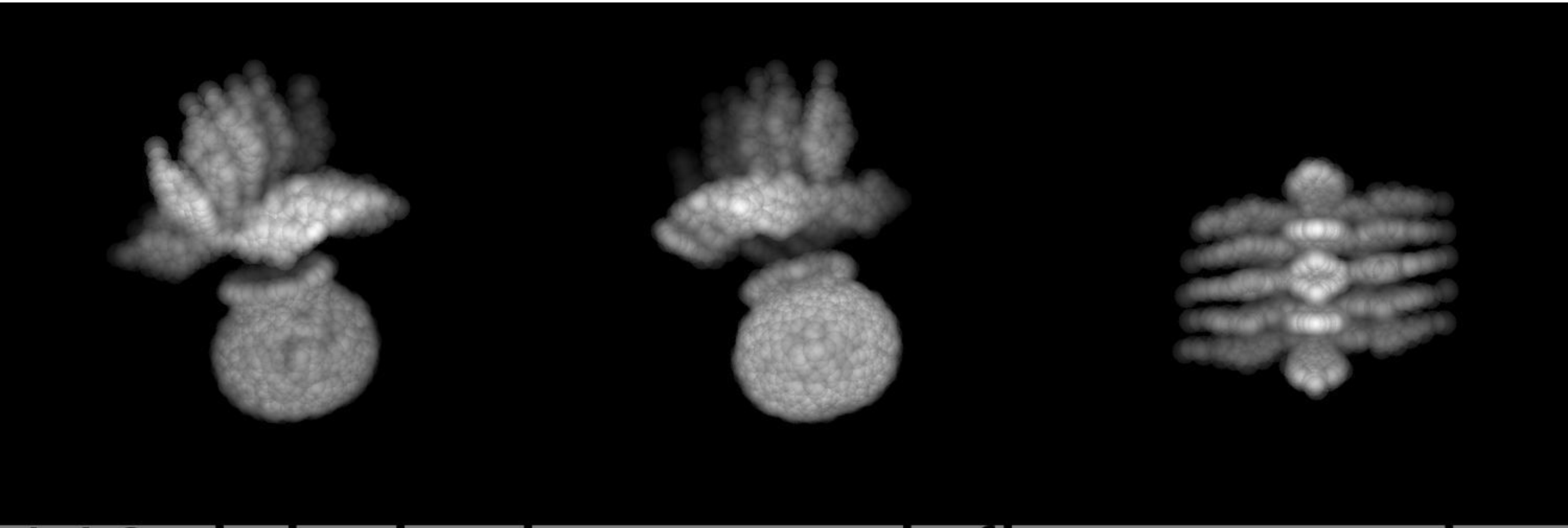
143_label_sink_pred_plant.jpg



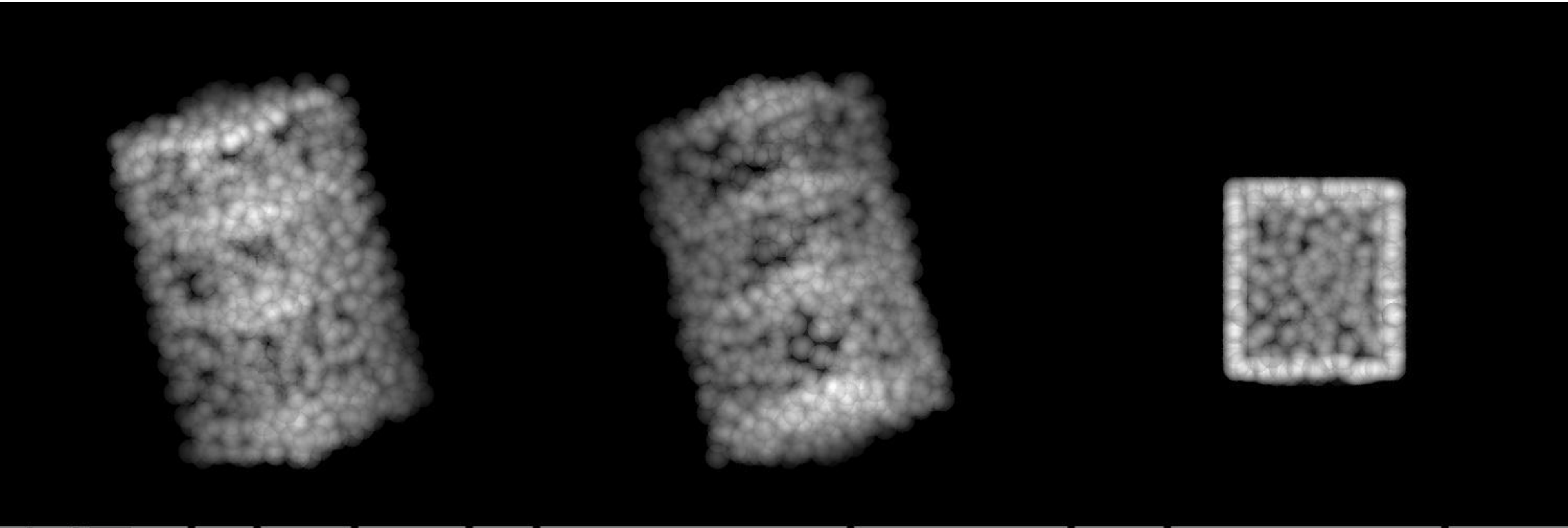
144_label_bookshelf_pred_wardrobe.jpg



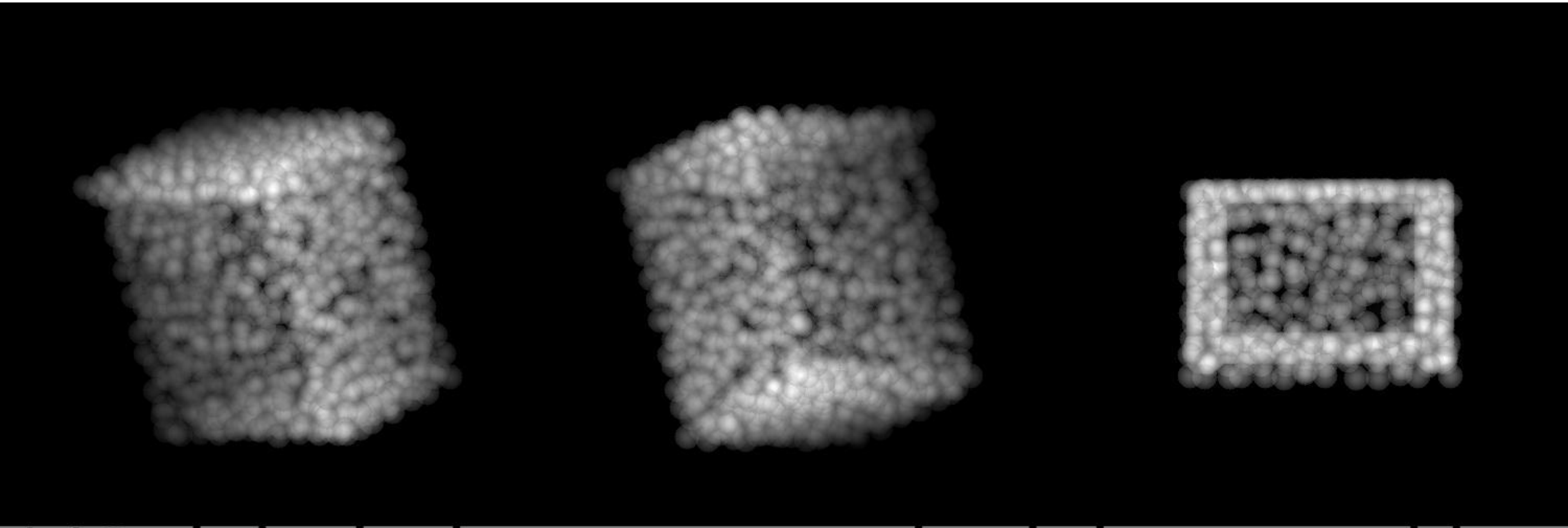
145_label_bathtub_pred_bed.jpg



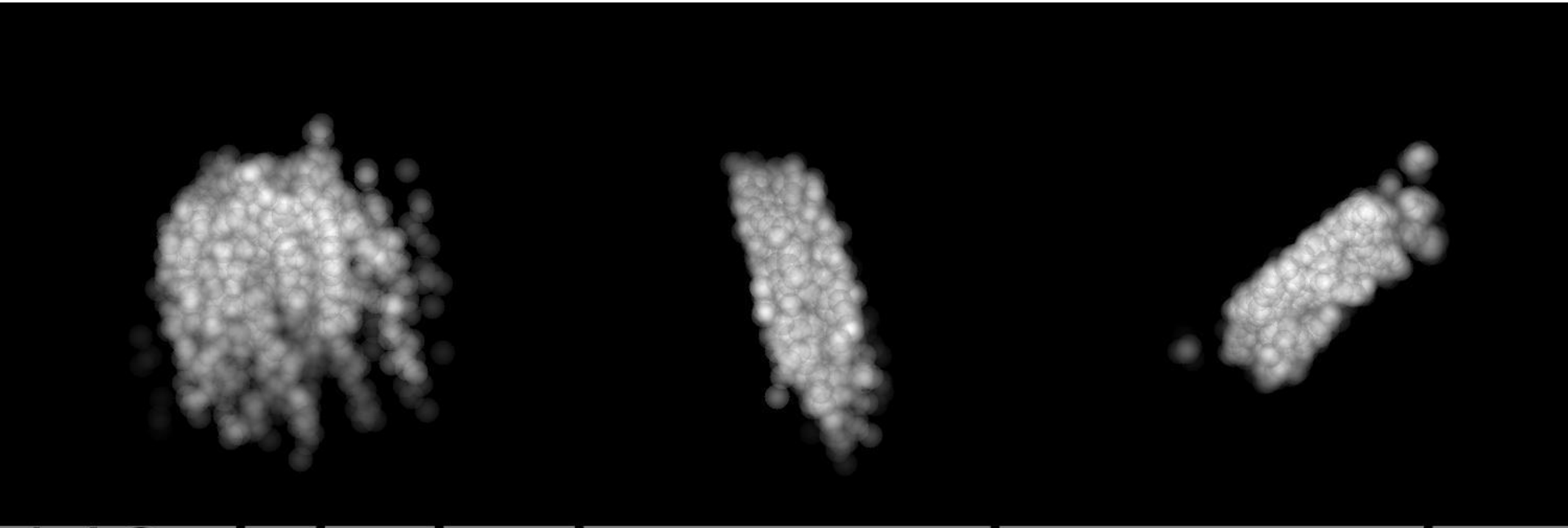
146_label_plant_pred_flower_pot.jpg



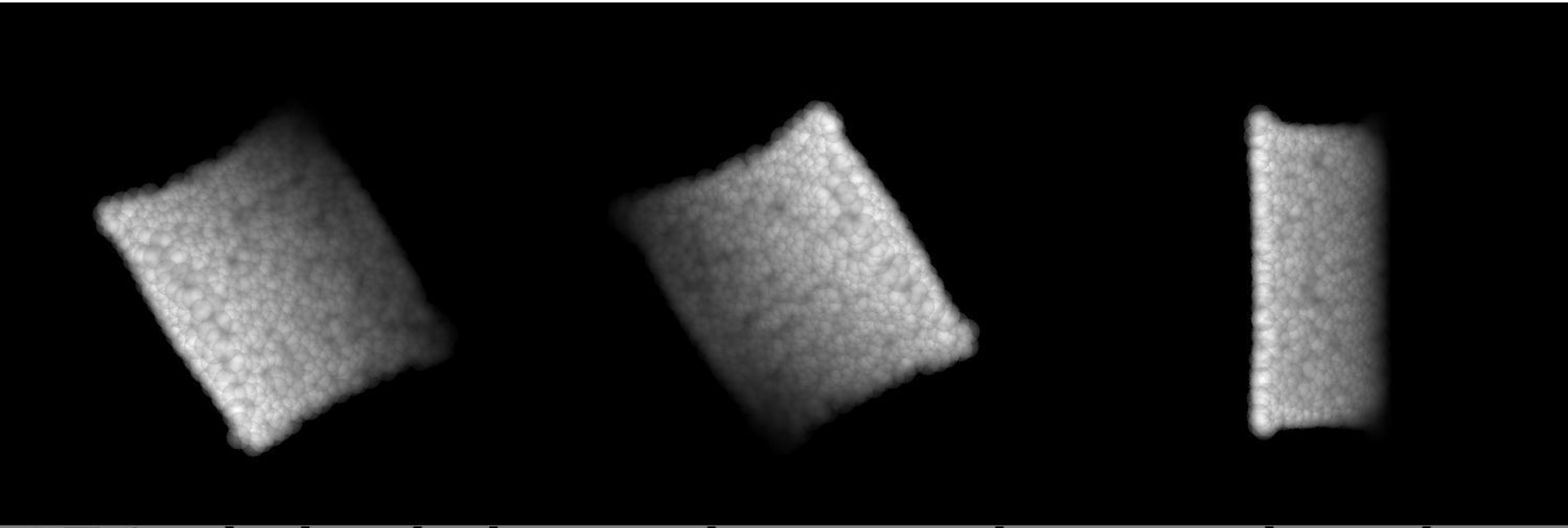
147_label_night_stand_pred_dresser.jpg



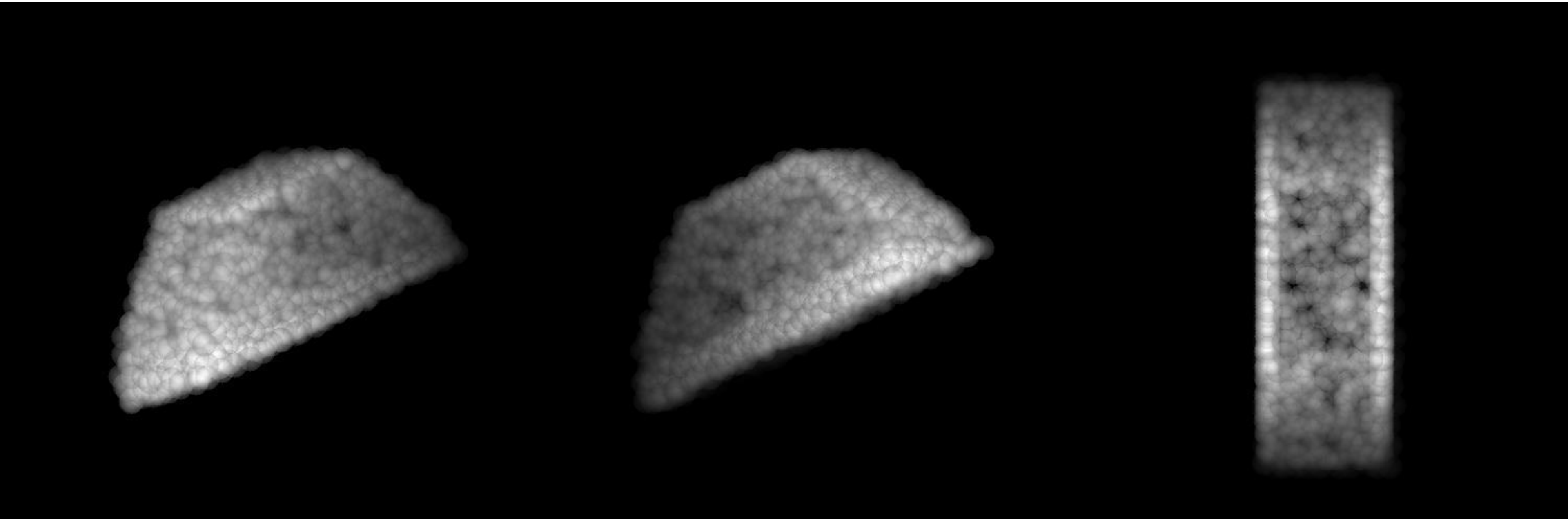
148_label_dresser_pred_night_stand.jpg



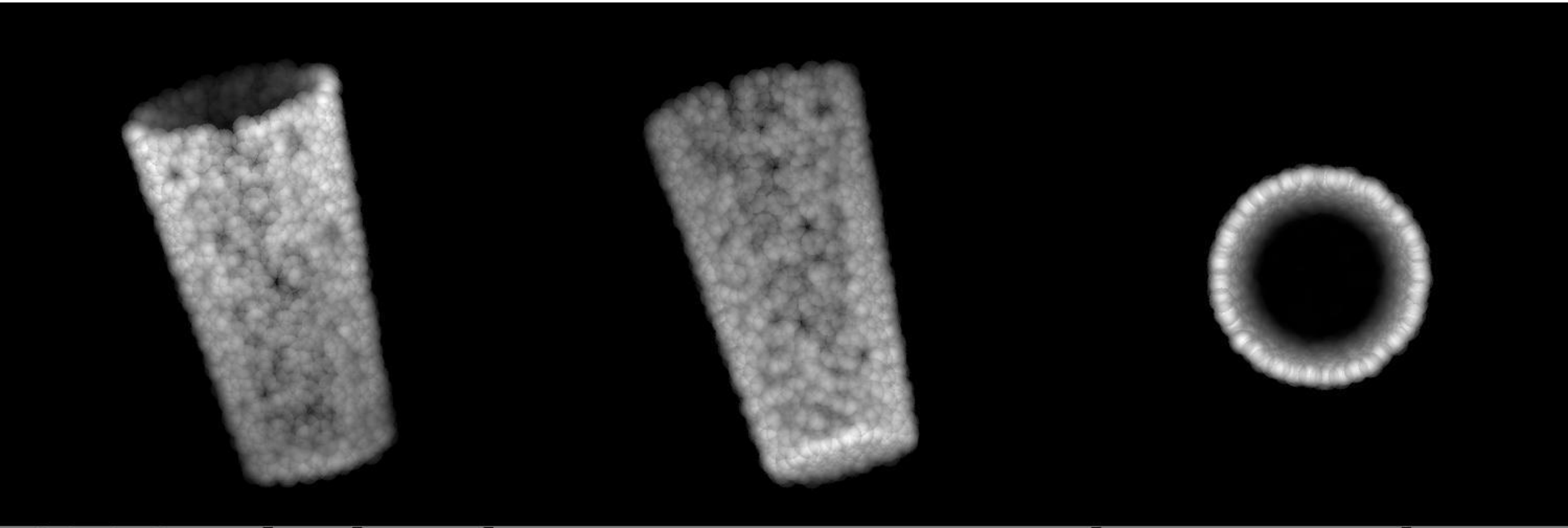
149_label_plant_pred_person.jpg



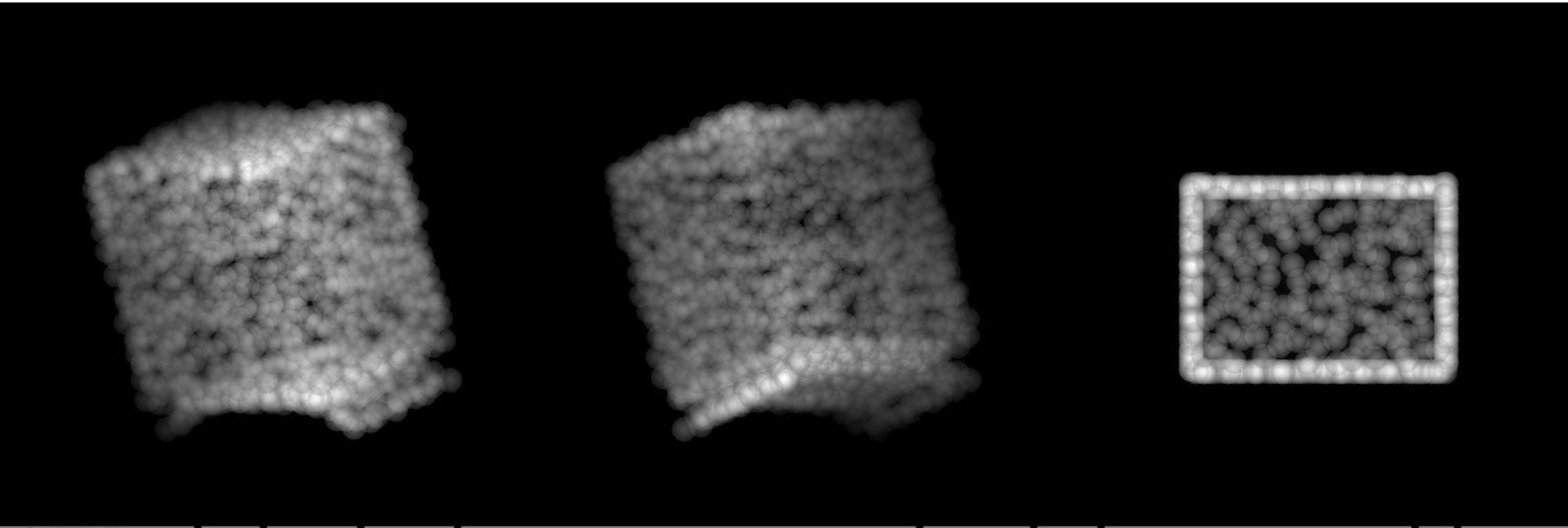
150_label_bench_pred_monitor.jpg



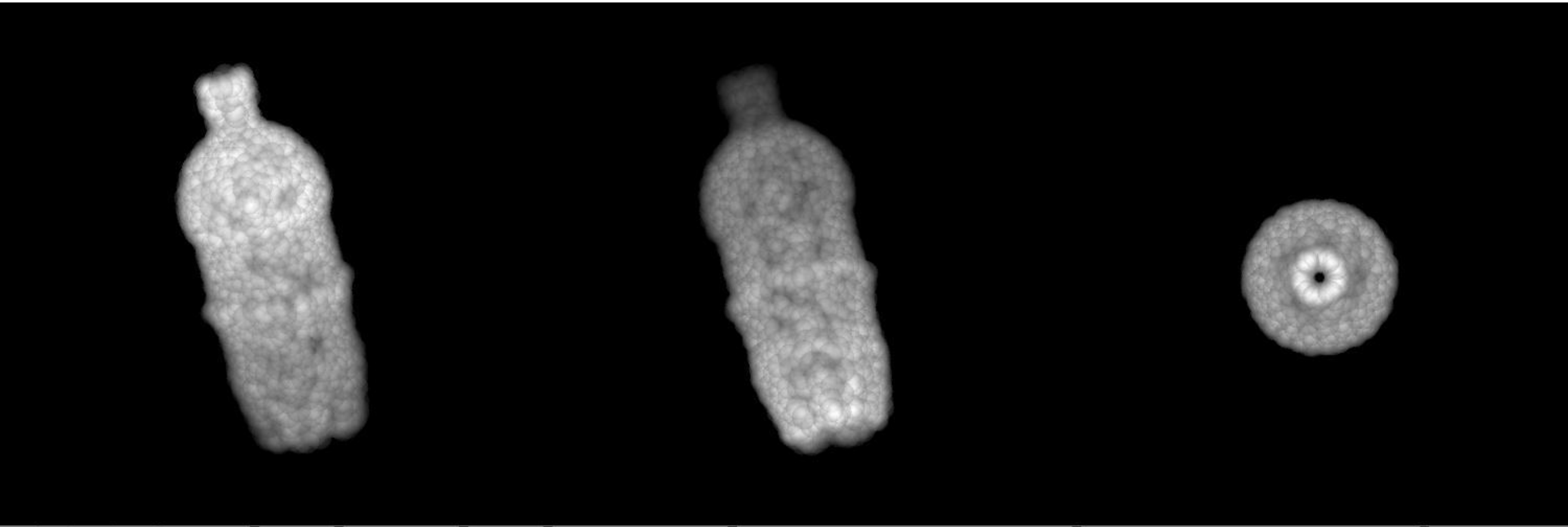
151_label_range_hood_pred_radio.jpg



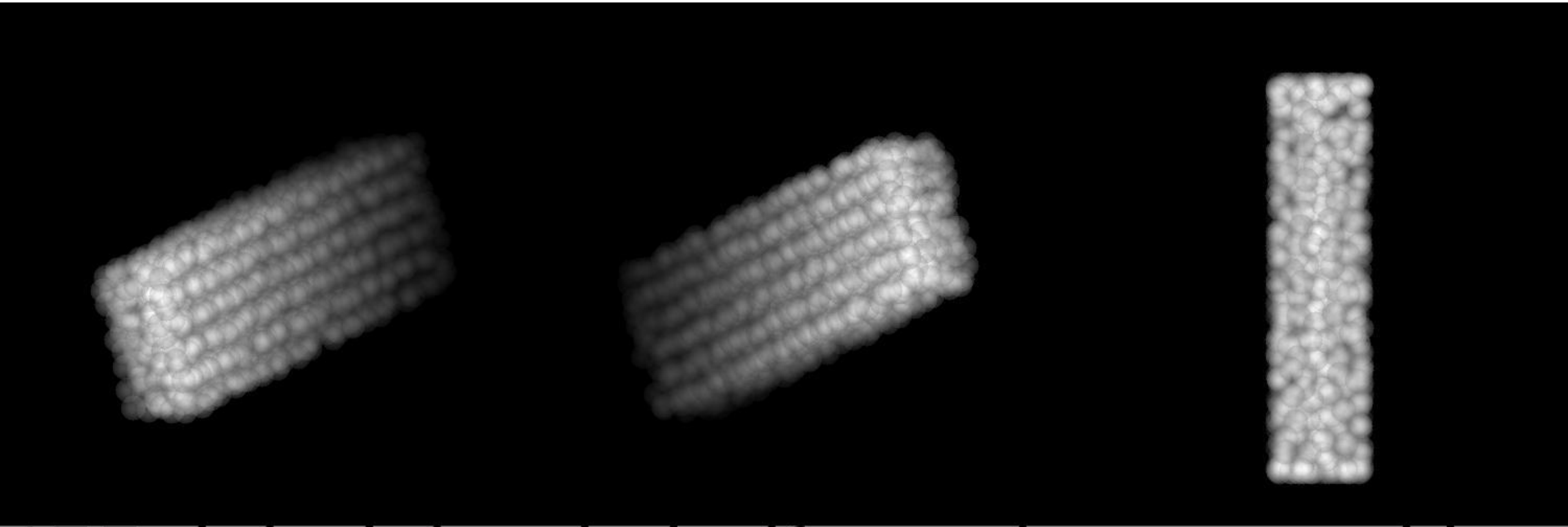
152_label_vase_pred_cup.jpg



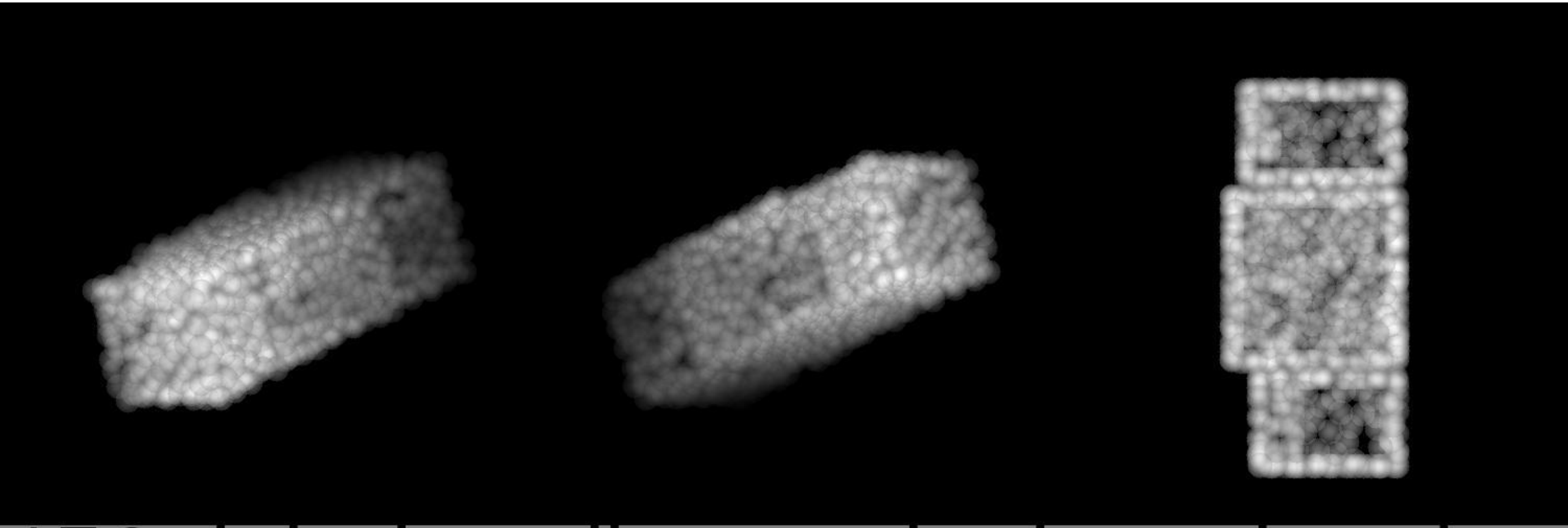
153_label_dresser_pred_night_stand.jpg

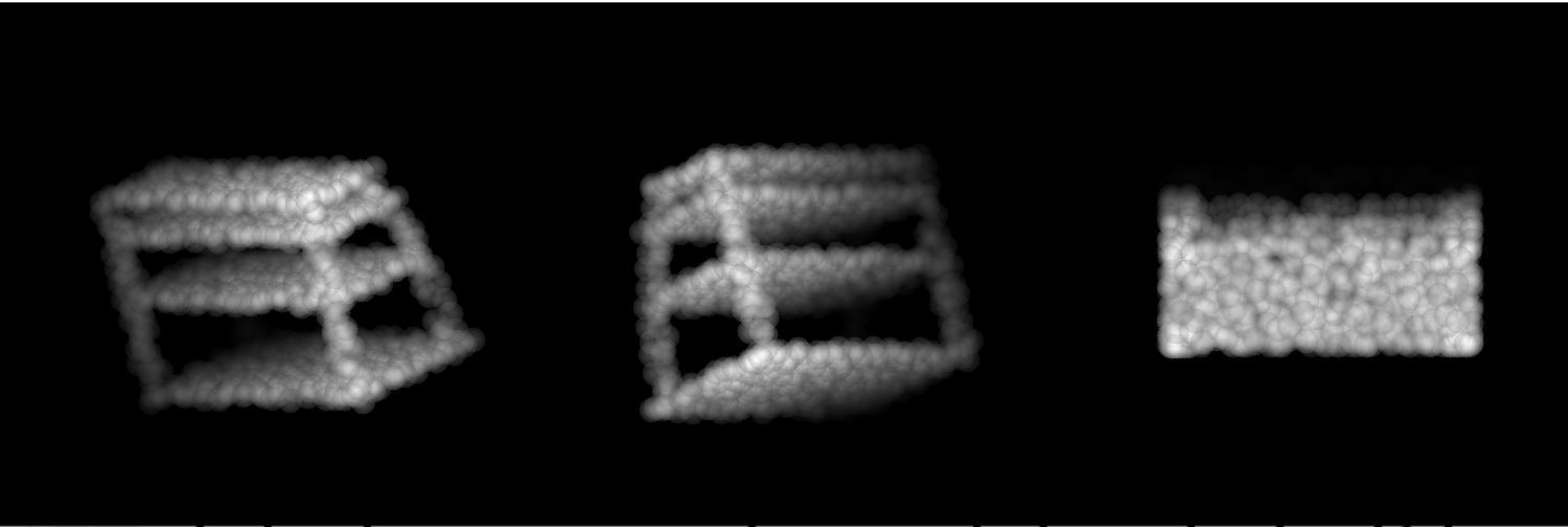


154_label_bottle_pred_vase.jpg

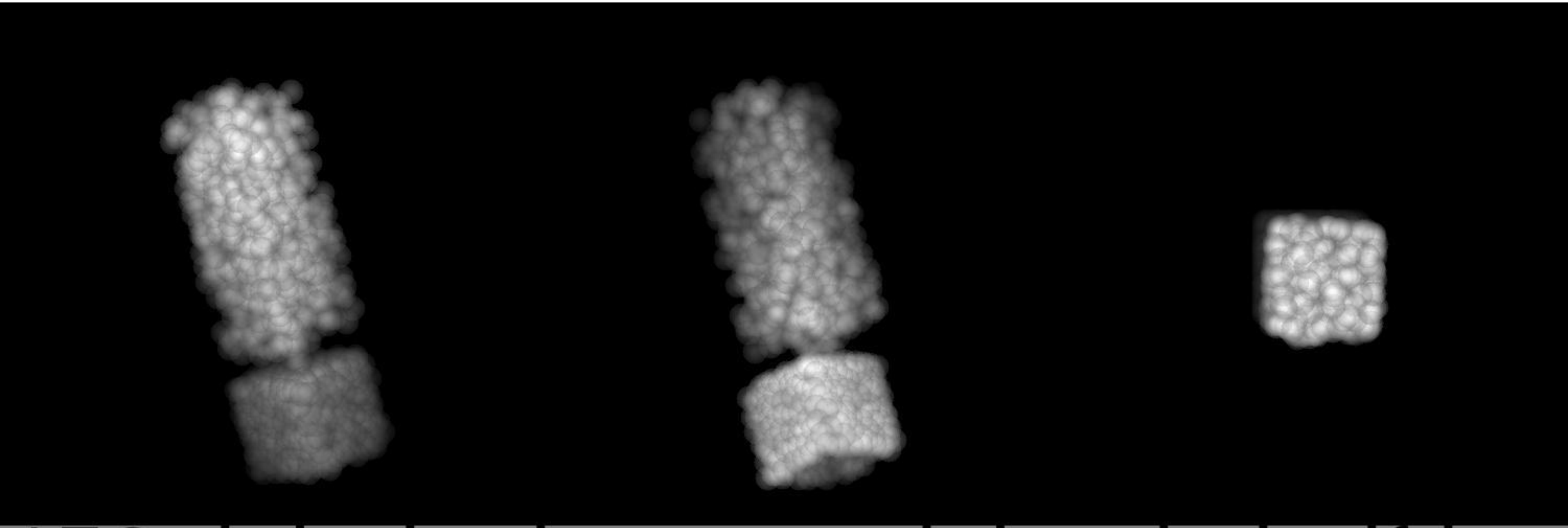


155_label_bookshelf_pred_tv_stand.jpg


156_label_radio_pred_glass_box.jpg

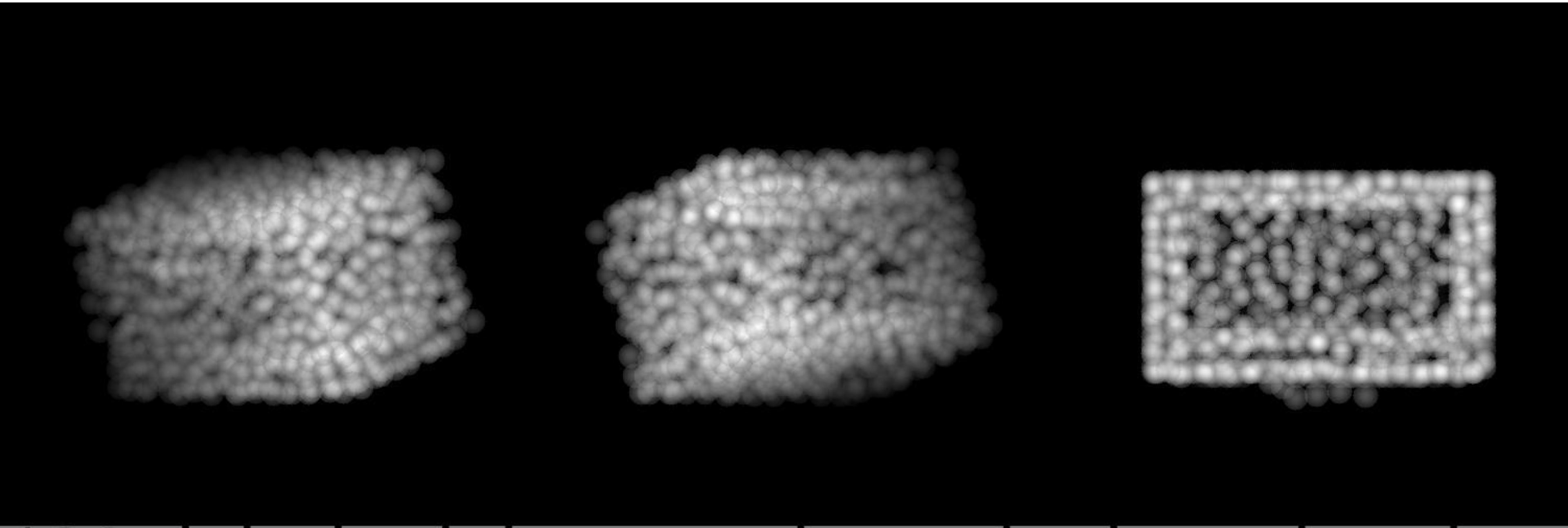


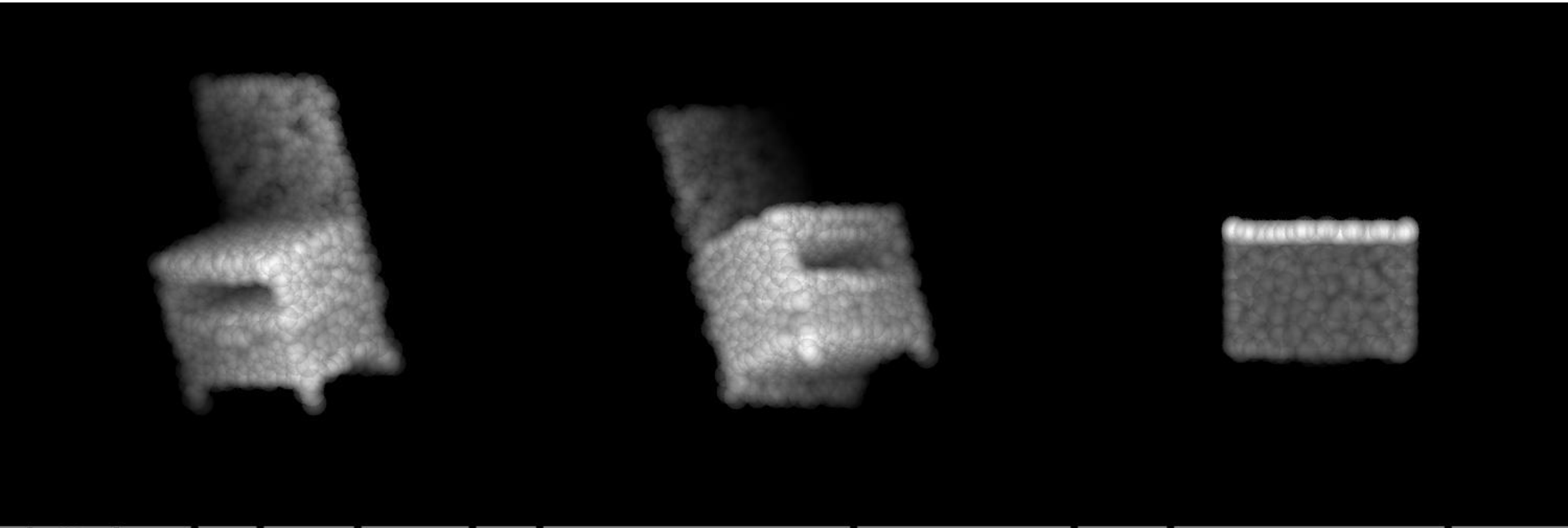
157_label_tv_stand_pred_bookshelf.jpg



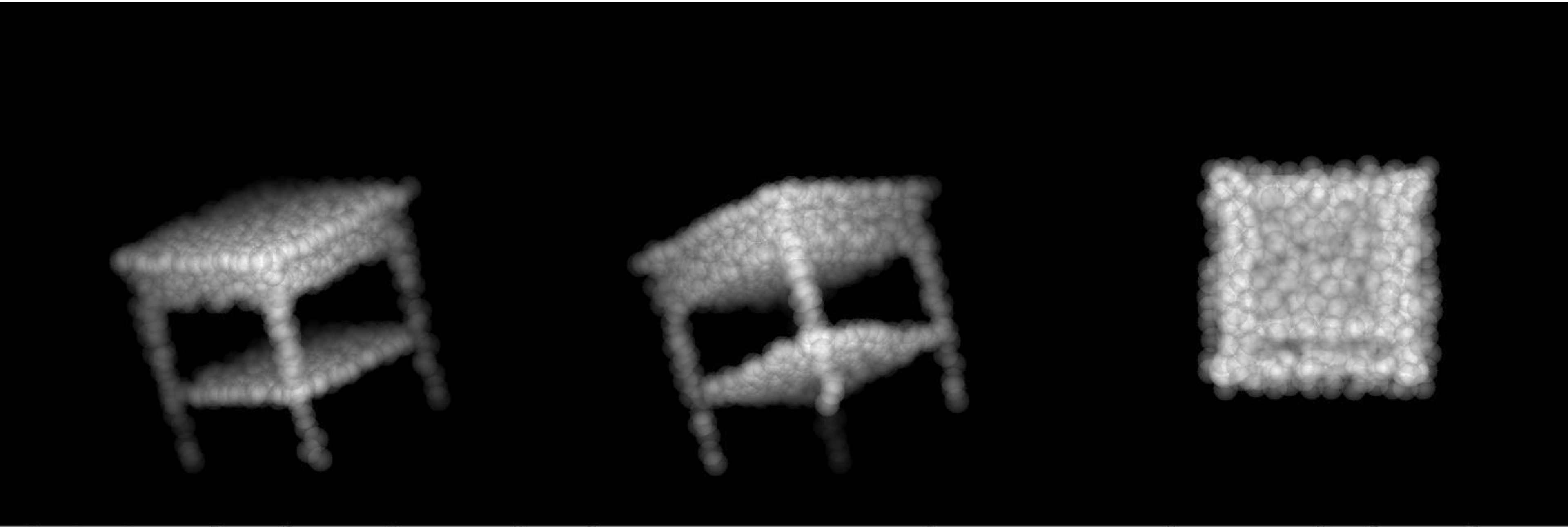
158_label_plant_pred_bookshelf.jpg

