SEMANTIC AMODAL VIDEO SEGMENTATION USING A SYNTHETIC DATASET

BY

KEXIN HUI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Assistant Professor Alexander Schwing

Abstract

In this work, we provide tools for annotating both object category and shot transitions for a new semantic modal instance-level object segmentation dataset. This new dataset provides ample opportunities to train models for instance-level segmentation, both modal and amodal. Moreover, in this work, we also present results for instance-level segmentation using ResNet-based DeepLab, a state-of-the-art semantic image segmentation model. We also develop a new semantic amodal instance-level video segmentation model based on DeepLab for the aforementioned dataset. Our model for amodal segmentation operates on a per-frame basis, and the model is guided by the modal mask estimated from the current frame and from previous frames delineating the object of interest. We demonstrate the efficacy of the proposed model on the new dataset.

Acknowledgments

I would like to express my deep gratitude to my advisor, Professor Alexander Schwing, for his valuable and constructive suggestions, enthusiastic encouragement, pioneering vision and generosity. All are crucial for the development of this work.

My grateful thanks are also extended to my colleague, Yuan-Ting Hu, for her guidance and assistance. This work is part of a larger project, which is under the supervision of Yuan-Ting Hu.

I would also like to extend my thanks to all my teachers, professors, mentors and advisers, for they shared their own knowledge that enlightened me all the way from kindergarten to graduate college.

Finally, I wish to thank my parents and my family for their unconditional love and support in whatever I pursue.

Contents

1. Introduction

Convolutional neural networks (CNNs) have revolutionized many visual tasks such as image classification [1], [2], object detection [3], [4], [5], and image segmentation [6], [7], [8]. Originally, segmentation tasks depended highly on the quality of annotators on a specific dataset. Namely, annotators lack clear guidance about how to complete their jobs so there will be some unavoidable inconsistencies among annotators when naming the class. In addition, this unrestricted segmentation dataset resulted in large numbers of classes which could be grouped together. Therefore, semantic segmentation is taking the place of unrestricted segmentation by assigning each pixel a unique class label among a fixed number of classes. Thanks to the boom of large-scale semantic image segmentation datasets these days [9], [10], [11], image segmentation has also rapidly developed using deep neural networks.

Yet most of the works on image segmentation only focus on modal segmentation, which perceives only the visible part of each object. What about those invisible or occluded parts of an object? Furthermore, what about video object segmentation as the world today prefers live videos instead of static images?

As we know, humans can effortlessly infer the whole structure of a physical object in a scene even though it is partially invisible. And if we are able to know the entire structure of an object, both visible and invisible parts, we have a better idea of how an object is related to its surroundings, how a scene is constructed and what information is behind the scene. However, few datasets exist in terms of amodal segmentation, which restricts the research on amodal segmentation. To our knowledge, Maire et al. released 100 meticulously annotated images in 2013 [12], and Zhu et al. released the COCOA dataset by annotating 5000 images from the MS-COCO dataset in 2015 [13]. Yet neither of these works on video amodal segmentation. In order to improve the situation, my colleagues have recently introduced a new dataset, semantic amodal instance-level video object segmentation (SAIL-VOS) recently. They generate video sequences using a photorealistic game engine, Grand Theft Auto V (GTA-V), by simulating different weather conditions, illuminance levels, scenes and scenarios with large numbers of characters and objects. SAIL-VOS provides 419 video sequences with over 82K objects, which is much larger than any other video segmentation dataset [14], [15]. Besides the RGB image of each frame, SAIL-VOS includes both modal masks and amodal masks of each instance in each frame. Since the game engine GTA-V is not able to assign a class label to each object, we develop a web-based object category annotation GUI using HTML/CSS and Python. The GUI program will display the frame image with highlighted object of interest for the annotator. And the annotator needs to provide the class label for each object. Our SAIL-

VOS dataset defines 148 classes in total. Furthermore, to promote the research on amodal segmentation, we also develop a GUI to find shot transitions in each video sequence since temporal consistency is lost during shot transitions and prediction between shot transitions makes no sense. In this case, only shot transitions with abrupt changes are annotated, and we neglect those gradual transitions. To speed up the process of shot detection workload, we also use a traditional computer vision method to preprocess potential frame pairs by obtaining the camera translation and rotation matrix of each frame. By calculating the difference of matrixes between the current frame and the previous frame, we will get a clue if the camera moves a lot between frames, in other words, if there is an abrupt shot transition between frames.

This new proposed SAIL-VOS dataset defines a new challenge for the research community. We further evaluate a new model on SAIL-VOS by providing a baseline result on video amodal segmentation. Inspired by DeepLab [16], we initially evaluate its performance on predicting amodal mask on the COCOA dataset with the guidance of modal mask. The results are quite promising with mean intersection over union better than 0.9. Similar to the previous works [17], [18], we do a semi-supervised training on our SAIL-VOS for each shot. The input is a combination of the RGB image of the current frame and the prediction amodal mask from the previous frame.

2. Related Work

2.1 Image Segmentation

Image segmentation has long been a popular visual task. Segmentation is fundamental and critical, and can be further applied to many other tasks, for example better feature extraction, proposing object regions and getting the segmented object. Traditional computer vision researchers can address the image segmentation through grouping adjacent pixels together into perceptually meaningful or similar regions using k-means and mean shift clustering [19], [20], [21], [22]. One popular method adapted from k-means clustering approach is simple linear iterative cluster (SLIC) [23]. Yet SLIC strongly relies on a good superpixel algorithm. Semantic instance-level image segmentation gives a more detailed view of how a scene can be decomposed into individual objects. It was first proposed by Shotton et al. by combining random forests with conditional random fields (CRFs) [6]. Recently, with the development of convolutional neural networks, image segmentation can be solved using deep neural networks, greatly improving the performance in terms of accuracy and speed [24], [25], [26], [27], [28].

One of the disadvantages of deep neural networks is that they depend on a large amount of training data, yet very few datasets are available now. Popular datasets for instance-level image segmentation include BSDS dataset [11], Pascal VOC 2007, Pascal VOC 2012 [9], MS-COCO [10], CityScapes [29], Berkeley Deep Drive [30], and KITTI [31]. The latter three are intensively used in autonomous driving industry.

2.2 Video Segmentation

Even though we have witnessed the thriving of image segmentation, instance-level video object segmentation has only been introduced recently. It is conceivable that temporal information is hard to incorporate. Current state-of-the-art methods more or less extend from visual tracking: they either interleave box tracking and segmentation [32], or they propagate ground truth annotation in space-time [33]. Khoreva et al. introduce MaskTrack as guided instance segmentation by feeding the network with the previous frame's estimate [18]. MaskTrack is able to refine the prediction frame by frame, along with the usage of optical flow and post-processing by CRFs. Caelles et al. get rid of the dependence of the previous frame estimate, and instead use a single annotated image (hence one shot), for example the first frame, to segment the rest of the frames [17]. They train at first a fully convolutional network (FCN) for the binary classification task between background and foreground object, and later fine-tune the network on the instance of interest.

