Supplementary Material

A. GRNNs implementation details

The input images, output images, and predictions (for view prediction) all have size 64×64. Our pre-unprojection 2D encoder-decoder network has encoder layers with 32, 64, 128, and 256 channels, respectively. The decoder layers are symmetric to the encoder layers. The sizes of these feature maps are 32×32, 16×16, 8×8 and 4×4 respectively, since each convolution has stride 2. For depth prediction, which is used for object detection, we use the same 2D encoder-decoder network as described in this pre-unprojection step.

View prediction We feed our feature tensor through a 3D encoder-decoder with skip-connections where the encoder has 64, 128, 256, 512, and 1024 channels respectively. The decoder is symmetric to the encoder. In our convLSTM decoder, we do not share weights across time steps. Indeed, allowing each step to have different weights was noted by [6] to improve performance. We use six generation steps, and we did not see noticeable improvement on image reconstruction quality when more steps were used. We used 256 channels at each step. Both our model and the baseline are deterministic networks, we did not include any stochastic units in any of the two to accelerate training. We trained each model for 24 hours, using a batch size of 8 and a learning rate of 10^{-4}. This resulted in roughly $800 \cdot 10^3$ steps of backpropagation for the tower (baseline) architecture and $160 \cdot 10^3$ steps for the GRNN architecture. We used the Adam optimizer.

Object detection/segmentation For object detection and ego-motion prediction, we feed the unprojected feature tensor through a 3D encoder-decoder with skip-connections where the encoder has 16, 32, 64, 128 channels receptively, and the decoder is again symmetric to the encoder. For ego-motion prediction, when we compare the current memory with the rotated feature tensors generated from the new view, we use outputs from all the layers in the decoder (feature tensors with size $4 \times 4 \times 4 \times 128$, $8 \times 8 \times 8 \times 64$, $16 \times 16 \times 16 \times 32$, $32 \times 32 \times 32 \times 16$) to compute cross-convolutions. For object detection, we use only the last feature tensor from the 3D encoder-decoder as input and pass it to another 3D encoder-decoder with skip-connections to predict positive anchor centroids and their corresponding adjustments for the box centers. The channels in the second 3D encoder-decoder are set to 16, 32, 64, 128 and the corresponding final output is a $32 \times 32 \times 32 \times 16$ feature tensor. We then pass this feature tensor to one more 3D convolutional layer with kernel size 3, stride 1 and channel size 7. The final prediction is a $32 \times 32 \times 32 \times 7$ tensor with the first channel indicating positive anchor centroids and the last 6 channels indicating 3D boxes adjustments at each centroid. The model is trained with Adam optimizer (learning rate is set to 10^{-4}) without further parameter tuning. We train the model for roughly 25K iterations with batch size 2. We first train for egomotion prediction, and next jointly train for egomotion prediction and object detection and segmentation losses.

B. Additional results

In Figures 8, 9 we show view prediction results on the `rooms_ring_camera` and `shepard-metlzer` dataset introduced in [6]. Since the camera intrinsics were not given, we used an estimated vertical and horizontal field of view of 60 degrees. Our model outperforms the baseline by a margin. In Figures 10, 11, we show more view prediction results on the ShapeNet arrangement dataset of [3]. In Figure 12, we show qualitative 3D object detection and segmentation results.

In Figure 7 we show, by using a *single* frame as input to our model, our model can generate completely novel views: our model can extrapolate the missing parts of the objects, which geometric SLAM methods [30, 14] cannot do. Our model not only remembers things it has seen in previous frames, but also learns to predict *invisible* parts of the scene.

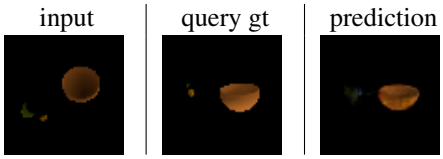

Figure 7. **View prediction** with a single image as input.