159_label_piano_pred_table.jpg

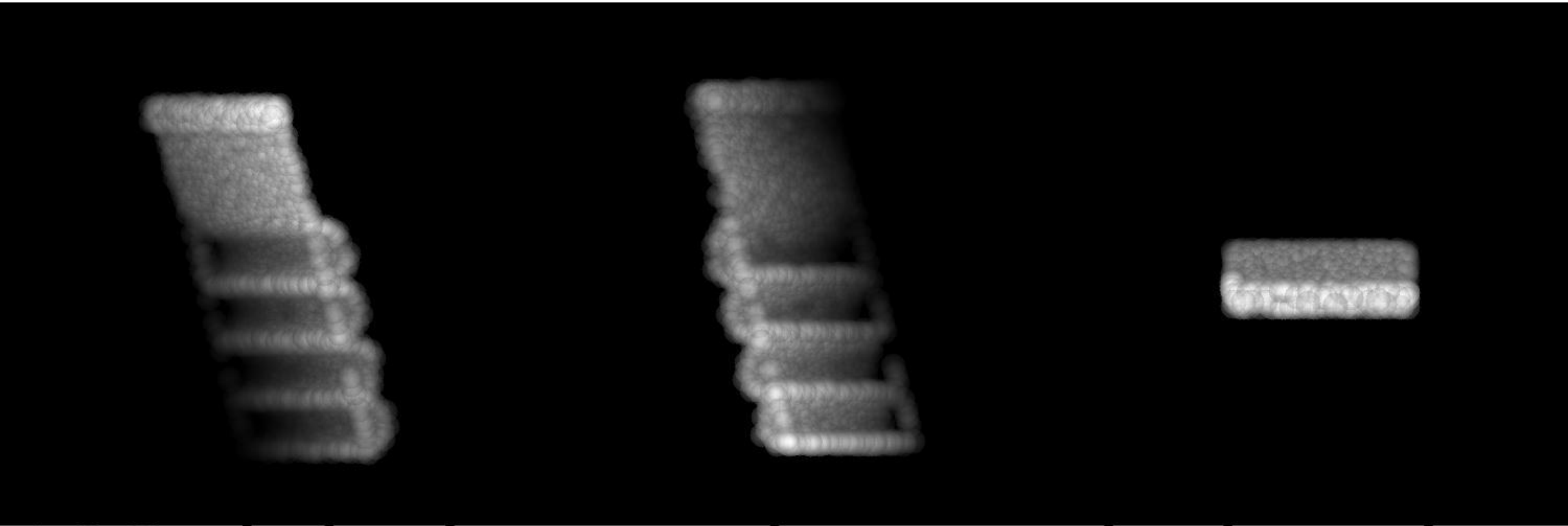

160_label_night_stand_pred_glass_box.jpg



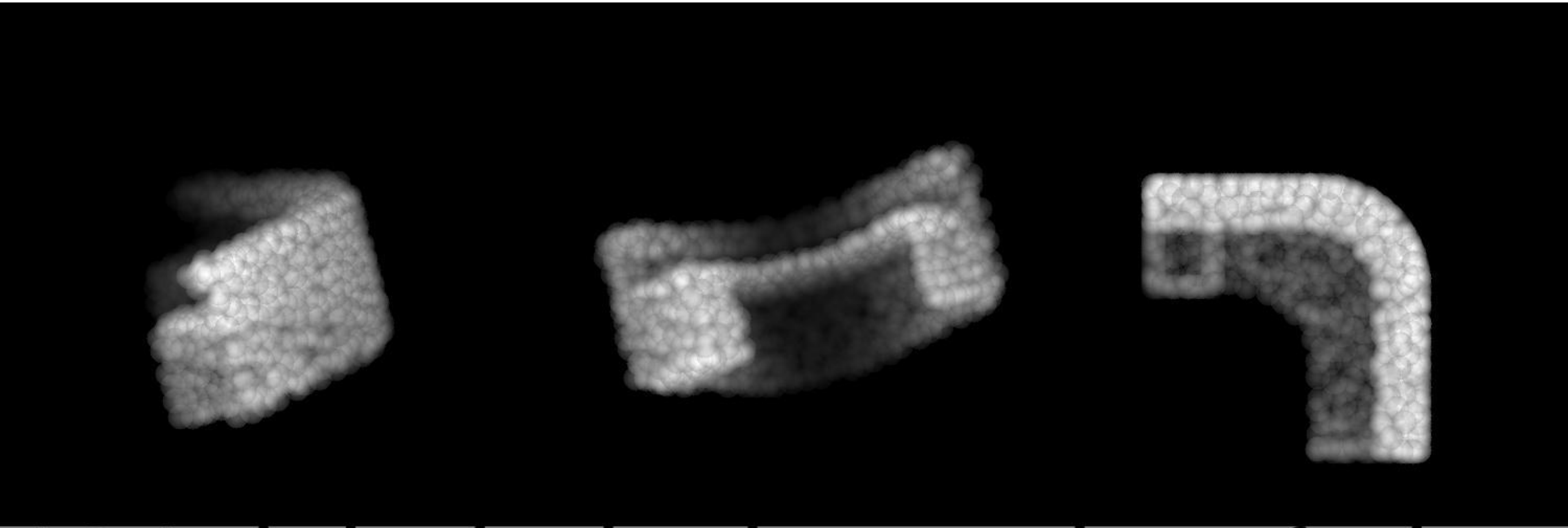
161_label_night_stand_pred_dresser.jpg



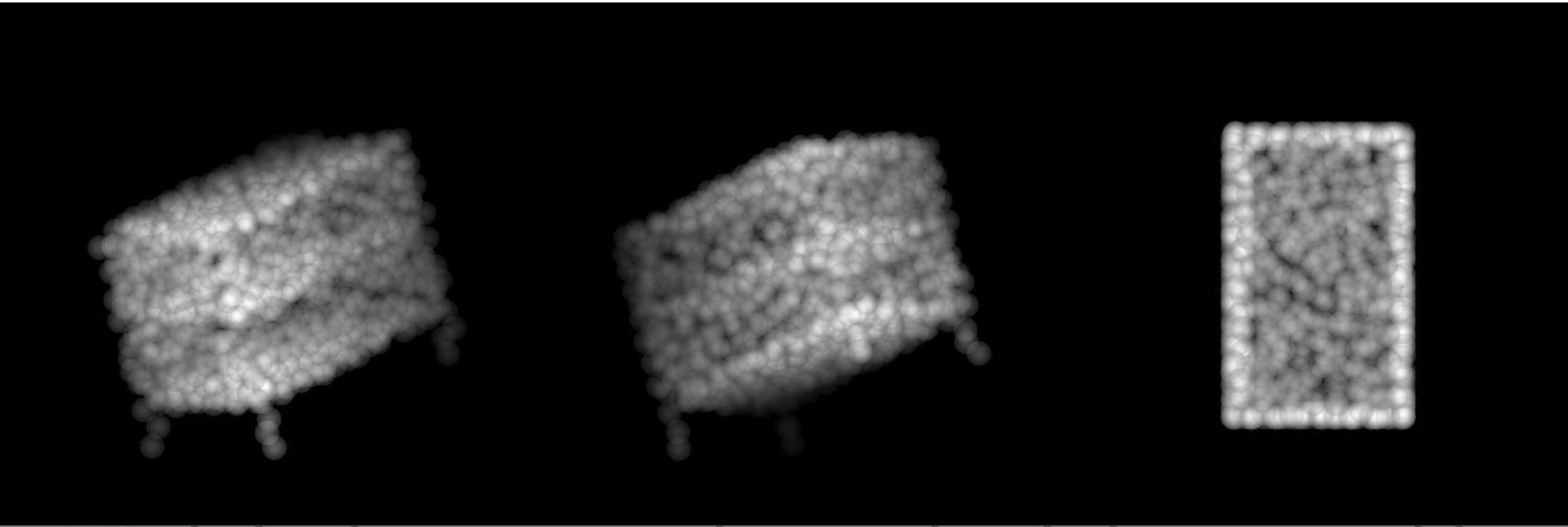
162_label_night_stand_pred_table.jpg

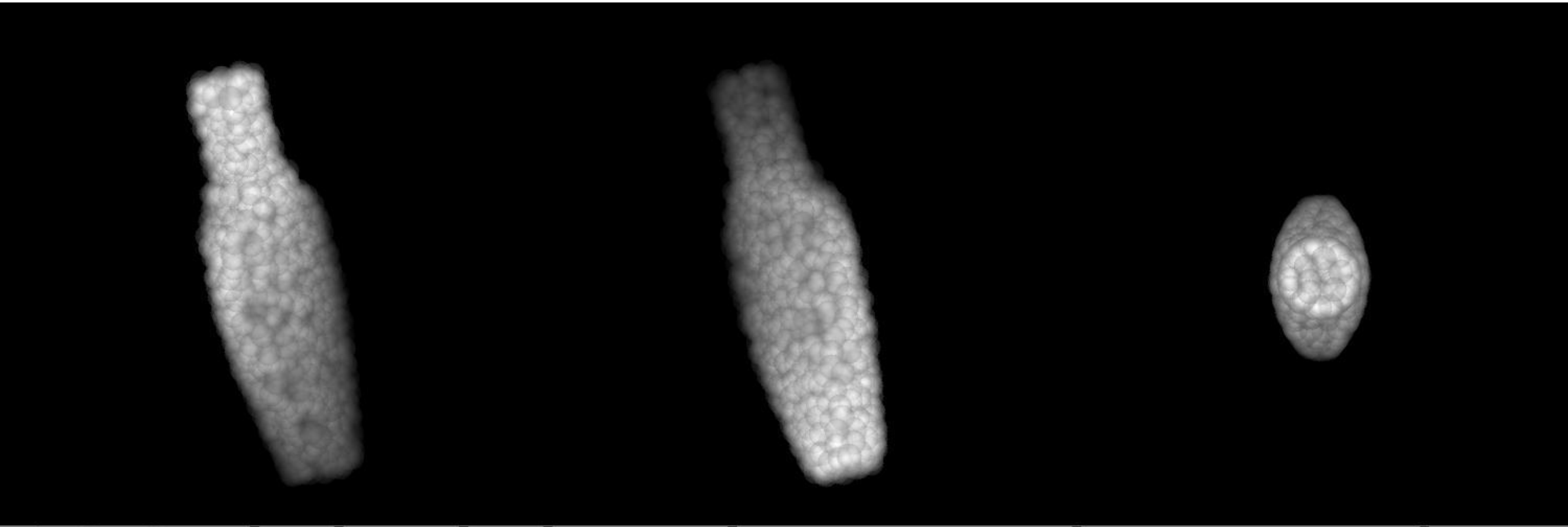


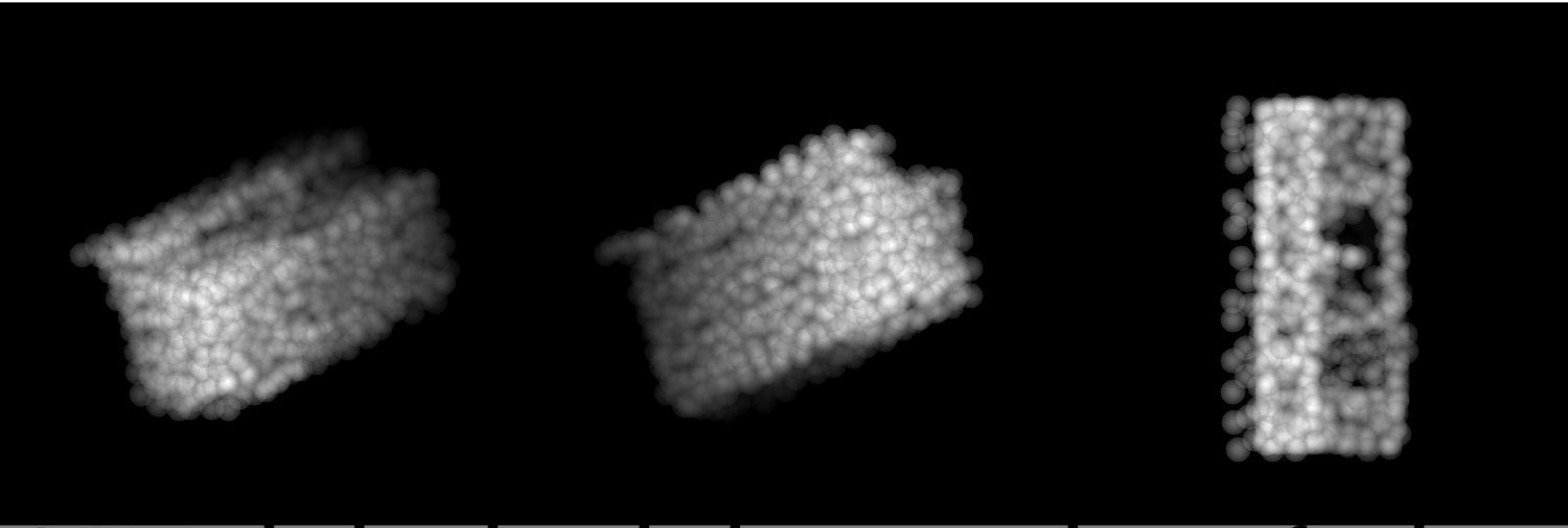
163_label_curtain_pred_door.jpg



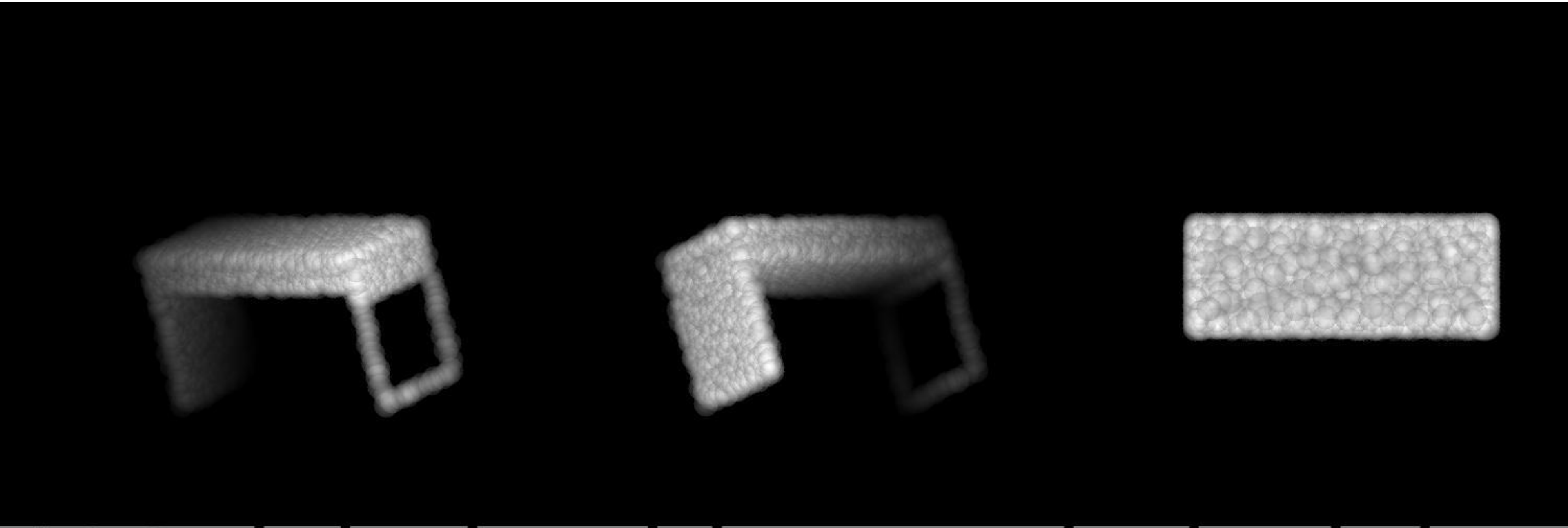
164_label_desk_pred_sofa.jpg


165_label_tv_stand_pred_night_stand.jpg

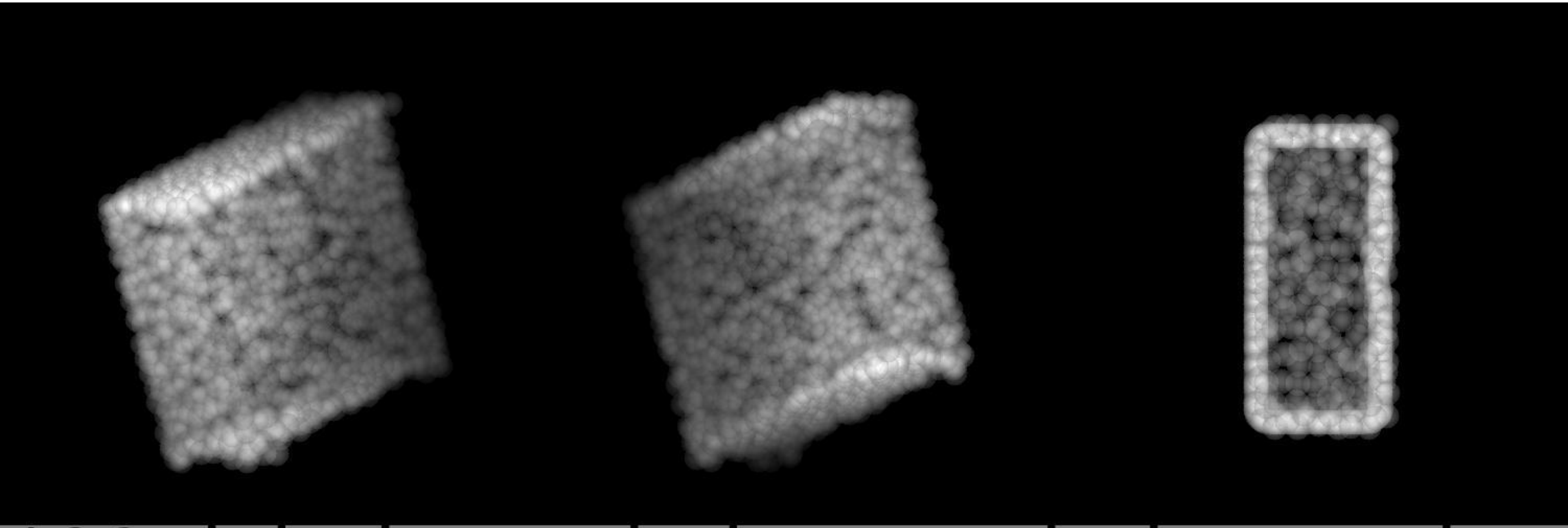

166_label_bottle_pred_vase.jpg



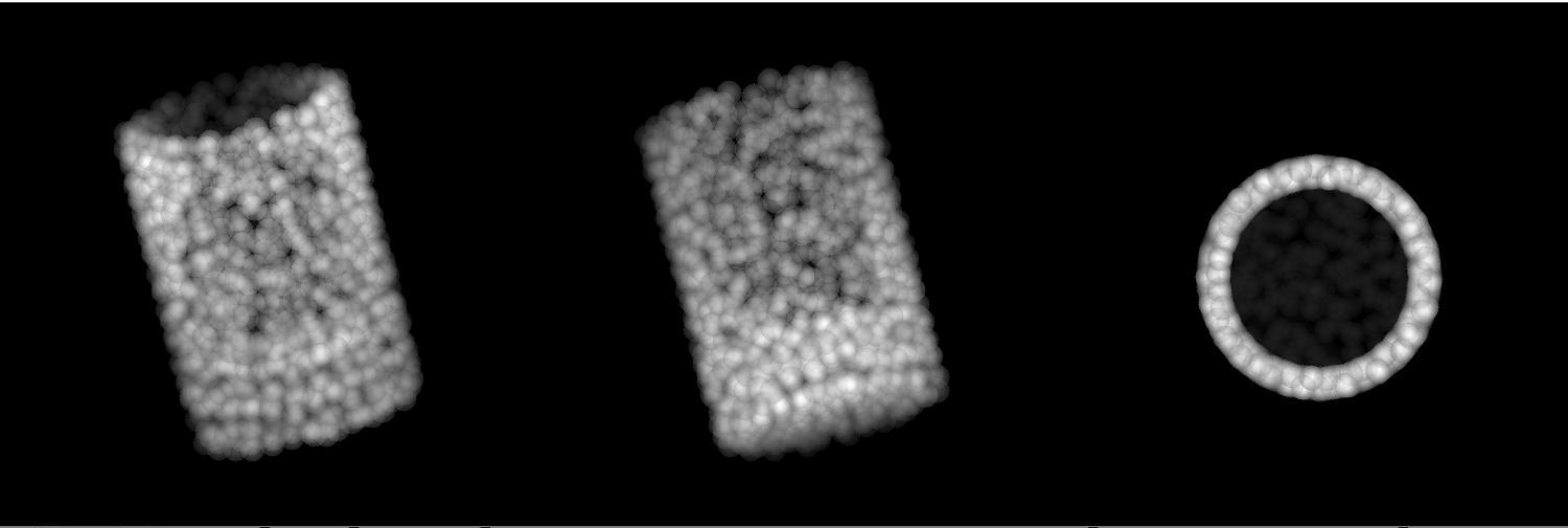
167_label_sink_pred_sofa.jpg



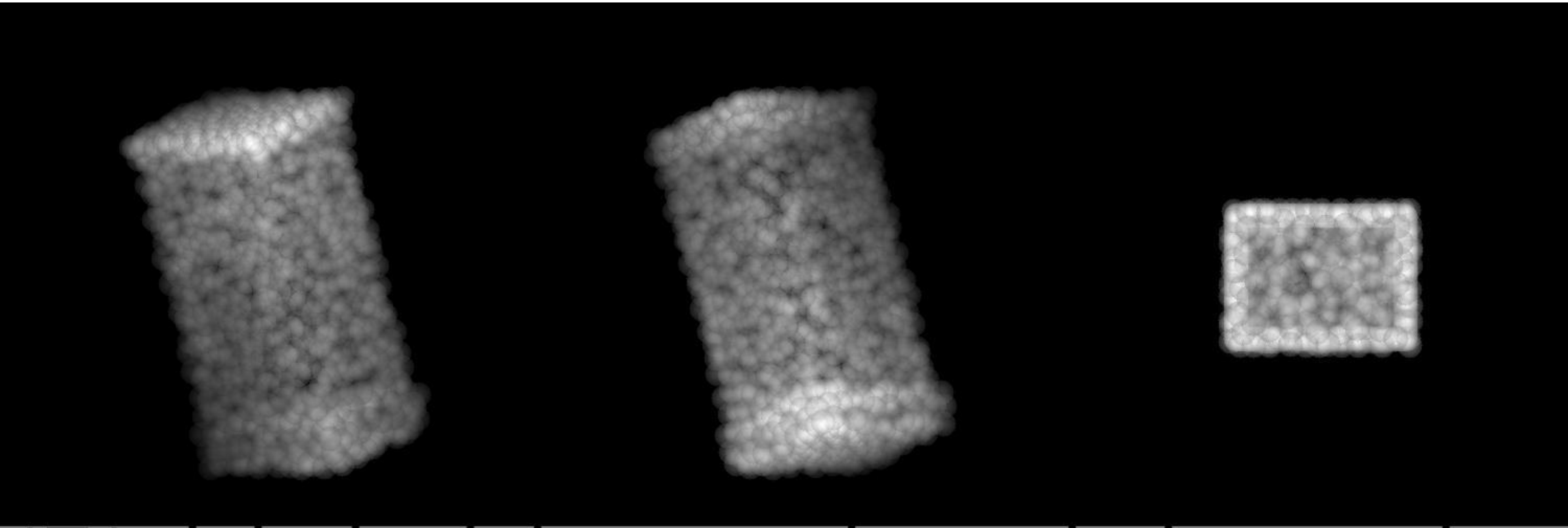
168_label_table_pred_desk.jpg



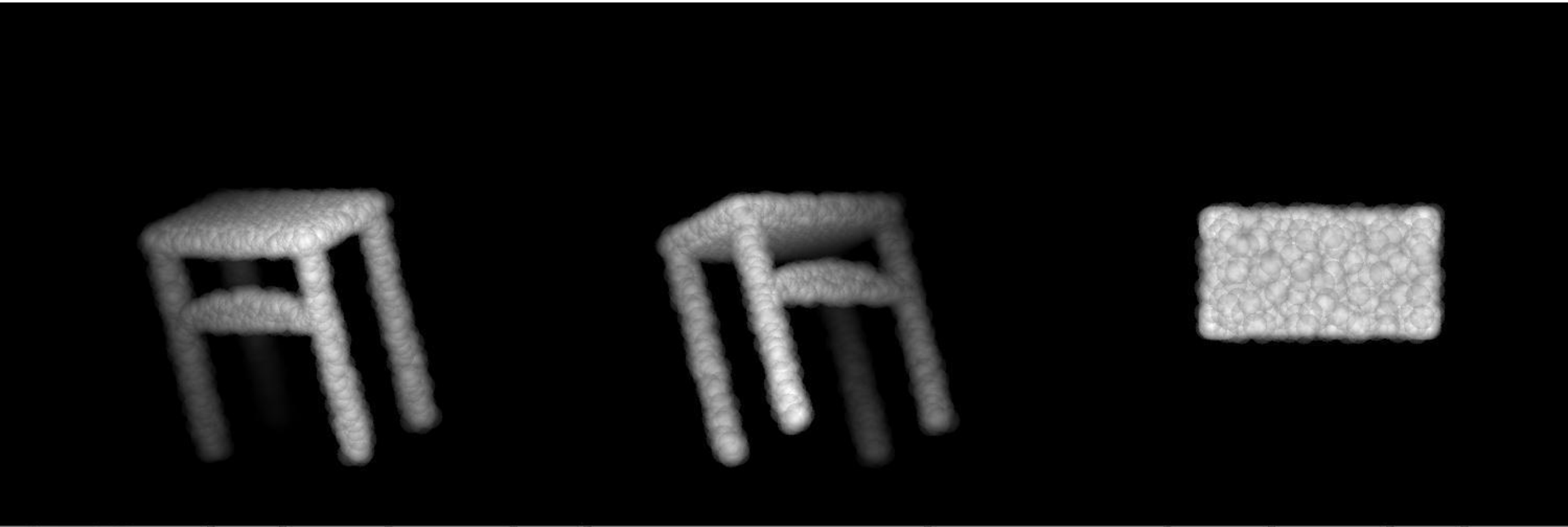
169_label_wardrobe_pred_dresser.jpg



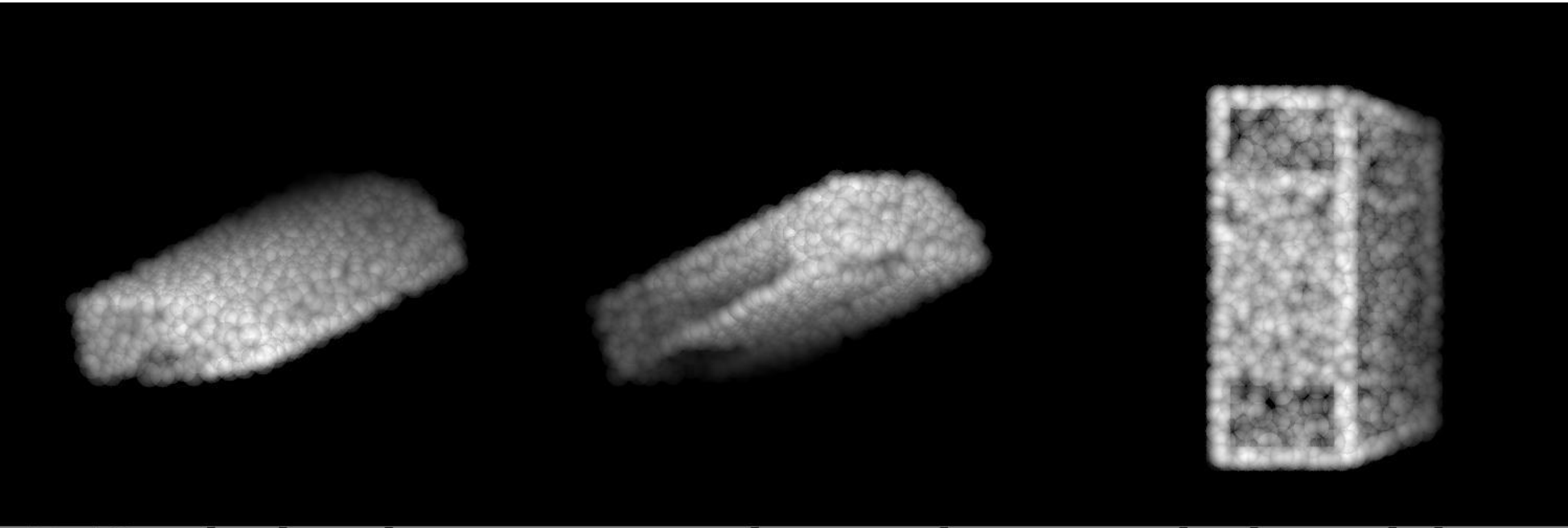
170_label_vase_pred_cup.jpg



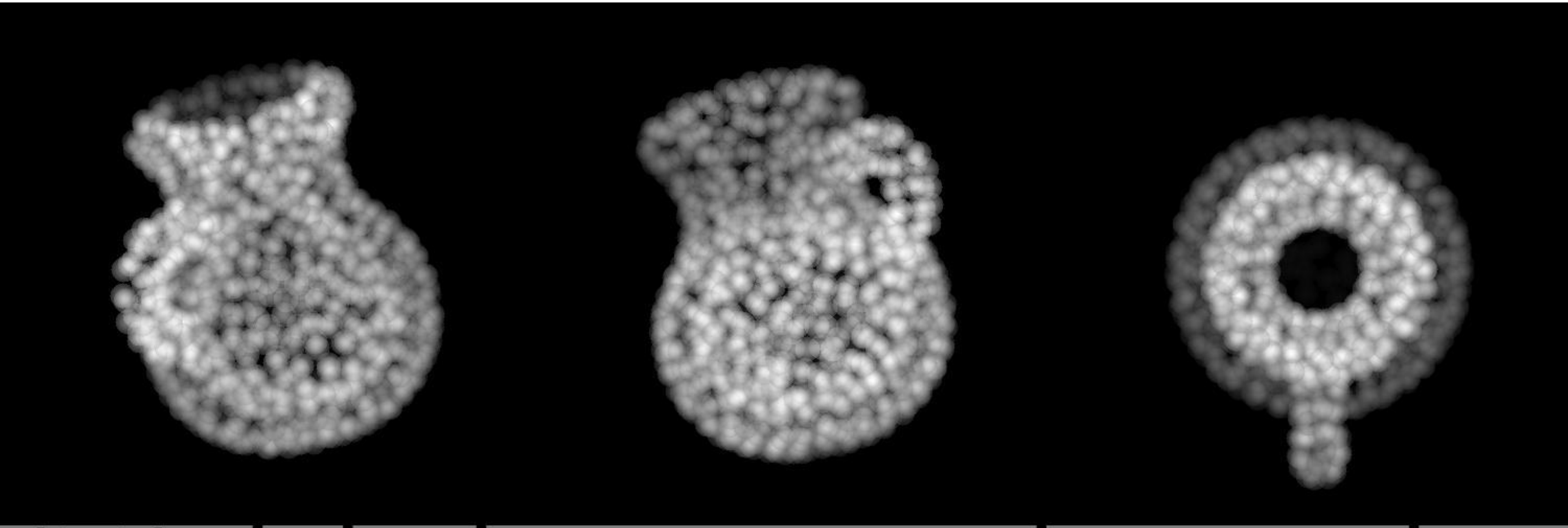
171_label_night_stand_pred_dresser.jpg



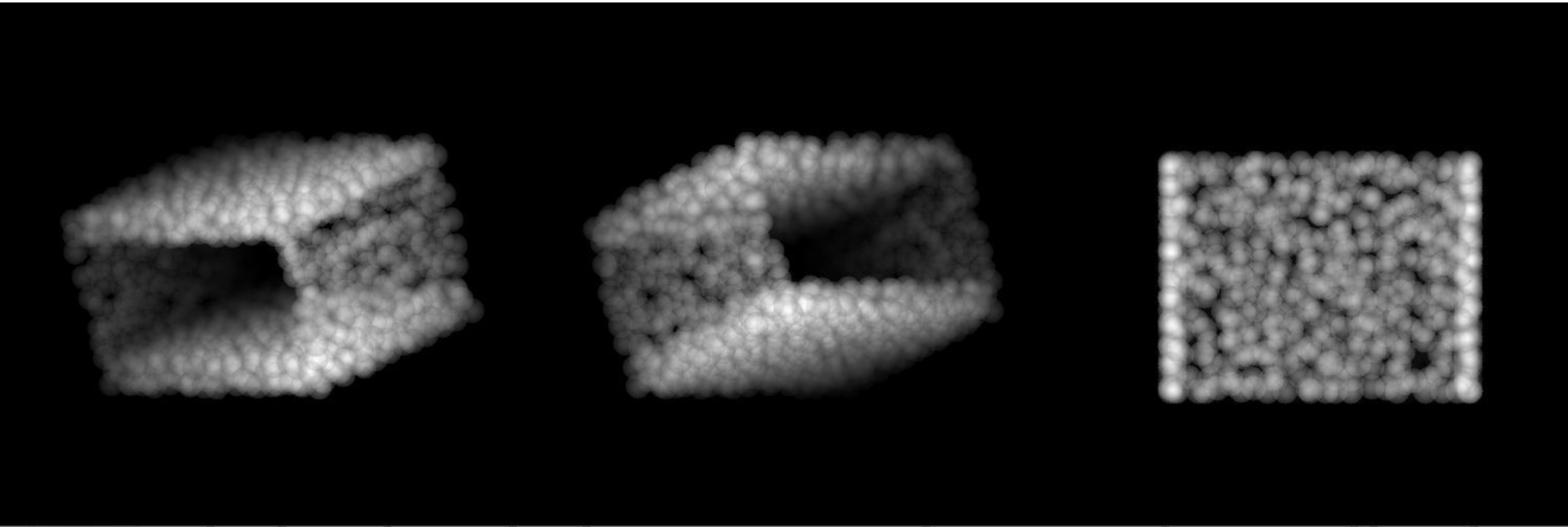
172_label_night_stand_pred_desk.jpg

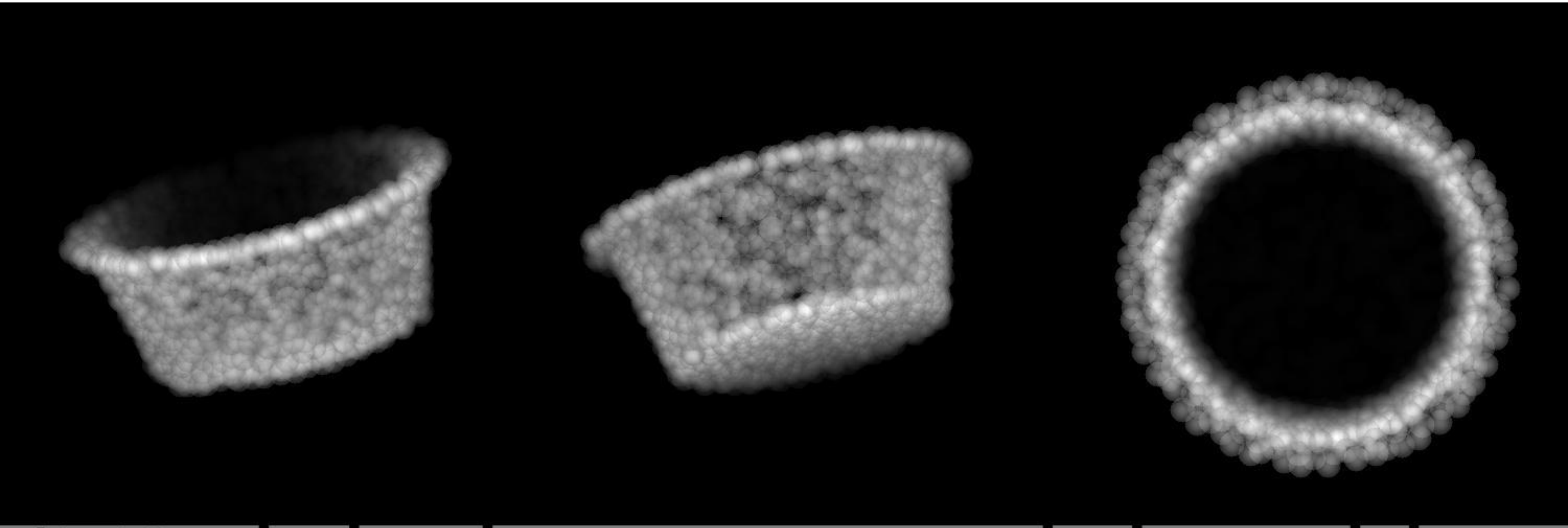


173_label_range_hood_pred_bed.jpg

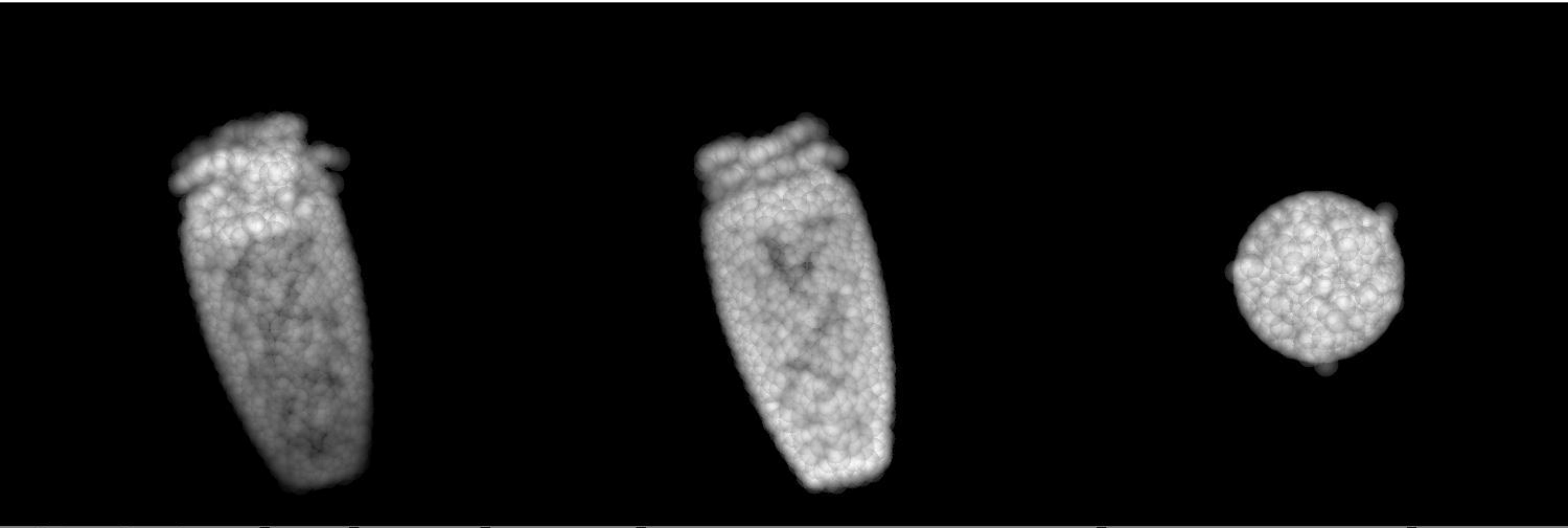


174_label_cup_pred_vase.jpg

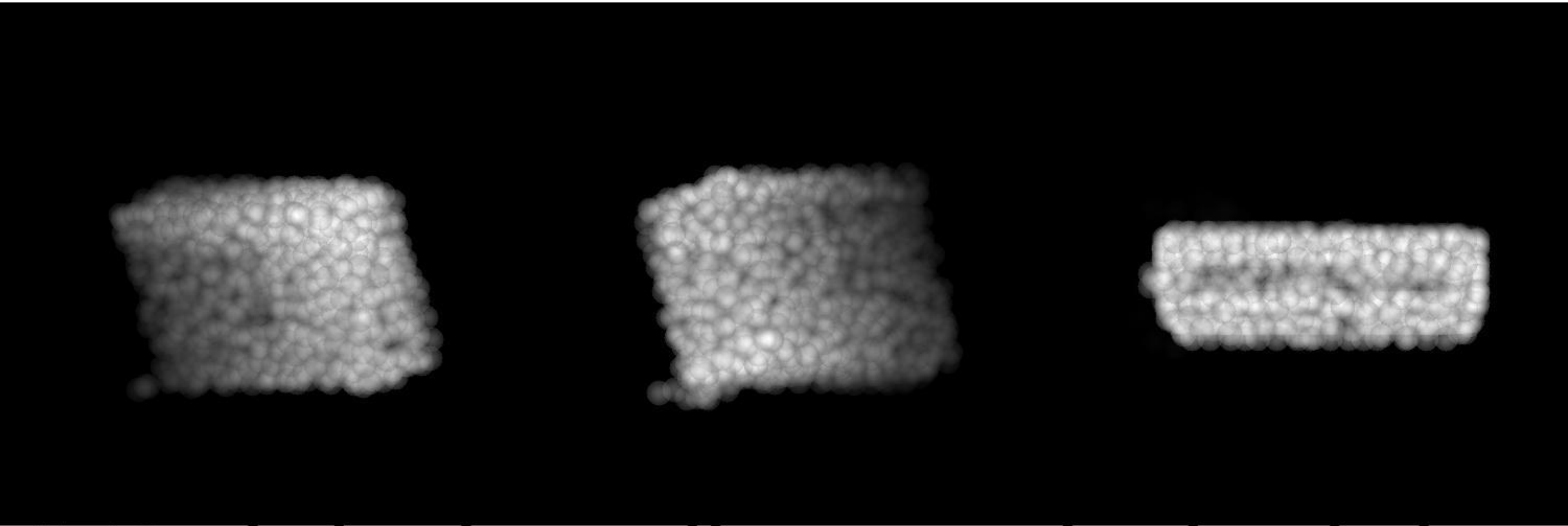