Two frequently used datasets for large-scale video object segmentation are DAVIS [15] and YouTube-objects [14]. Yet none of these address instance-level video amodal segmentation.

2.3 Amodal Segmentation

As opposed to the fact that modal segmentation is quite prevalent today, amodal segmentation has only been studied intensively in psychophysical literature [34]. It is obvious that instead of perceiving only the visible parts, humans can easily perceive the whole of a physical structure even though it is partially occluded. Inspired by human perception, amodal segmentation focuses on the full extent of each object, not just the visible part. There has been some work on amodal segmentation these days [35], [36]. Among these, Li et al. tackle the problem by training a model to expand the modal mask [37]. Inspired by this, Zhu et al. propose a new COCOA dataset [13] based on the MS-COCO dataset [10], and set up baseline results using AmodalMask adapted from Li's work [37]. To our knowledge, there are no other large-scale amodal segmentation datasets today.

3. Data Annotation

To address the problem that few video datasets release semantic amodal instance-level video object segmentation, my colleagues have proposed a new synthetic dataset, SAIL-VOS. They utilize a photorealistic game engine GTA-V to generate the dataset since it is easy to obtain a lot of useful ground truth information, such as exact positions, in a game simulator.

The raw dataset is composed of 419 densely sampled video sequences in a total of more than 130,000 frames at a resolution of 800 × 1280, covering different weather conditions, illumination, scenes and scenarios. For each frame, we have an original RGB image, clean background RGB image with only one object that is present at that frame including its corresponding modal and amodal masks, and also the key coordinates of every object that is present at that frame. Figure 1 shows an example of the SAIL-VOS dataset. RGB images, and both modal and amodal masks, can be useful for the video segmentation task. There are over 82k instances in the SAIL-VOS dataset, under three major categories: people, properties and vehicles. Figure 2, Figure 3, and Figure 4 show the example objects from these three categories respectively. For human objects, we also provide both 2D coordinates in the image plane and 3D coordinates in the world plane, that can be used to construct human pose: 18 points on the body and 30 points on both hands (15 points per hand). For other objects under the category of vehicles and properties, we record the 2D and 3D coordinates for the object center.

3.1 Object Category Annotation

As stated before, there are over 82k instances in SAIL-VOS. The game engine, GTA, only provides us with a unique object ID for each object, under three major categories, “Ped”, “Prop”, and “Vehicle”. For example, we will get an object with its unique ID “Ped_000000103106535”, yet we are not able to tell whether this object is a person, a dog, or a cat. Similarly, we have an object with its unique ID “Prop_prop_laptop_01a”, and we might guess it is a laptop, but we do not know exactly what it is. Therefore, we will need to manually label the object category, which results in 10 times more annotation work than existing datasets.

3.1.1 Annotation GUI

The brute force way is to look at all the objects in SAIL-VOS dataset and assign a corresponding category to each object. The problem with this method is that we are not able to keep track of what object we have annotated and what category we have assigned. And we may end up in annotating one object multiple times and having large numbers of categories in which some can actually be grouped or merged together.