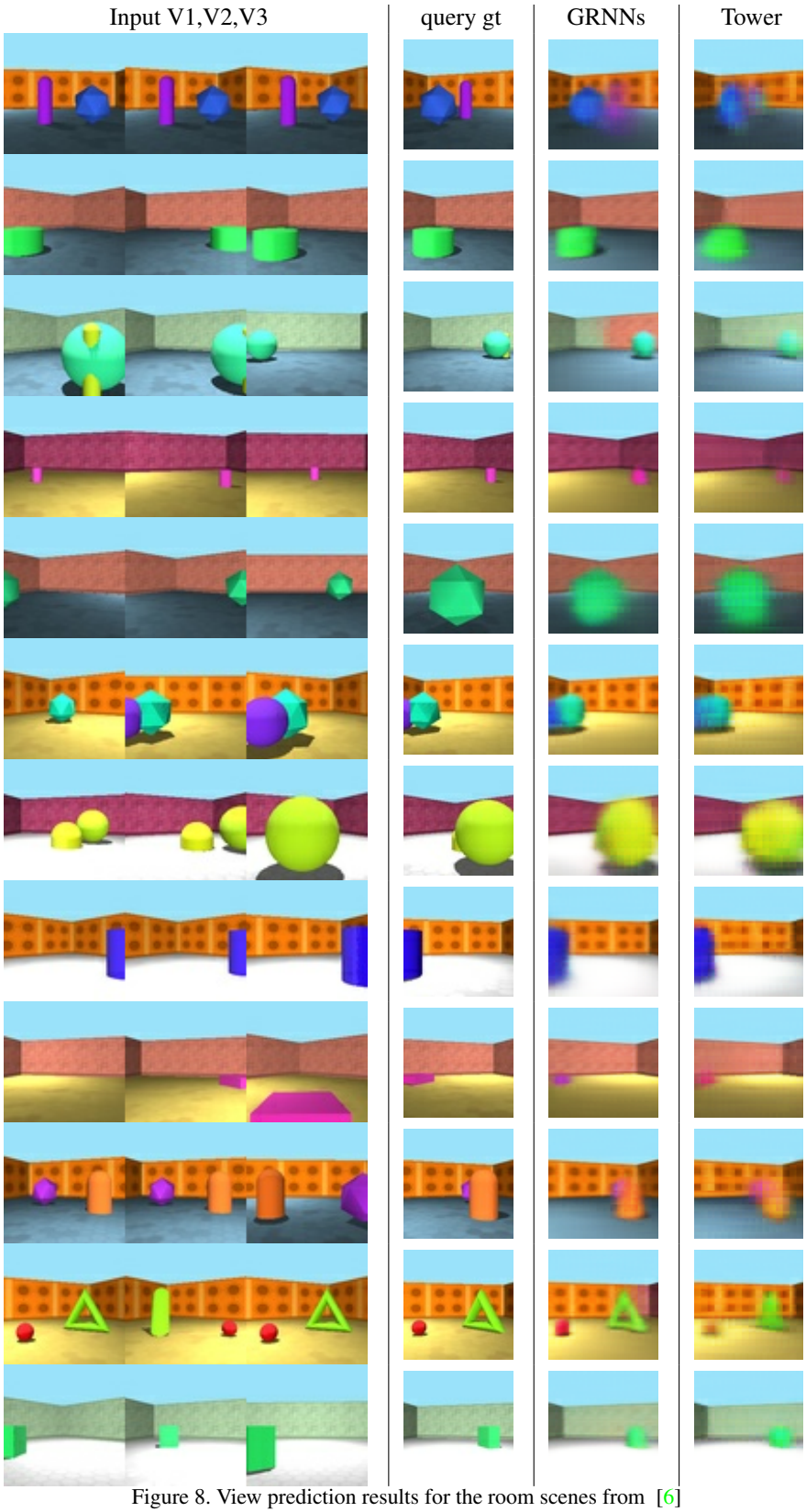

Figure 8. View prediction results for the room scenes from [6]