175_label_night_stand_pred_radio.jpg



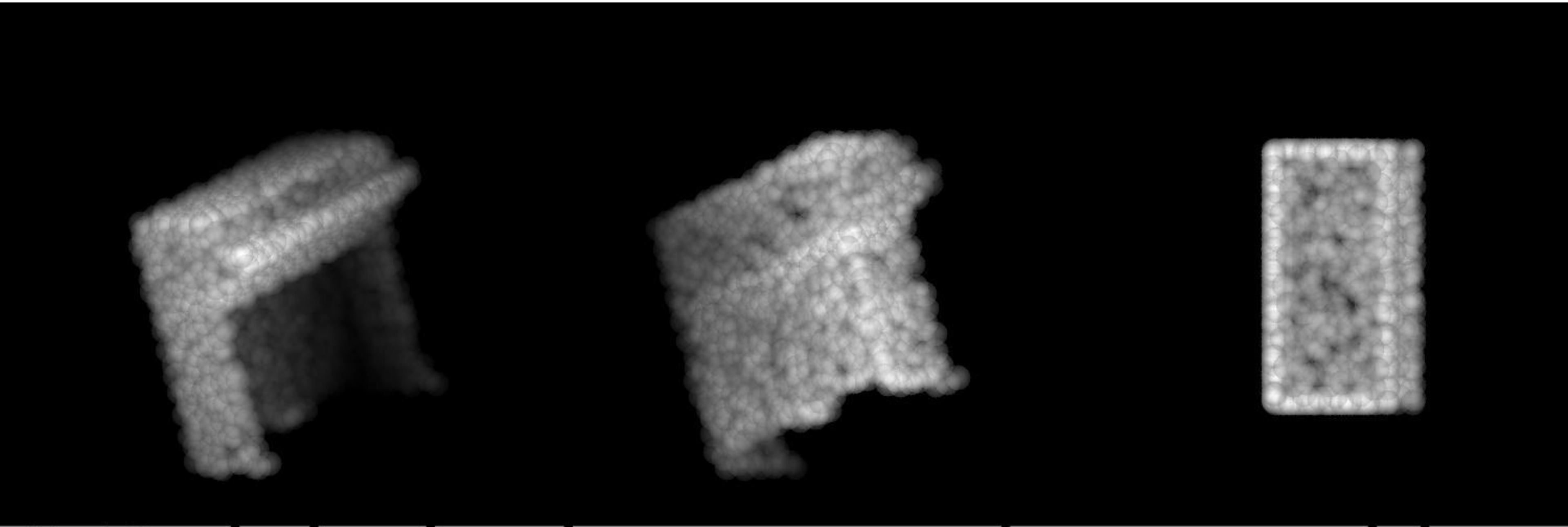
176_label_cup_pred_bowl.jpg



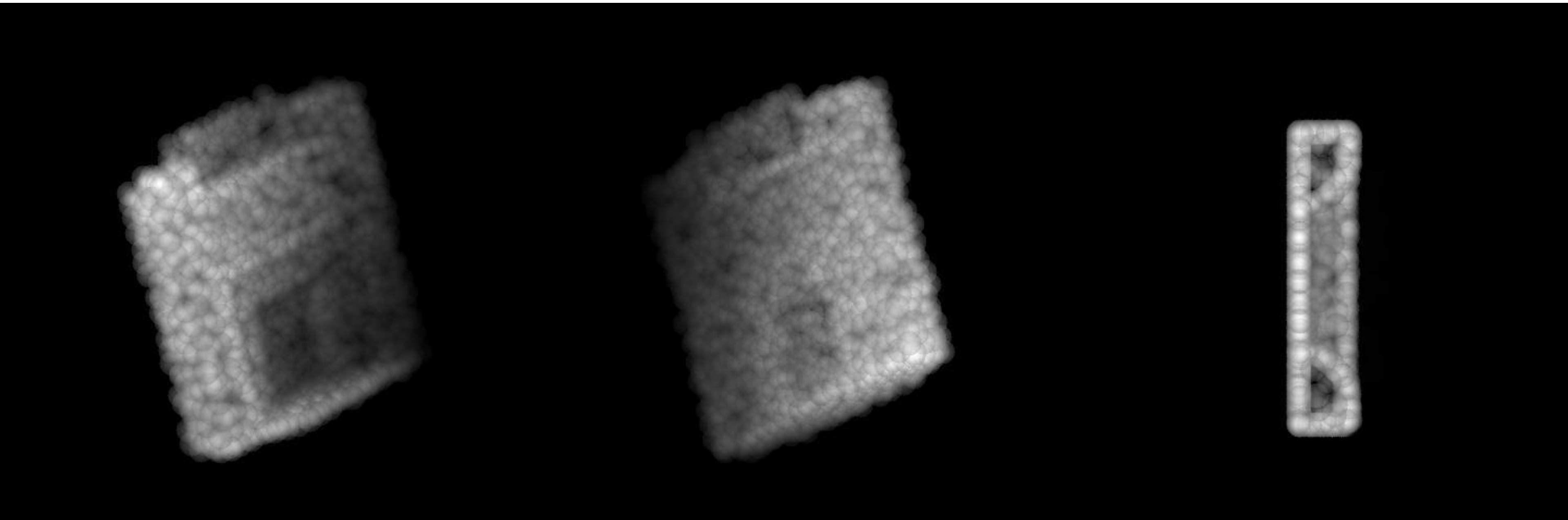
177_label_plant_pred_cup.jpg



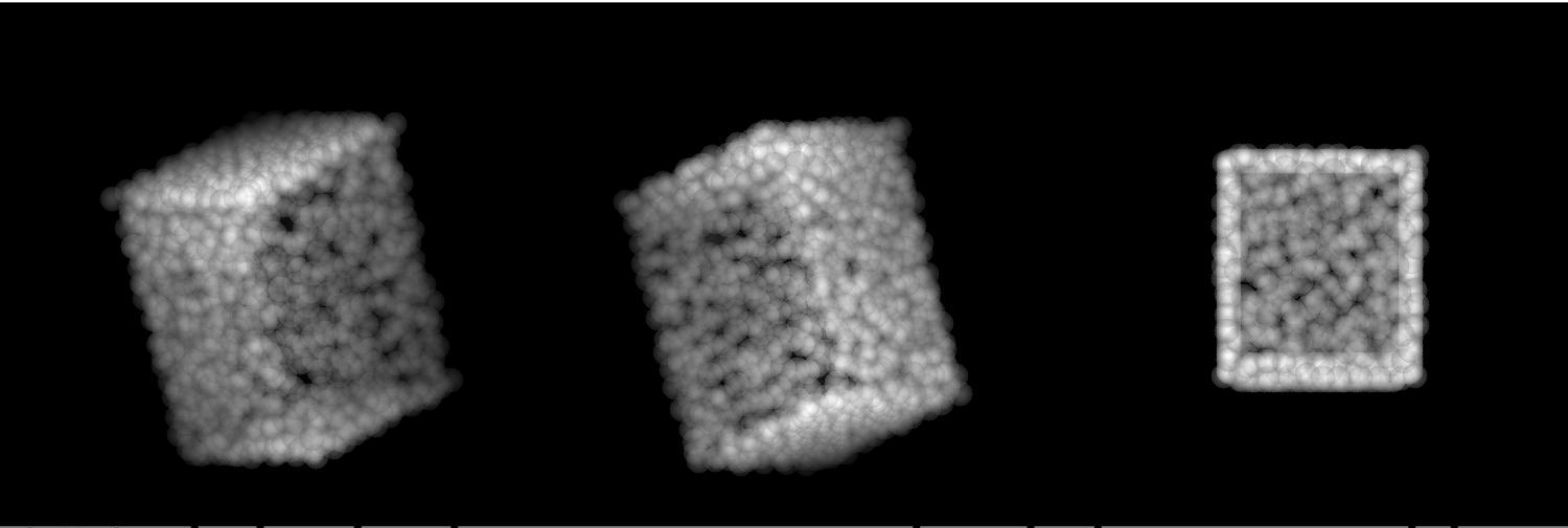
178_label_radio_pred_desk.jpg



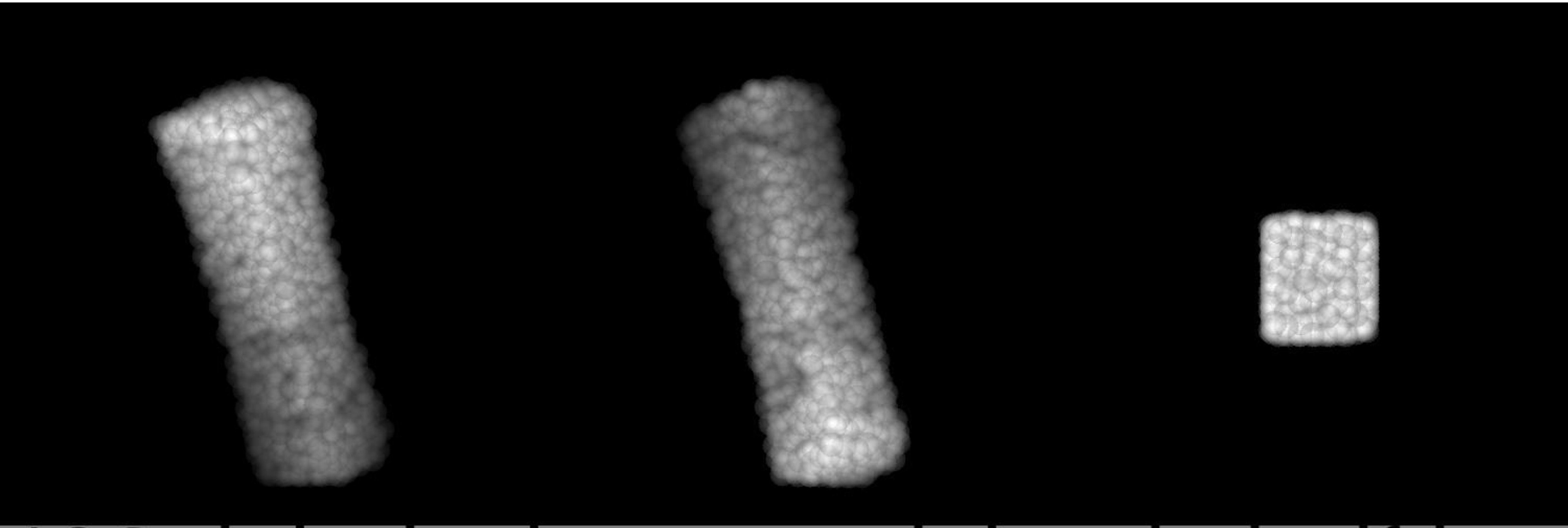
179_label_piano_pred_mantel.jpg



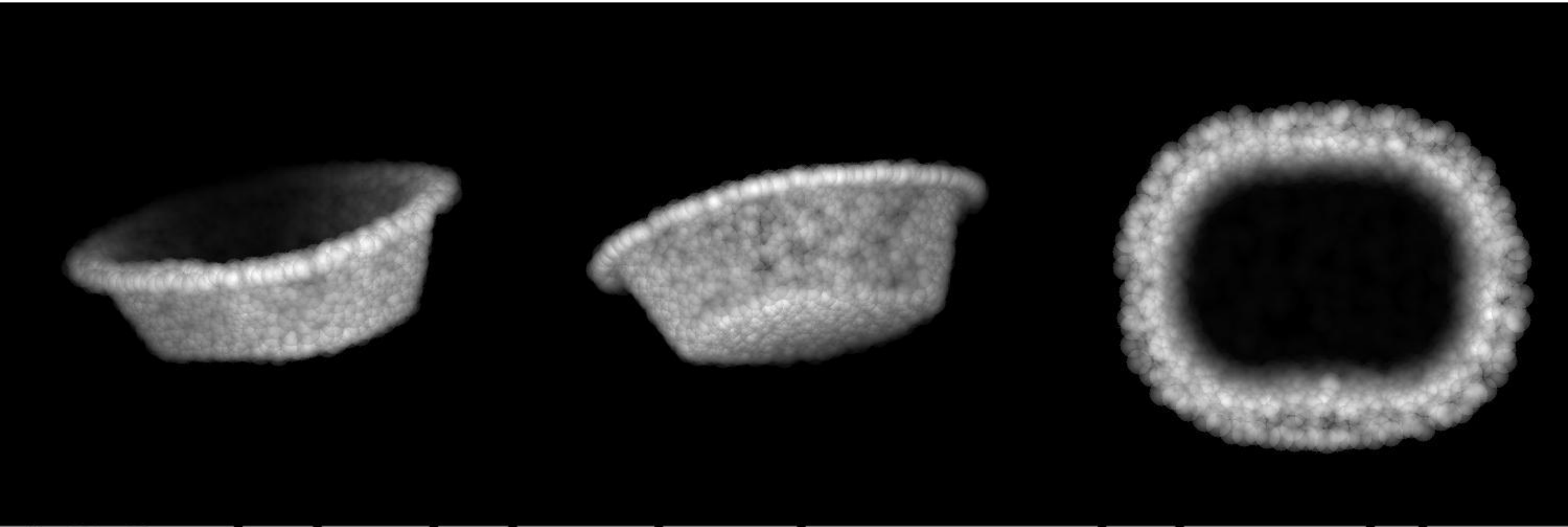
180_label_mantel_pred_bookshelf.jpg



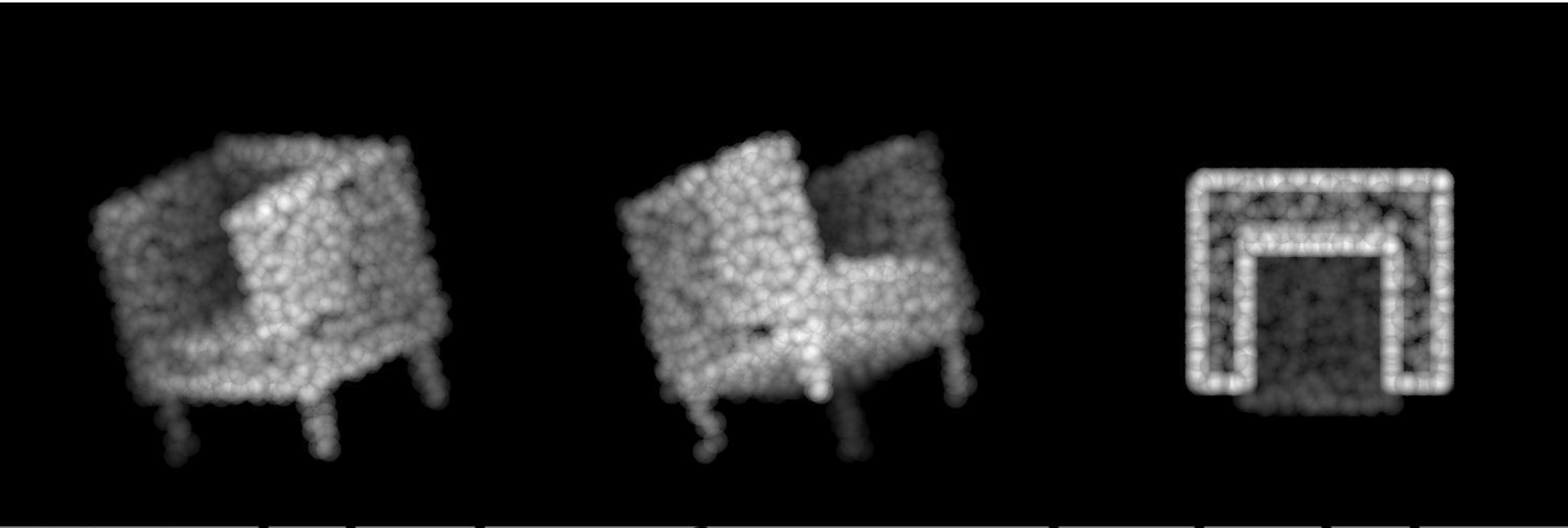
181_label_dresser_pred_night_stand.jpg



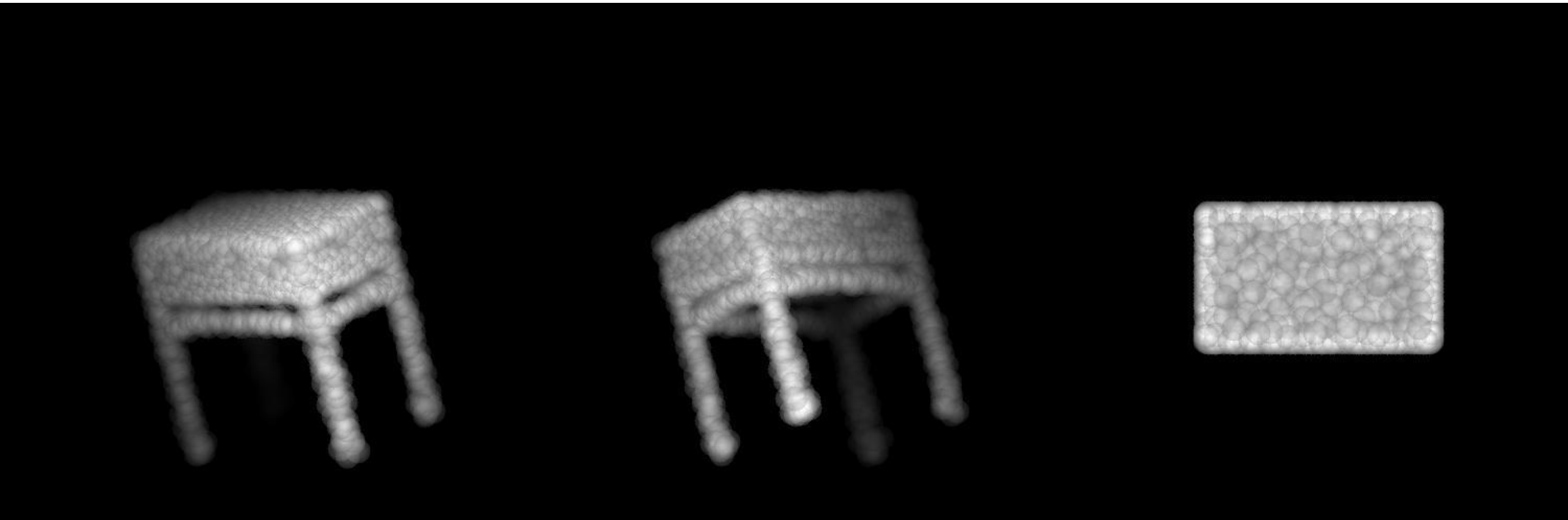
182_label_xbox_pred_bookshelf.jpg



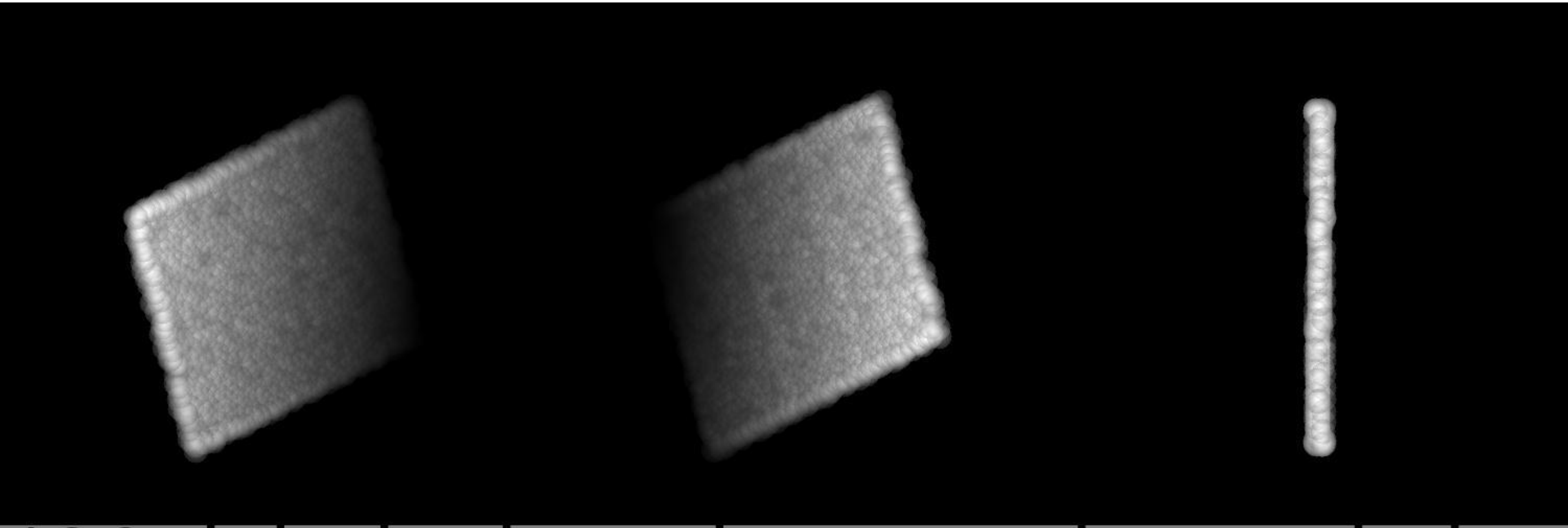
183_label_bathtub_pred_bowl.jpg



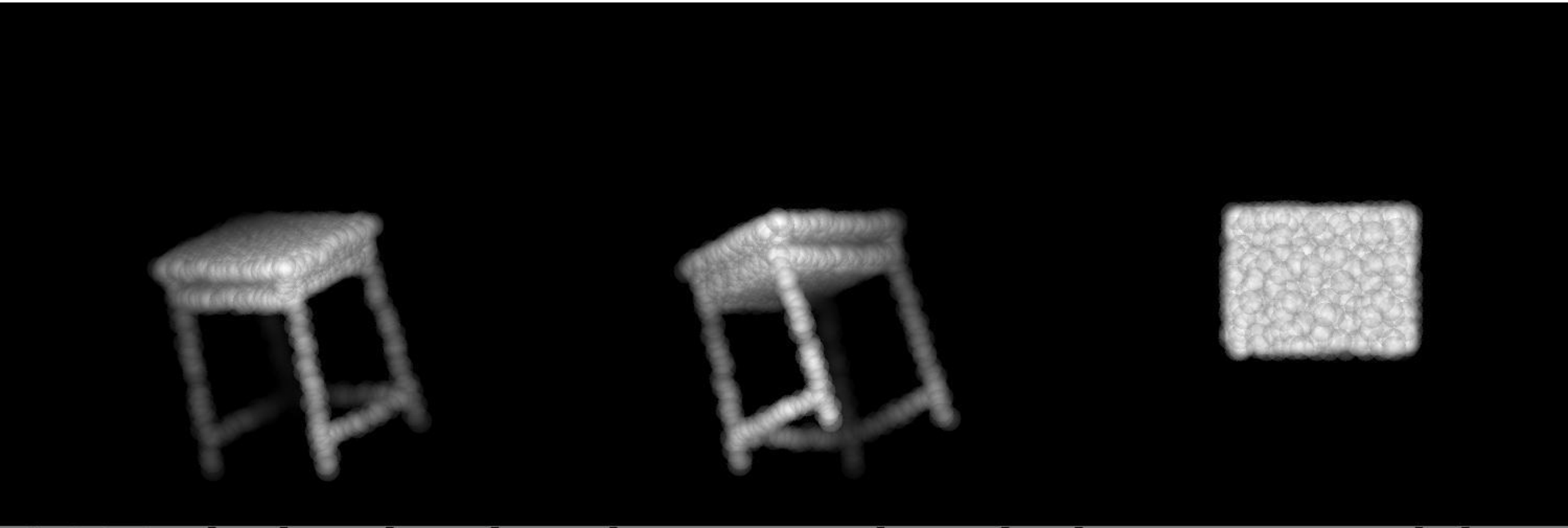
184_label_sofa_pred_chair.jpg



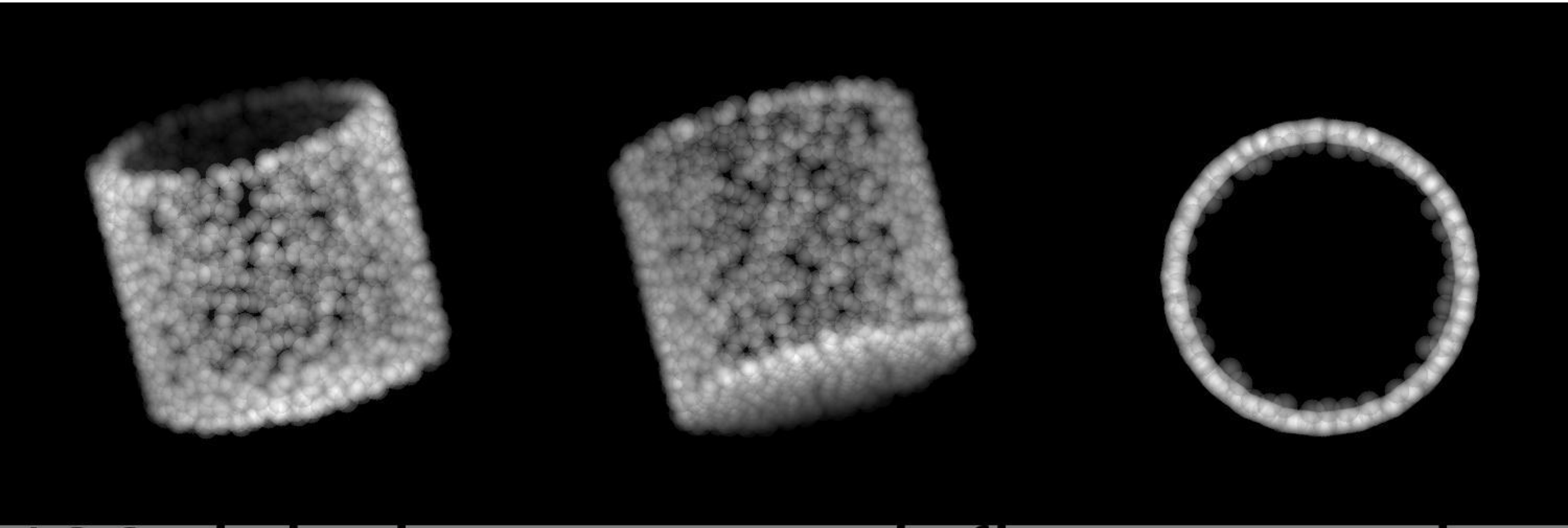
185_label_night_stand_pred_table.jpg



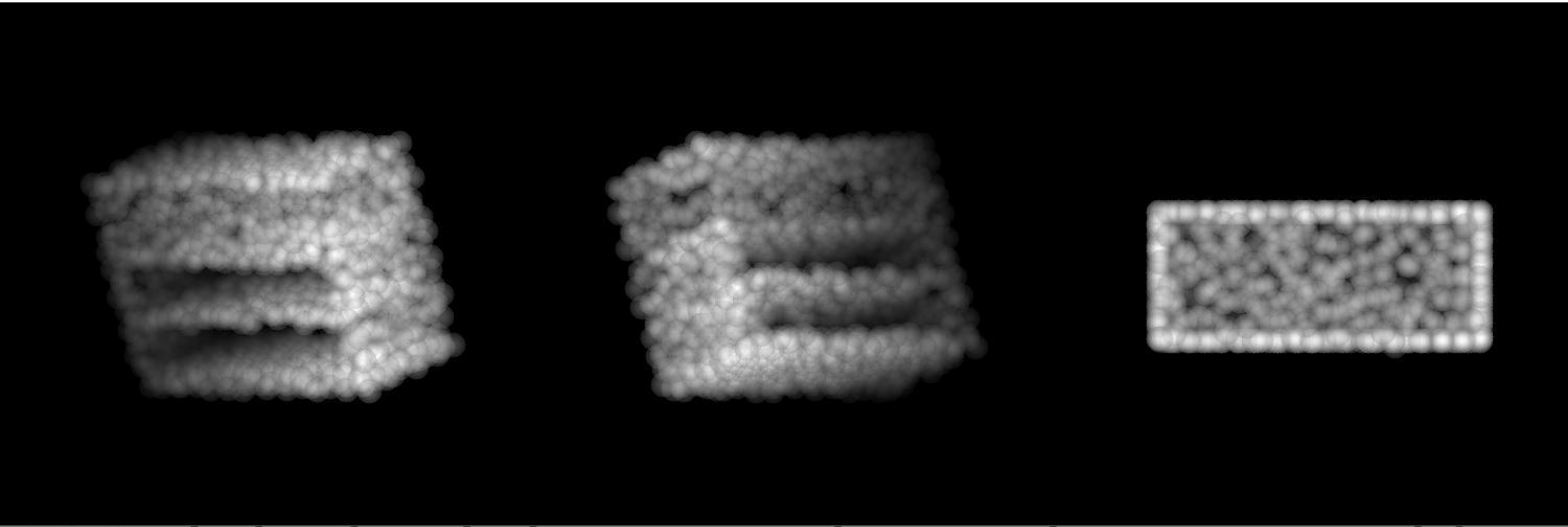
186_label_glass_box_pred_curtain.jpg



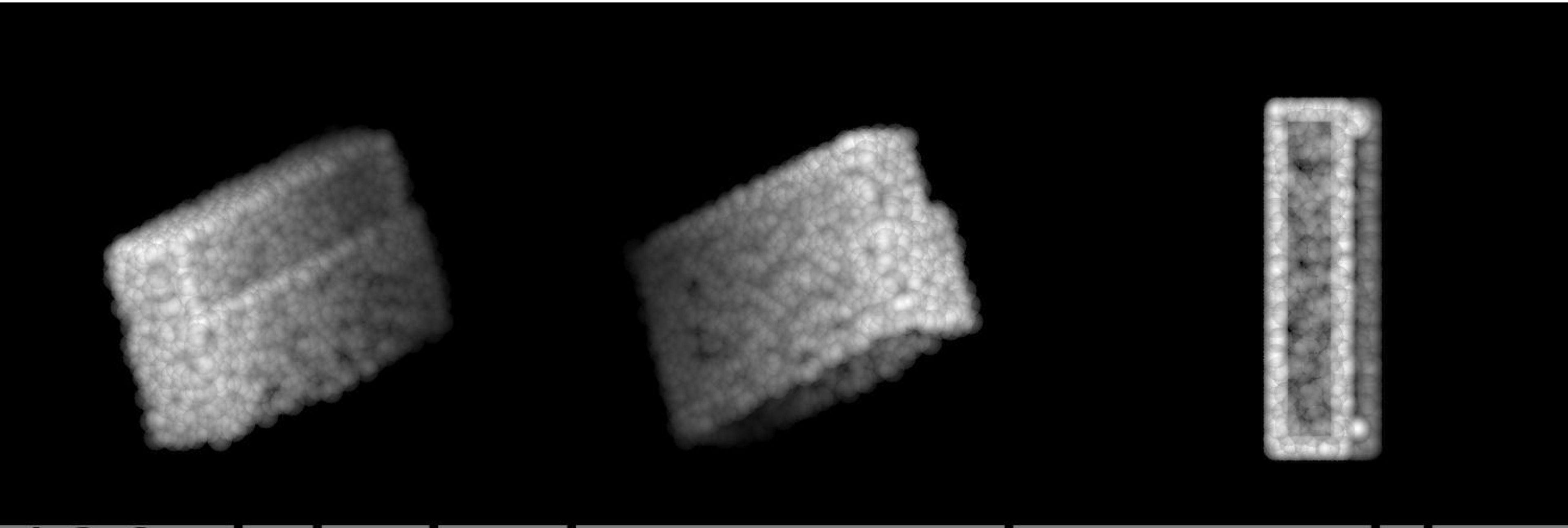
187_label_desk_pred_night_stand.jpg



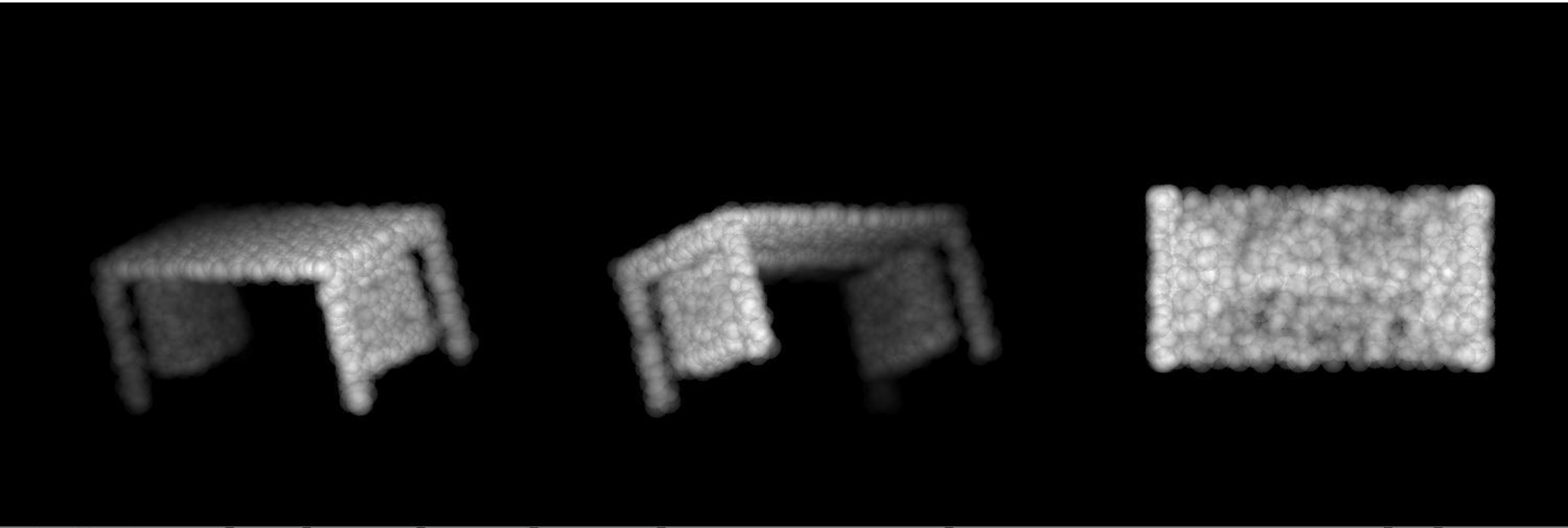
188_label_vase_pred_flower_pot.jpg



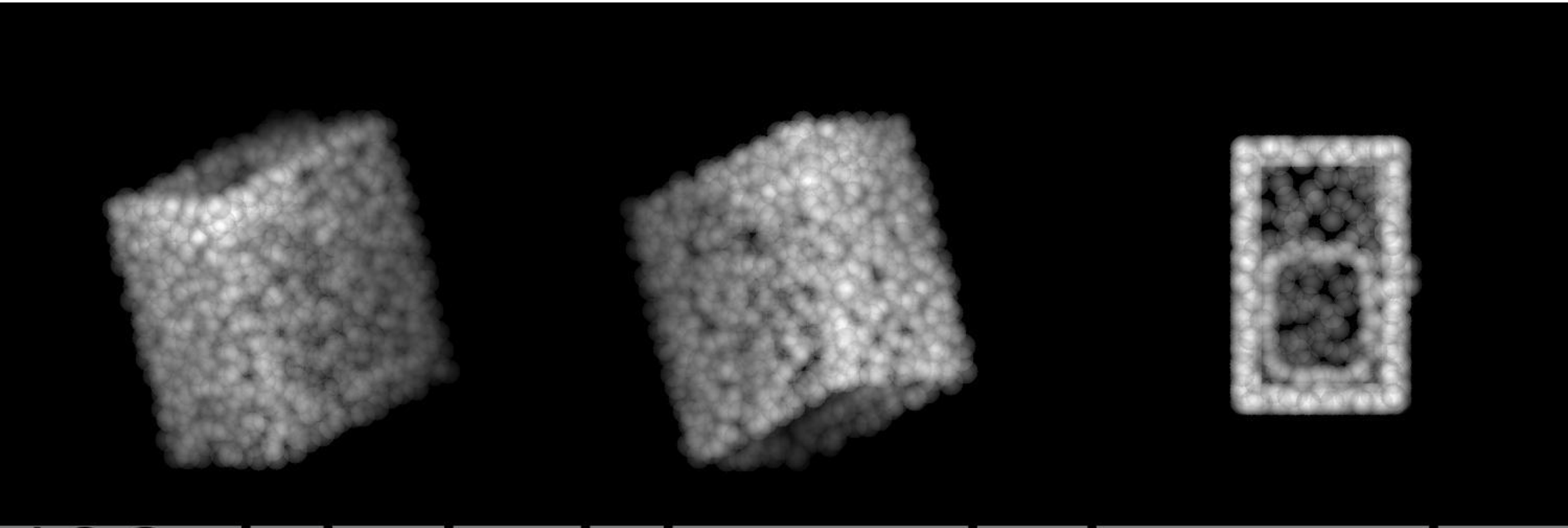
189_label_night_stand_pred_tv_stand.jpg



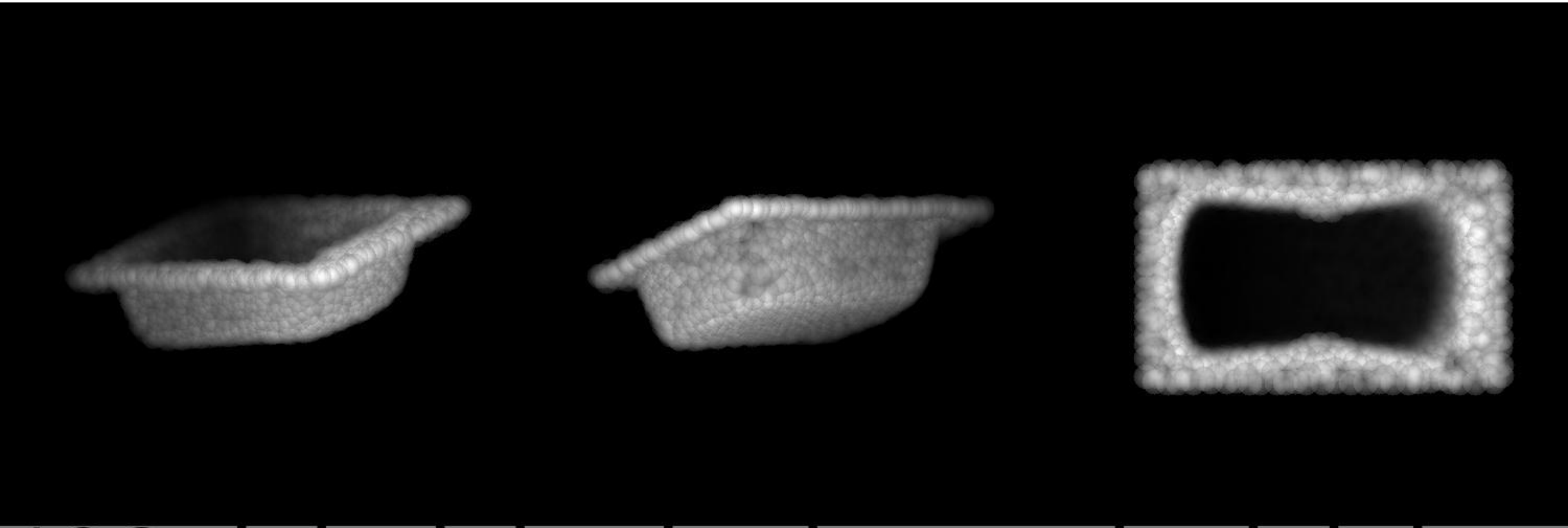
190_label_piano_pred_mantel.jpg



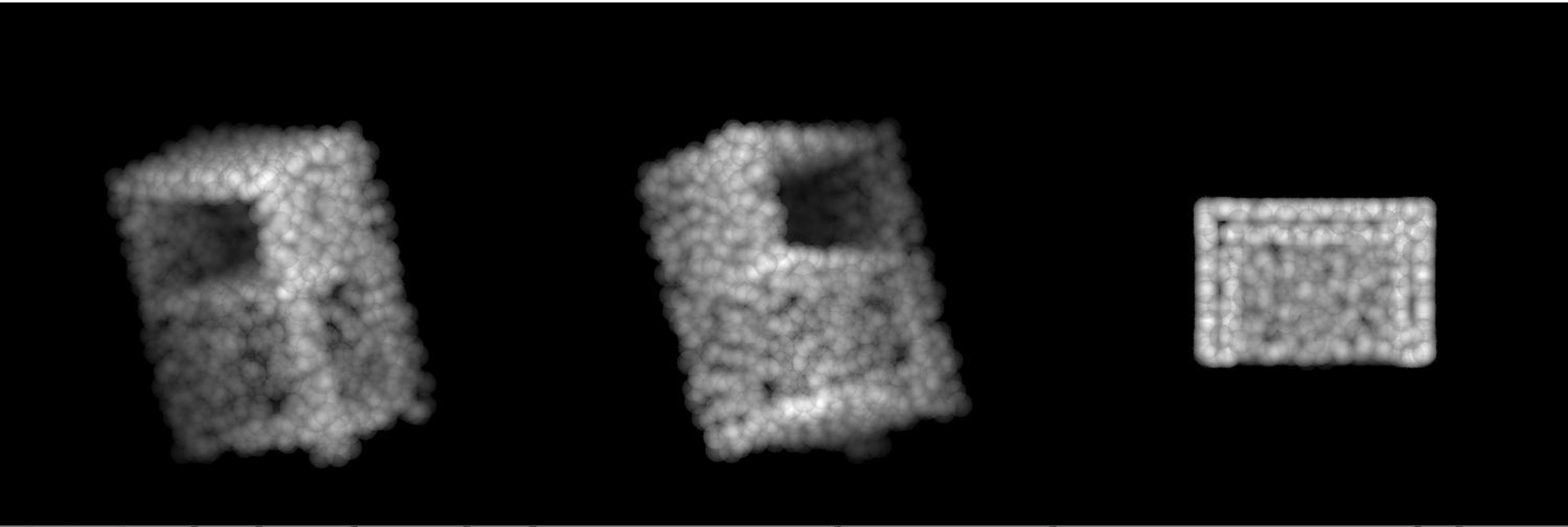