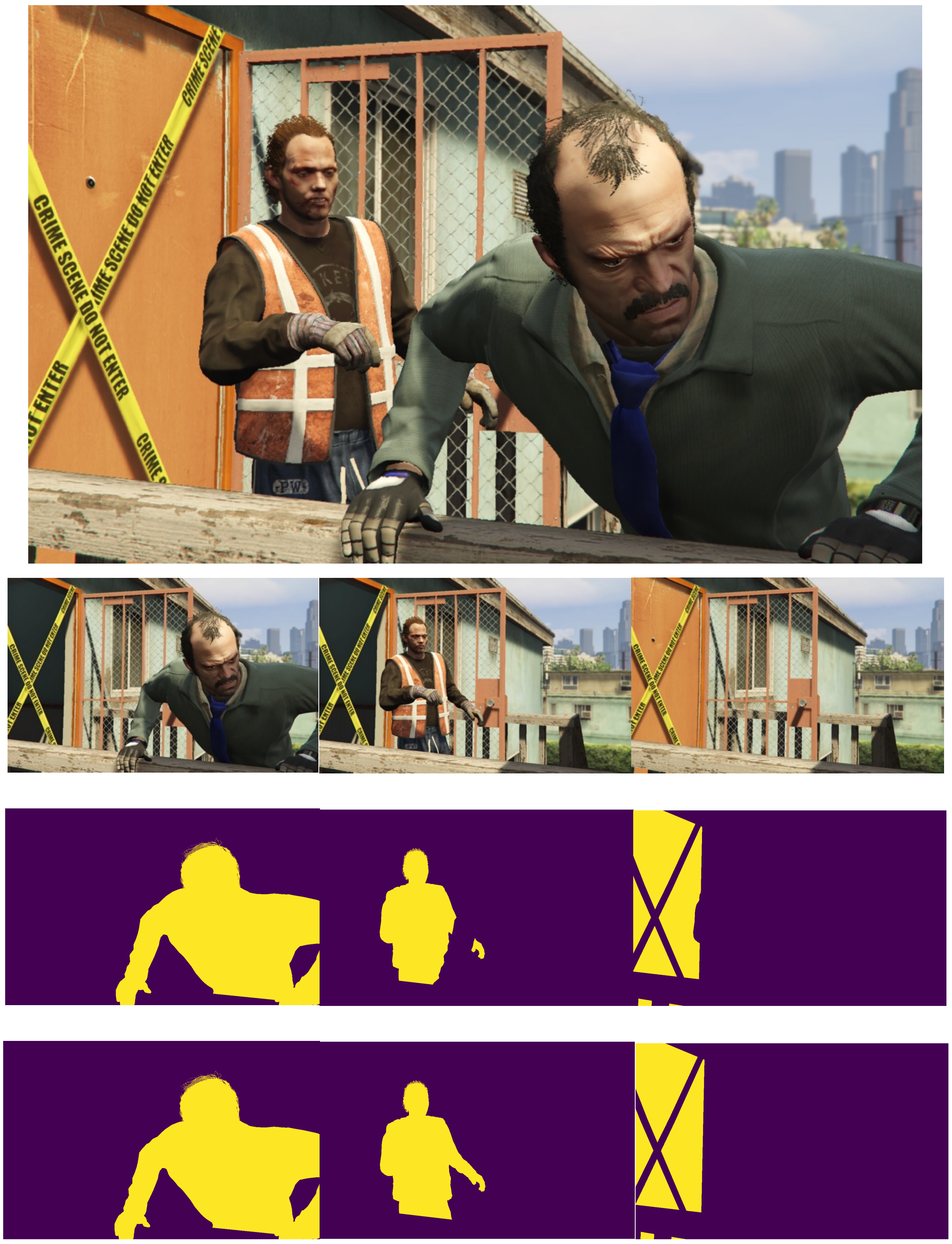
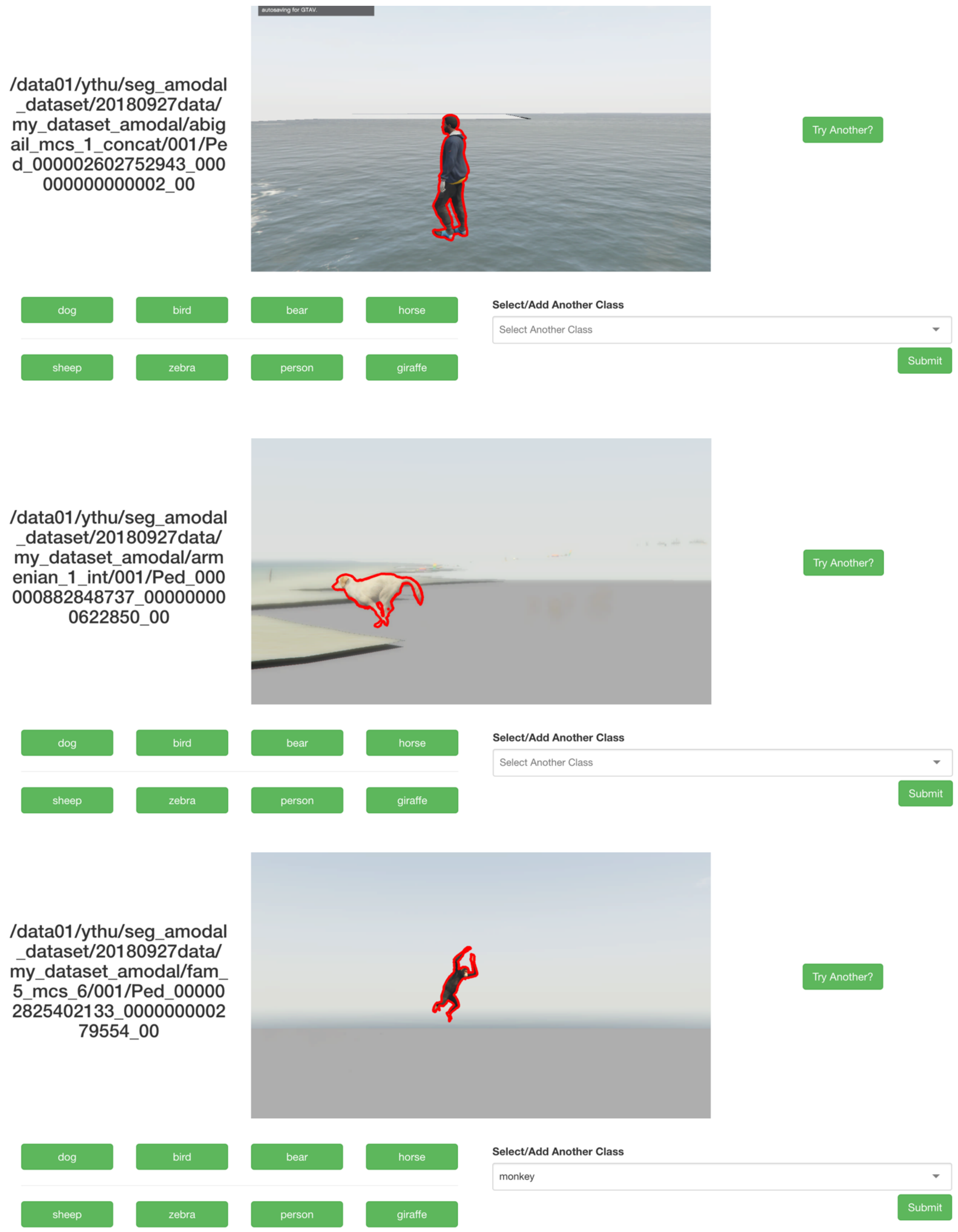


Figure 1 Example Data in the SAIL-VOS Dataset. **First Row:** Original RGB Image. **Second Row:** Clean Background Image with Only One Object Present at that Frame, from Left to Right are Human 1, Human 2, and Object "door". **Third Row**: Modal Masks for Each Object at that Frame. **Fourth Row**: Amodal Masks for Each Object at that Frame.


/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/abigail_mcs_1_concat/001/Ped_000002602752943_000000000000002_00

Select/Add Another Class

Select Another Class

/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/armenian_1_int/001/Ped_000000882848737_000000000622850_00

Select/Add Another Class

Select Another Class

/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/fam_5_mcs_6/001/Ped_000002825402133_000000000279554_00

Select/Add Another Class

monkey

Figure 2 Examples of “Ped” Object. **First Row**: Person. **Second Row**: Dog. **Third Row**: Monkey.