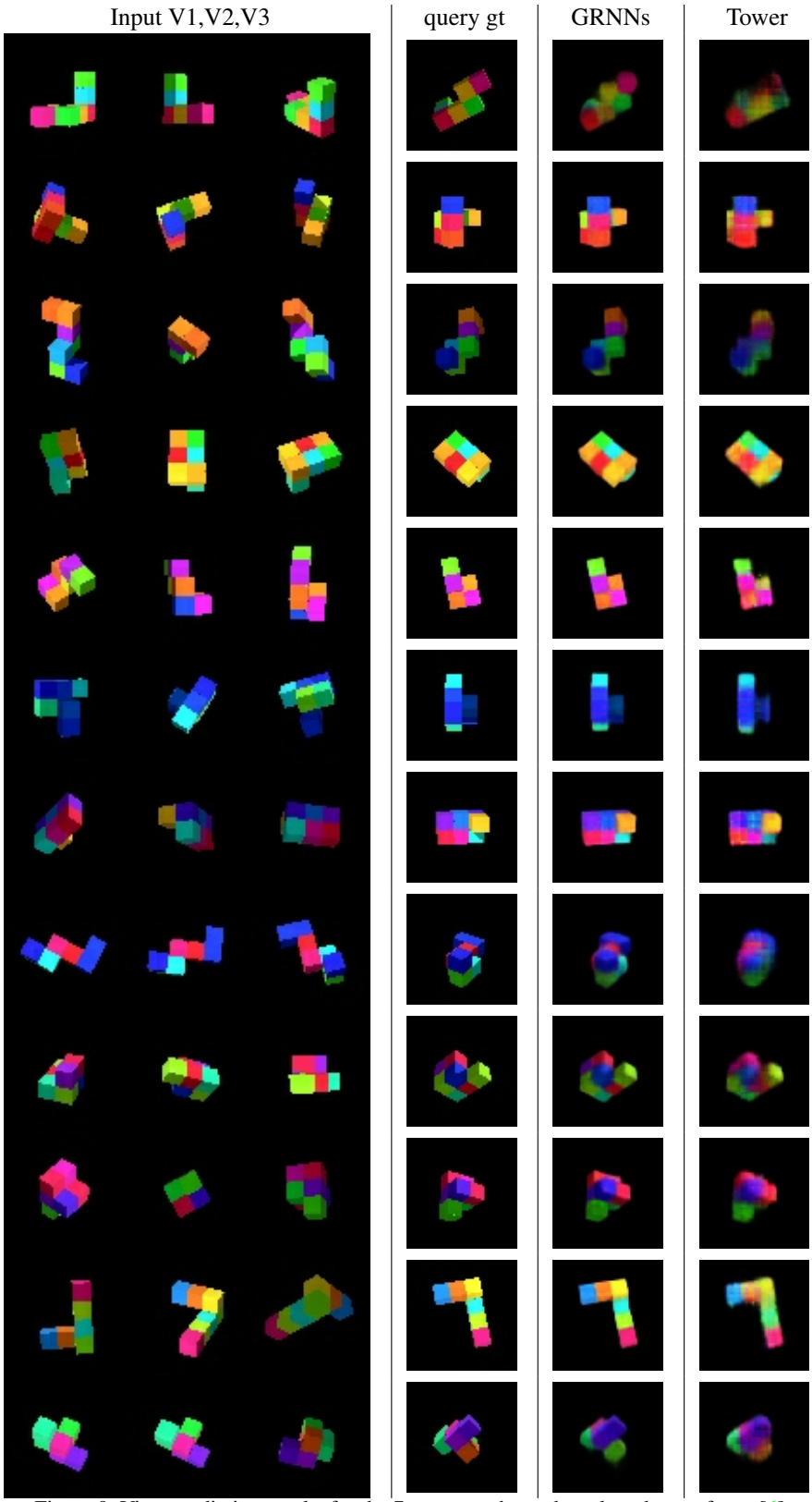

Figure 9. View prediction results for the 7-segment shepard-metlzer dataset from [6]

