Supplementary Material for EvAC3D: From Event-based Apparent Contours to 3D Models via Continuous Visual Hulls

Ziyun Wang*, Kenneth Chaney*, and Kostas Daniilidis

University of Pennsylvania, Philadelphia PA 19104, USA

Supplemental video, code, and data are available at
https://www.cis.upenn.edu/~ziyunw/evac3d/

1 Additional Handheld Results

In Figure 1, we show more examples of carving of EvAC3D on ground truth Apparent Contour Events.

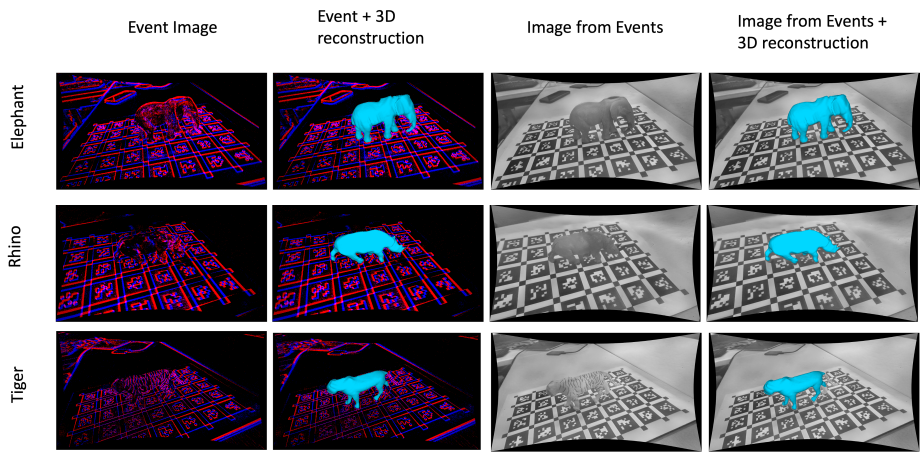

Fig. 1. Reconstruction with Apparent Contour Events of three animals with handheld trajectory. From Left to right: event image by binning (red and blue indicate polarities), event image overlaid with EvAC3D reconstruction, grayscale image from events, grayscale images from events overlaid with EvAC3D reconstruction.

2 Carving Algorithms

We include the two algorithms for carving based on Apparent Contour Events. In Algorithm 1, we show how individual events can be used to carve the volume as tangent rays. In Algorithm 2, we show how to extract the occupancy values and obtain a mesh.

Algorithm 1 Event Carving Algorithm

Input V volume initialized to zero
Input E active contour events
Input ${}^{\mathrm{w}}\mathbf{R}_{\mathrm{c}}(\mathrm{t}), {}^{\mathrm{w}}\mathbf{p}_{\mathrm{c}}(\mathrm{t})$ camera trajectories

procedure CarveEvents(V, E, ${}^{\mathrm{w}}\mathbf{R}_{\mathrm{c}}(\mathrm{t})$, ${}^{\mathrm{w}}\mathbf{p}_{\mathrm{c}}(\mathrm{t})$)
  for $i \leftarrow 1, |E|$ **do**
    $(x_i, y_i, t_i, p_i) \leftarrow E_i$
    ${}^{V}T_{C(t_i)} \leftarrow {}^{V}T_{W}\,{}^{W}T_{C(t_i)}$
    $O_i \leftarrow {}^{V}\mathbf{R}_{W}\,{}^{w}\mathbf{p}_{c}(t_i) + {}^{V}\mathbf{t}_{W}$
    $D_i \leftarrow {}^{V}\mathbf{R}_{W}\,{}^{w}\mathbf{p}_{c}(t_i)\,{}^{w}\mathbf{x}_{c}(t_i)$
    $V_i \leftarrow bresenham3D(O_i, D_i, bounds(V))$
    $V[V_i] += 1$
  end for
  return V
end procedure

Algorithm 2 Segmentation of the probability volume

Input V volume with large values representing vacancy

procedure SegmentVolume(V)
  $Occupancy \leftarrow max(V) - V$
  $center \leftarrow \dfrac{(\Sigma_{V_i} V_i Occupancy[V_i])}{(\Sigma_{V_i} Occupancy[V_i])}$
  $\epsilon \leftarrow Otsu(Occupancy)$
  $CC \leftarrow LabelConnections(Occupancy < \epsilon)$
  $ID \leftarrow CC[center]$
  $Mesh \leftarrow MarchingCubes(CC == ID)$
  return $Mesh$
end procedure

3 MOEC-3D Dataset Details

In this section, we describe the details of MOEC-3D, the first real event dataset for single object reconstruction.