191_label_desk_pred_tv_stand.jpg



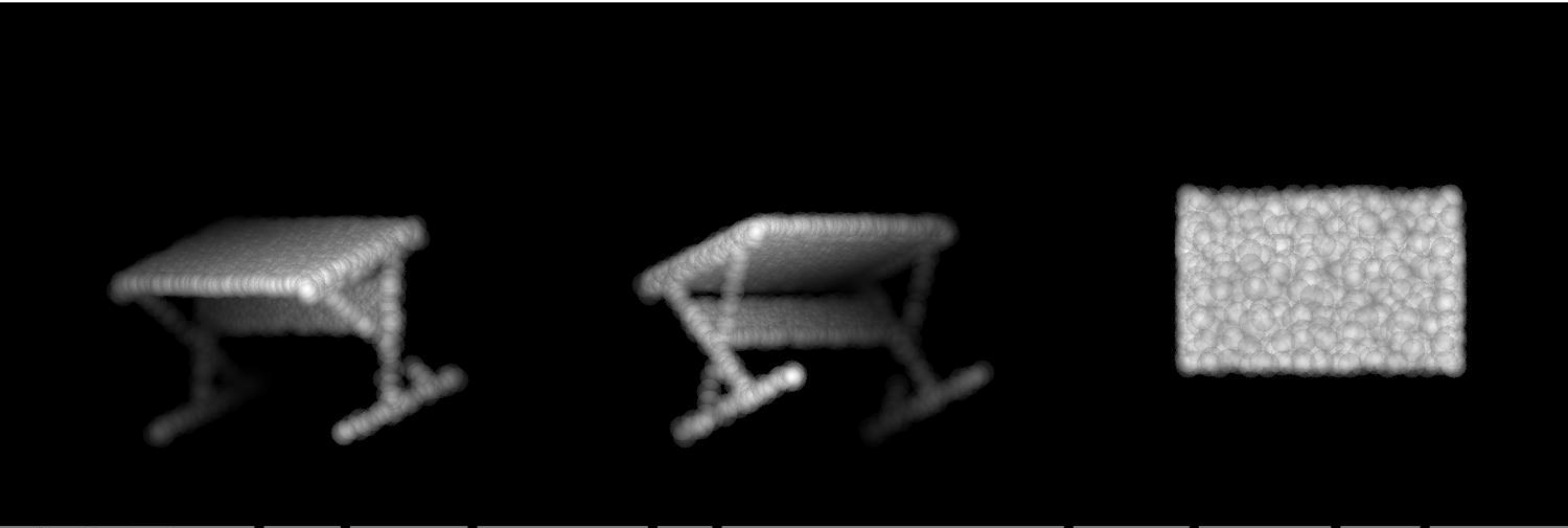
192_label_sink_pred_dresser.jpg



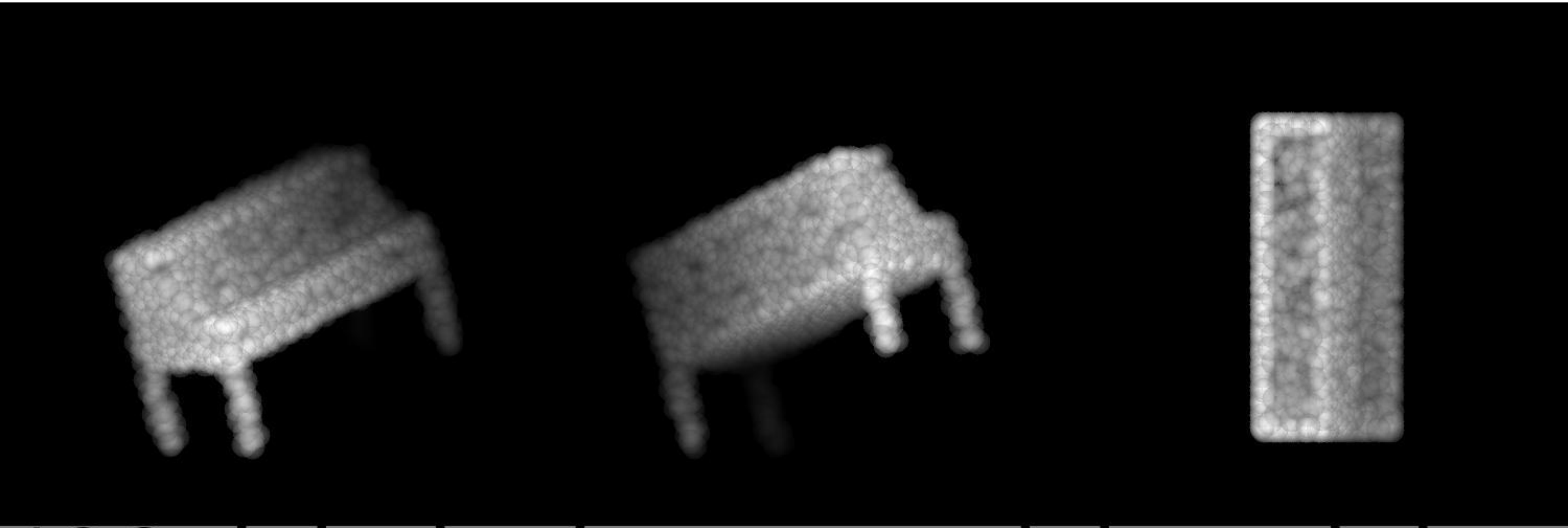
193_label_bathtub_pred_sink.jpg



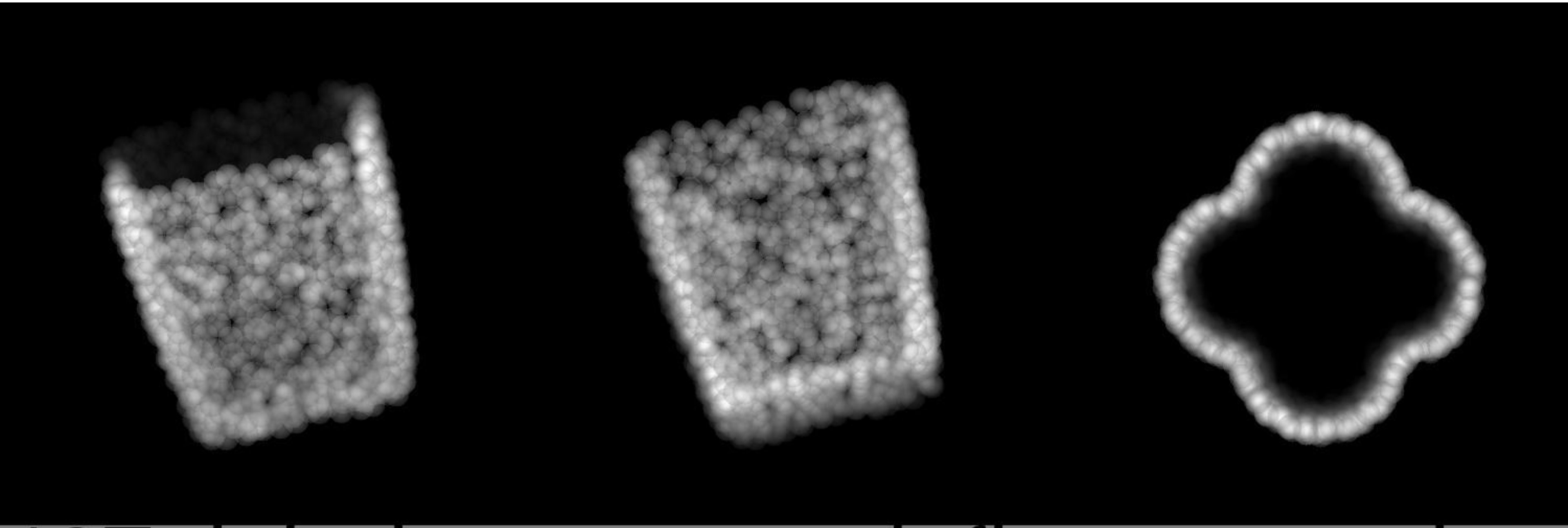
194_label_night_stand_pred_tv_stand.jpg



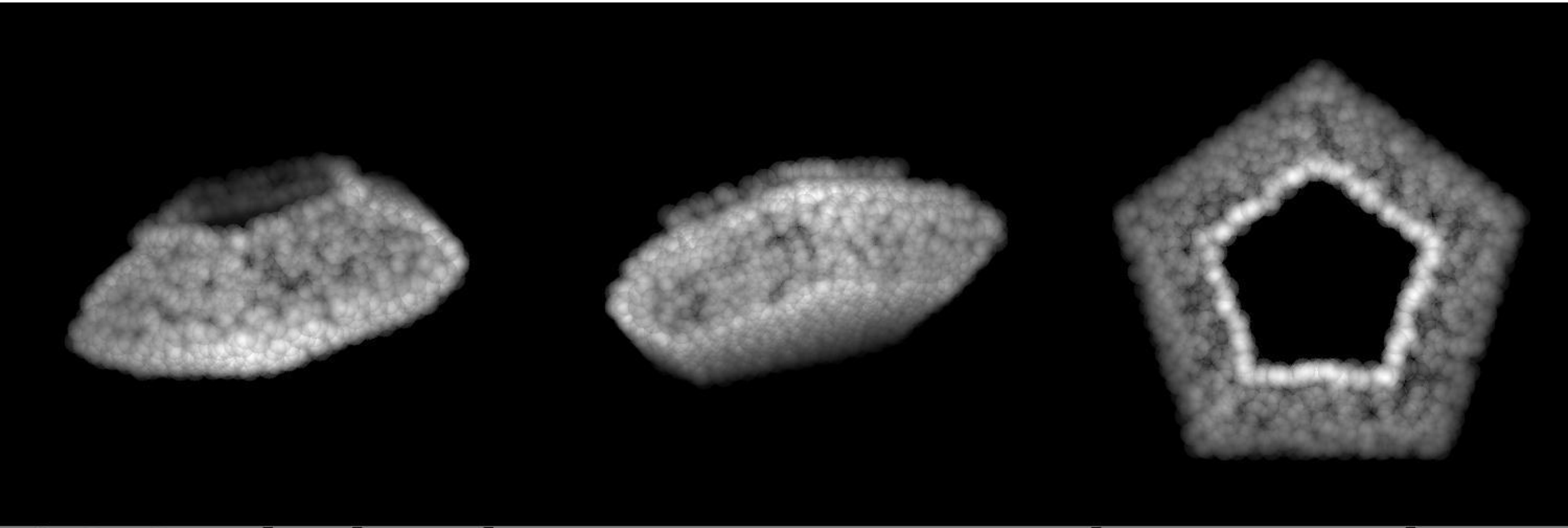
195_label_table_pred_desk.jpg



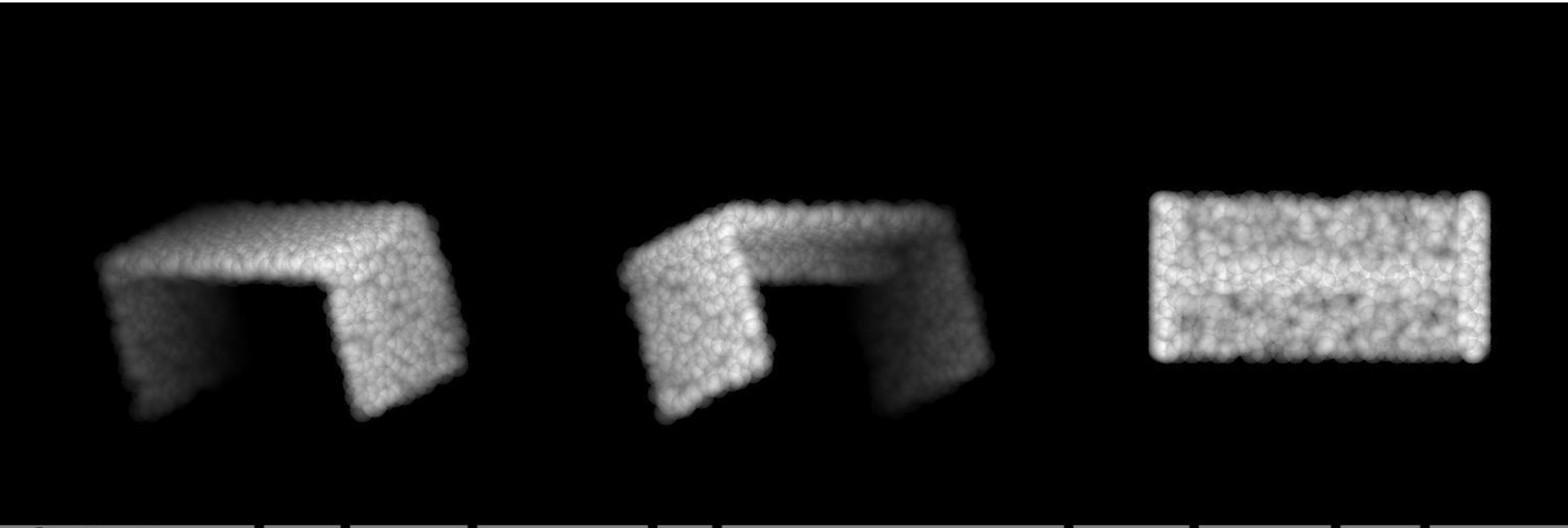
196_label_piano_pred_bench.jpg



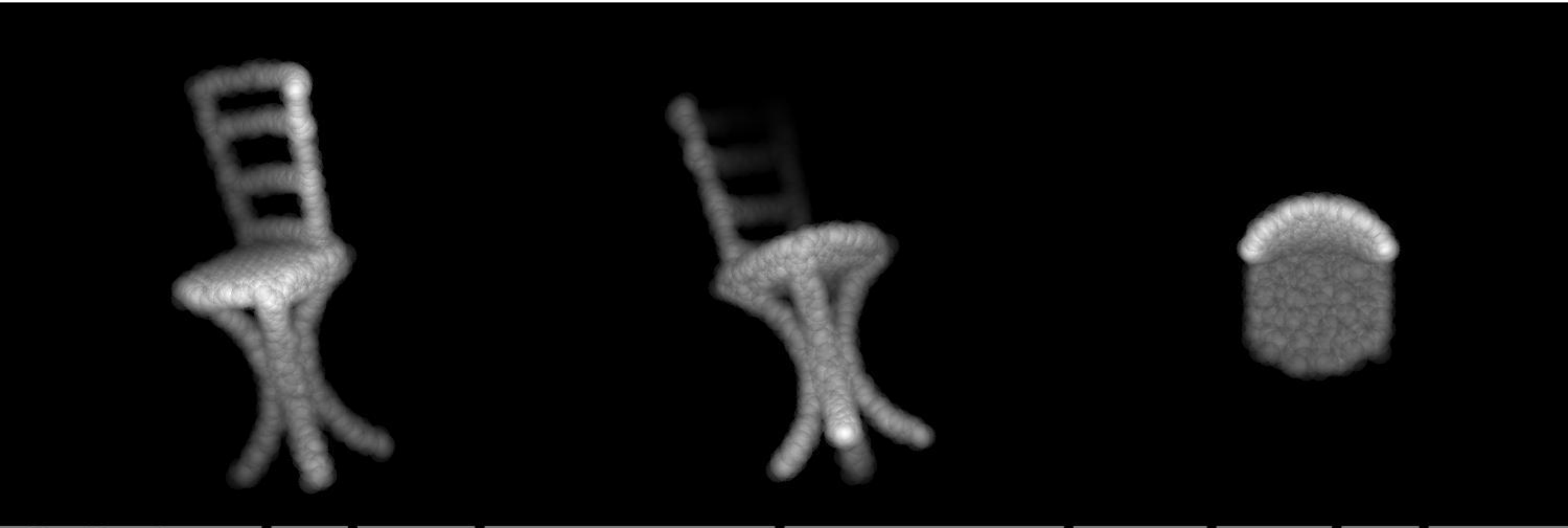
197_label_vase_pred_flower_pot.jpg

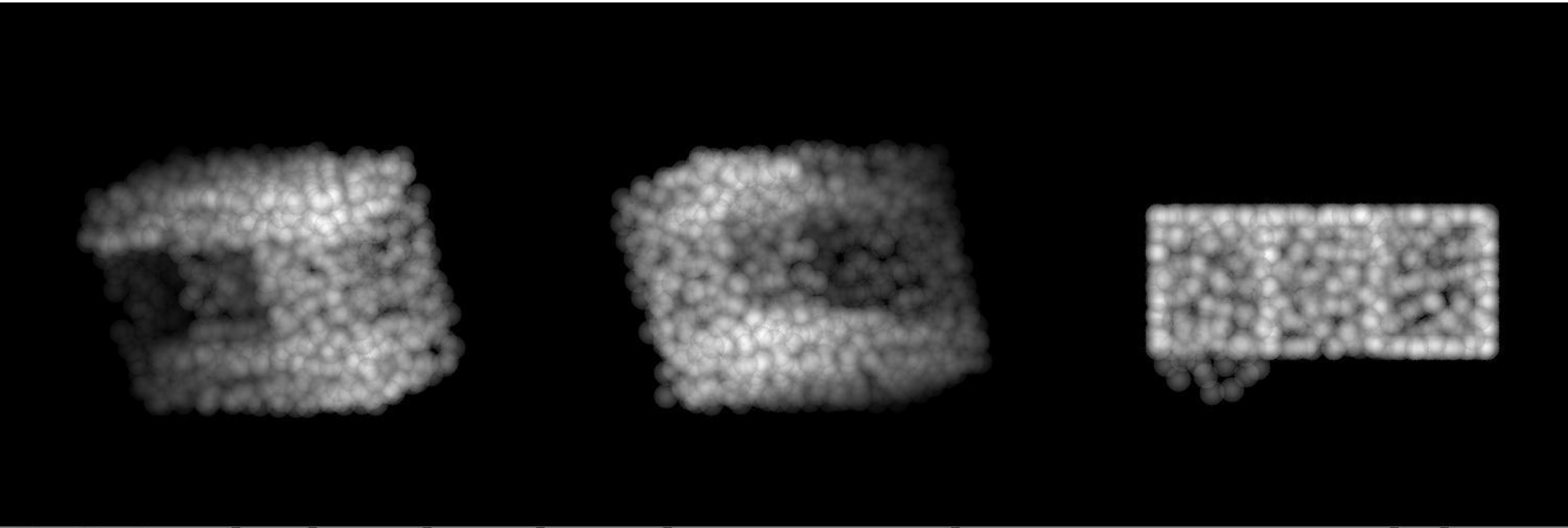


198_label_vase_pred_tent.jpg

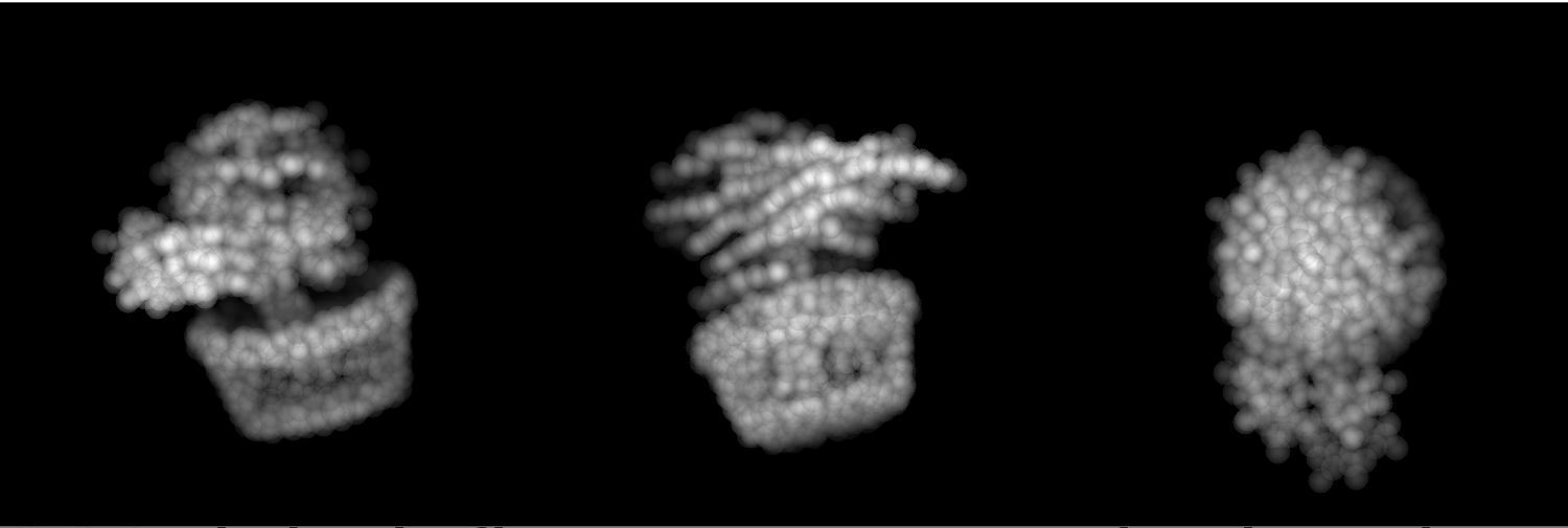


199_label_table_pred_desk.jpg

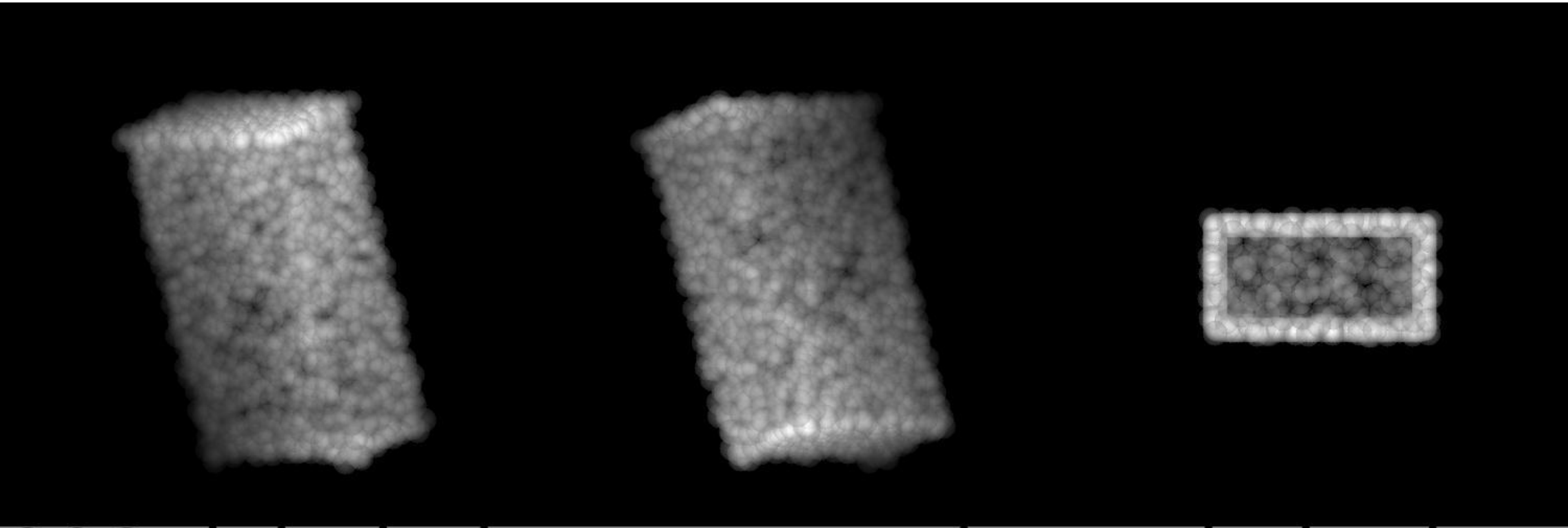

200_label_stool_pred_chair.jpg

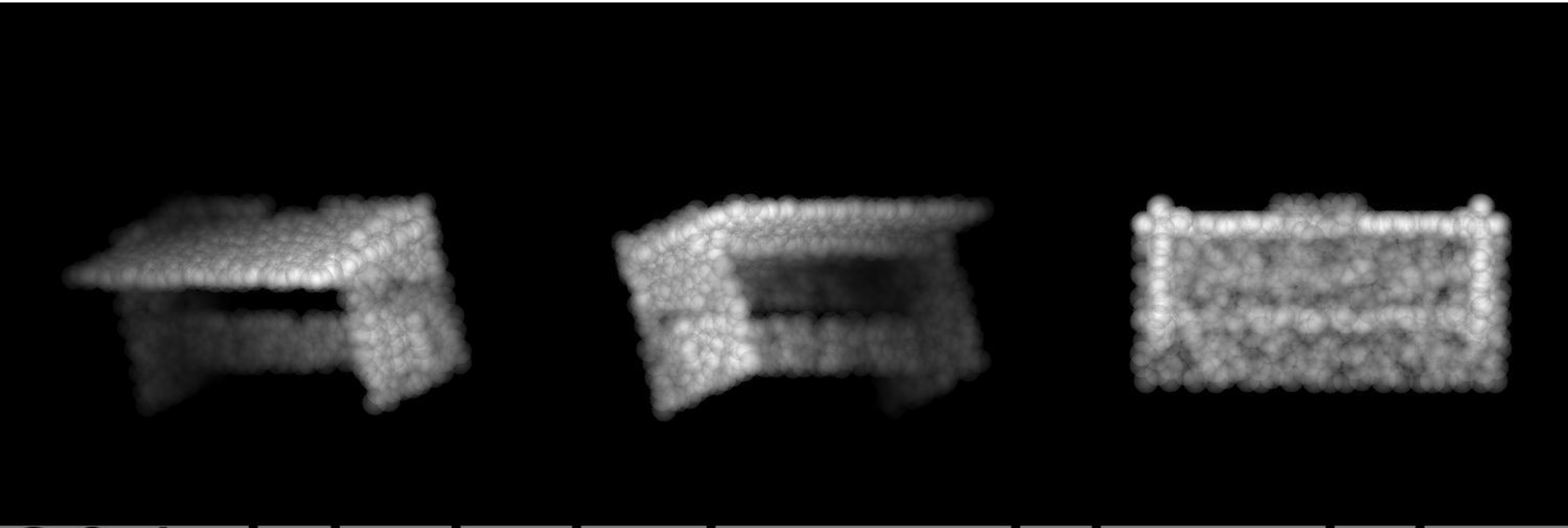


201_label_desk_pred_tv_stand.jpg

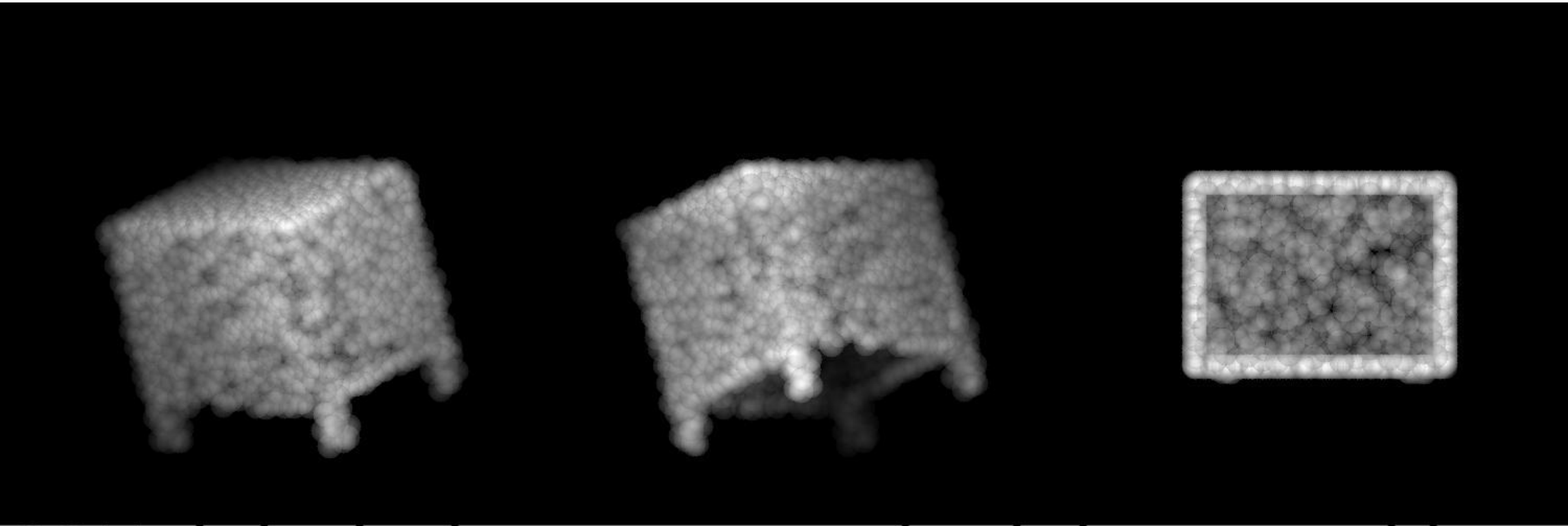


202_label_flower_pot_pred_plant.jpg

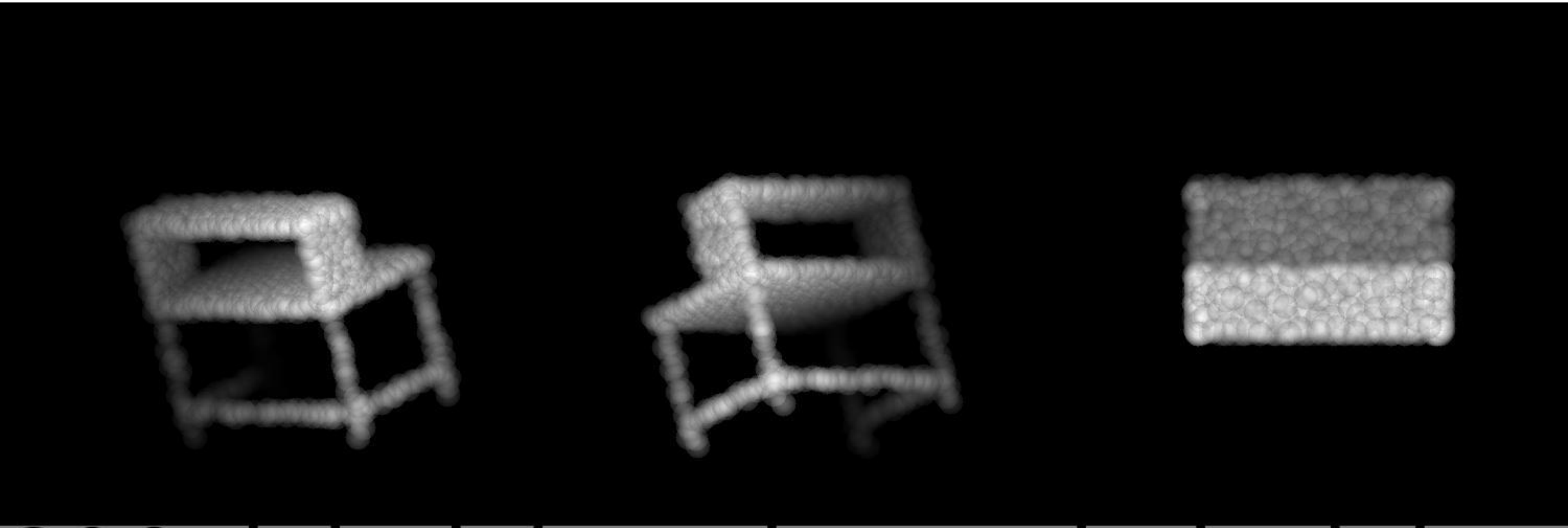

203_label_dresser_pred_wardrobe.jpg

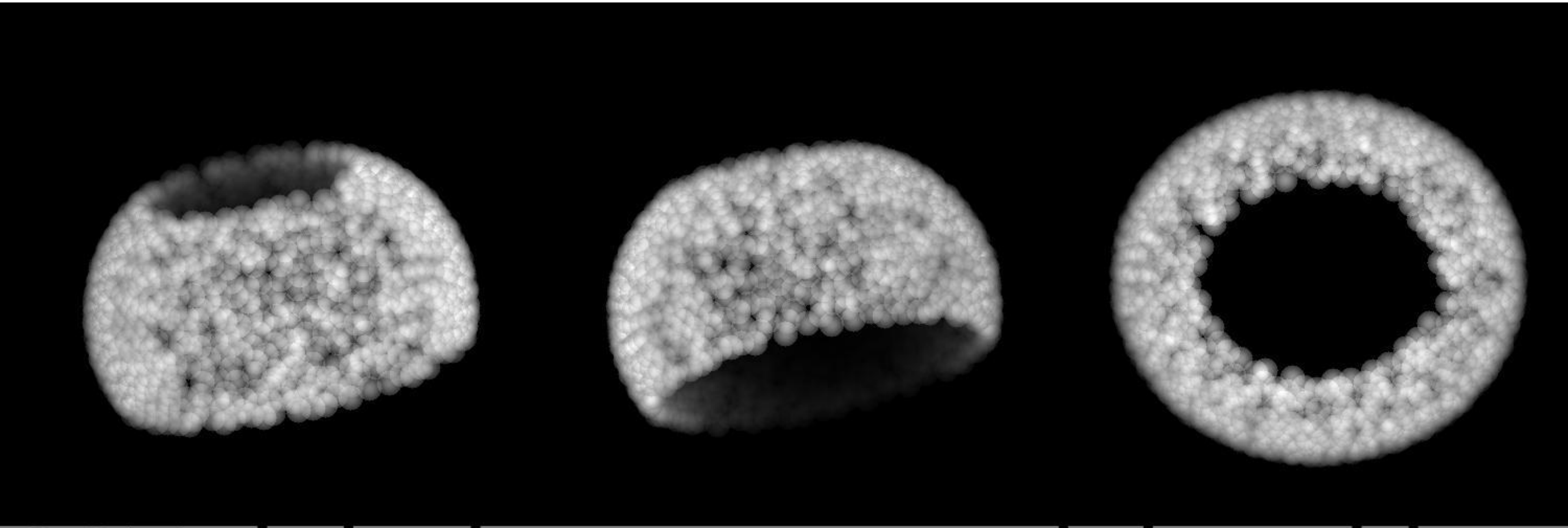


204_label_desk_pred_bench.jpg

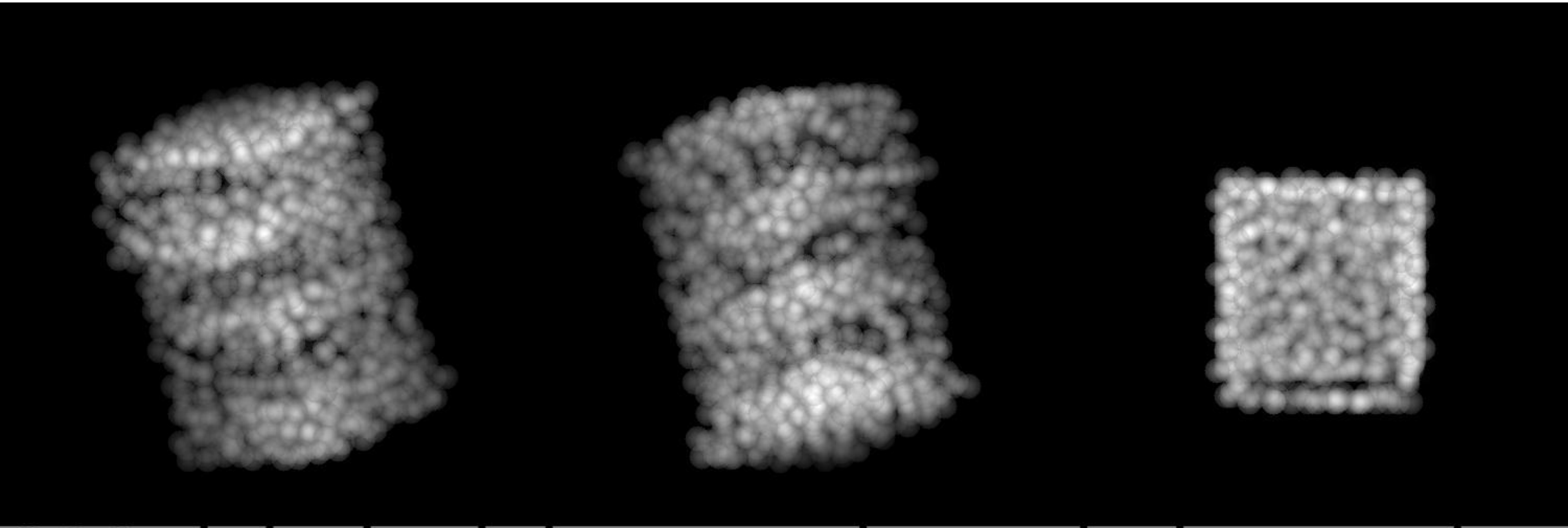


205_label_dresser_pred_night_stand.jpg

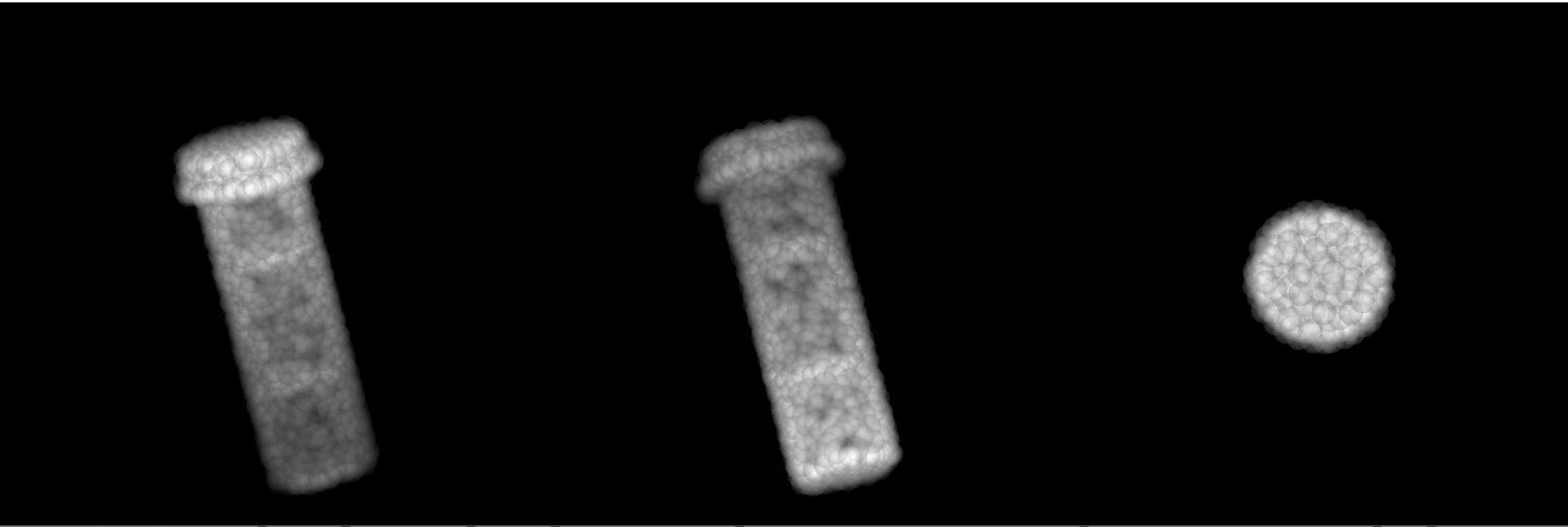

206_label_bench_pred_desk.jpg



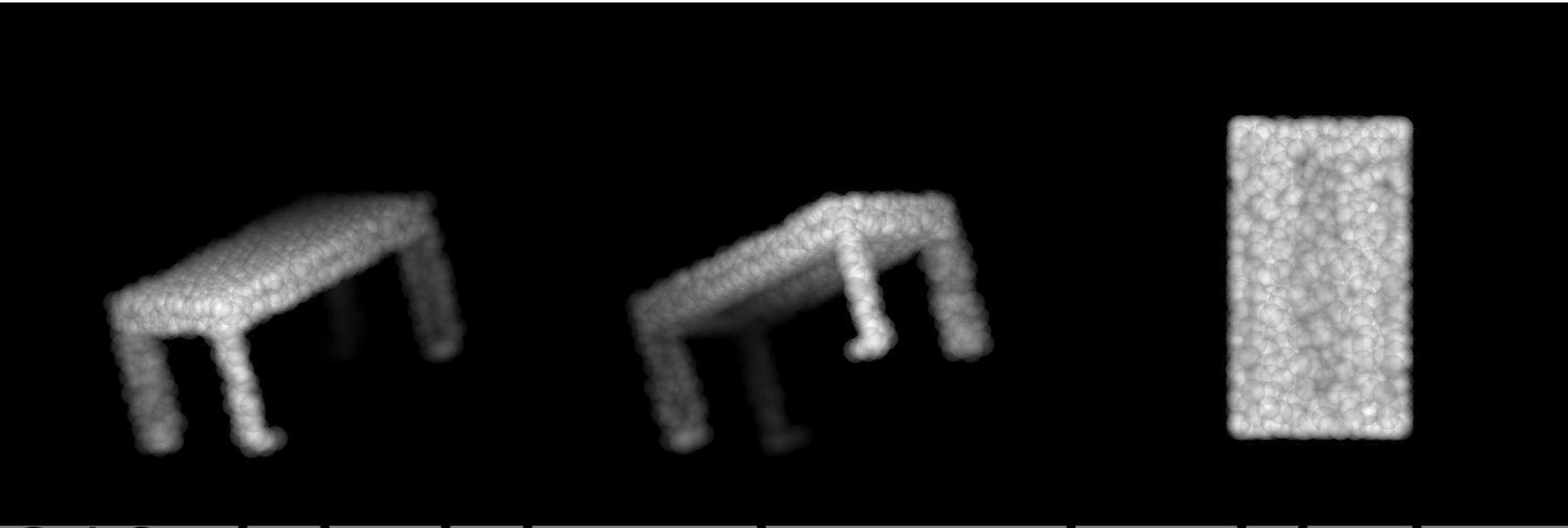
207_label_vase_pred_bowl.jpg