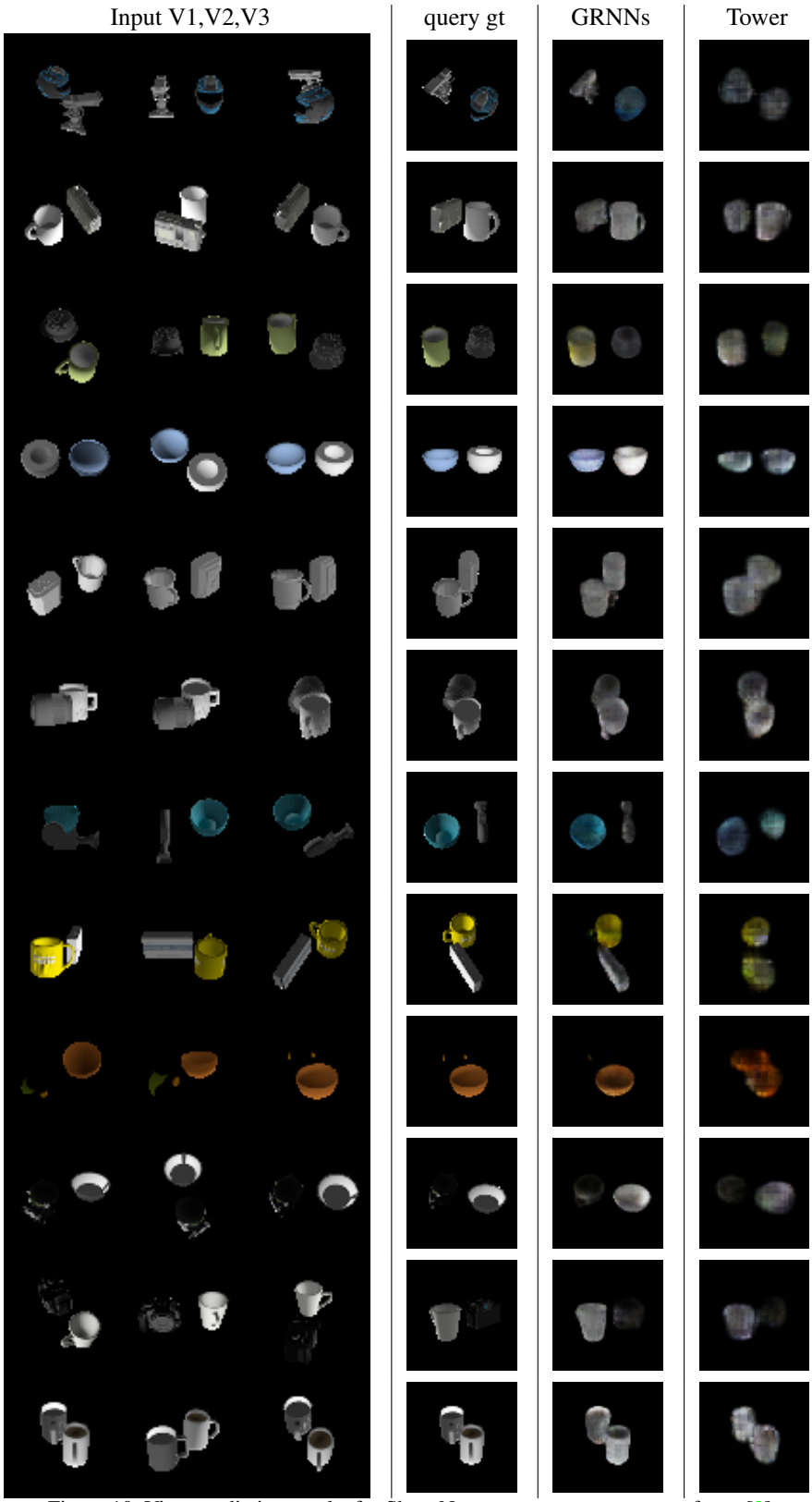

Figure 10. View prediction results for ShapeNet arrangement test scenes from [3]