3.1 Hardware Setup

There are two stages in collecting MOEC-3D:

1. Scanning 3D Ground Truth Models
2. Capturing Events with Estimated Pose

Scanning 3D Ground Truth Models In MOEC-3D, we pay close attention to the accuracy of the scanned models because the dataset is designed to test the algorithm's ability to accurately reconstruct 3D models with fine details. A naive way to scan the models is using an RGB-D sensor, e.g. RealSense d415, to capture partial scans of the object from multiple viewpoints and fuse them into a full 3D model. In addition, the pose of the object in the camera frame would be naturally obtained when fusing different views. This approach has been previously used in YCB [1], a well-known dataset for object manipulation. However, the mesh models in YCB have limited accuracy due to the sensor precision. For instance, Intel RealSense d415 has an estimated error of 2 percent of the measured depth. In our data collection, this error rate translates to roughly 1 millimeter of error in 3D mesh reconstruction in our dataset. Therefore, we choose to scan objects using a more accurate industrial-grade 3D scanner with global registration to generate higher accuracy object meshes. For this task, we use an Artec Space Spider high-precision scanner, a high-precision scanner with a rated accuracy of 0.05 millimeters. We show the visualization of our scans in Figure 5.

There are two types of trajectories in our dataset for objects and animals. For objects, we use a turntable to rotate the object. The pose of the object is obtained by mapping the joint angles of the turntable to a pre-computed pose lookup table. The pose lookup table is computed by putting an AprilTag on the turntable. We manually select good segmentation masks from multiple views and run Iterative Closest Points (ICP) to obtain the object pose. However, we recognize that a turntable motion is an oversimplified model of the camera motion. Therefore, we collect additional data with animal models and handheld trajectories. In these sequences, we placed an AprilTag grid on the ground and use reconstructed grayscale frames from events for pose estimation. A sample trajectory is shown in Figure 3. Since animals have more complex geometry, we design a more robust object pose estimation method. Precision alignment of the animal objects was accomplished through aligning known features on the AprilTag Grid and the object. A precision square was used to provide a positive stop at a known corner within the grid. Next, the convex hull of the contact patches was analyzed to find the longest flat surface for alignment on one axis. This provides one degree of freedom left which is constrained by aligning the foot furthest in the remaining movement with the second axis of the square. In 2D this fully constrains the location of the object.

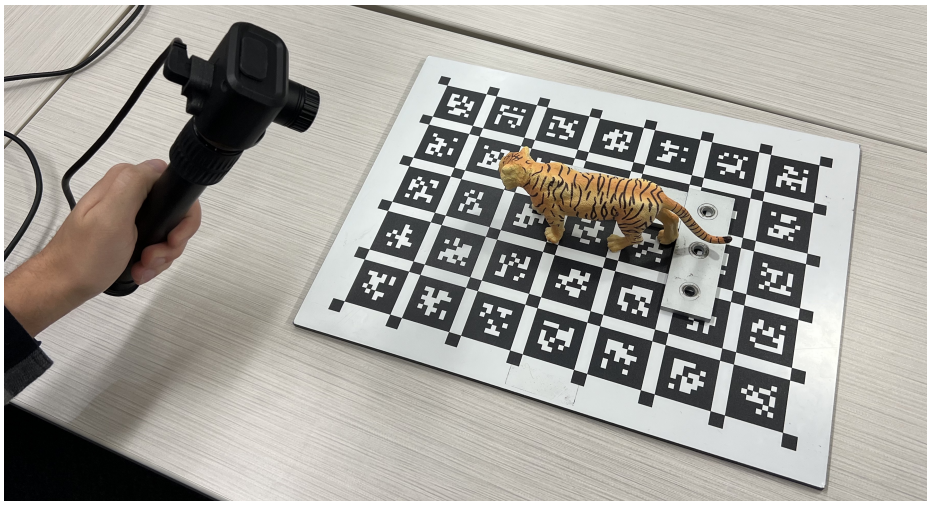

Fig. 2. Data collection setup of a handheld trajectory around an animal model.