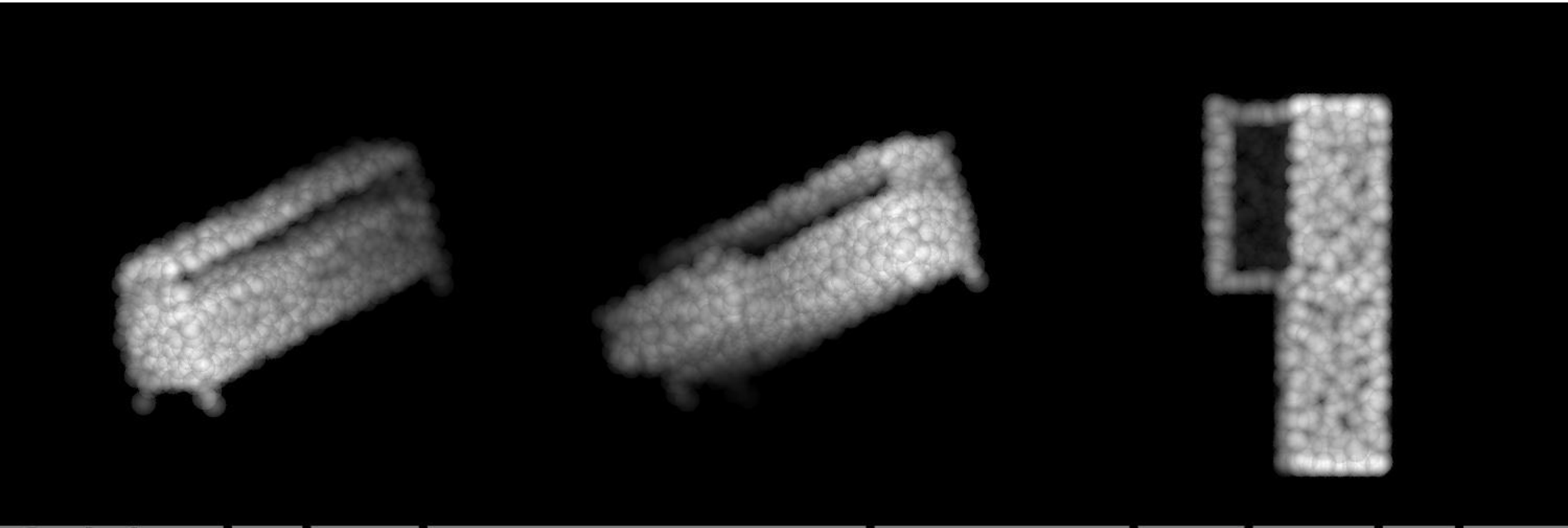
208_label_night_stand_pred_dresser.jpg



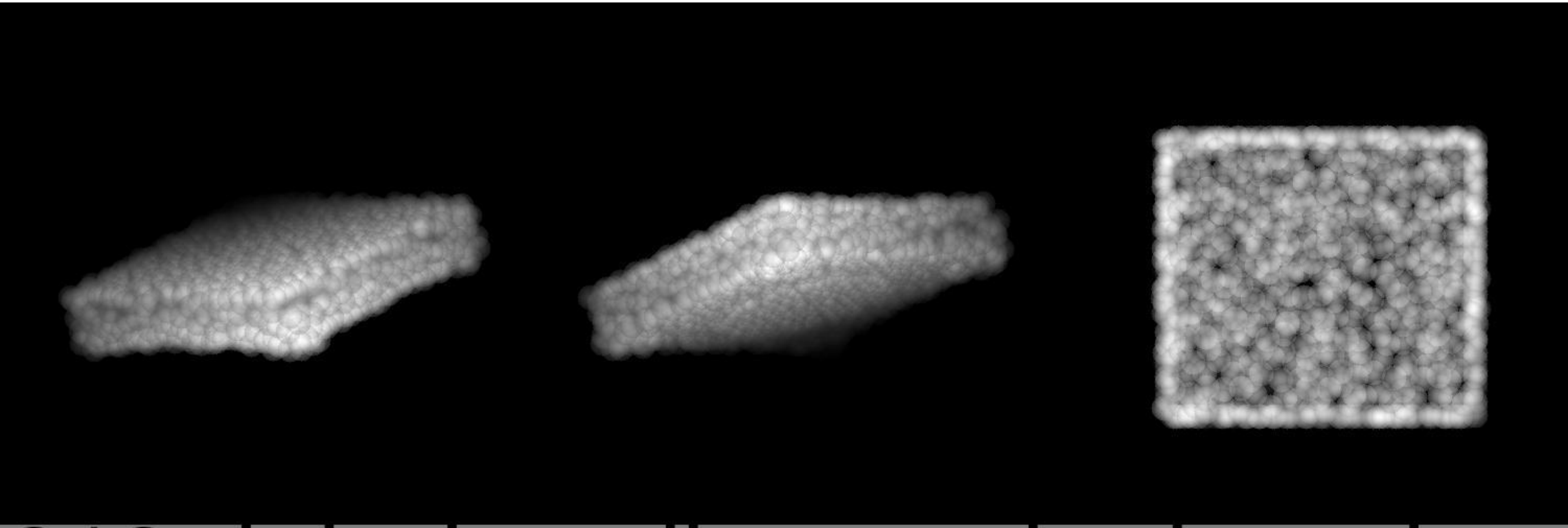
209_label_bottle_pred_stool.jpg

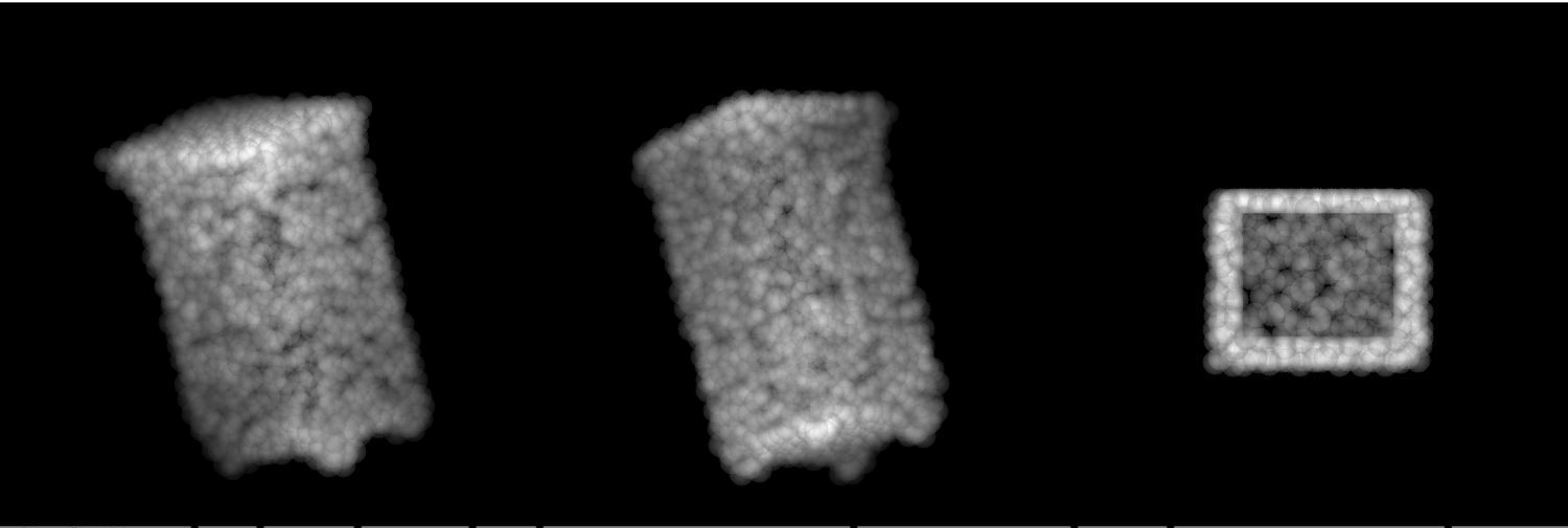


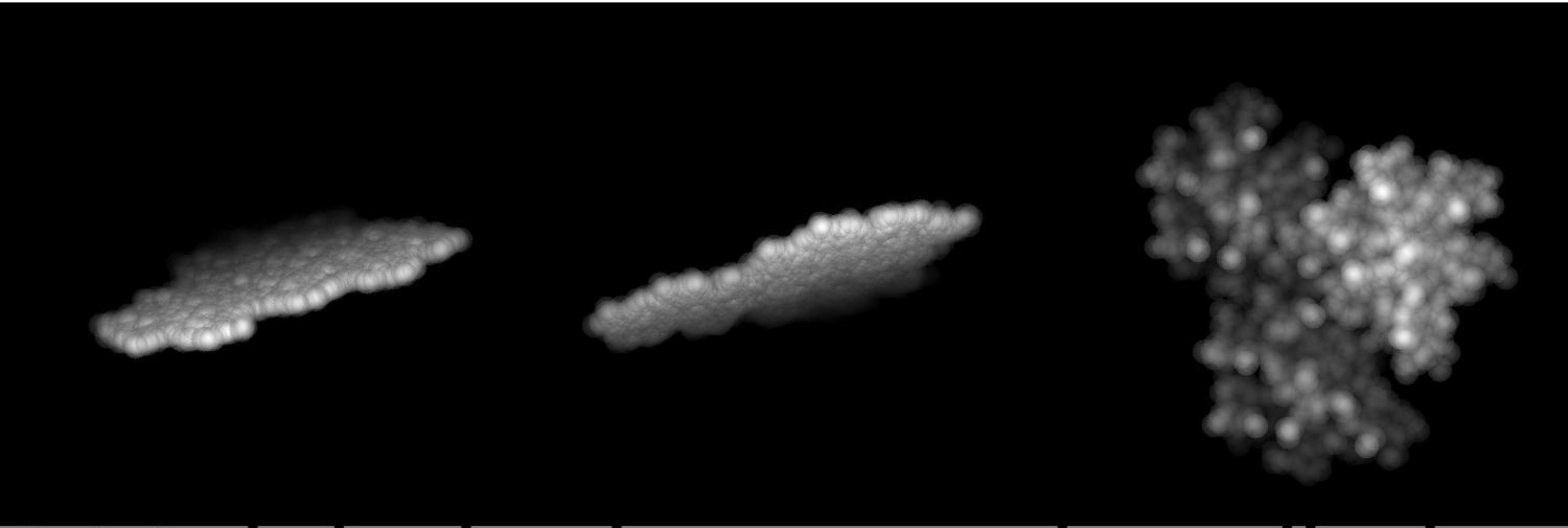
210_label_bench_pred_table.jpg

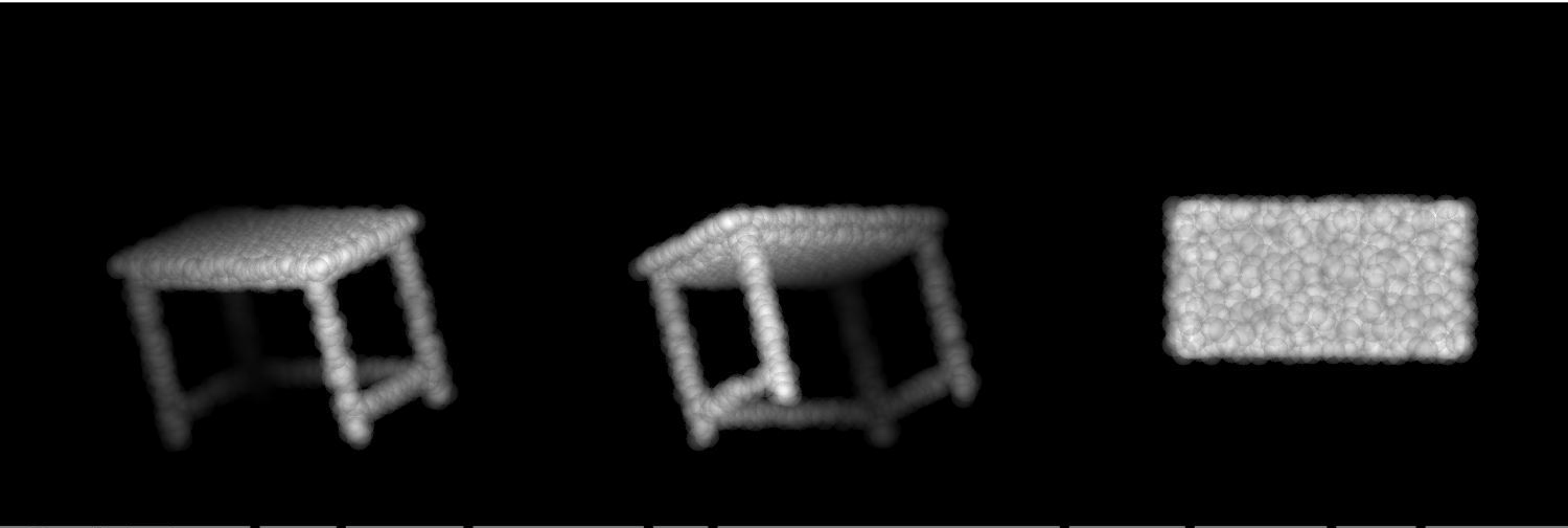


211_label_tv_stand_pred_desk.jpg

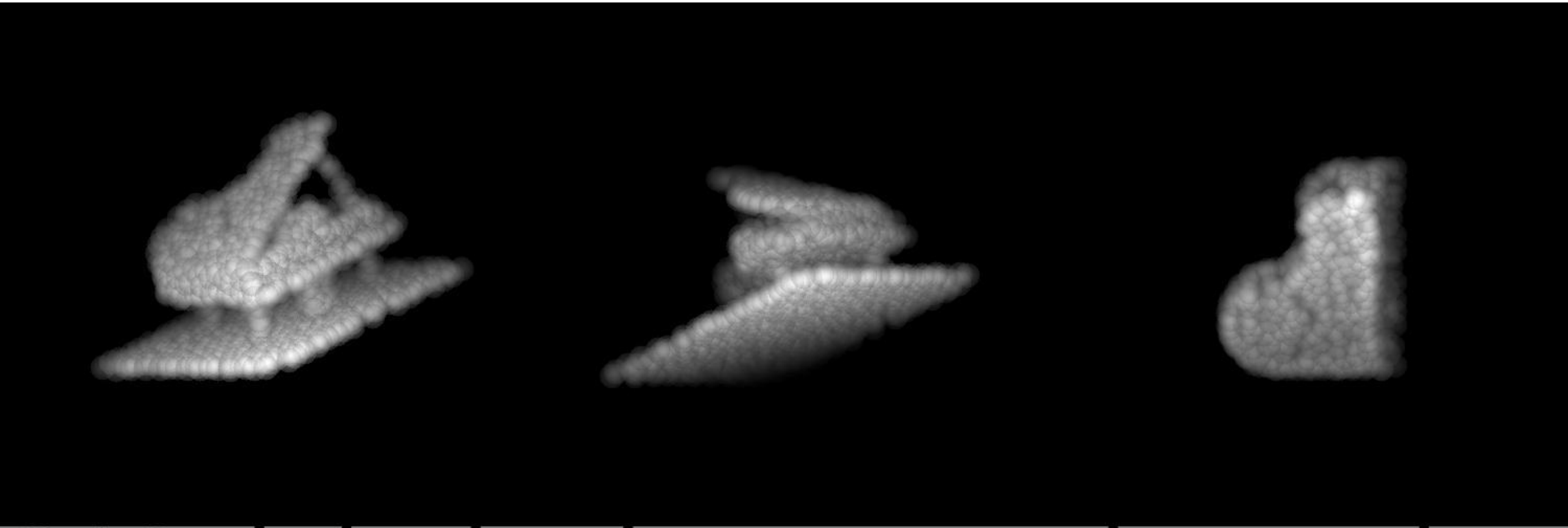

212_label_radio_pred_piano.jpg


213_label_night_stand_pred_dresser.jpg

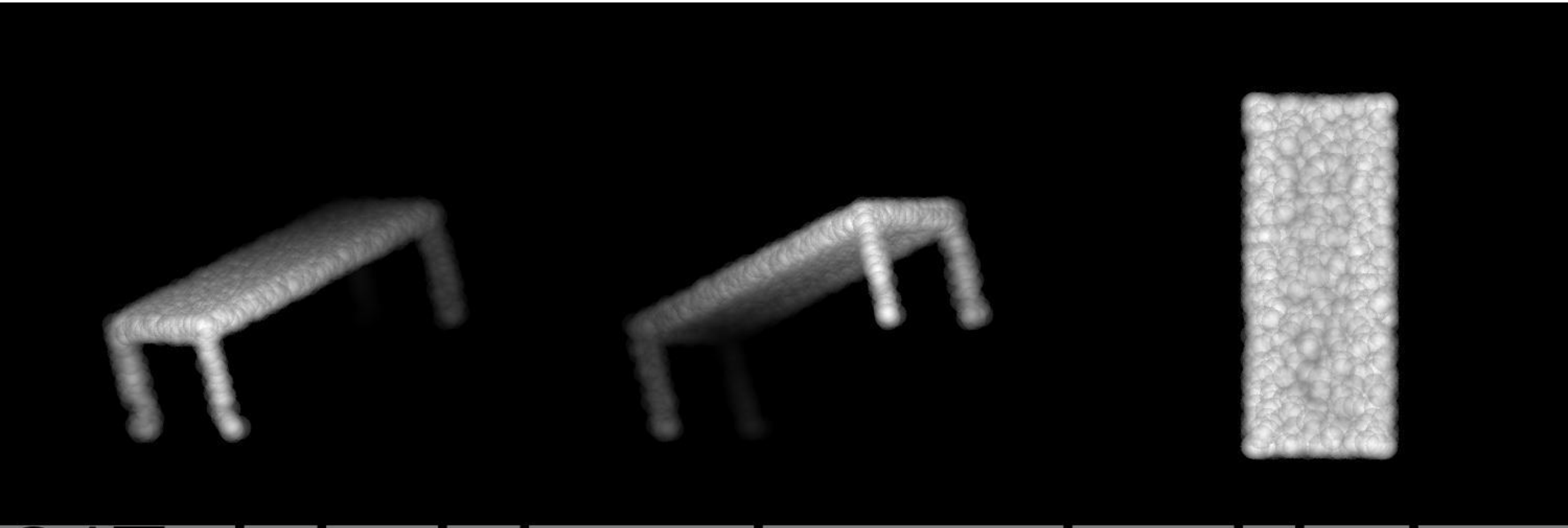

214_label_plant_pred_radio.jpg



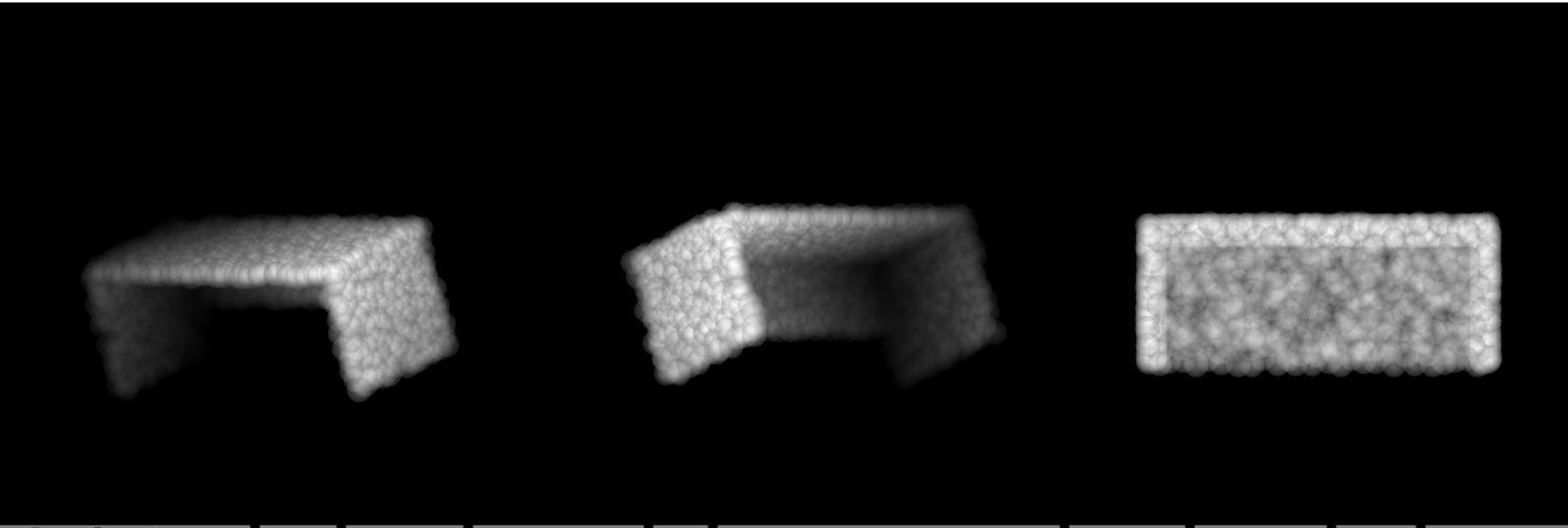
215_label_table_pred_desk.jpg



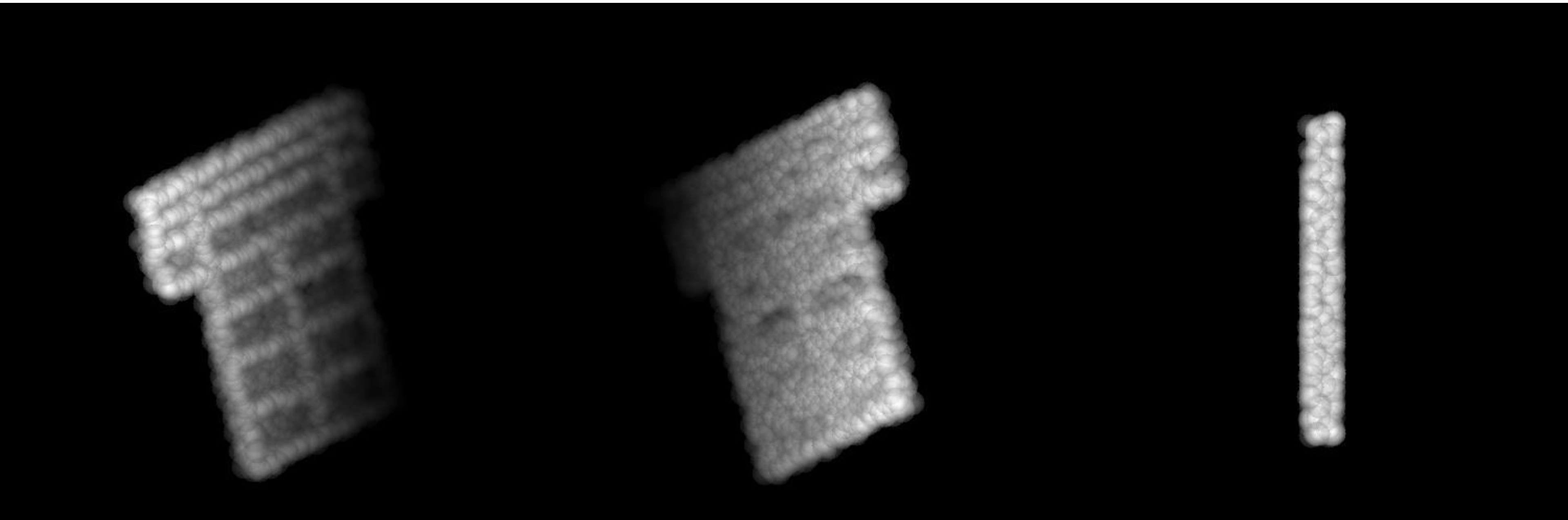
216_label_piano_pred_tent.jpg



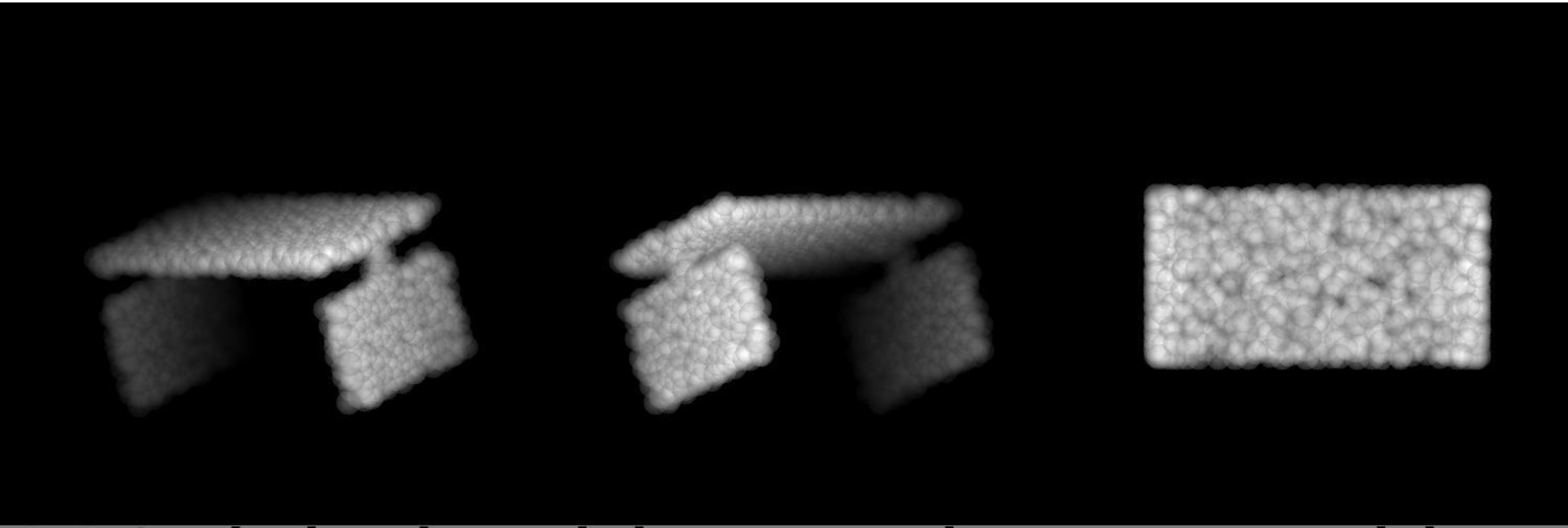
217_label_bench_pred_table.jpg



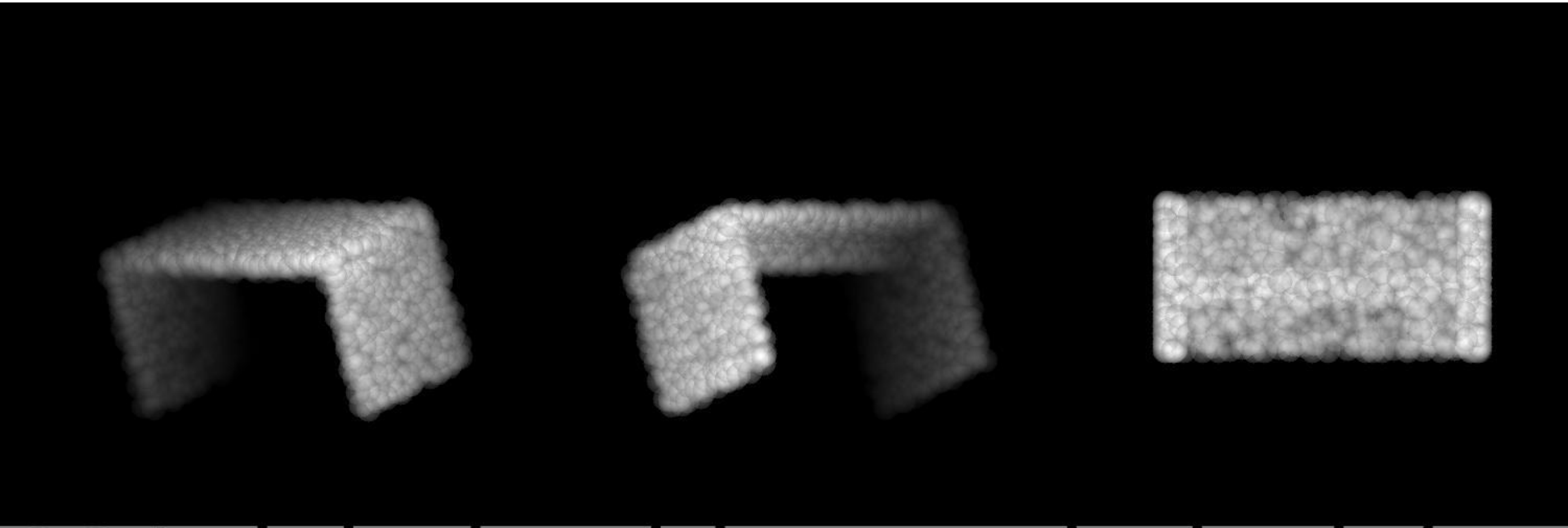
218_label_table_pred_desk.jpg



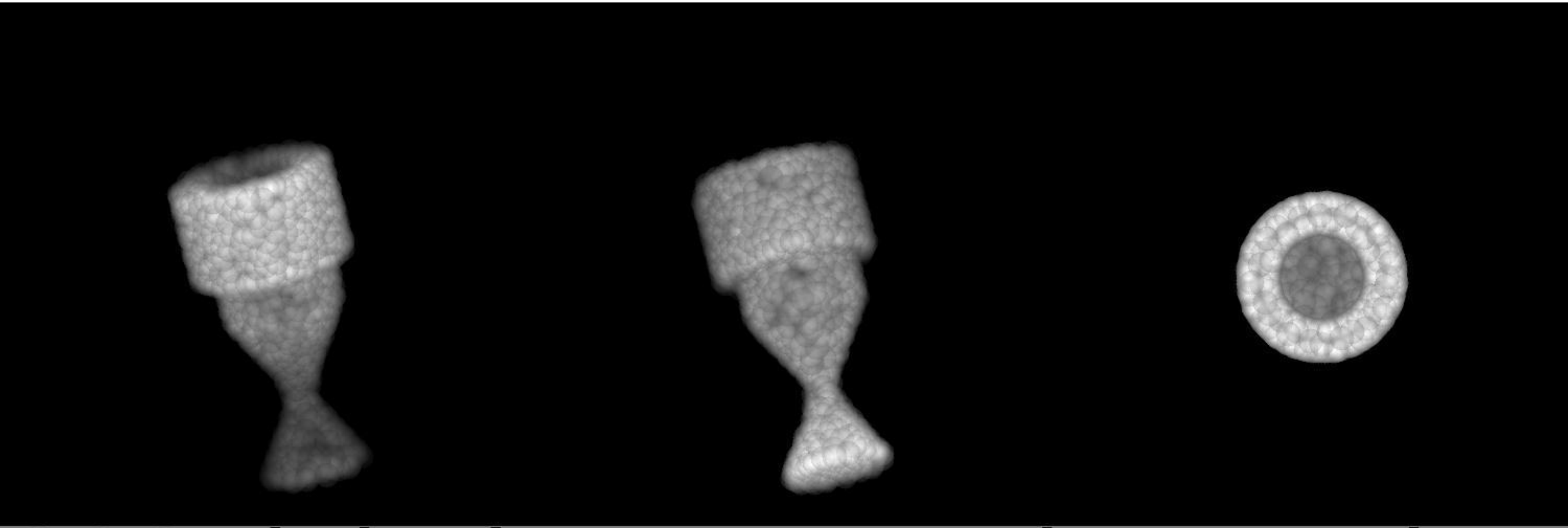
219_label_bookshelf_pred_curtain.jpg



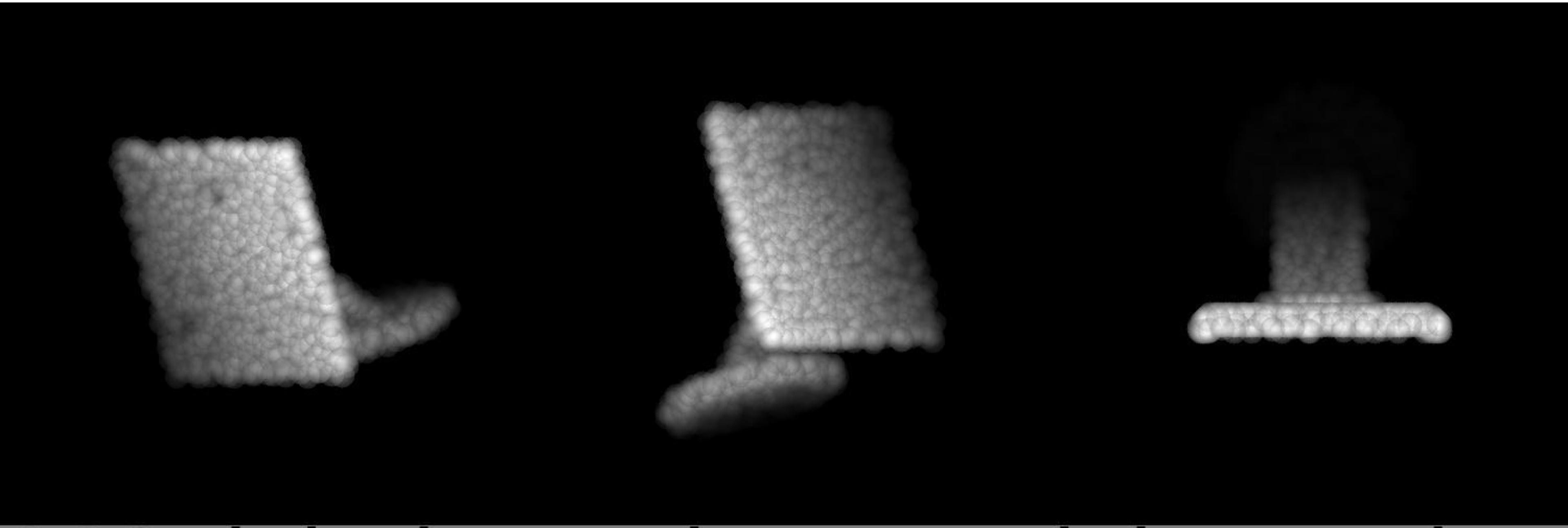
220_label_table_pred_tv_stand.jpg



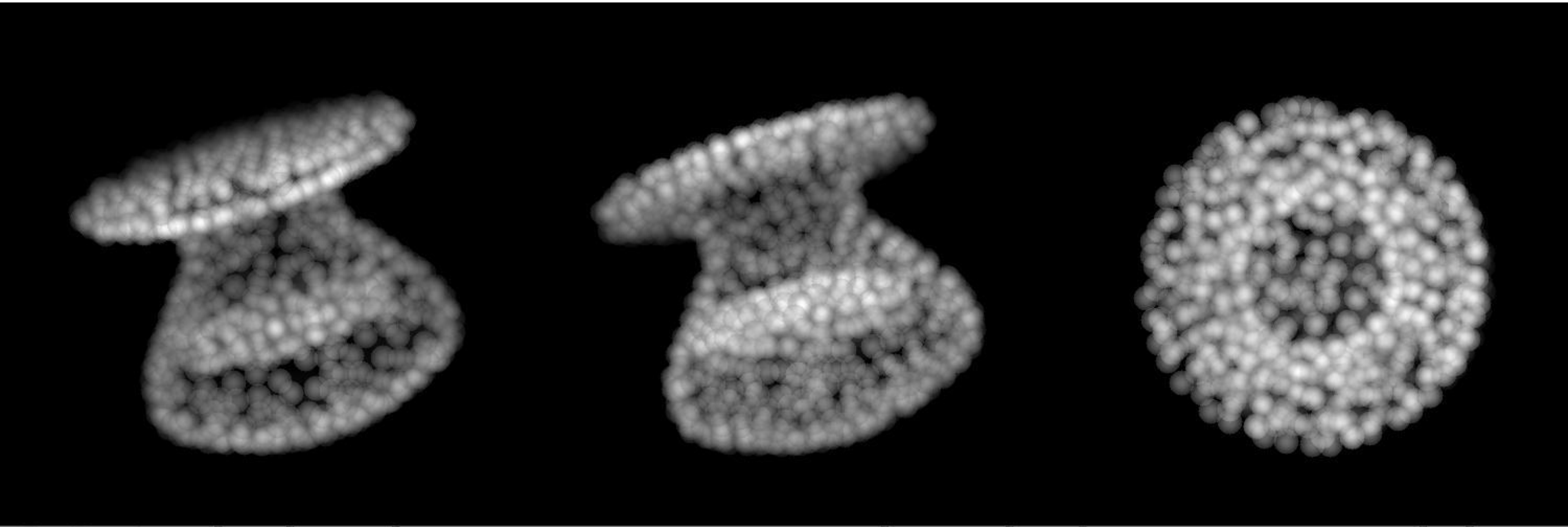
221_label_table_pred_desk.jpg



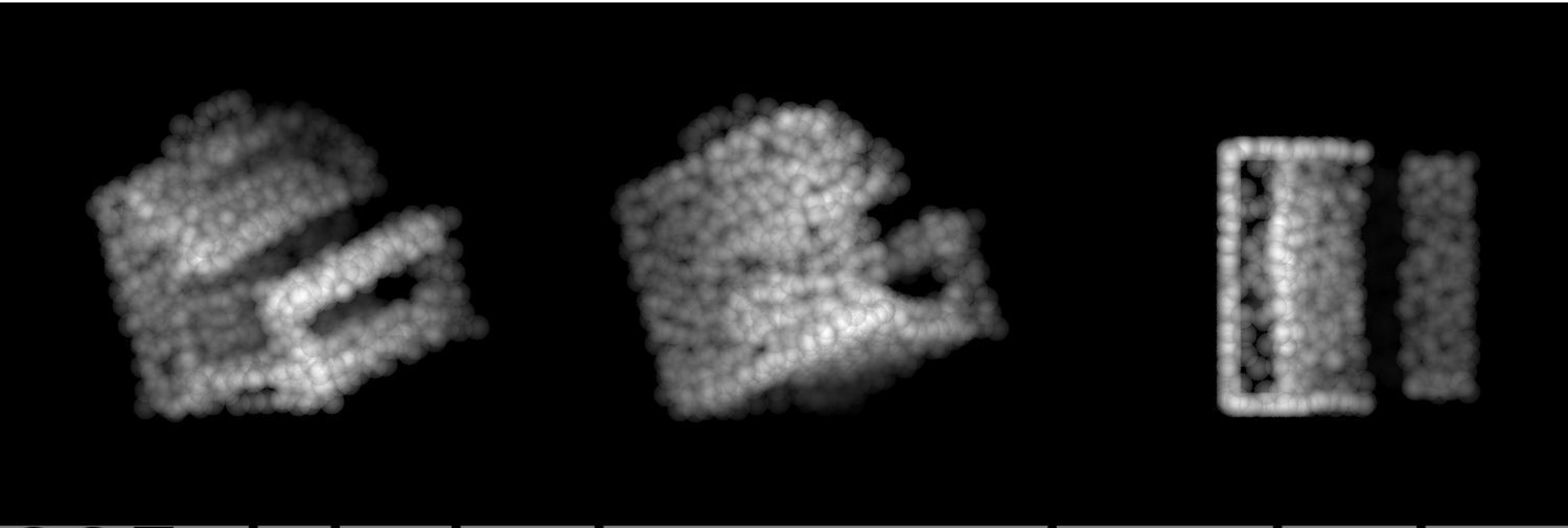
222_label_cup_pred_vase.jpg



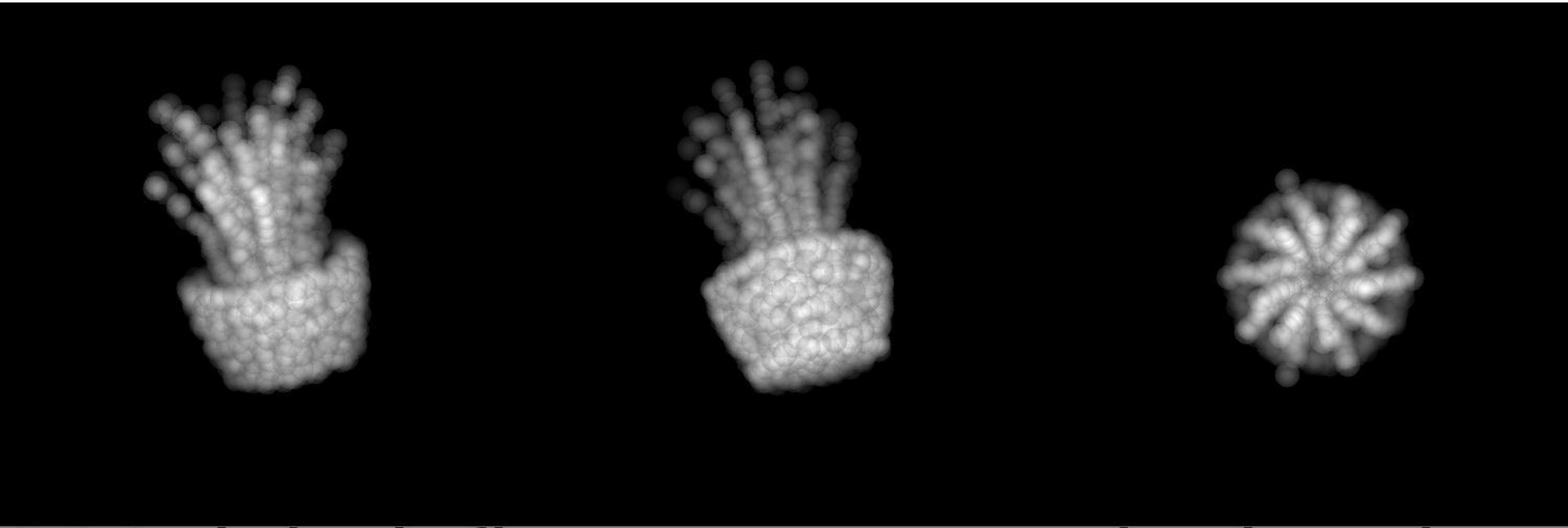
223_label_monitor_pred_laptop.jpg



224_label_vase_pred_night_stand.jpg