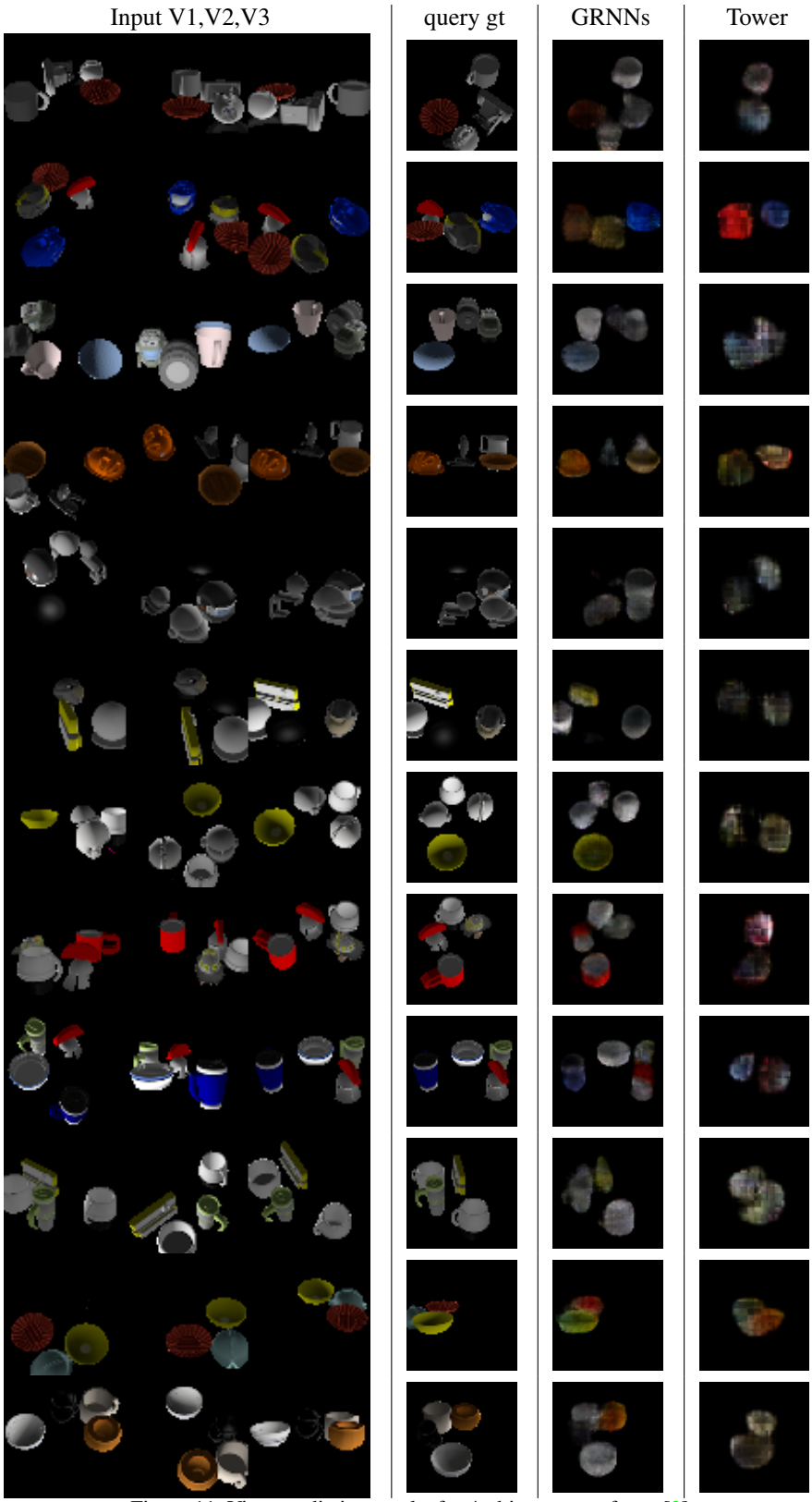

Figure 11. View prediction results for 4-object scenes from [3]