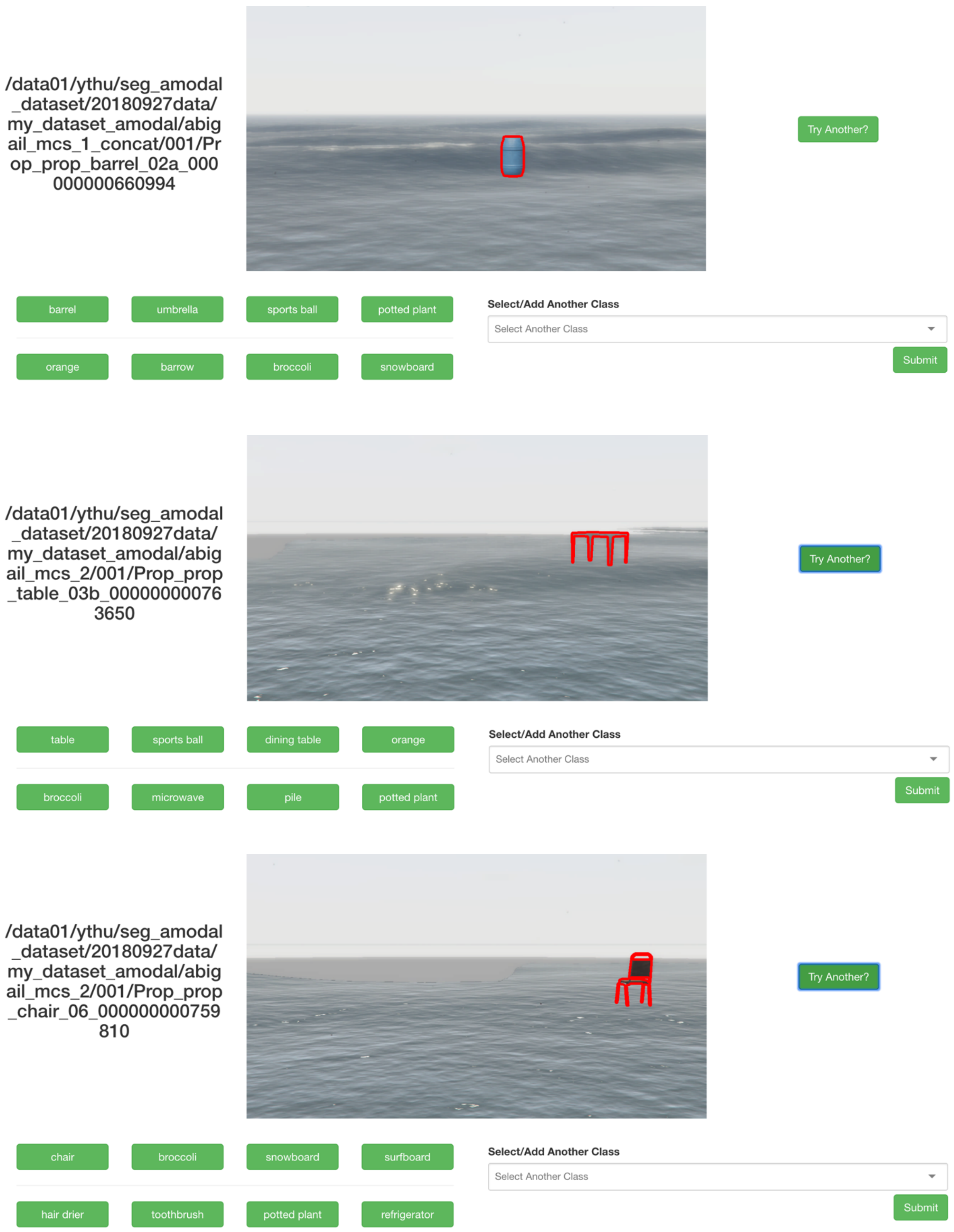
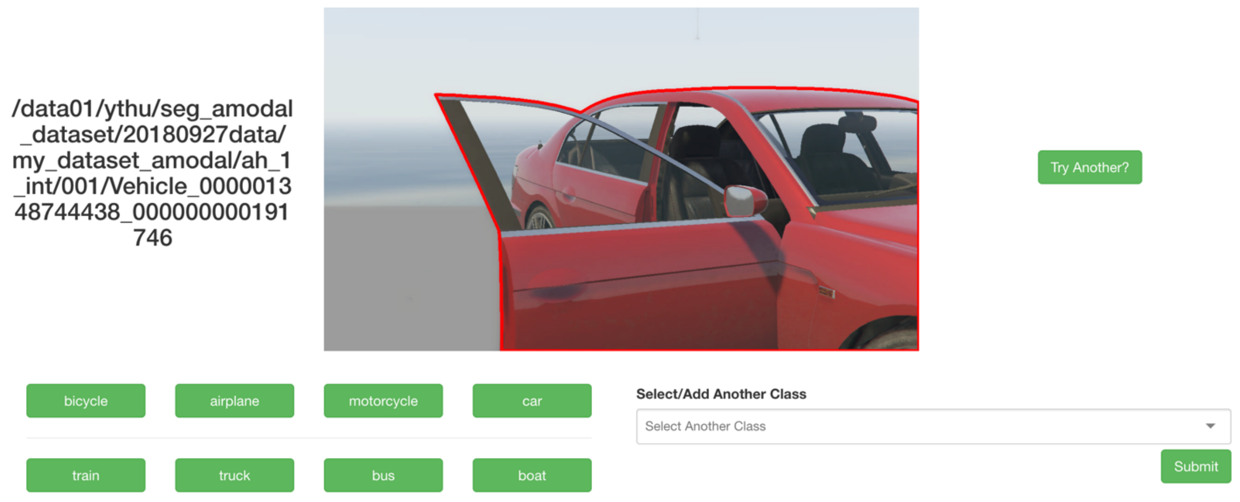
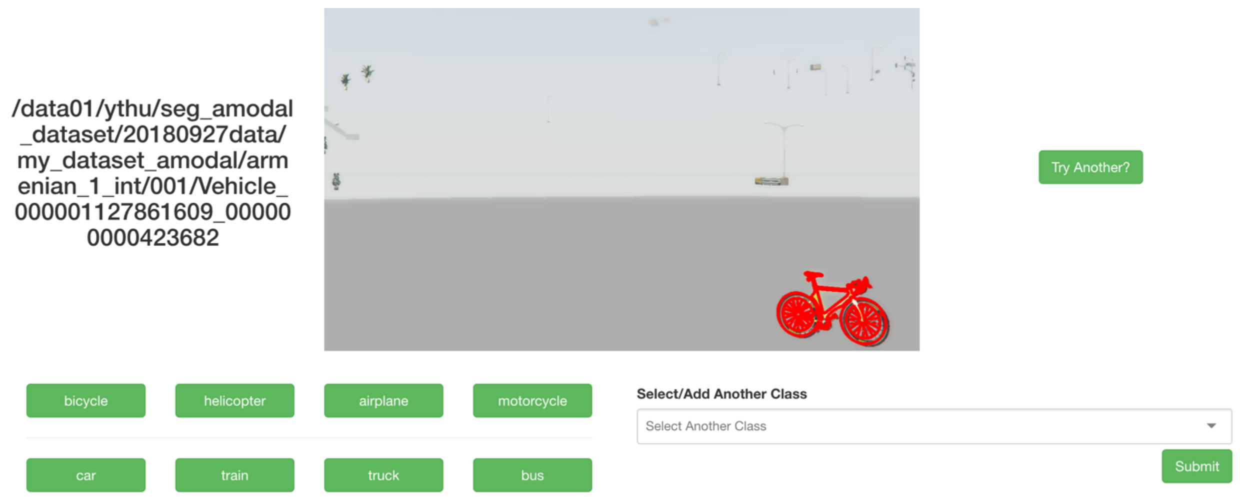


Figure 3 Examples of “Prop” Object. **First Row**: Barrel. **Second Row**: Table. **Third Row**: Chair.