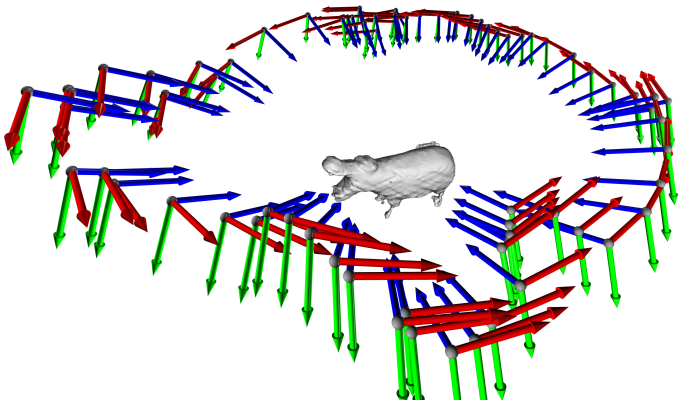

Fig. 3. A sample handheld trajectory from MOEC-3D.

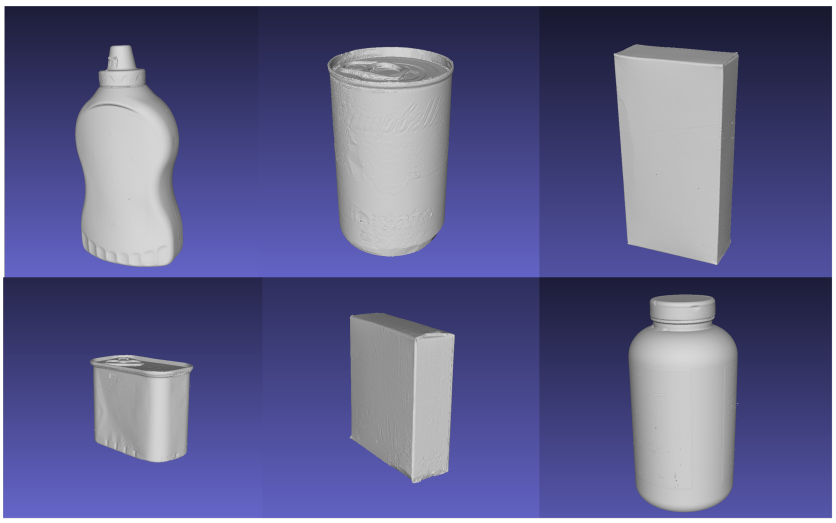

Fig. 4. Select real object scans from the Artec Space Spider scanner.

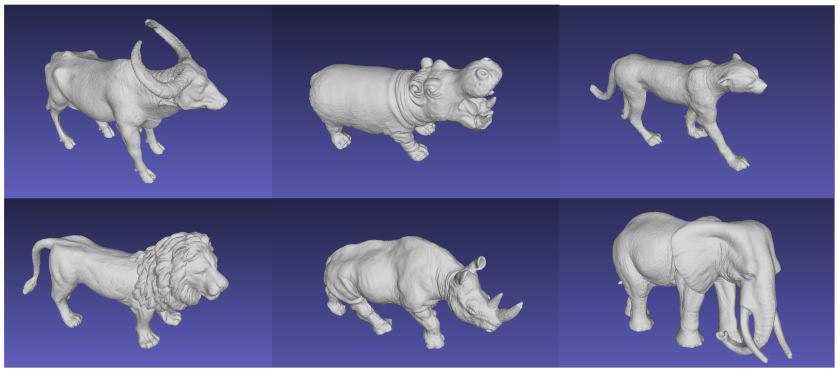

Fig. 5. Select real animal model scans from the Artec Space Spider scanner.