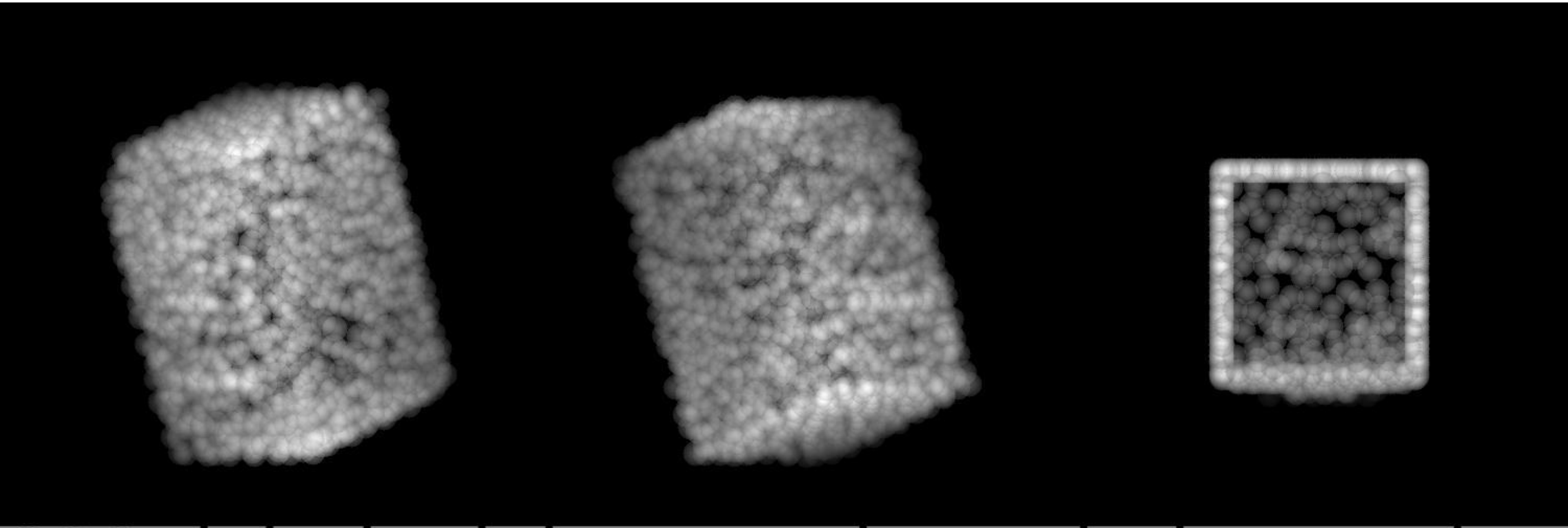


225_label_piano_pred_stairs.jpg

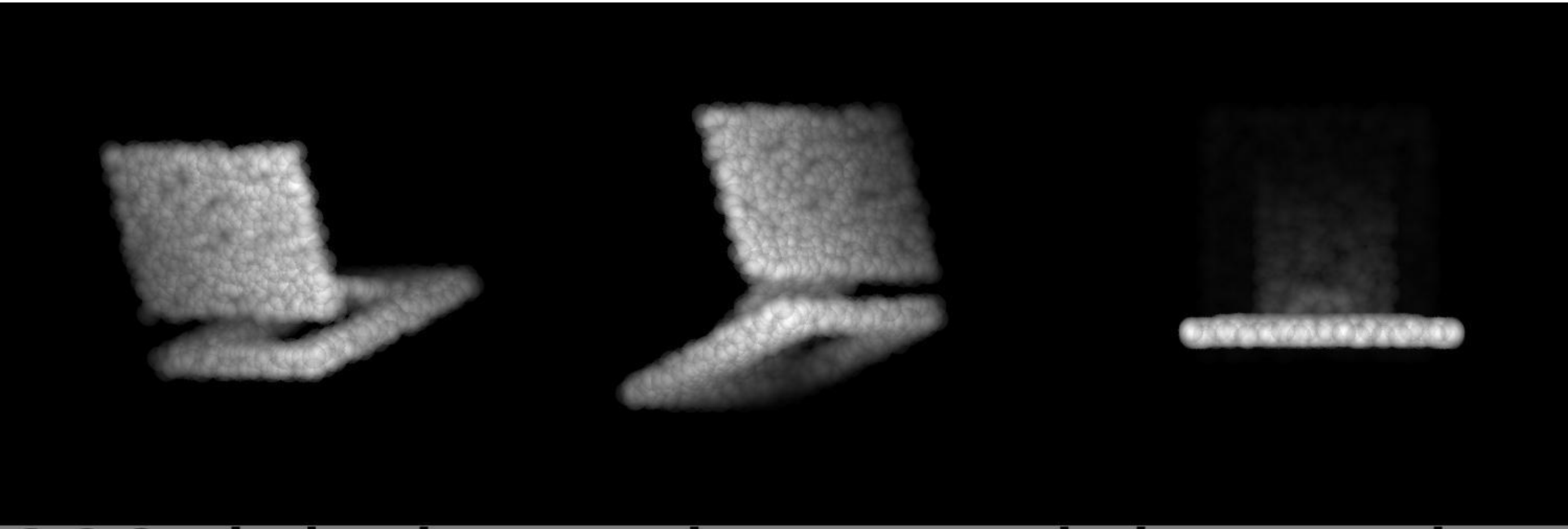


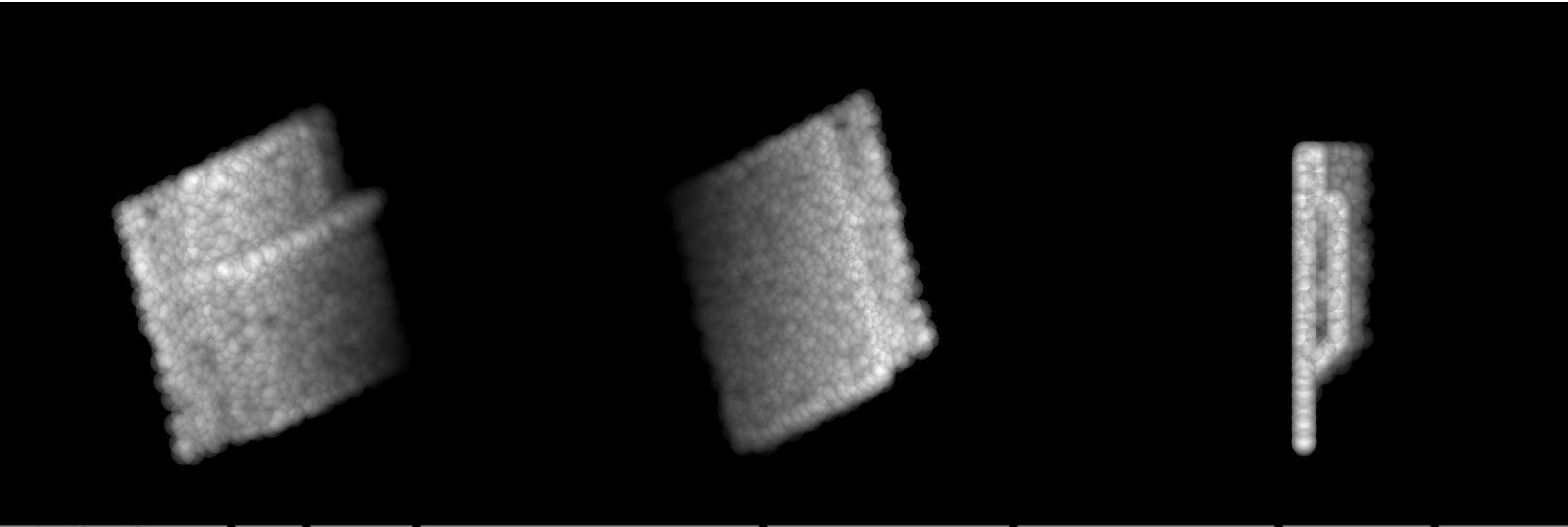
226_label_flower_pot_pred_plant.jpg

227_label_plant_pred_flower_pot.jpg

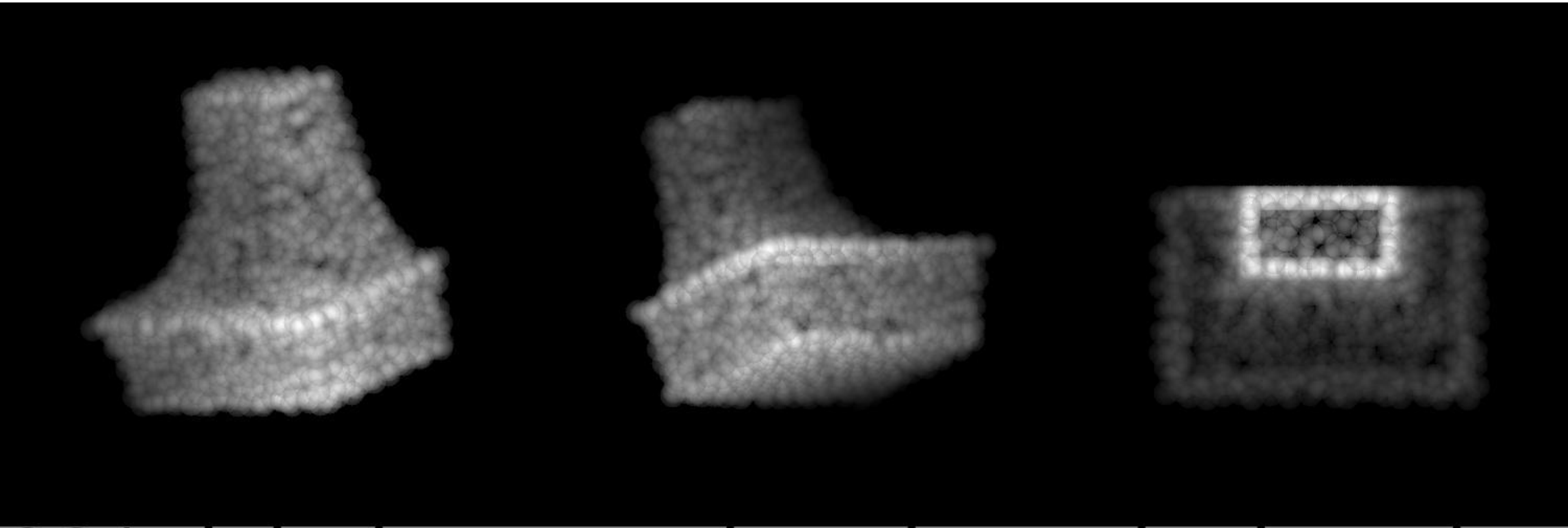


228_label_night_stand_pred_dresser.jpg

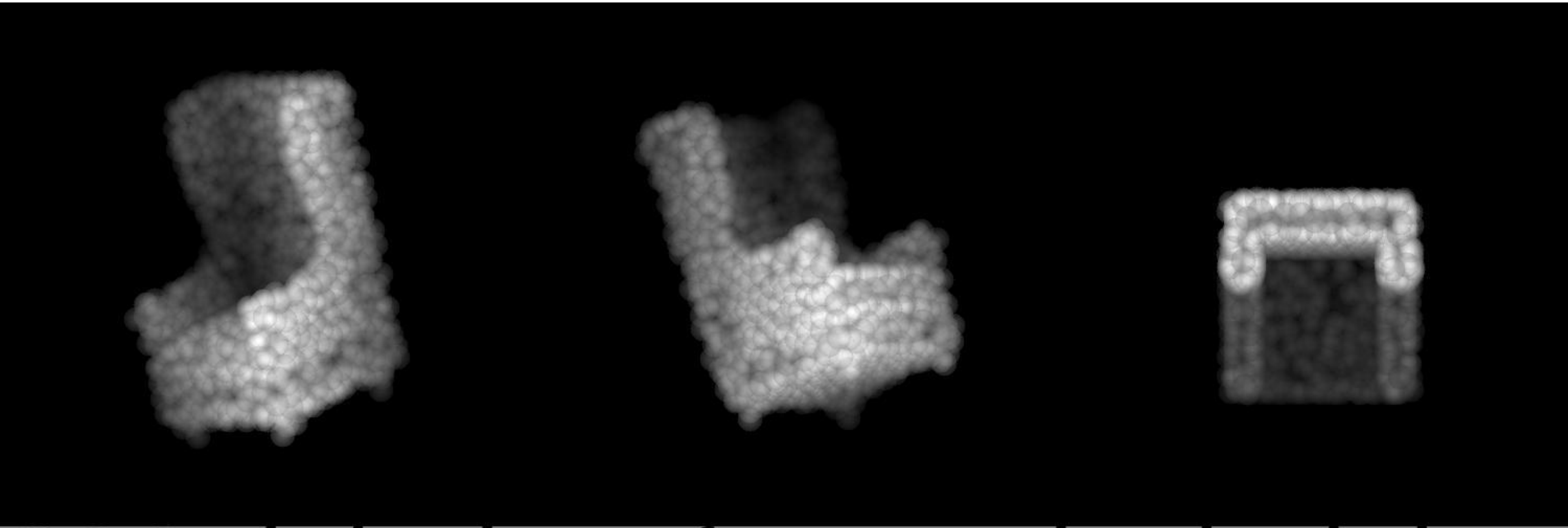

229_label_monitor_pred_laptop.jpg



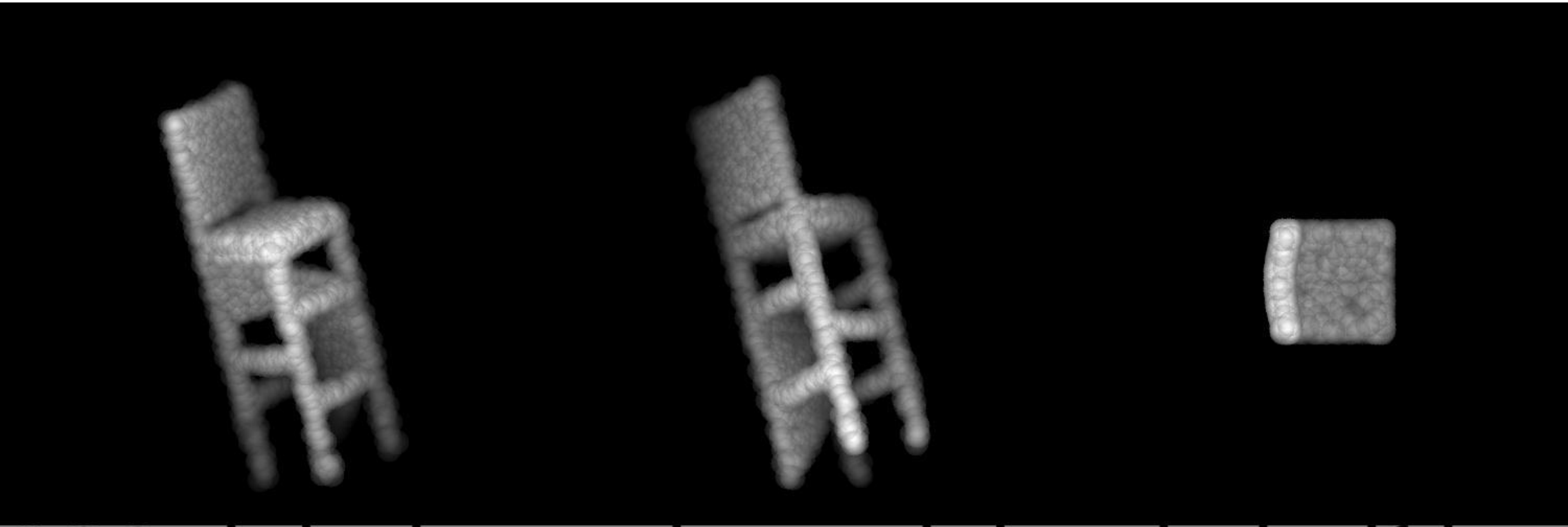
230_label_mantel_pred_monitor.jpg



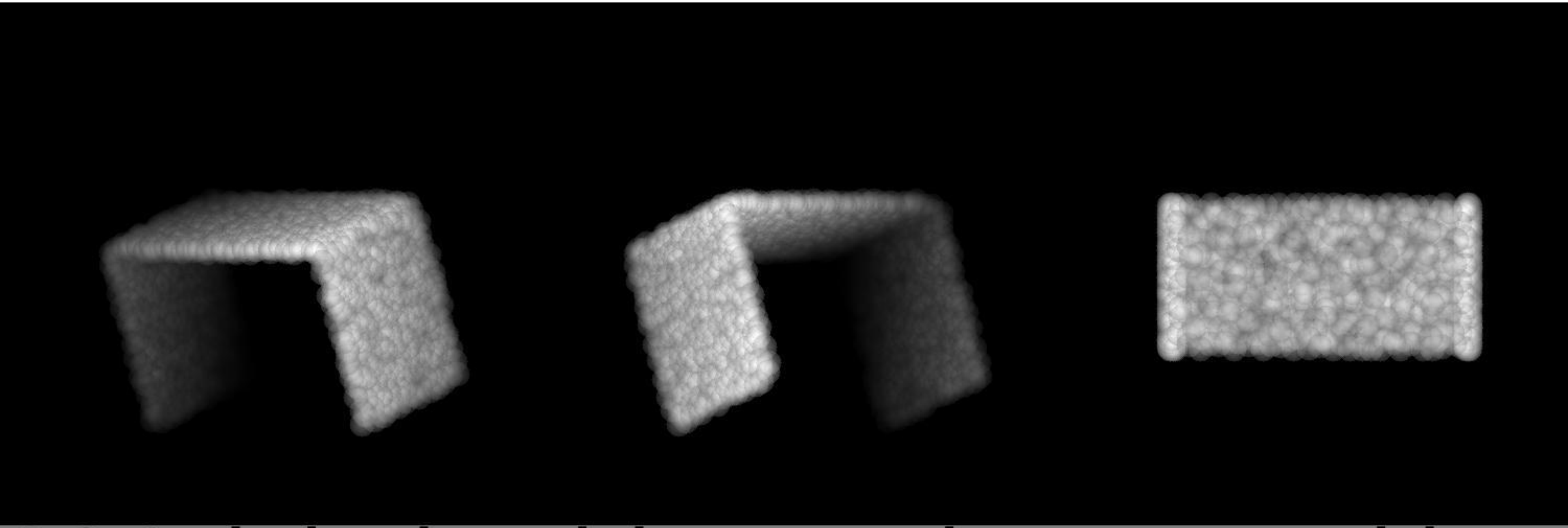
231_label_range_hood_pred_piano.jpg



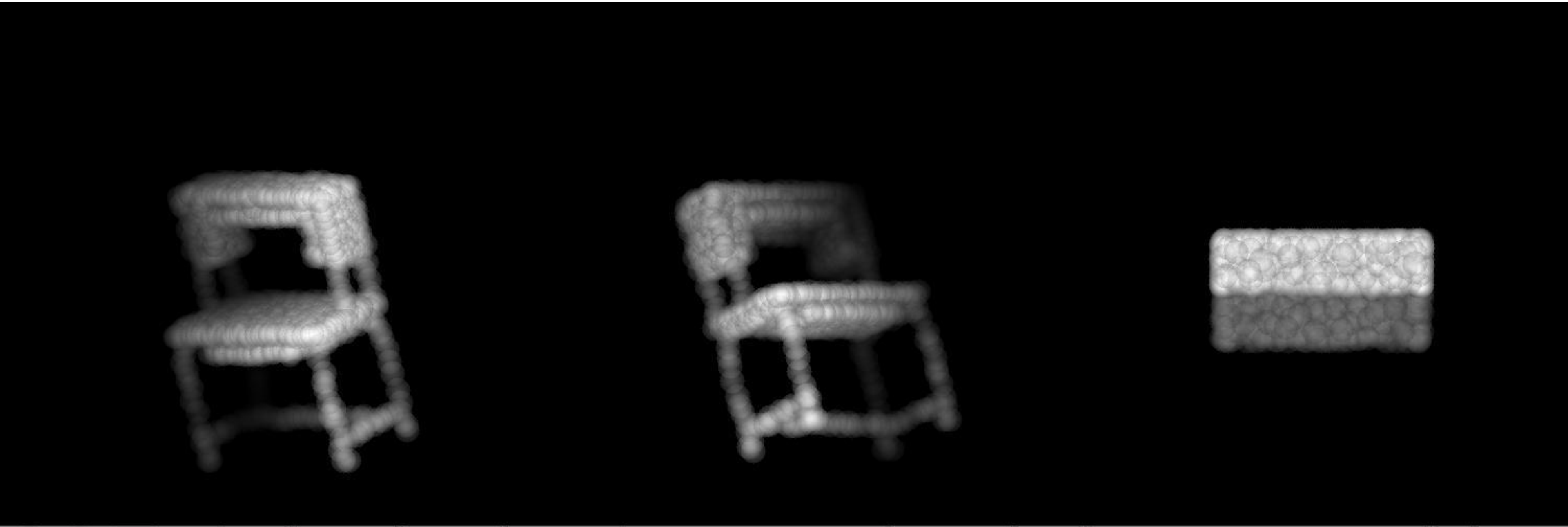
232_label_sofa_pred_chair.jpg



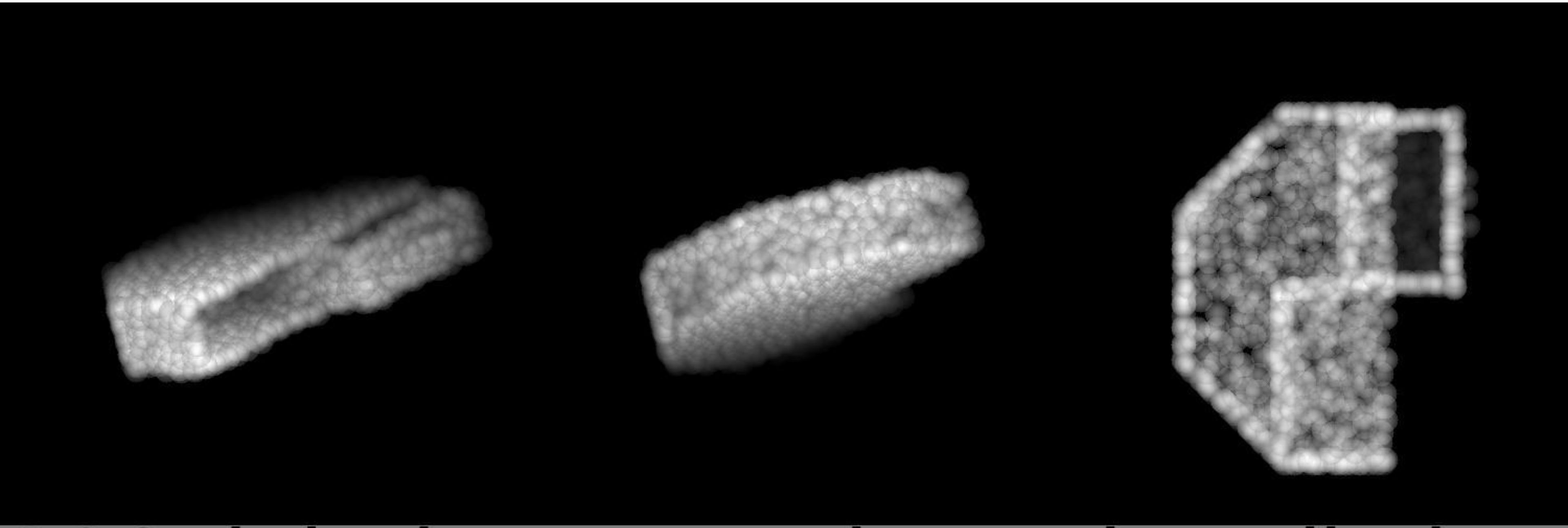
233_label_stool_pred_bookshelf.jpg

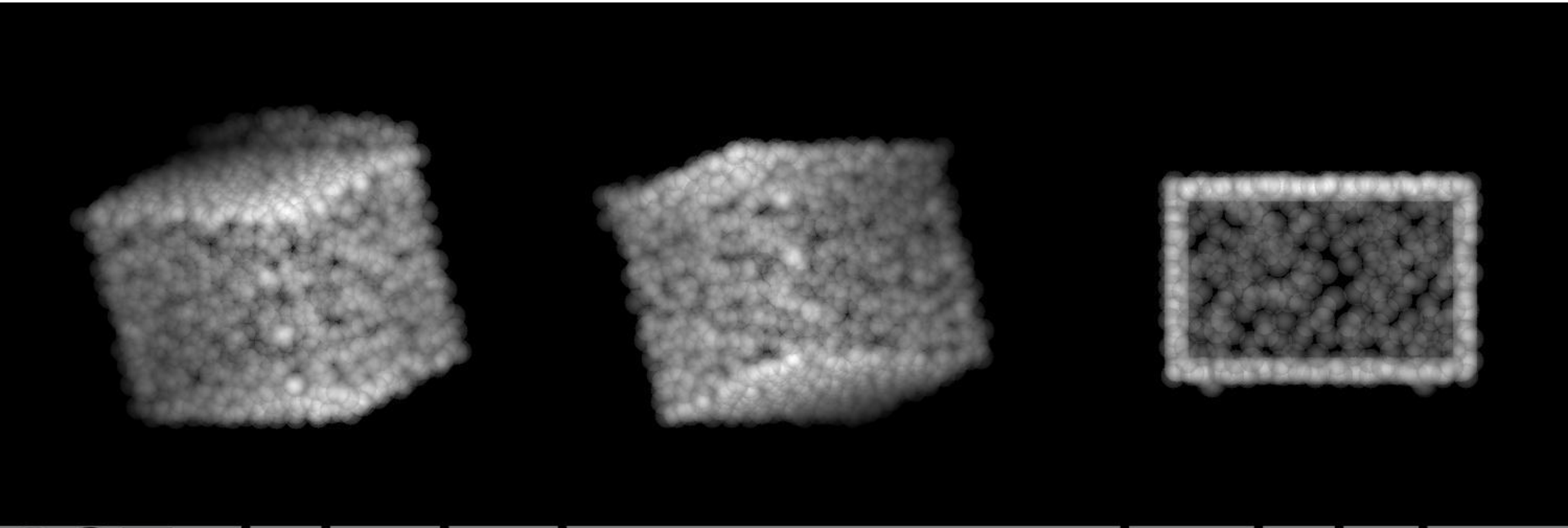


234_label_table_pred_tv_stand.jpg

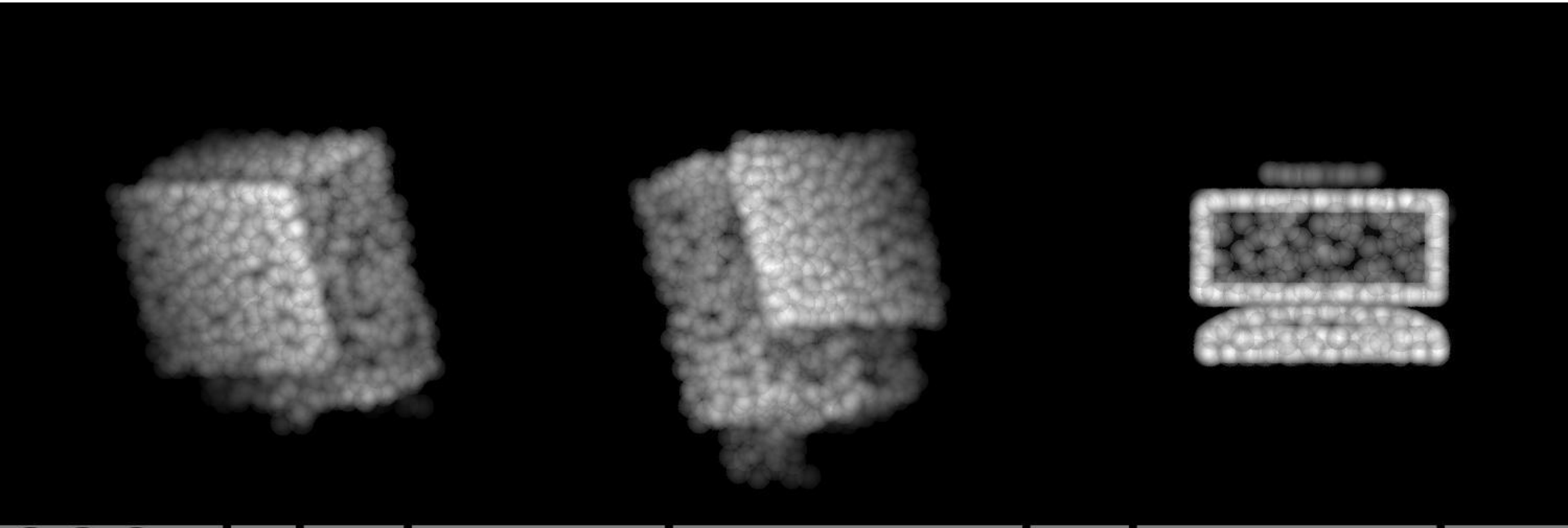


235_label_desk_pred_night_stand.jpg

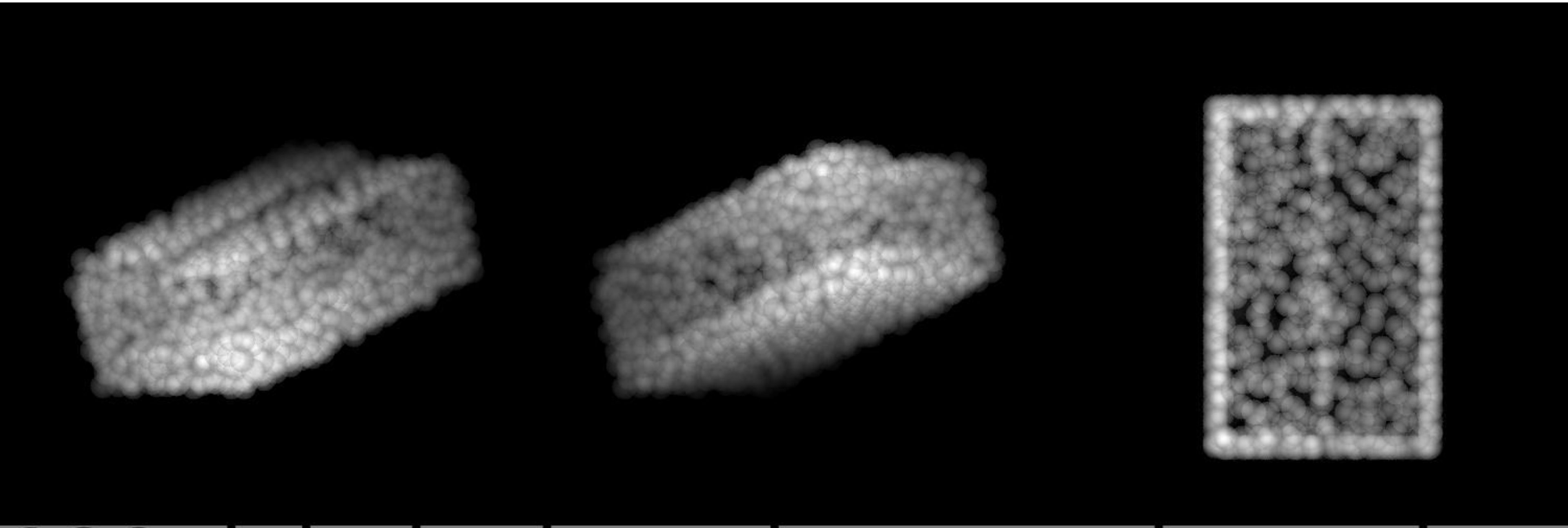

236_label_tv_stand_pred_radio.jpg



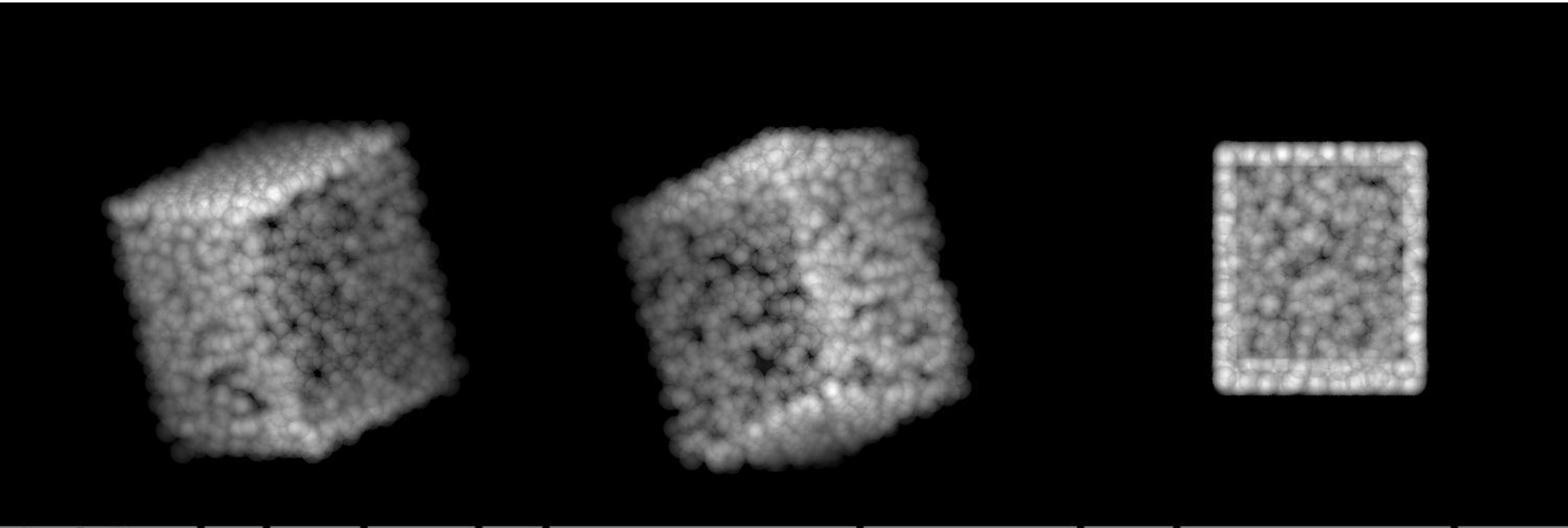
237_label_dresser_pred_sink.jpg



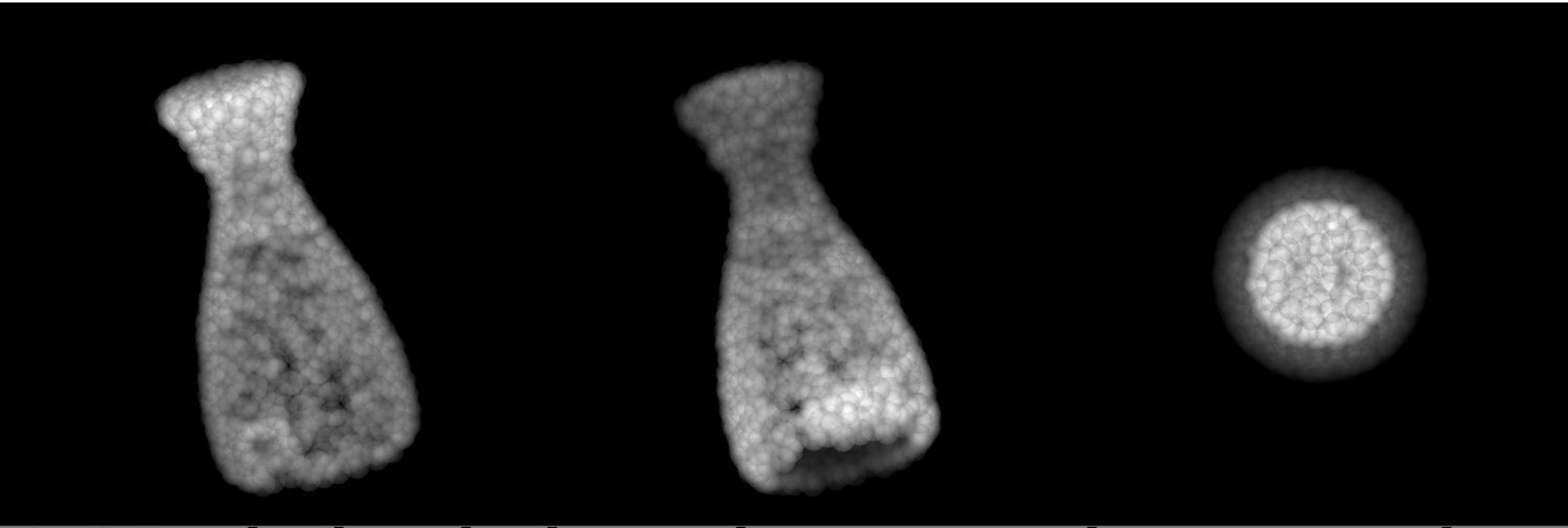
238_label_monitor_pred_dresser.jpg



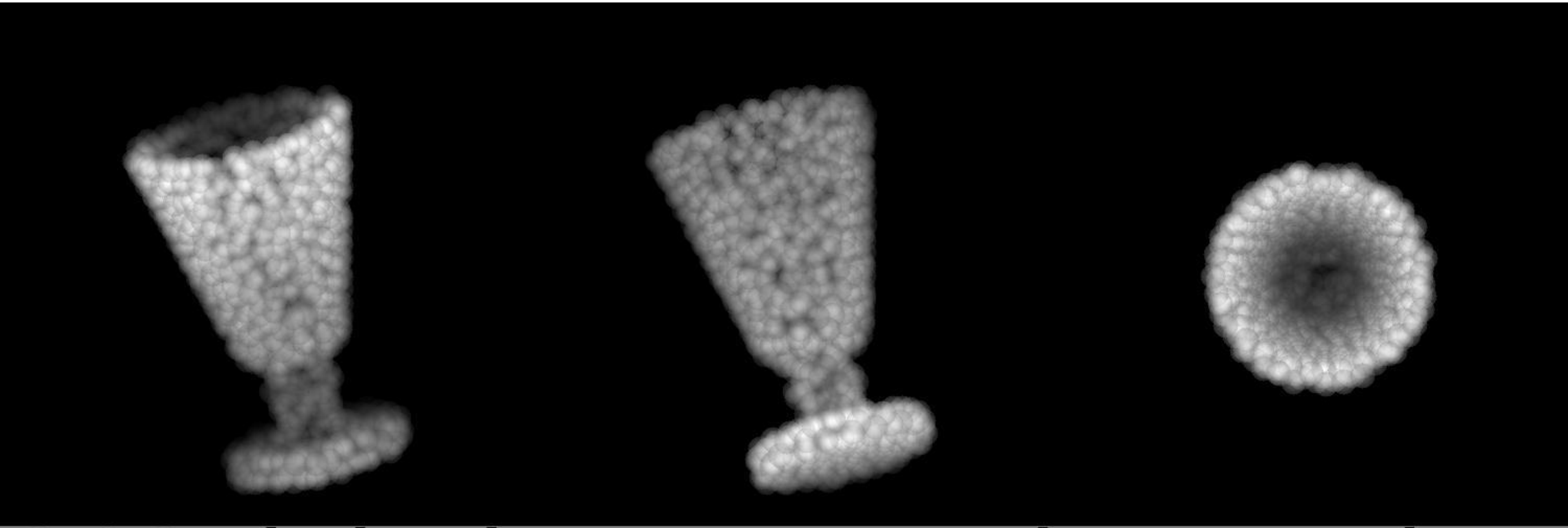
239_label_glass_box_pred_tent.jpg



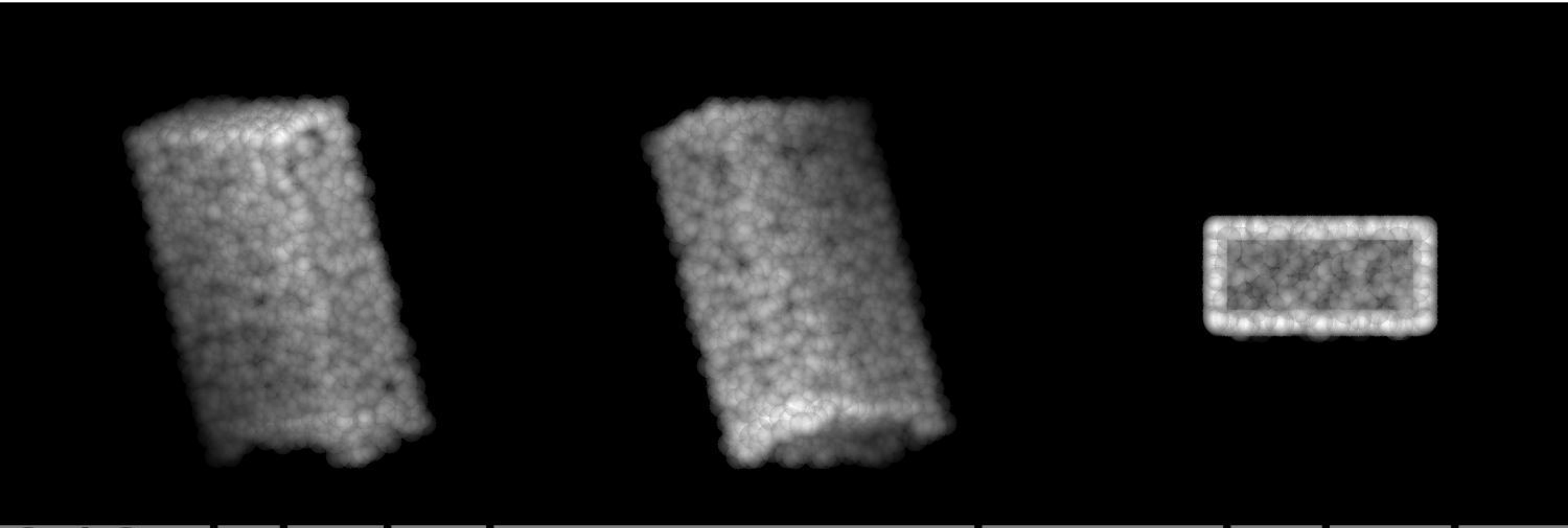
240_label_night_stand_pred_dresser.jpg