/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/ah_1_int/001/Vehicle_000001348744438_000000000191746


/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/armenian_1_int/001/Vehicle_000001127861609_000000000423682

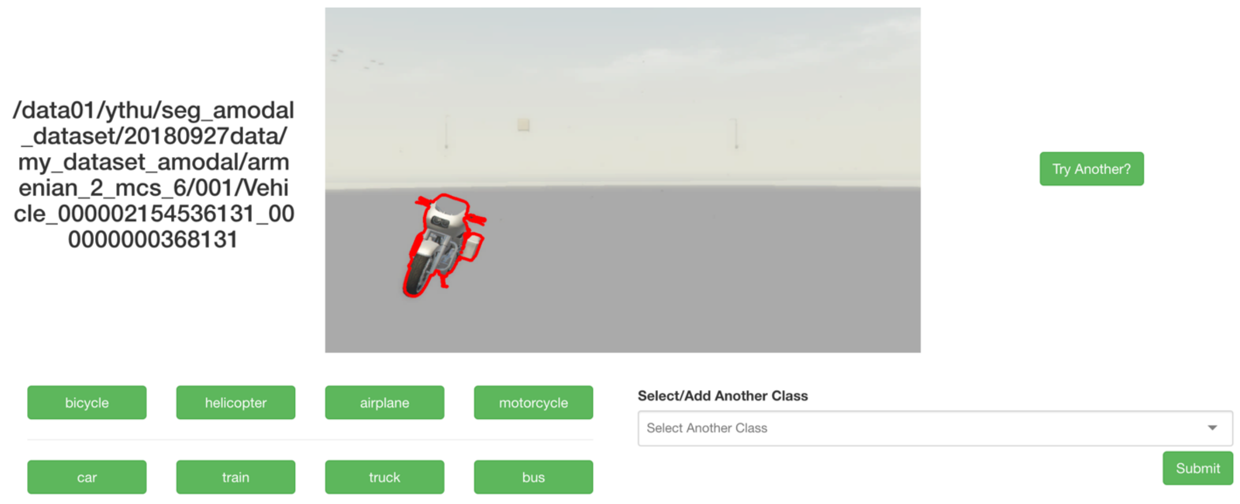
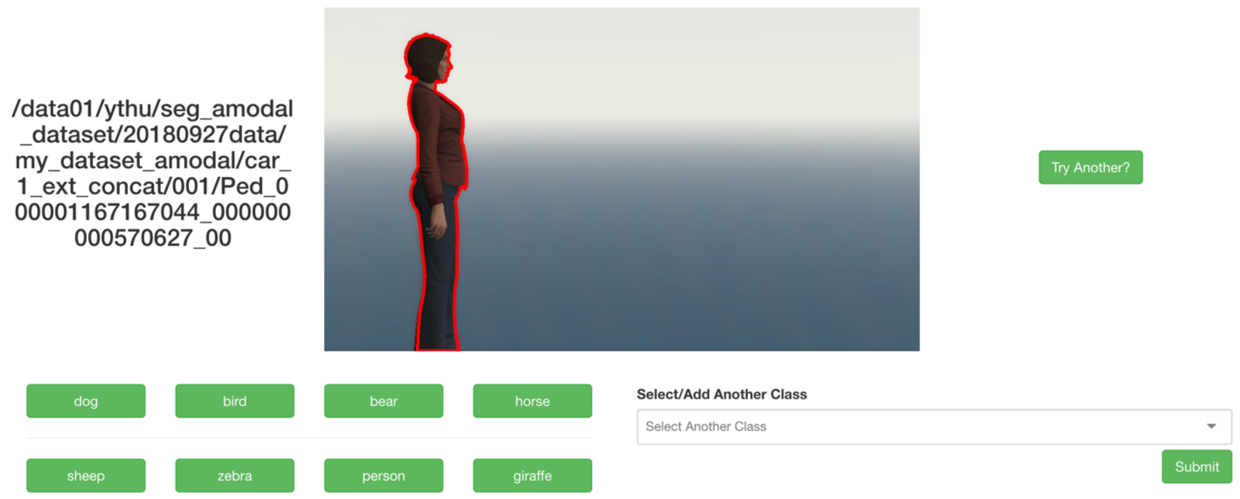

/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/armenian_2_mcs_6/001/Vehicle_000002154536131_000000000368131

Figure 4 Examples of "Vehicle" Object. **First Row**: Car. **Second Row**: Bicycle. **Third Row**: Motorcycle.

Instead, we designed a web-based graphical user interface (GUI) to help annotate the object category. We first iterate all the video sequences, and store each object with its corresponding frame and video. For each object, we will get a list of frames in which it appears in all the video sequences. In order to avoid the annotator checking one object multiple times, we just want to display one image with that object. The frame will be chosen as the image with the highest probability that the annotator can tell exactly which category this object belongs to. Intuitively, we choose the frame with the object taking the largest area. This decision is made by looking at the corresponding amodal mask of that object in that particular frame and computing the area of that object, which is the number of all the 1s in the amodal mask. We sort all the frames by object area from large to small and always give the annotator, which is the frame with the largest object area, the best. In addition, in order to emphasize the object that we want to annotate in that frame, we draw a red contour of the object in that frame using its amodal mask. In addition, we want to give the annotator some recommendations for the object category to speed up the annotation process.

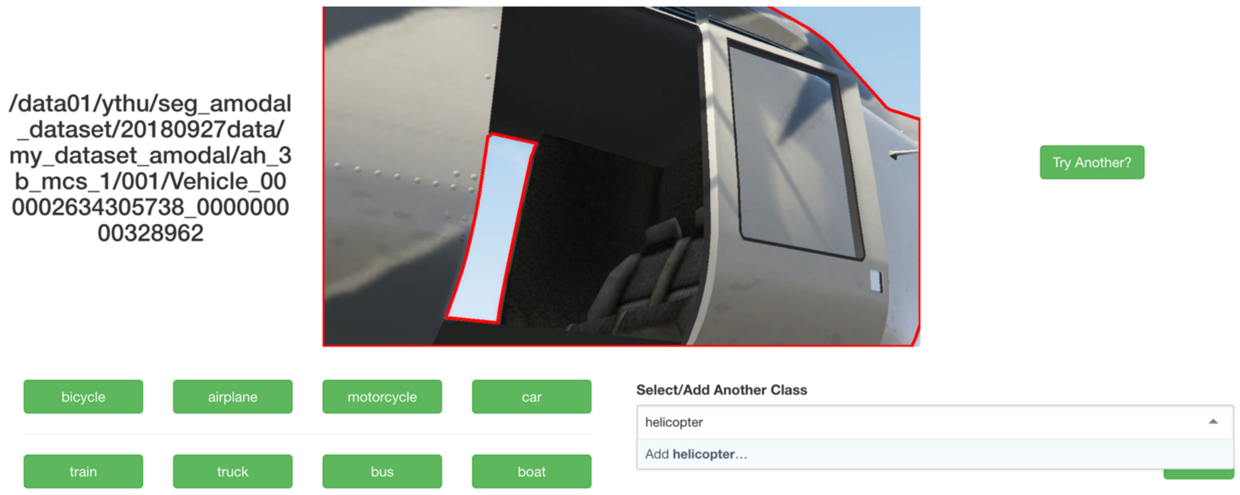
We extend from the 80 categories from the MS-COCO dataset [10], which is a popular and standard instance-level image segmentation dataset. We split the 80 categories from MS-COCO into three major categories, “Ped”, “Prop” and “Vehicle”, which is consistent with default names generated by GTA simulator. Every time we annotate a new object, we only give the annotator the options that correspond to its major category. For a “Prop” object, we will also find the most similar category name by matching the pattern between the current object name and the candidate names in that major category since the unique ID of a “Prop” object also suggests its potential category.