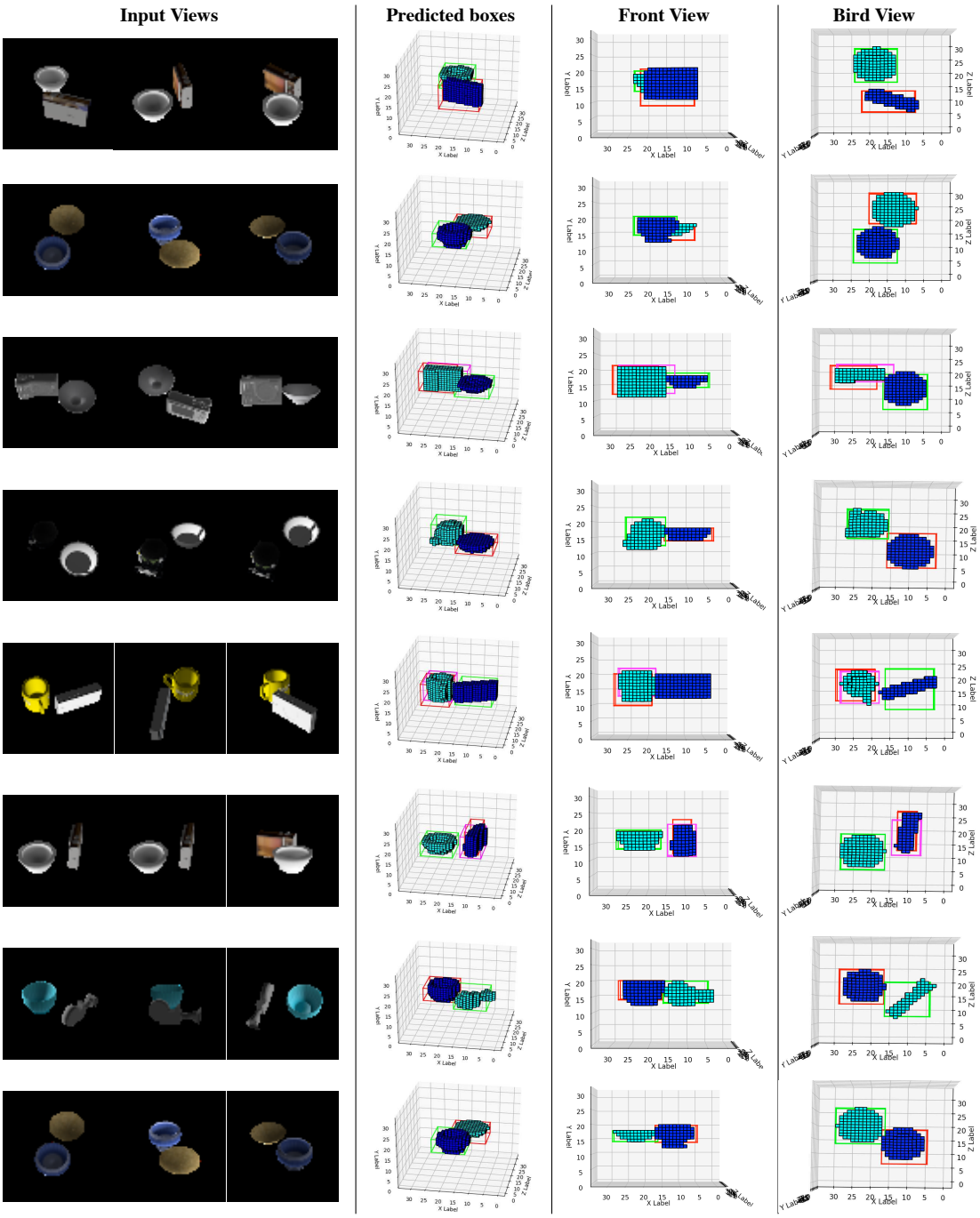

Figure 12. **Object detection and segmentation results.** Blue and light blue grids in the last three columns show **groundtruth** voxel occupancy for the two objects present in the scene. 3D bounding boxes with different colors (red, green and magenta) are **predicted** from the proposed 3D MaskRCNN.