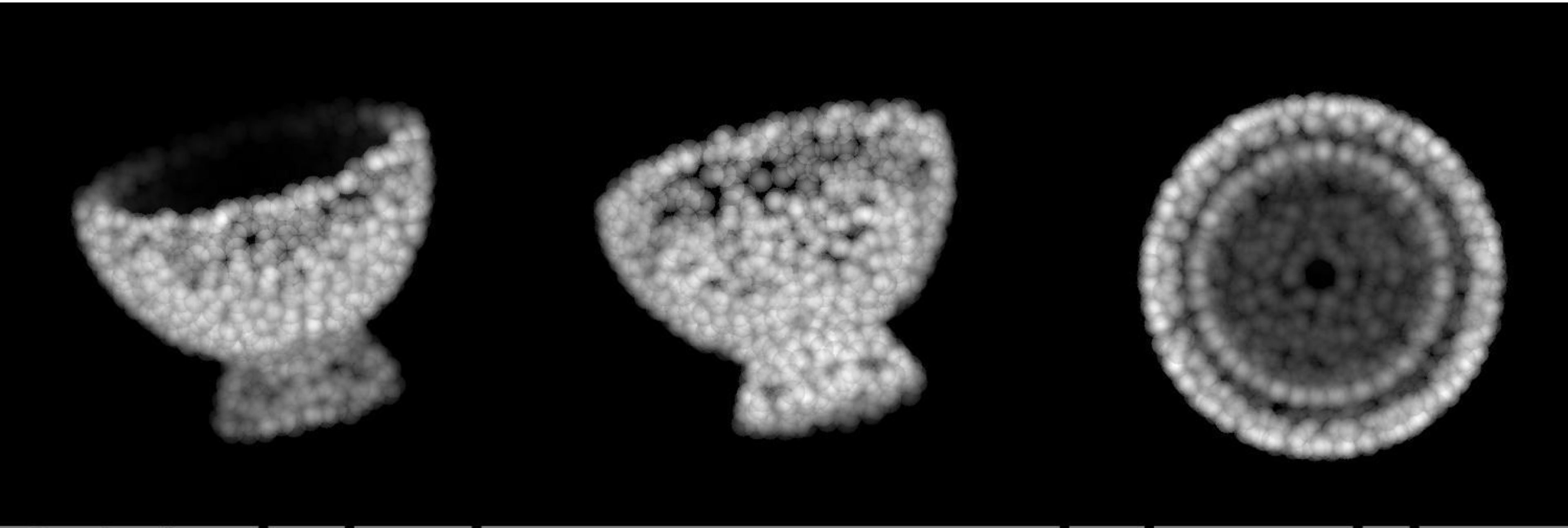
241_label_bottle_pred_vase.jpg



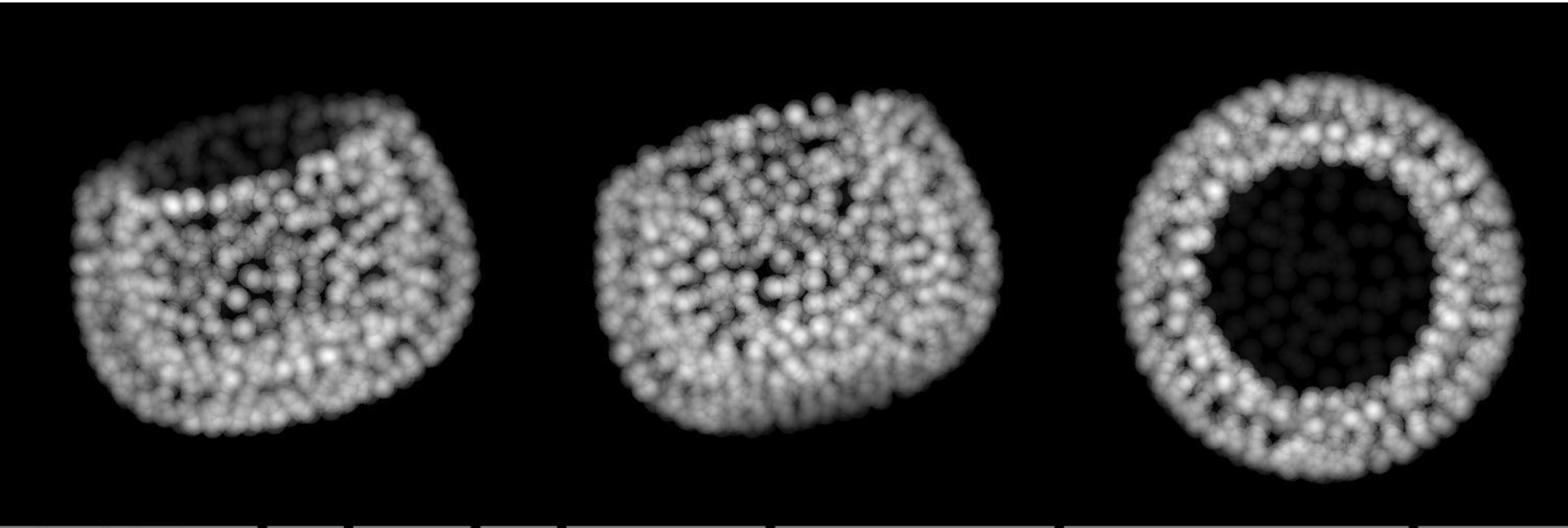
242_label_cup_pred_vase.jpg



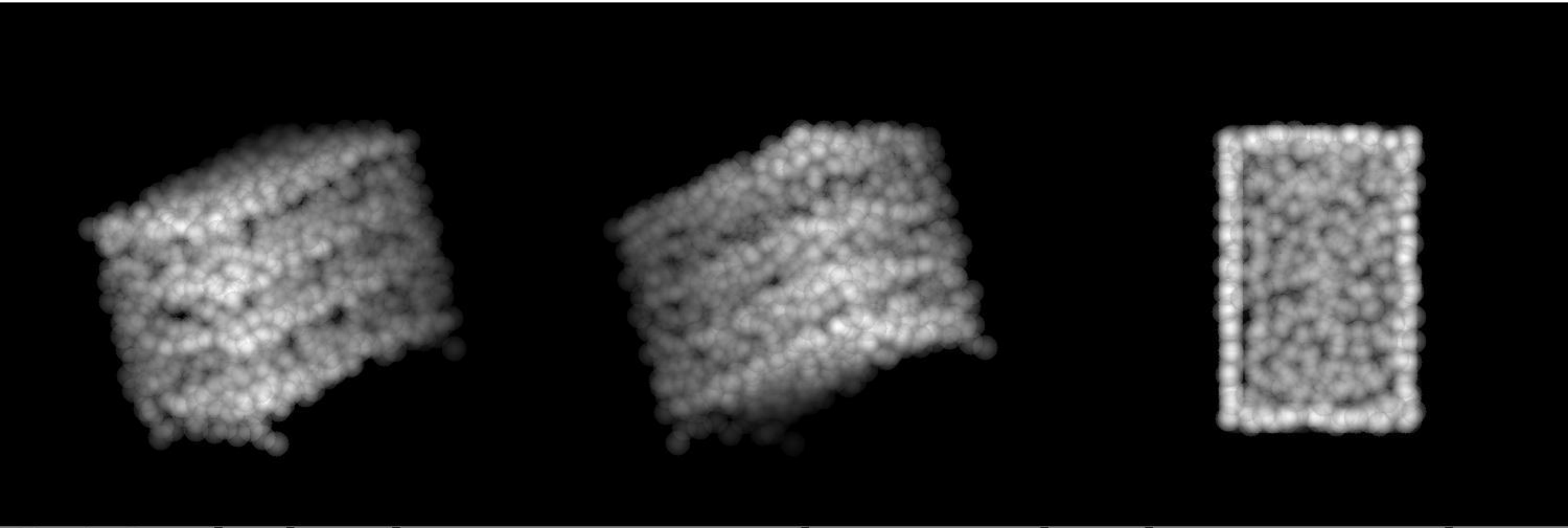
243_label_dresser_pred_wardrobe.jpg



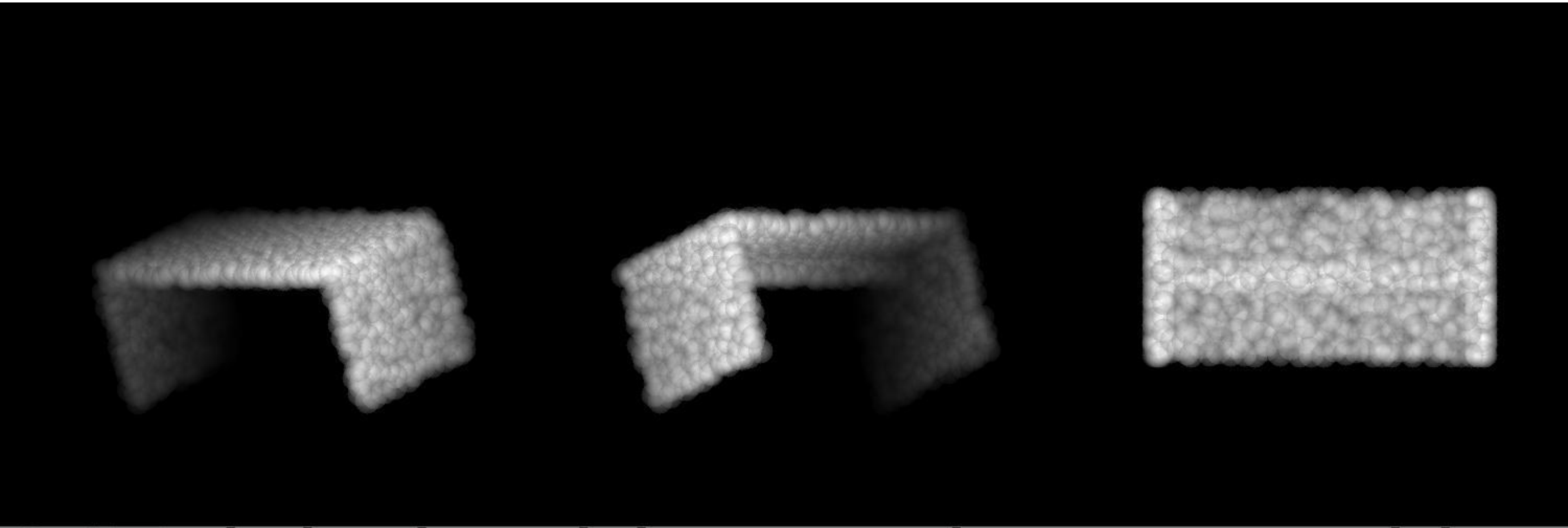
244_label_vase_pred_bowl.jpg



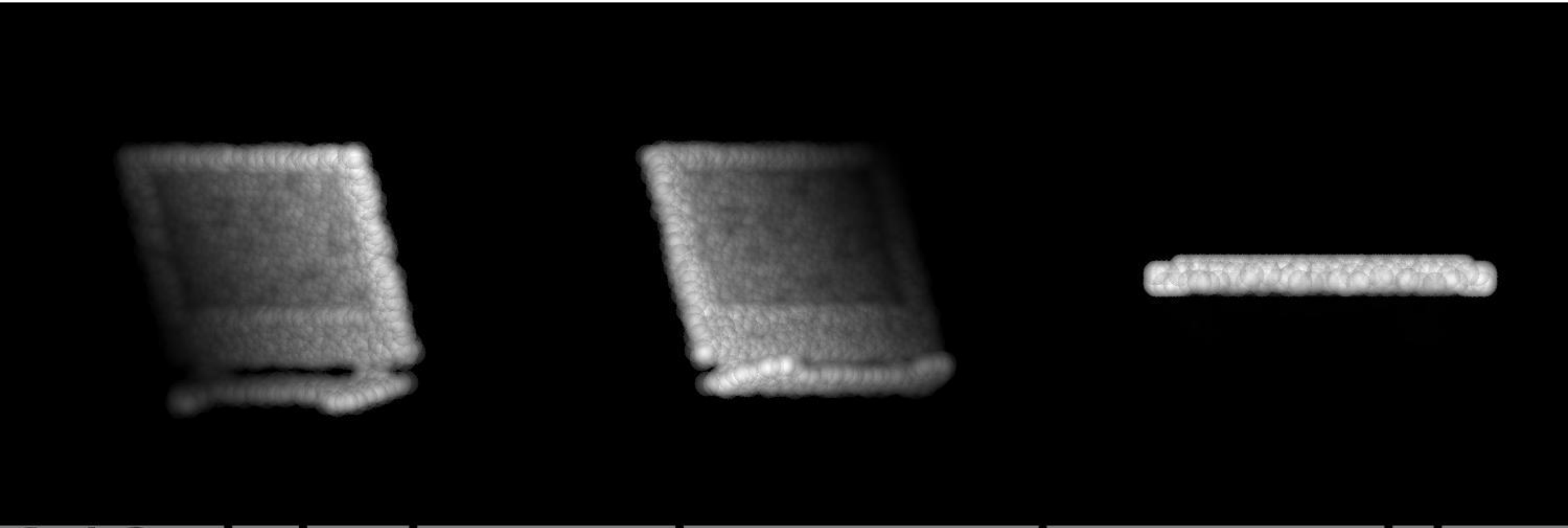
245_label_bowl_pred_vase.jpg



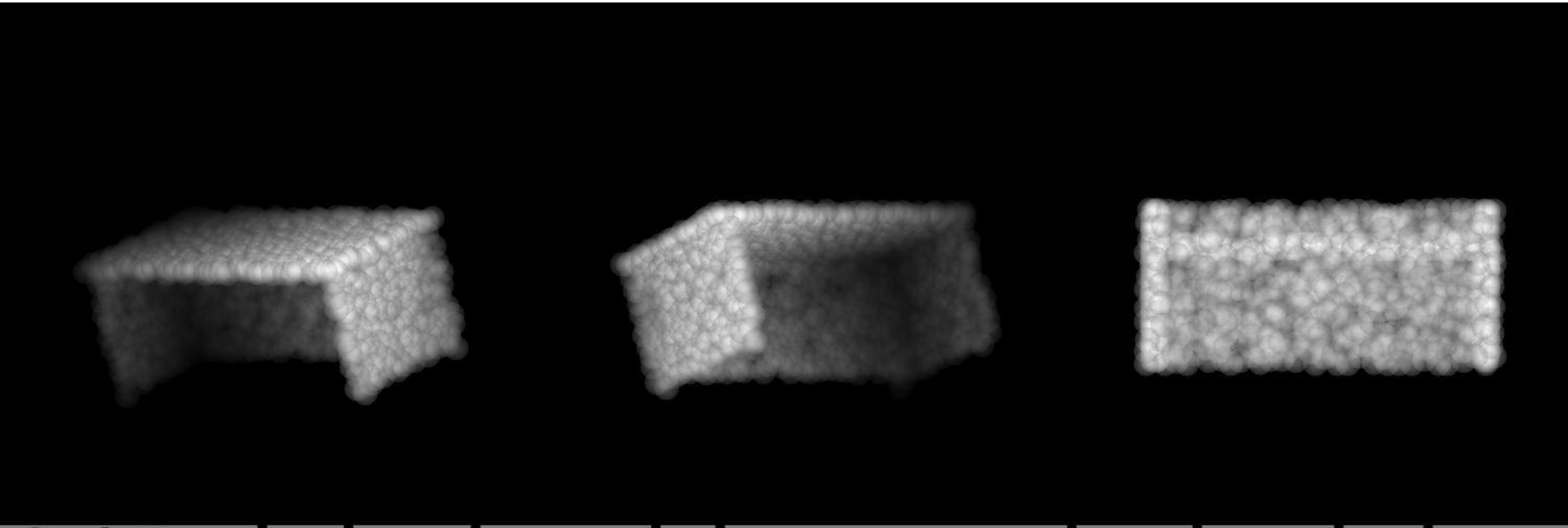
246_label_tv_stand_pred_dresser.jpg

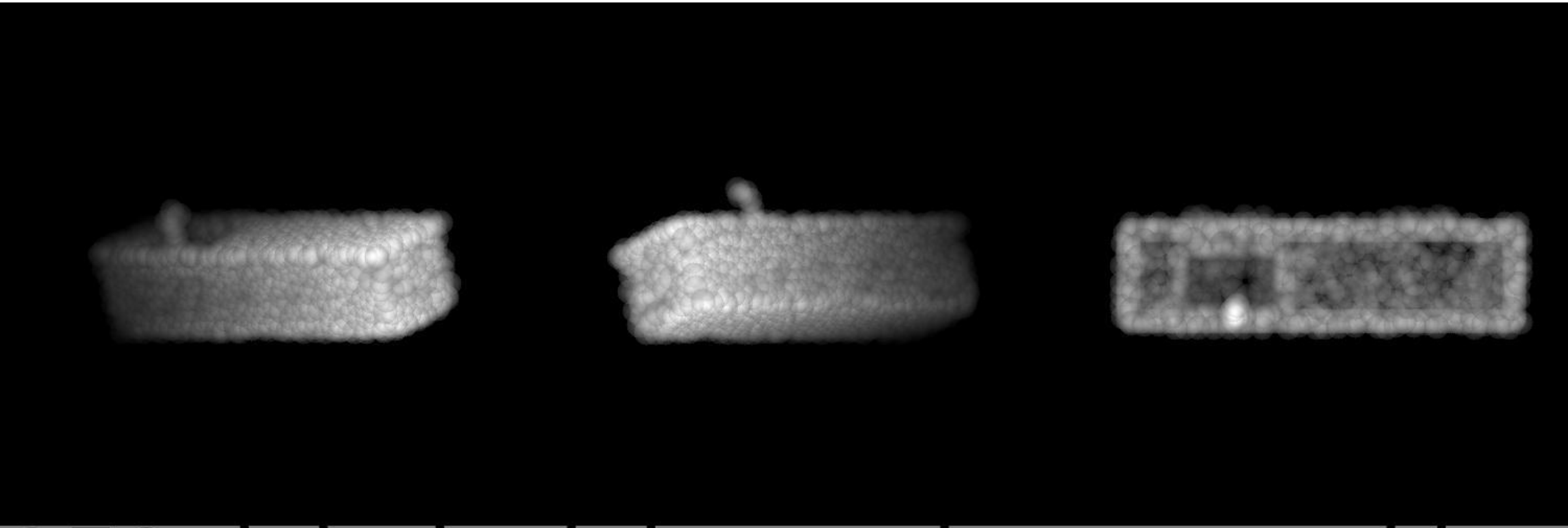


247_label_table_pred_tv_stand.jpg

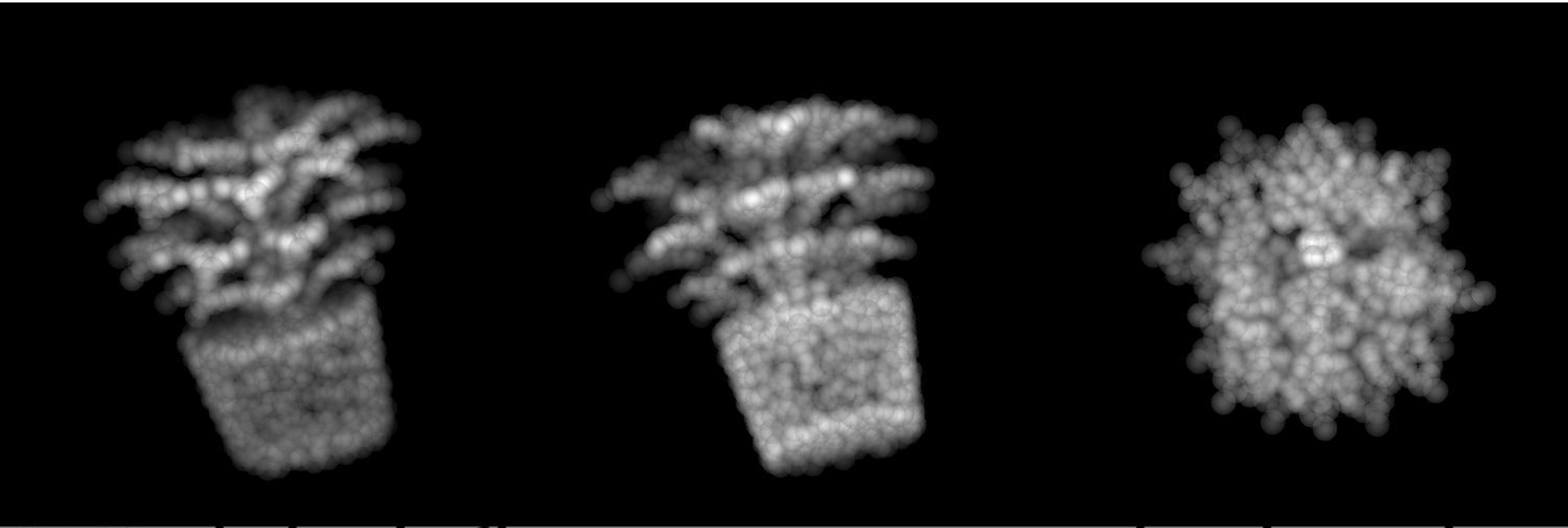


248_label_monitor_pred_mantel.jpg

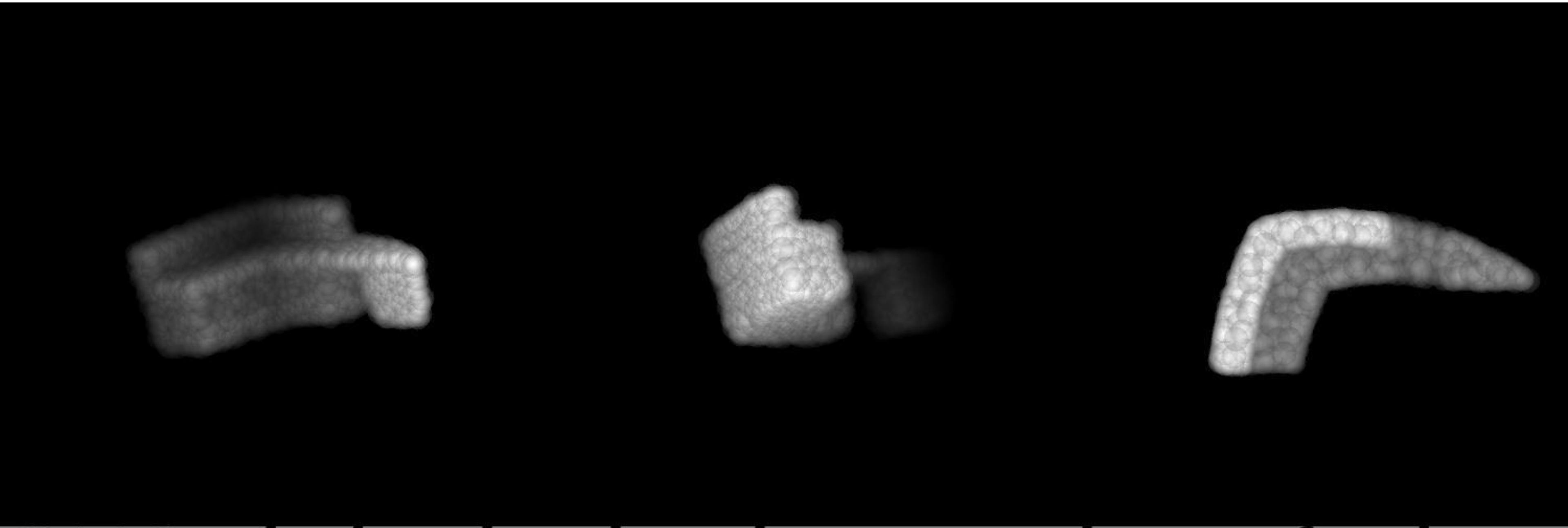

249_label_table_pred_desk.jpg



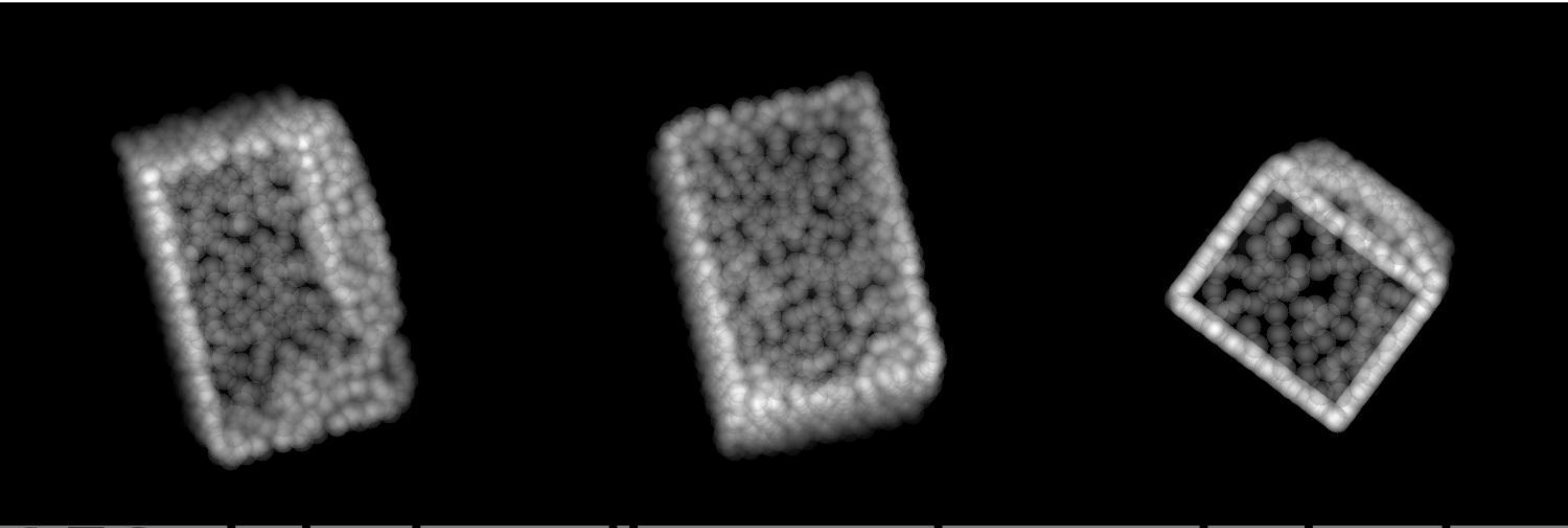
250_label_sink_pred_tv_stand.jpg



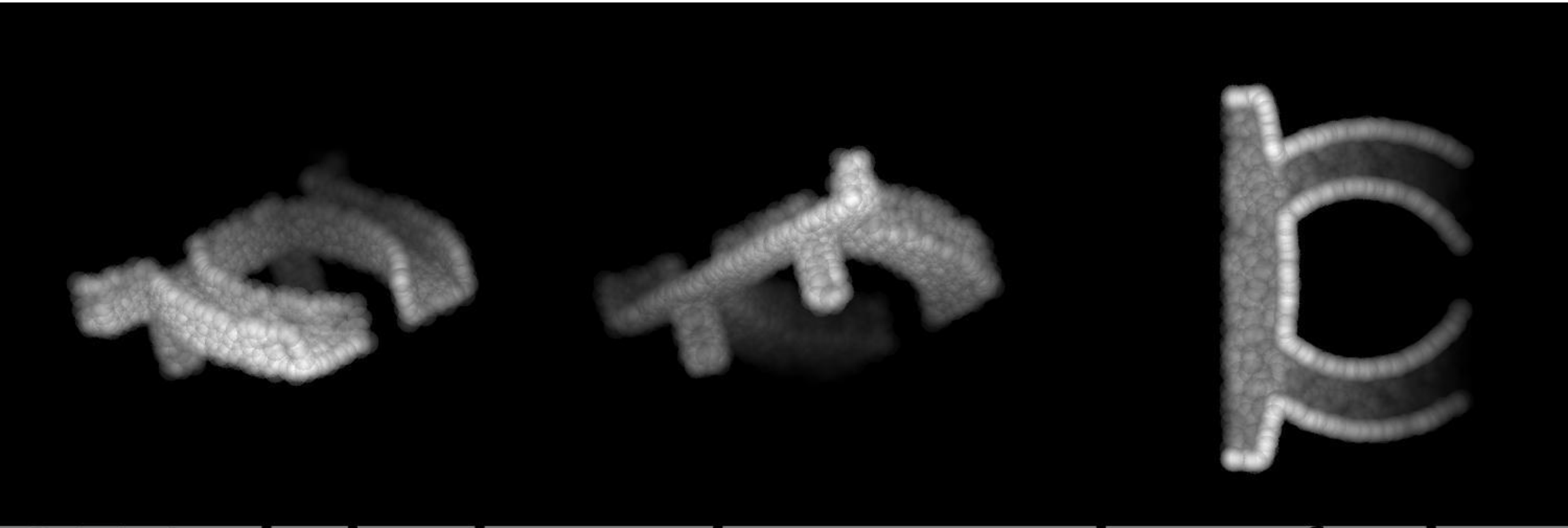
251_label_flower_pot_pred_plant.jpg



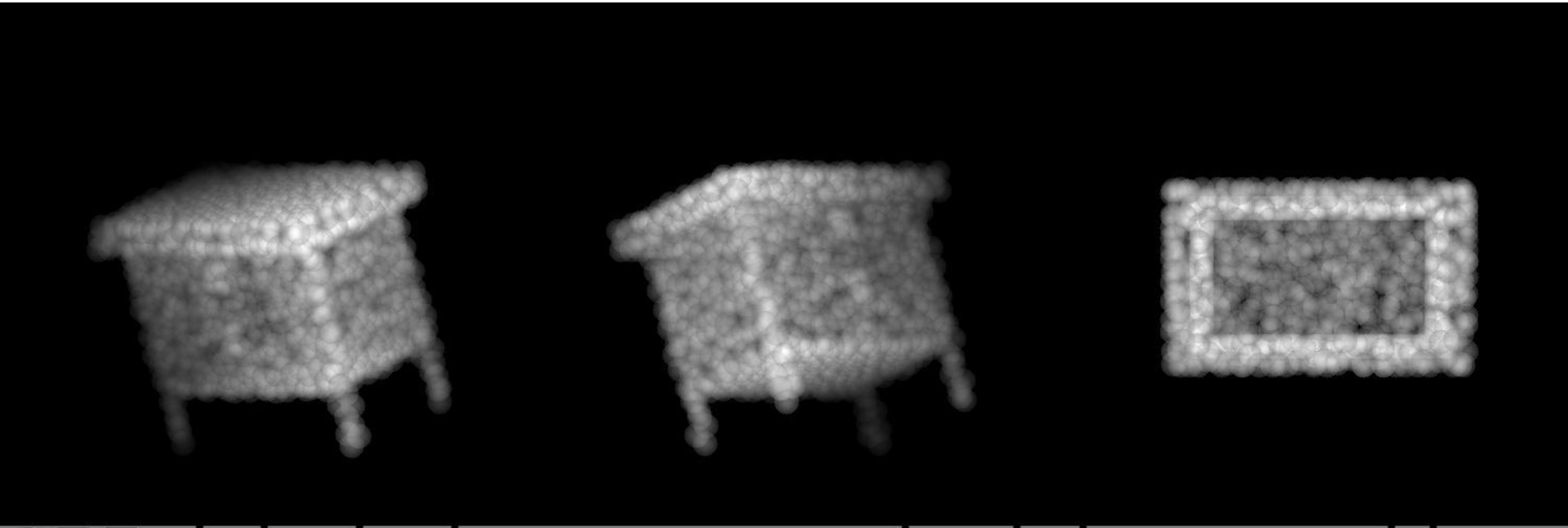
252_label_desk_pred_sofa.jpg



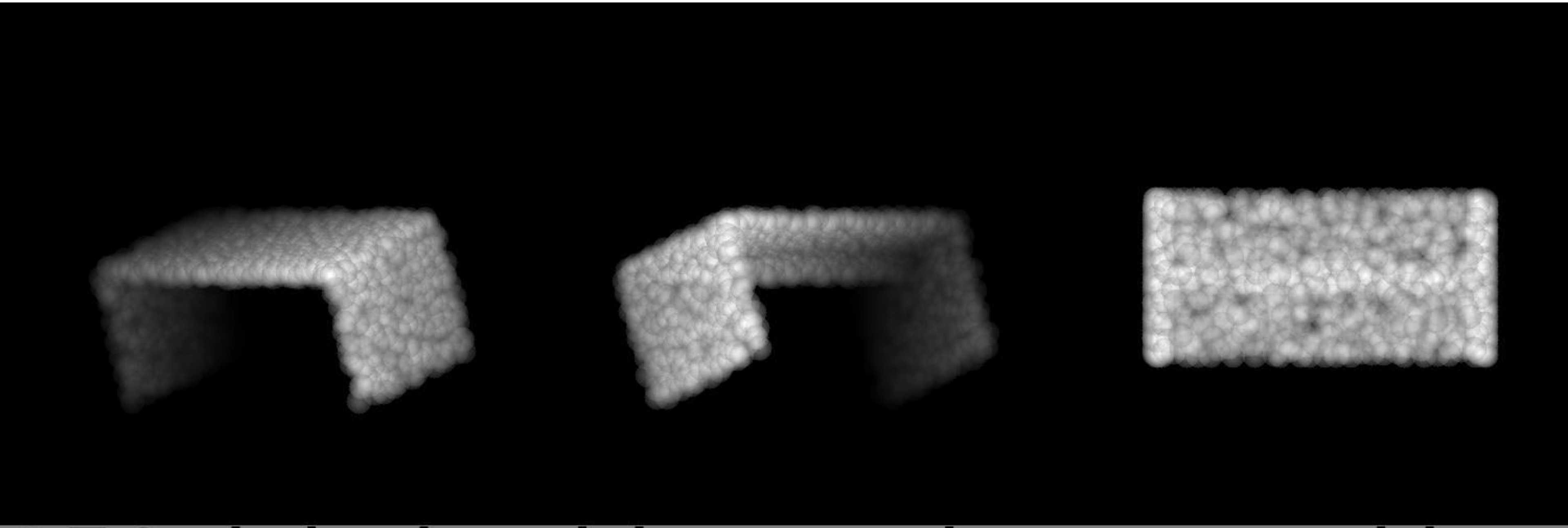
253_label_radio_pred_wardrobe.jpg



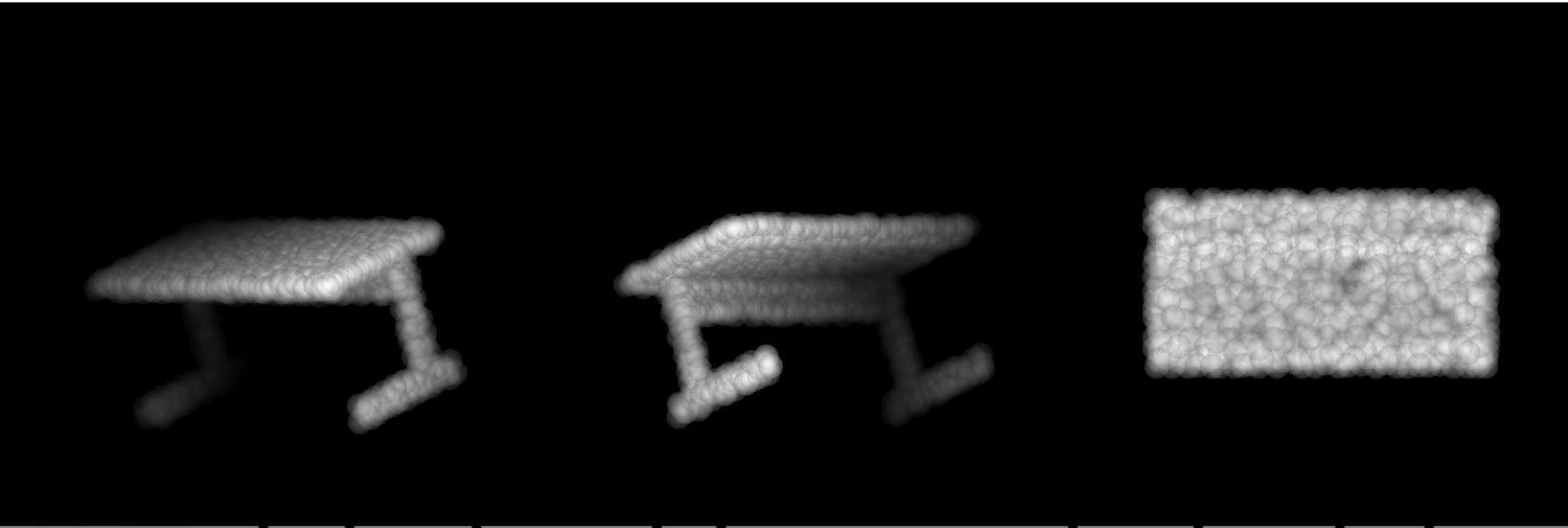
254_label_stairs_pred_sofa.jpg

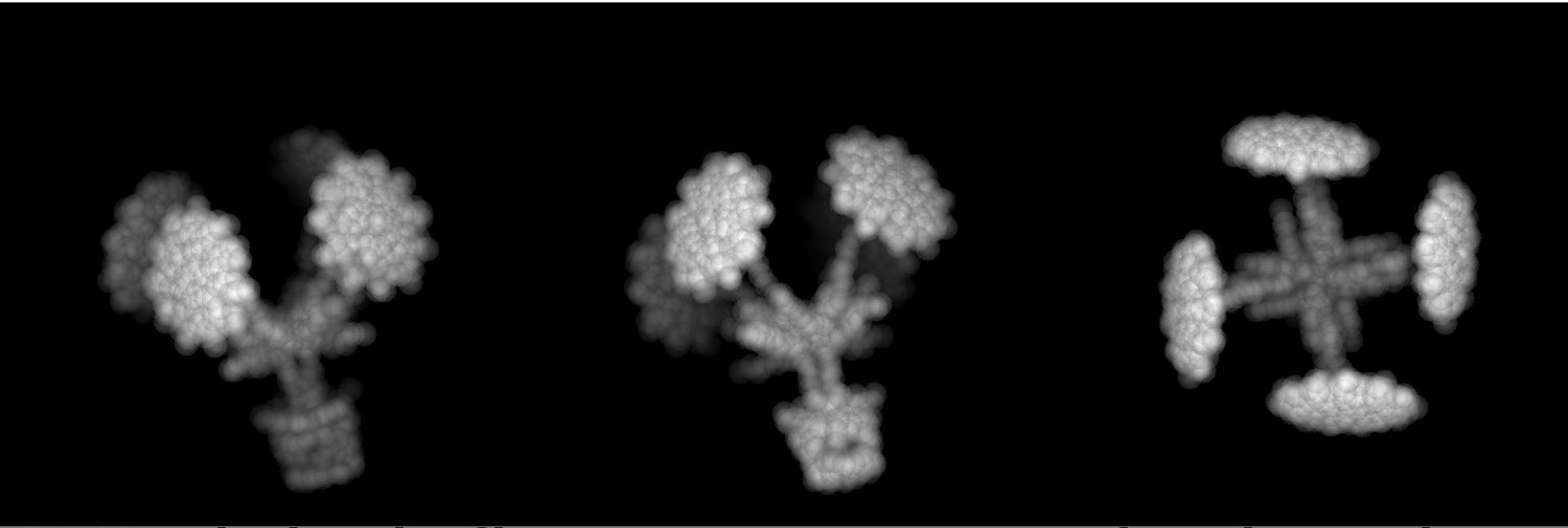


255_label_dresser_pred_night_stand.jpg

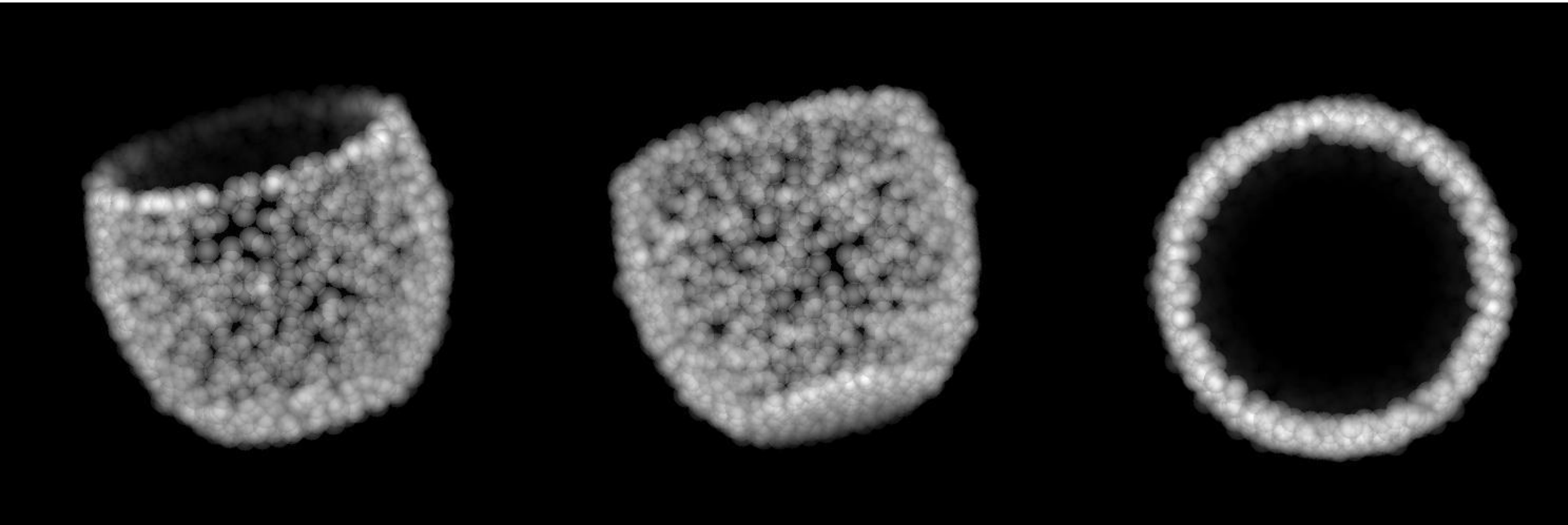


256_label_table_pred_tv_stand.jpg

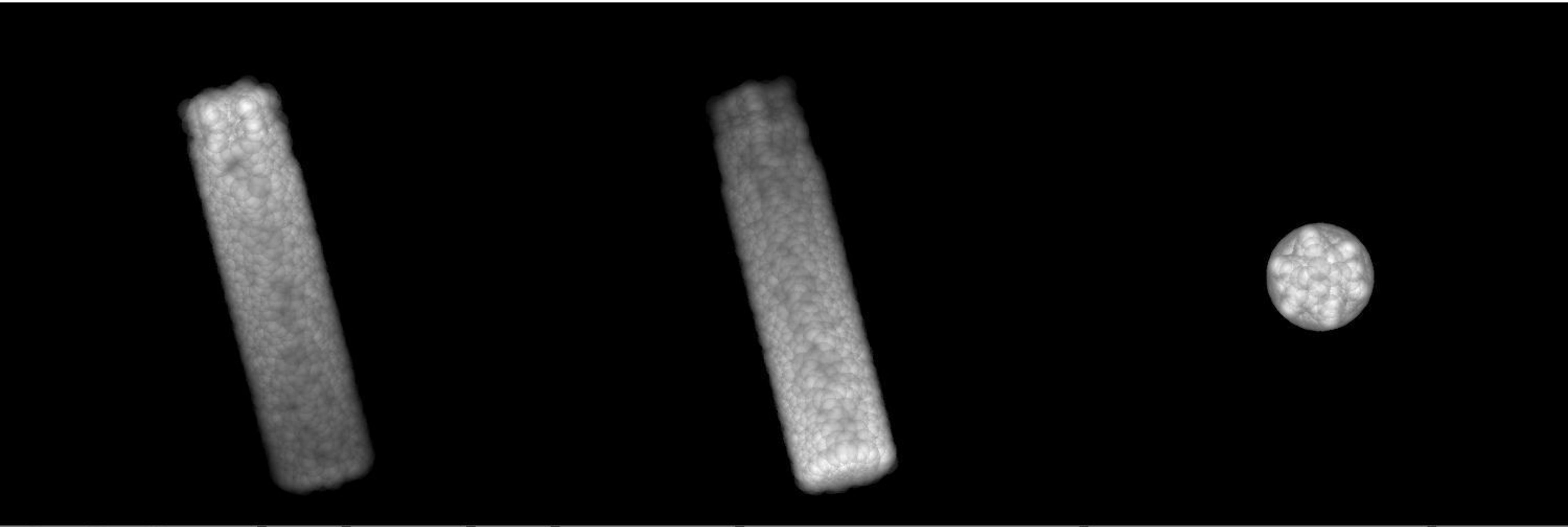