A sample interface of object category annotator can be found in Figure 5. The frame image will be displayed in the middle with a clean background and an object emphasized by red contours. The name of the image and its corresponding path will be shown on the left. We only display 8 highly likely categories below the image based on our recommendation strategy. If the current object does not belong to the 8 direct recommendations, there is a scroll-down list in which the annotator can look for all the options in that major category. It is worth mentioning that our SAIL-VOS object category is not limited to the 80 categories from the MS-COCO dataset [10]. As can be seen from the example in Figure 6, the current object is a helicopter, which cannot be found in the existing categories of “Vehicle”. Therefore, we need to add this category “helicopter” manually in the scroll-down list by just typing its name and submitting it. After “helicopter” is successfully added as a new object category, the program will memorize and store this new object category under “Vehicle”. In the future, when a new “Vehicle”



/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/car_1_ext_concat/001/Ped_000001167167044_000000000570627_00

Figure 5 Example of Object Category Annotation GUI



/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/ah_3b_mcs_1/001/Vehicle_000002634305738_000000000328962

Figure 6 Example of Adding a New Object Category in Annotation GUI

object needs annotating, "helicopter" will also appear as an option for the annotator. It can also be the case that the current frame is not eligible for the annotator to tell which category it belongs to. Several factors cause this problem: one is that the camera is so close to the object that the current frame only reveals part of it; another is that the capturing angle makes it difficult for the annotator to tell which category it belongs to. In this case, we add a button "Try Another?" to help the annotator select other frames to see if they can decide the category as shown in Figure 7. The program will select the frame with the next largest object area based on pre-computation. Figure 8 shows the case when there are no more frames available. "Unknown" category is used to annotate those objects for which we have trouble assigning a category.

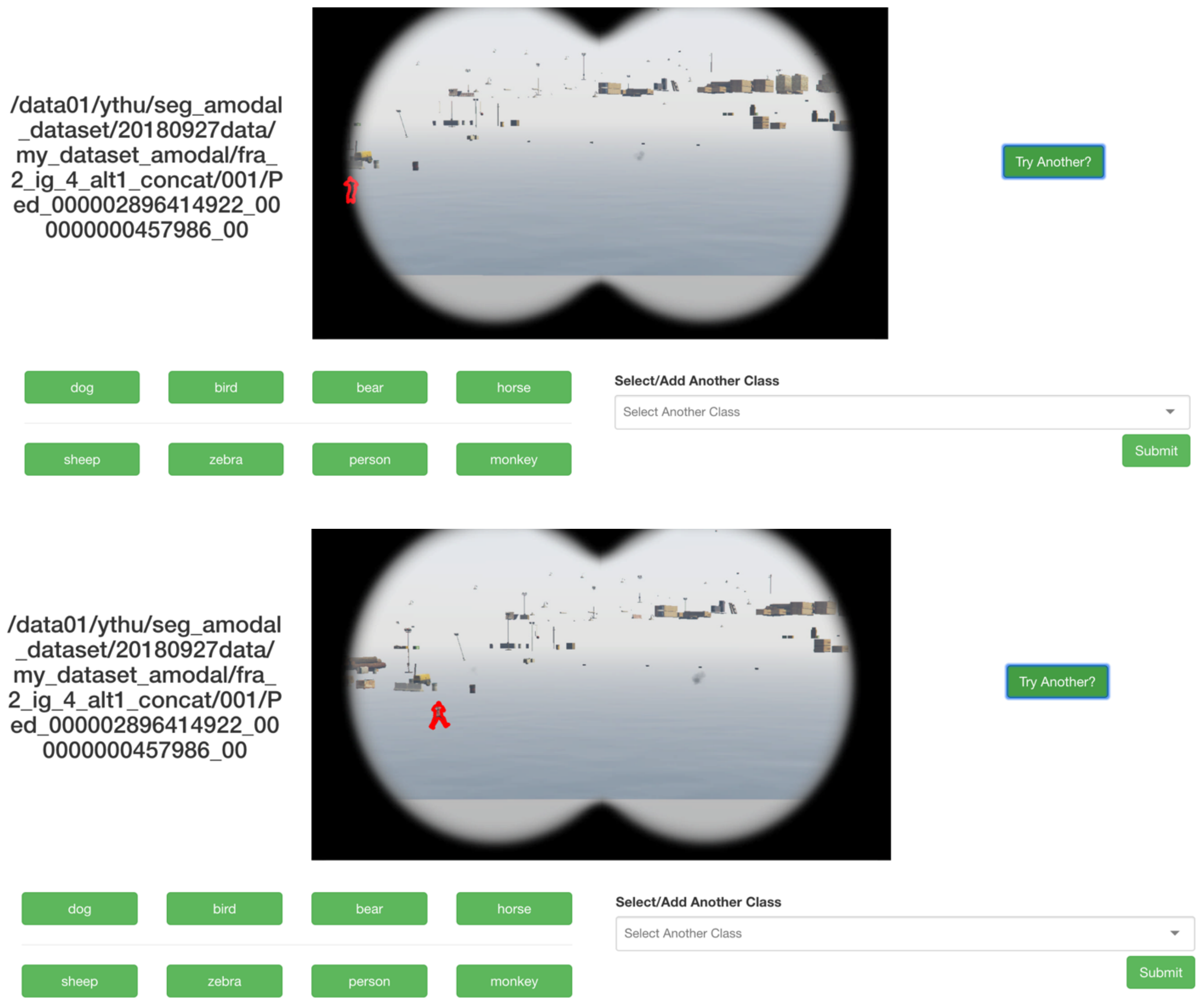
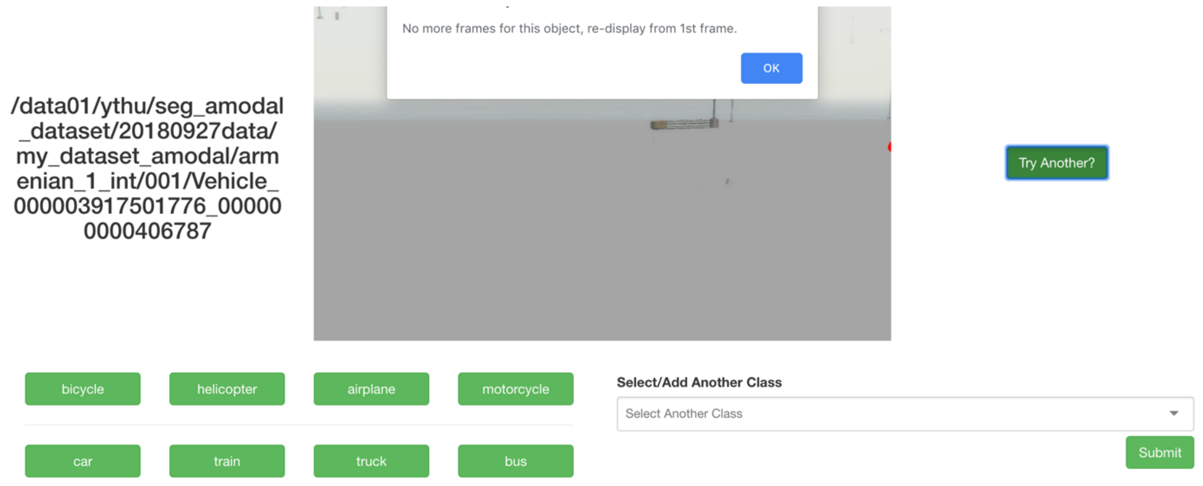