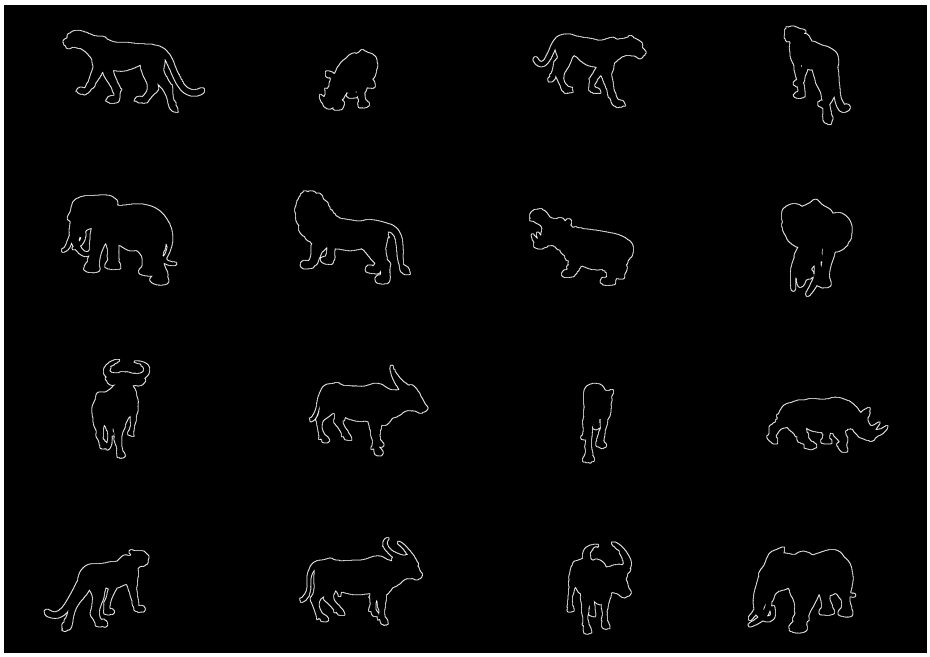

Fig. 6. Example animal apparent contours obtained from projecting the ground truth scan into the event camera.

3.2 Generating Apparent Contour Events

For the two types of trajectories explained above, we know both the camera poses and the object poses. We place an virtual camera in Open3D [6] with the same intrinsics as the real camera. Then the object depth map is rendered using this camera. We then threshold the detph map to obtain the object masks. The final step for obtaining the object apparent contour is straightforward: we dilate the object mask with a 5×5 kernel and then subtract the original mask from the dilated mask. Since we have calibrated the Prophesee camera, the intrinsics of the camera can be used to achieve accurate projection. In figure 6, we show an example of the apparent contours that we obtain through the procedure described above.

ACE Classification Performance A key component of our pipeline is the Apparent Contour Events classification network. We report the test classification performance of each class of the MOEC-3D dataset in Table. 1.

Table 1. Per-class classification performance

Class	Classification Accuracy
Coffee can	0.967
Jello box big	0.954
Jello box small	0.958
Mustard bottle	0.955
Soda can	0.957
Soup can	0.957
Spam can	0.958
Sugar box	0.955
Tuna can	0.956
Vitamin bottle	0.956
All	0.956

Training Details To predict the labels for each event, we map each event directly to a binary label. Since a single event does not contain enough information about the contour, we condition the mapping function on a fixed number of events in the past. In practice, we condition our network on the previous 100,000 events. During training, we predict the label of 50,000 events as a single batch. We attempted to use a PointNet [4] to encode the previous events; however, we found a PointNet encoder unable to perform well enough for this task. Instead, we use an modified image encoder to directly process the event volume generated with the previous events. We use a modified ResNet-18 encoder to encode the previous events, where the first convolutional layer is replaced with a 2D convolutional layer whose input channel matches the time channel of the event volume. Following Occupancy Networks [3], we use several Conditional Batch

Normalization (CBN) layers that apply an affine transformation based on α and β, which are computed from the conditional code (output from the encoder). For each CBN layer, we compute the α and β parameters from a fixed size 128-dimensional vector from the encoder. We use the open source implementation of Occupancy Networks [3] for the decoder network (DecoderCBatchNorm2).

The decoder network consists of one 1D convolution and 5 CBN blocks followed by one more 1D convolutional layer. The x-y coordinates are normalized between -1 and 1 before passing into the decoder network. We use Rectified Linear Unit (ReLU) as the activation functions. Adam optimizer is used with a learning rate of 0.0002 with no learning rate decay. The batch size for training is 32. For fair comparisons, all baseline networks described in the main manuscript are trained 50 epochs. We experimented with multiple loss functions including Focal loss [2] and Dice loss [5], but we found Binary Cross Entropy loss had the best performance in our experiments.

References

1. Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., Dollar, A.M.: The ycb object and model set: Towards common benchmarks for manipulation research. In: 2015 international conference on advanced robotics (ICAR). pp. 510–517. IEEE (2015) 3
2. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017) 8
3. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019) 7, 8
4. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017) 7
5. Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Cardoso, M.J.: Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: Deep learning in medical image analysis and multimodal learning for clinical decision support, pp. 240–248. Springer (2017) 8
6. Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018) 7