257_label_table_pred_desk.jpg



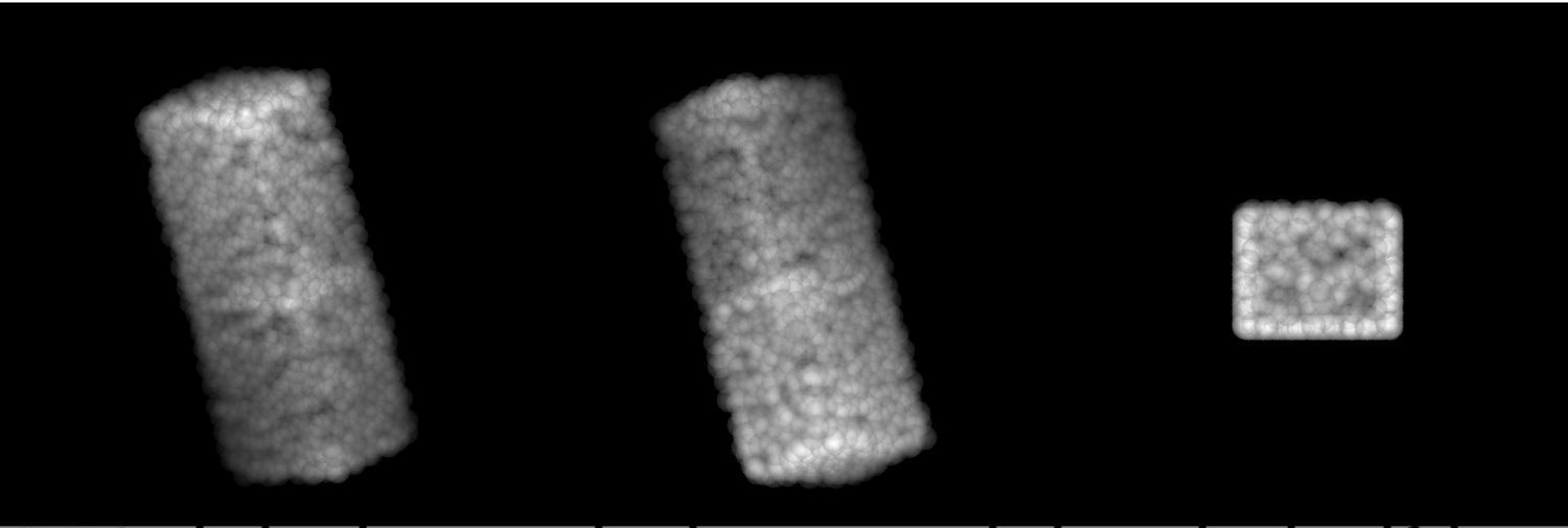
258_label_flower_pot_pred_plant.jpg



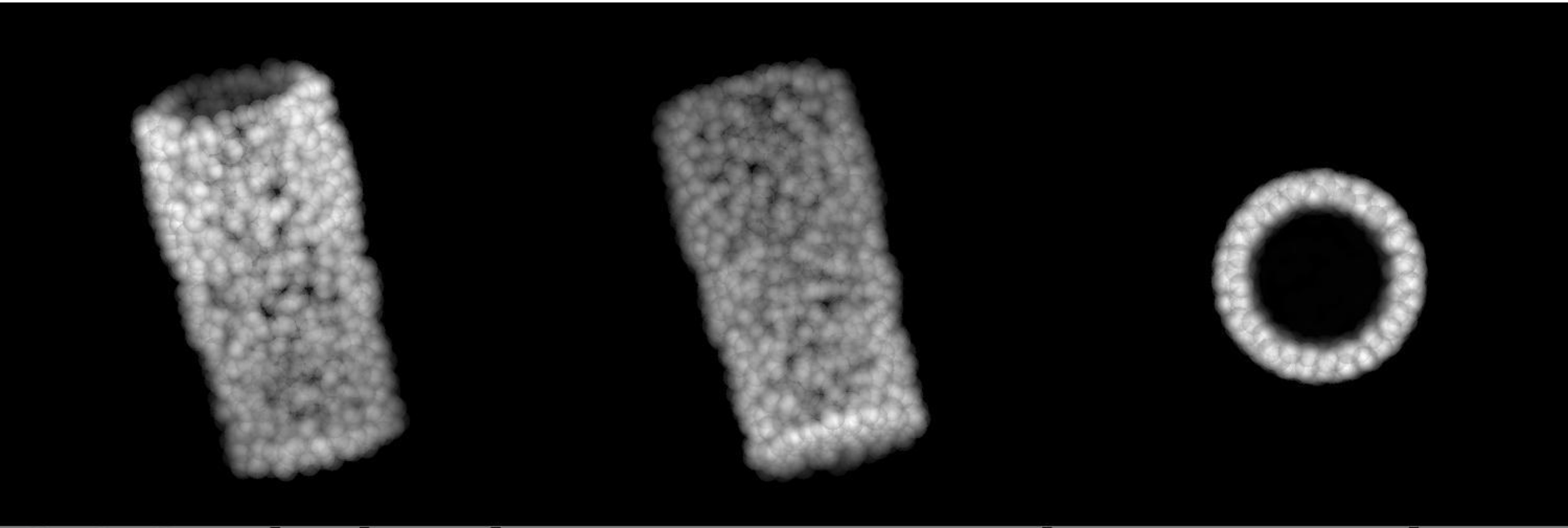
259_label_flower_pot_pred_vase.jpg



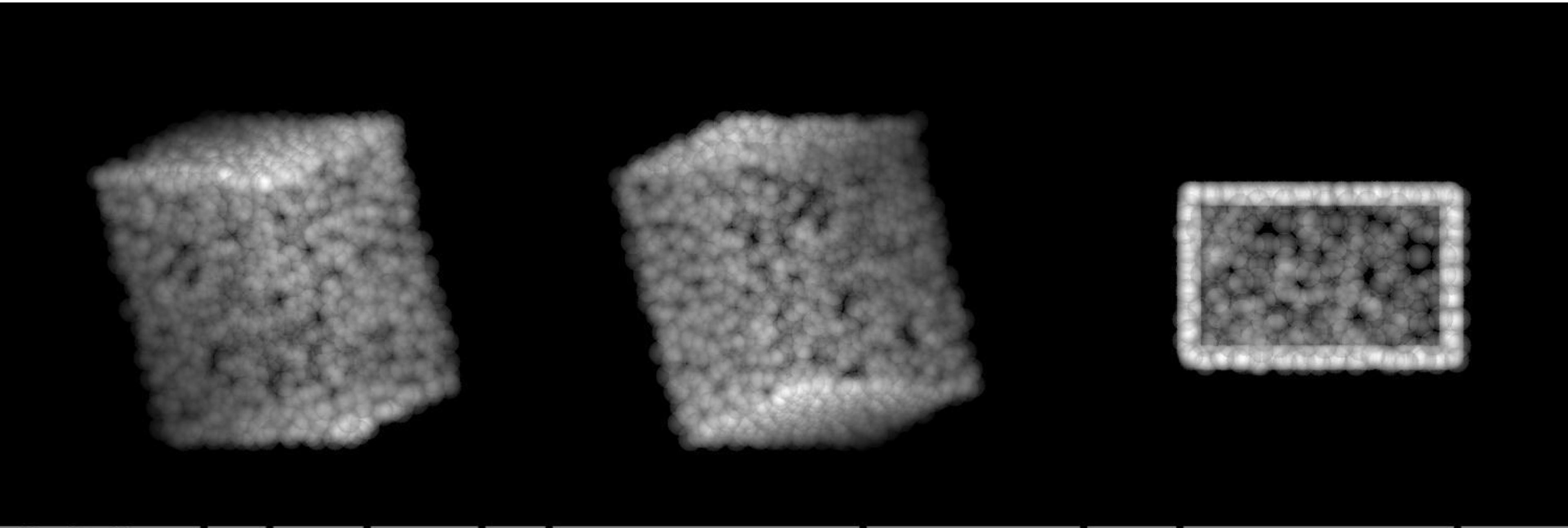
260_label_bottle_pred_vase.jpg



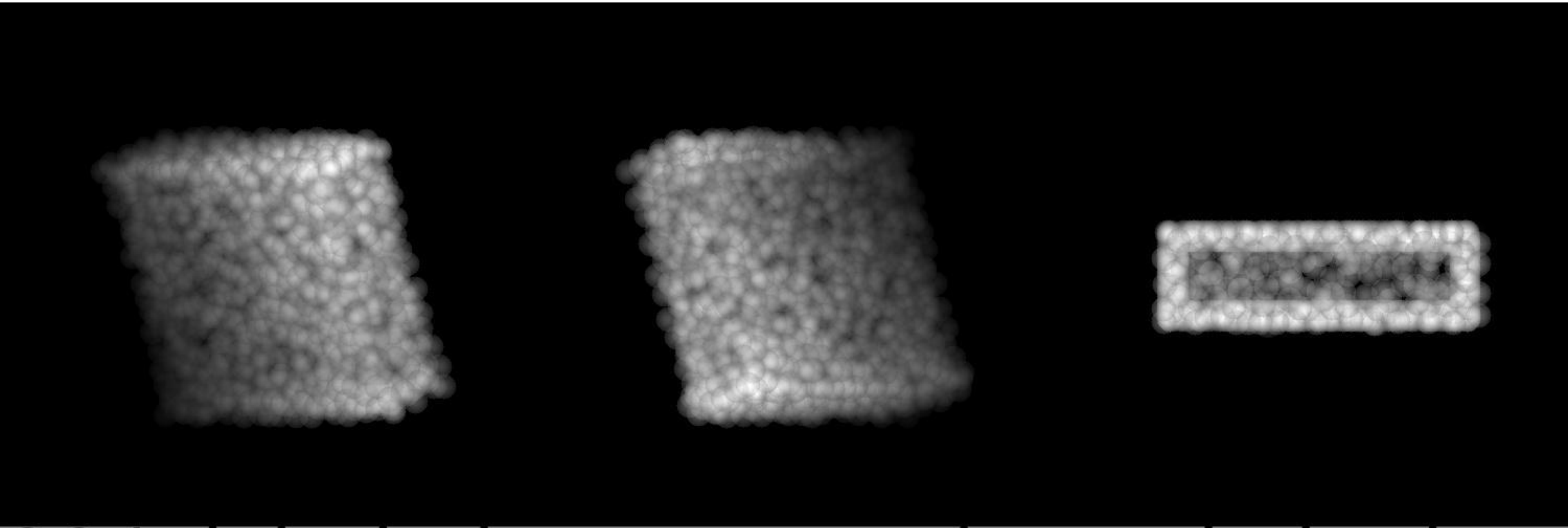
261_label_wardrobe_pred_bookshelf.jpg



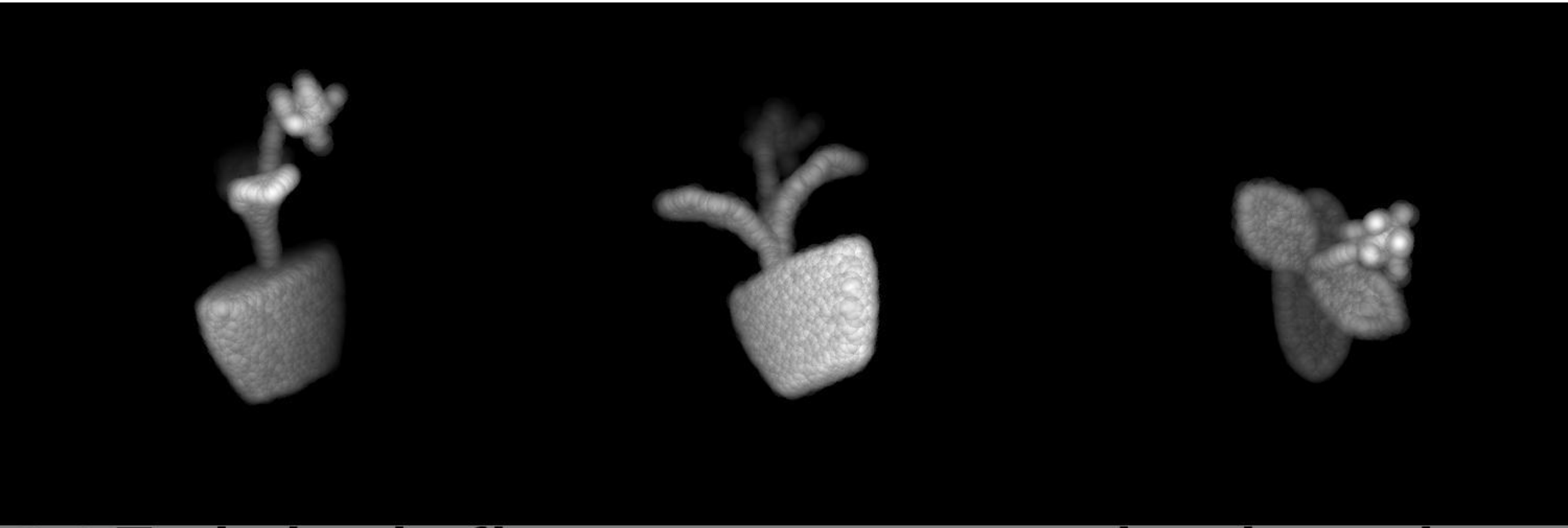
262_label_cup_pred_vase.jpg



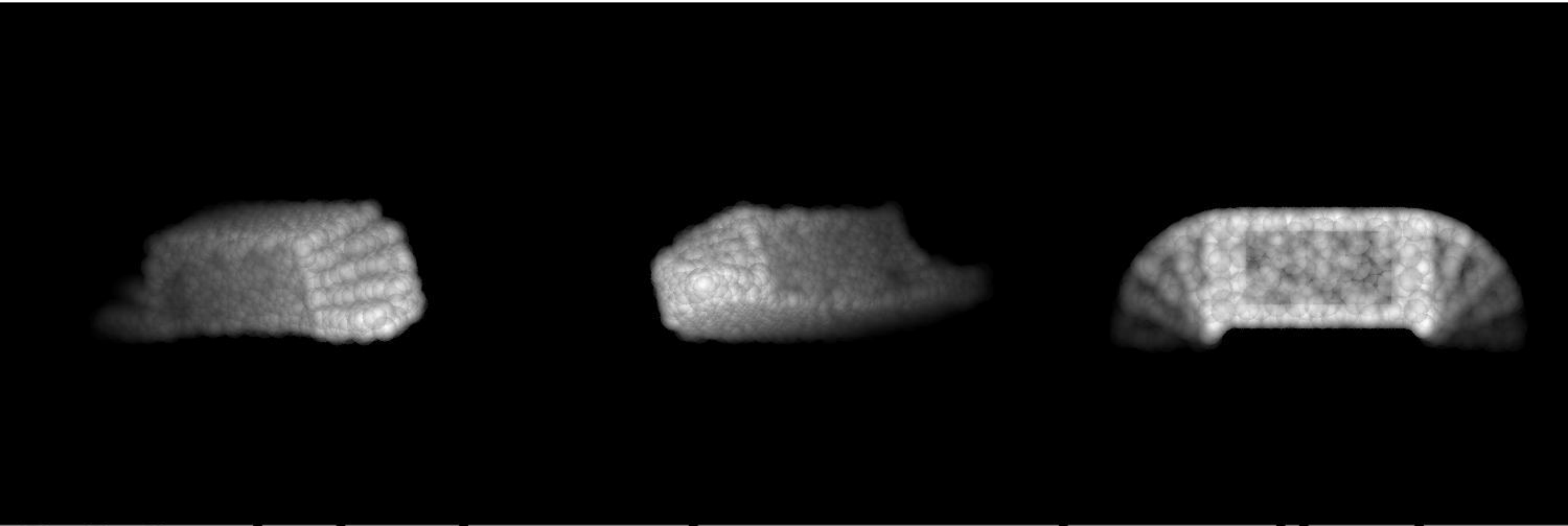
263_label_night_stand_pred_dresser.jpg



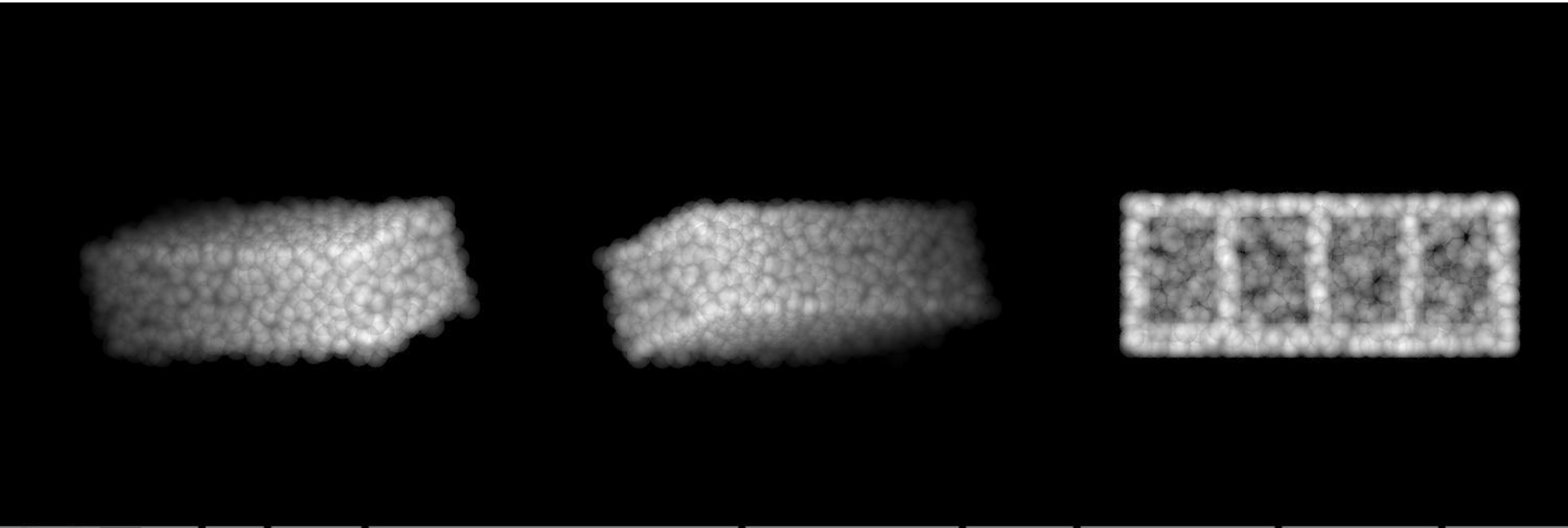
264_label_dresser_pred_wardrobe.jpg



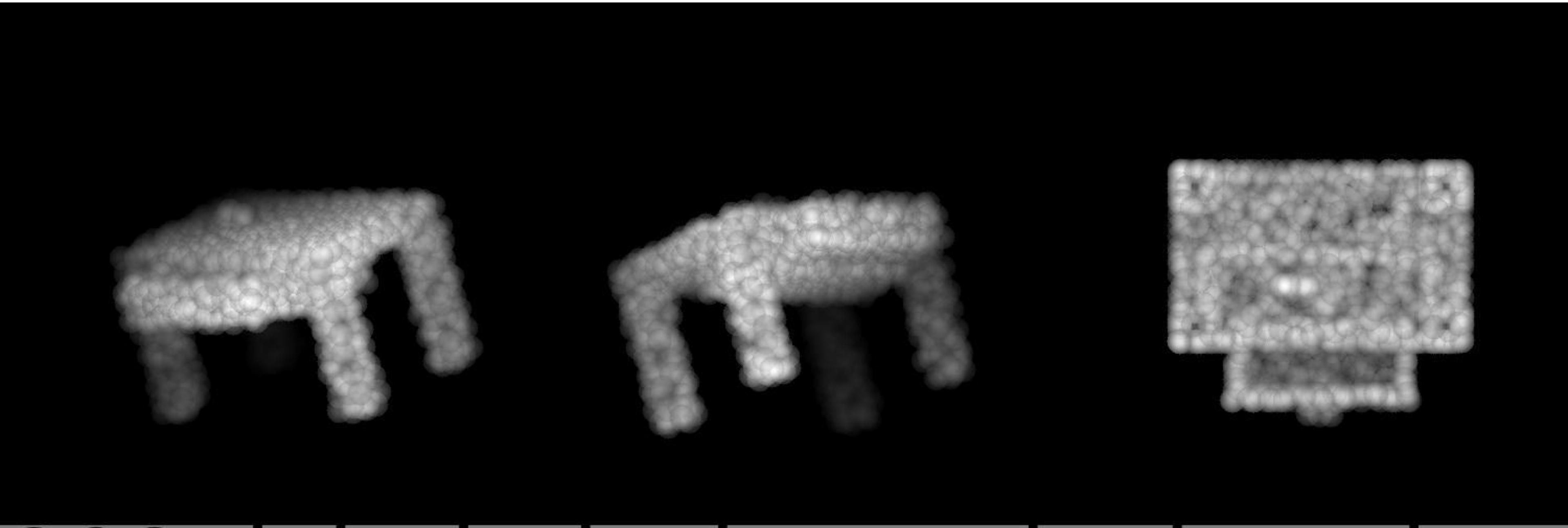
265_label_flower_pot_pred_plant.jpg

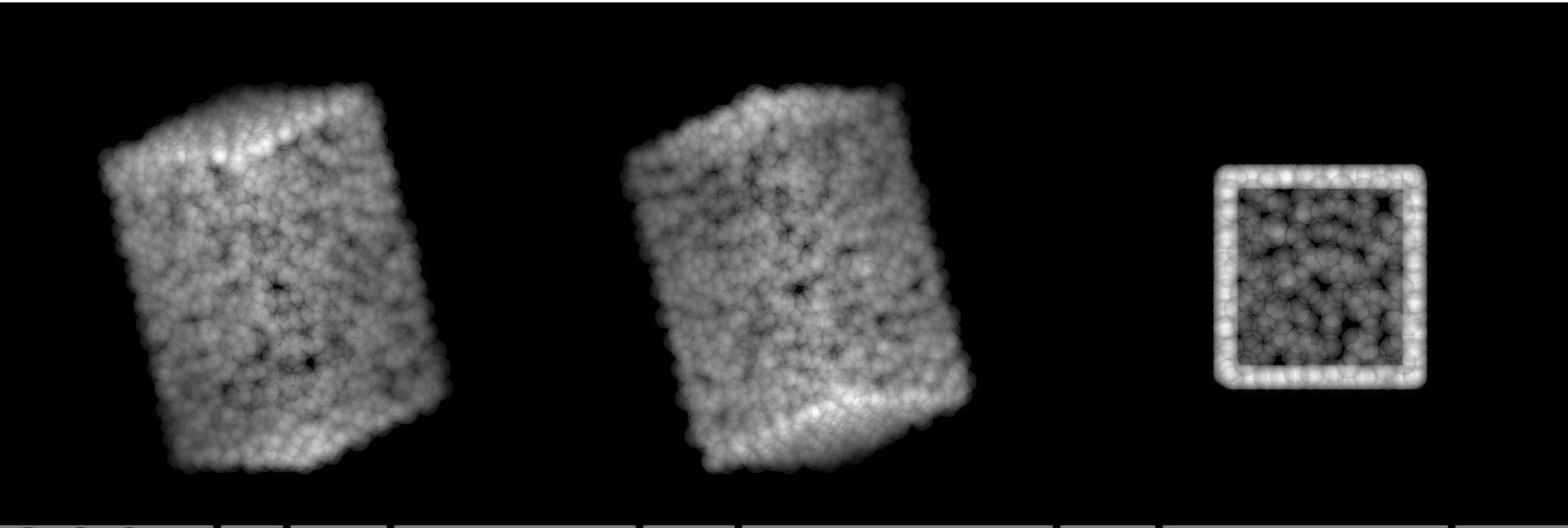


266_label_stairs_pred_radio.jpg

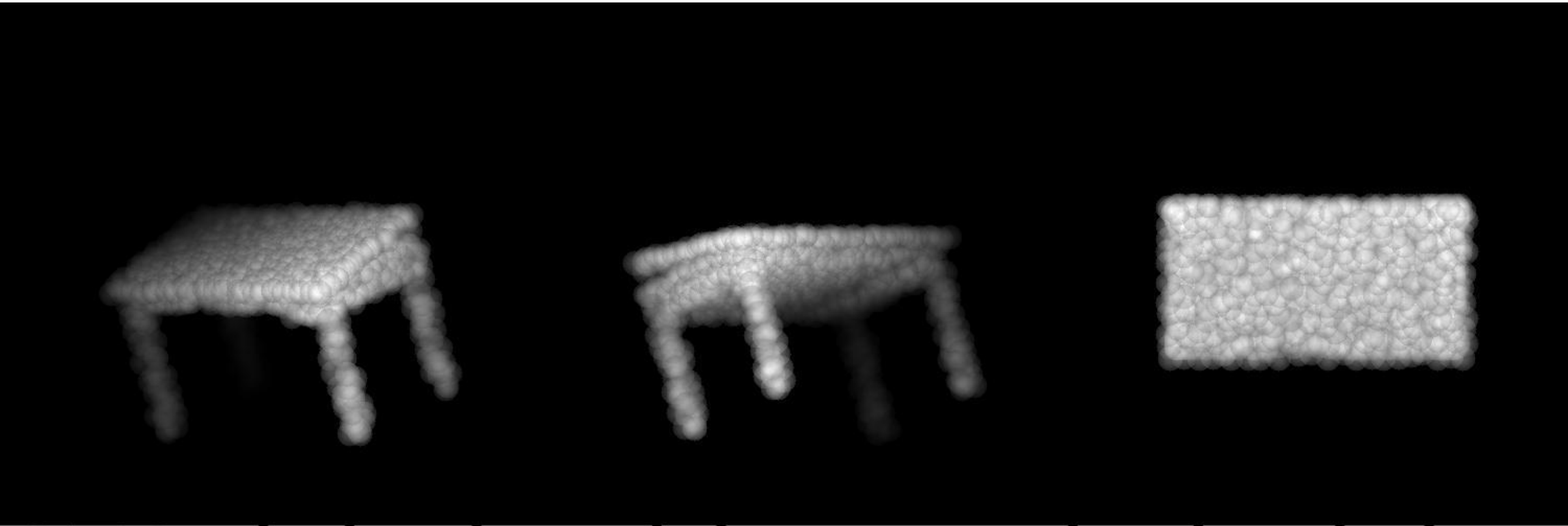


267_label_tv_stand_pred_glass_box.jpg

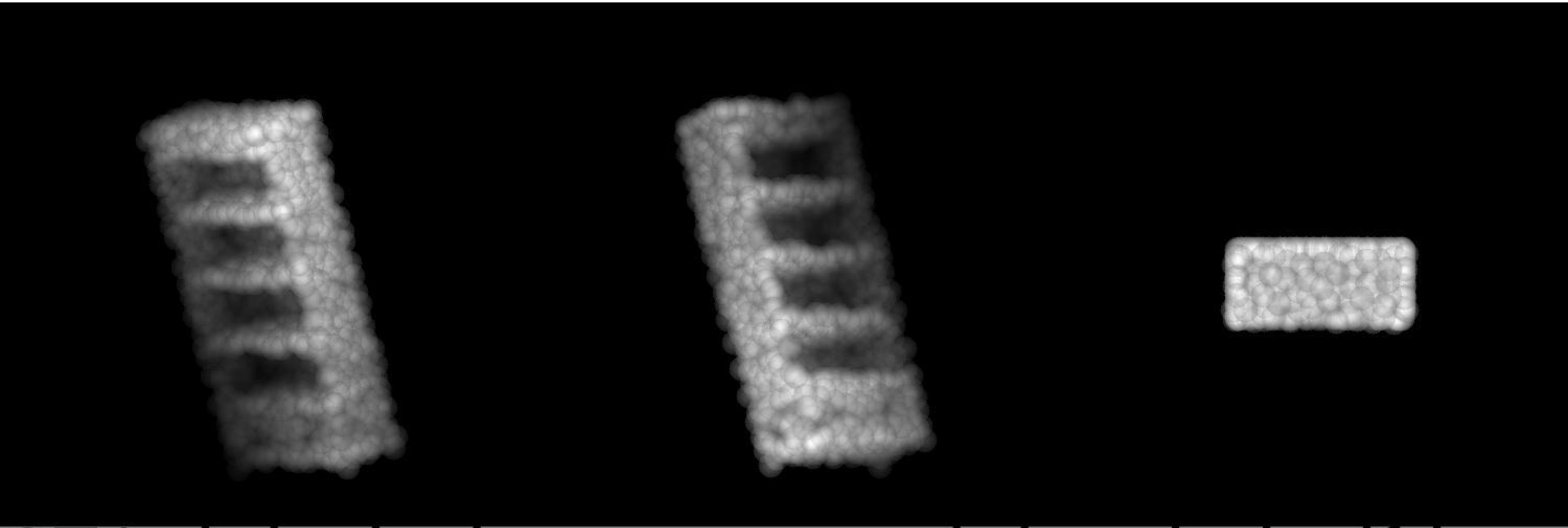

268_label_desk_pred_piano.jpg



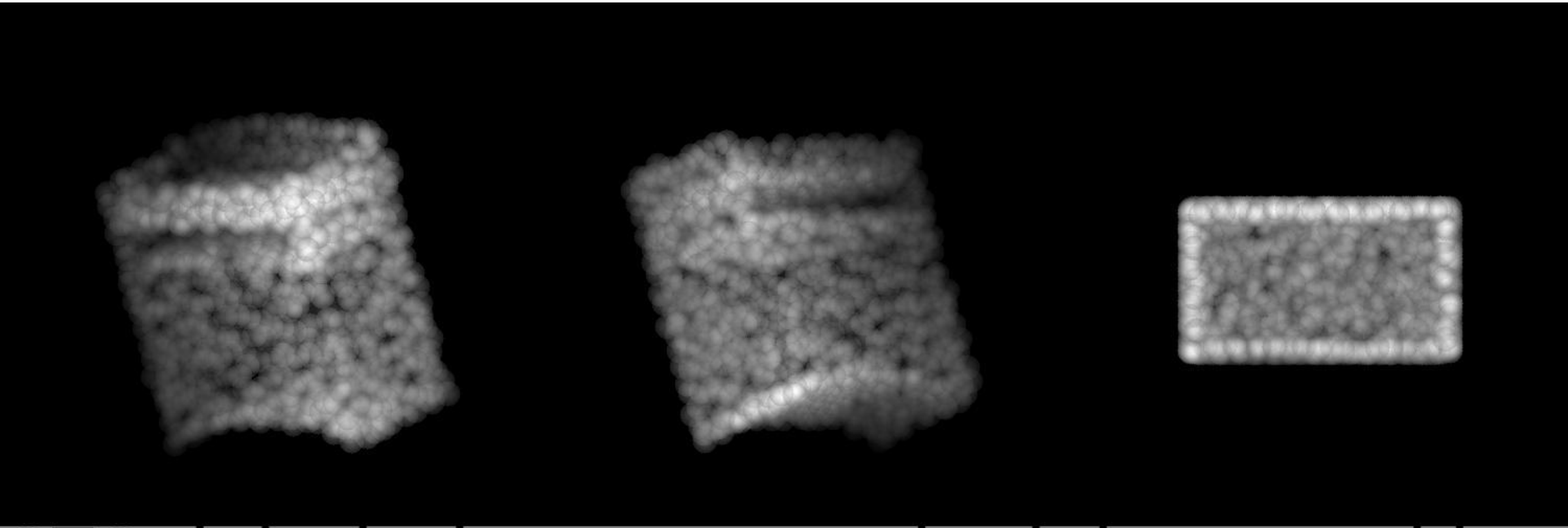
269_label_wardrobe_pred_dresser.jpg



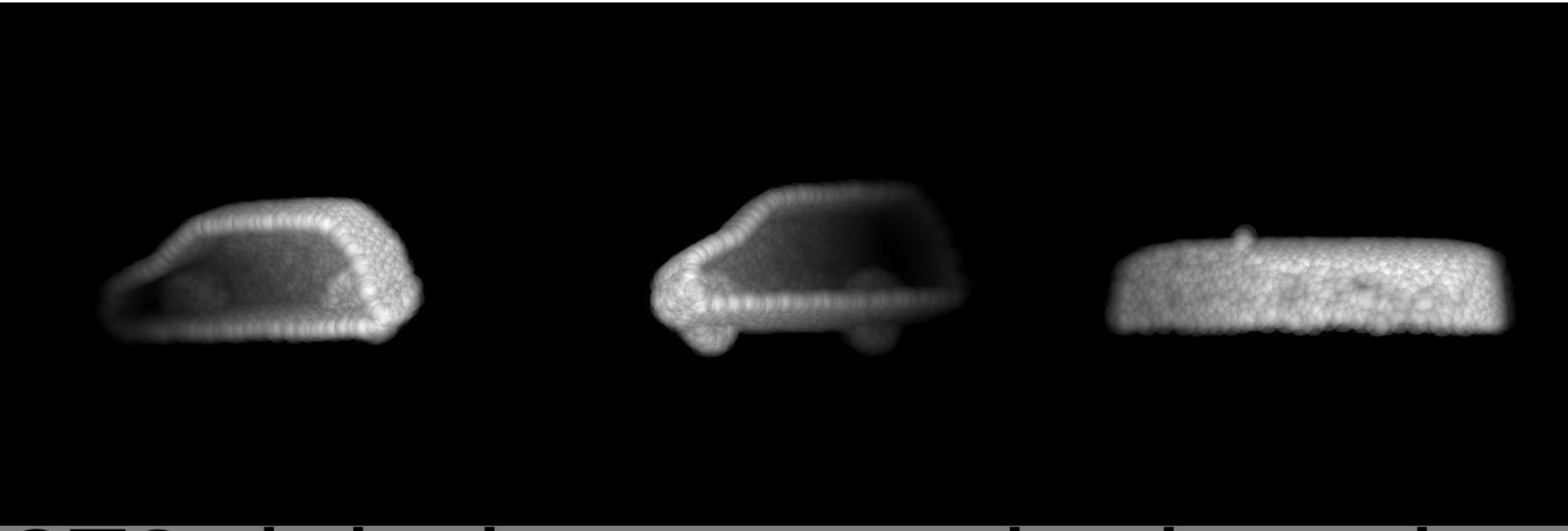
270_label_table_pred_desk.jpg



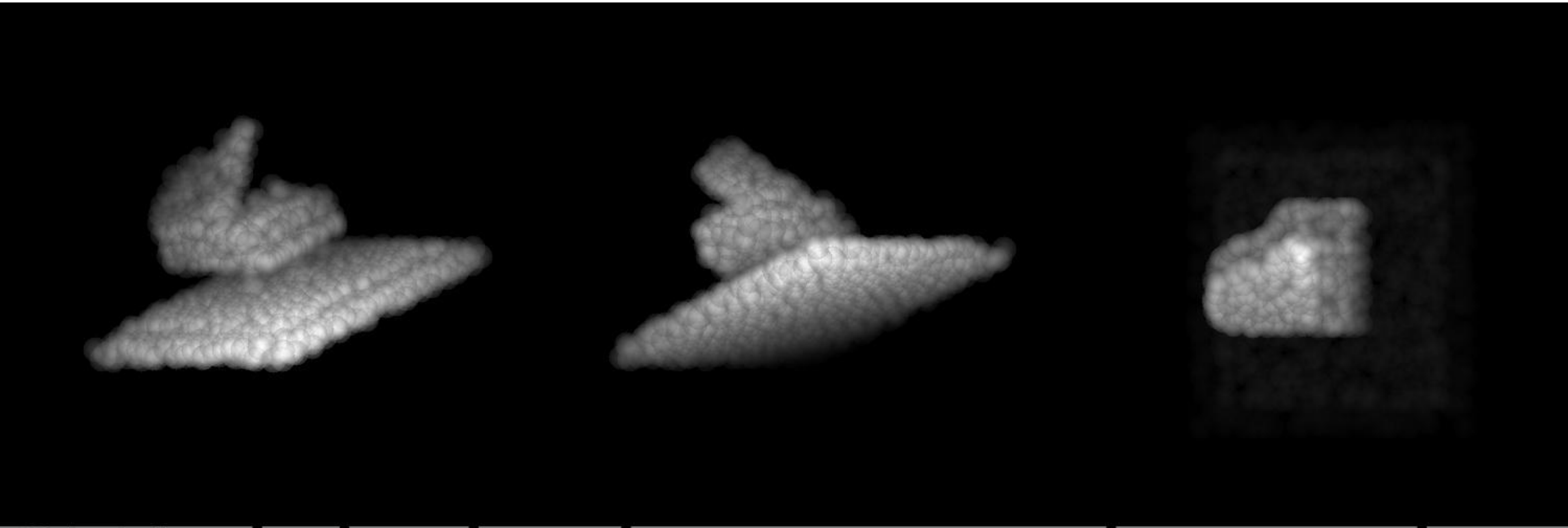
271_label_dresser_pred_bookshelf.jpg



272_label_dresser_pred_night_stand.jpg



273_label_car_pred_piano.jpg



274_label_piano_pred_tent.jpg