/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/fra_2_ig_4_alt1_concat/001/Ped_000002896414922_000000000457986_00
Try Another?
dog
bird
bear
horse
sheep
zebra
person
monkey
Select/Add Another Class
Select Another Class
Submit
/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/fra_2_ig_4_alt1_concat/001/Ped_000002896414922_000000000457986_00
Try Another?
dog
bird
bear
horse
sheep
zebra
person
monkey
Select/Add Another Class
Select Another Class
Submit

Figure 7 Example of “Try Another?” is Used. **First Row**: Hard to Assign a Category. **Second Row**: Confirmed Person.


No more frames for this object, re-display from 1st frame.

OK
/data01/ythu/seg_amodal_dataset/20180927data/my_dataset_amodal/armenian_1_int/001/Vehicle_000003917501776_000000000406787
Try Another?
bicycle
helicopter
airplane
motorcycle
car
train
truck
bus
Select/Add Another Class
Select Another Class
Submit

Figure 8 Example of Object Category Annotation When No More Frames Are Available

3.1.2 Results and Statistics

We assign a class label to each object that appears in SAIL-VOS. In total, 825,186 instances are found in our SAIL-VOS dataset, defining a total of 148 classes, which is much more than the number of classes in the MS-COCO dataset [10]. Figure 9 shows the number of instances per class in our SAL-VOS dataset. Among our 148 classes, 55 overlap with MS-COCO dataset. In other words, 68.75% of the classes in MS-COCO dataset can be found in SAIL-VOS. Ninety-three classes can be found only in our SAIL-VOS, but not in MS-COCO, and 25 classes exist only in the MS-COCO dataset but not in our dataset. Classes that exist only in MS-COCO include giraffe, kite, and hair drier. Classes that exist only in our dataset include lamp, pot and cabinet.

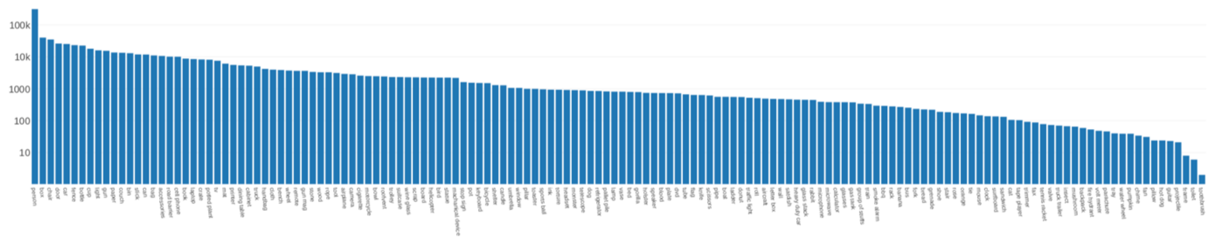


Figure 9 Number of Instances Per Class in SAIL-VOS

3.2 Shot Transition Annotation

Shot transition detection, also known as shot detection, aims to detect shots in a video sequence. A shot is defined as a series of interrelated consecutive pictures taken accordingly by a single camera and representing a continuous action in time and space. Obviously, a video can contain more than one shot since the camera may move, rotate, or focus on different objects in a video sequence. There are two kinds of shot transitions: abrupt transition where the camera abruptly changes focus on another object, and gradual transitions where the camera is moving or rotating gradually and slowly.

In our dataset, we want to differentiate only those abrupt transitions, also known as hard cuts, since different shots do not share any common background or objects which will be difficult for video segmentation. For those gradual transitions, we can still infer or get a clue for the current frame based on the previous frames as we can safely assume that objects between frames will not move fast.

3.2.1 Shot Detection

The brute force way to annotate the shot transition is to look at every frame of all the video sequences and manually label if each frame pair, namely the current frame and the previous frame, is a shot transition. However, it is too time-consuming and expensive to do that as we will expect over 3 days to finish the annotation if we take 2 seconds to look at each frame. In this case, we certainly want to filter out those frame pairs that are not shot transitions. Next, we can annotate manually the shot transitions from those left candidates. Then the problem becomes how we know if a frame pair is a shot transition.

We address the problem through solving a camera model using 2D and corresponding 3D coordinates provided in our dataset. As we know, a camera works as a mapping of an object in the 3D world coordinate system to a 2D image plane in the film. In other words, a camera projects the 3D world onto a 2D image. We are using a homogeneous coordinate system in Figure 10 for convenience. Using homogeneous coordinates, we can formulate the camera model as $\mathrm{x} = \mathrm{K}\,[\mathrm{R}\ \mathrm{t}]\mathrm{X}$ as shown in Figure 11. The matrix K is the camera matrix.

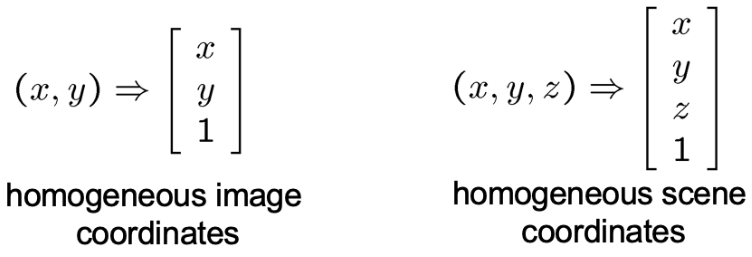

$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
homogeneous image coordinates
$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$
homogeneous scene coordinates

Figure 10 Homogeneous Coordinate System [38]

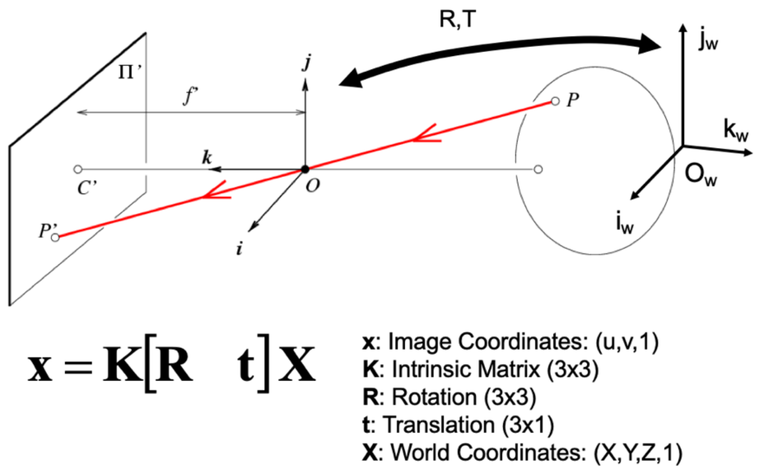
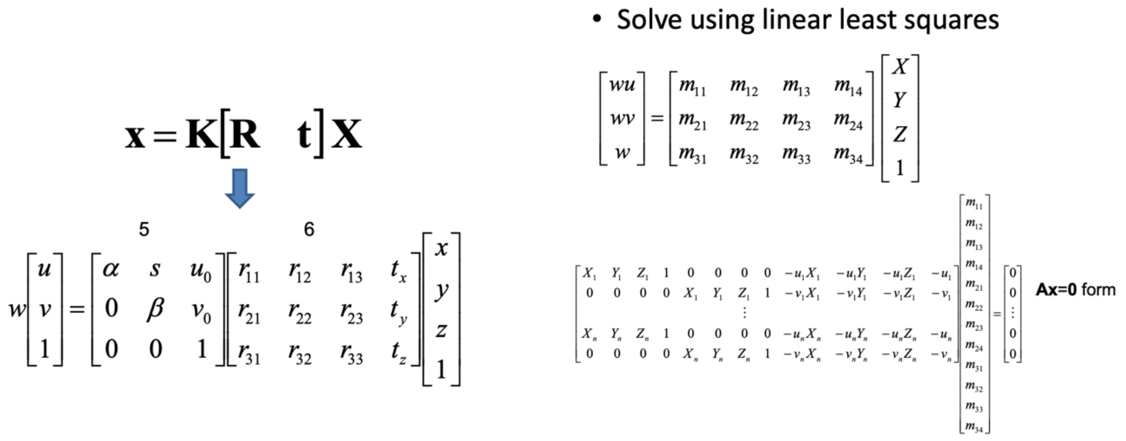

R,T
j
j_w
Π'
f'
P
k_w
k
C'
O
O_w
P'
i
i_w
$\mathbf{x} = \mathbf{K}[\mathbf{R}\quad \mathbf{t}]\mathbf{X}$
x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

Figure 11 Camera Model [38]

This matrix contains some of the critical parameters that are useful to characterize a camera model. We are assuming no skew and distortion for the camera model [39]. Rotation matrix R and translation matrix t are required to relate points from the world coordinate system to the camera reference system. In that sense, K can be referred to as the camera intrinsic matrix, and the matrix $[\mathrm{R}\ \mathrm{t}]$ can be referred to as the camera extrinsic matrix. Thus, we can get the overall projection matrix P as $\mathrm{P} = \mathrm{K}\,[\mathrm{R}\ \mathrm{t}]$. Overall, we find that the 3 × 4 projection matrix P has 11 degrees of freedom: 5 from the intrinsic camera matrix, 3 from extrinsic rotation matrix and 3 from extrinsic translation matrix [39]. As a result, we will need at least 6 pairs of 2D and 3D coordinates to solve the linear equation since each pair of points will construct two equations as shown in Figure 12.



$$\mathbf{x} = \mathbf{K}[\mathbf{R}\quad \mathbf{t}]\mathbf{X}$$

$$w\begin{bmatrix}u\\v\\1\end{bmatrix} = \underset{5}{\begin{bmatrix}\alpha & s & u_0\\0 & \beta & v_0\\0 & 0 & 1\end{bmatrix}}\underset{6}{\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

$$\begin{bmatrix}wu\\wv\\w\end{bmatrix} = \begin{bmatrix}m_{11} & m_{12} & m_{13} & m_{14}\\m_{21} & m_{22} & m_{23} & m_{24}\\m_{31} & m_{32} & m_{33} & m_{34}\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

$$\begin{bmatrix}X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1\\0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1\\ & & & & & & \vdots & & & & & \\X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n\\0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n\end{bmatrix}\begin{bmatrix}m_{11}\\m_{12}\\m_{13}\\m_{14}\\m_{21}\\m_{22}\\m_{23}\\m_{24}\\m_{31}\\m_{32}\\m_{33}\\m_{34}\end{bmatrix} = \begin{bmatrix}0\\0\\\vdots\\0\\0\end{bmatrix}$$

Figure 12 Solving Camera Model through Linear Method [38]

In our case, for each frame of a video sequence, we read 2D and 3D coordinates into two large matrixes by checking all objects that are present at that specific frame, and we get pos_2d (N × 2) and pos_3d (N × 3) where N is the number of point pairs for that frame. Next, we construct a large A matrix as Equation 1 using all the corresponding pairs from pos_2d and pos_3d. As we discussed before, if the number of

$$\begin{bmatrix}\mathrm{X} & \mathrm{Y} & \mathrm{Z} & 1 & 0 & 0 & 0 & 0 & -\mathrm{x}\cdot\mathrm{X} & -\mathrm{x}\cdot\mathrm{Y} & -\mathrm{x}\cdot\mathrm{Z} & -\mathrm{x}\\0 & 0 & 0 & 0 & \mathrm{X} & \mathrm{Y} & \mathrm{Z} & 1 & -\mathrm{y}\cdot\mathrm{X} & -\mathrm{y}\cdot\mathrm{Y} & -\mathrm{y}\cdot\mathrm{Z} & -\mathrm{y}\end{bmatrix}$$

Equation 1 Linear Equation Using World Coordinate $[\mathbf{X\ Y\ Z}]$ and Image Coordinate $[\mathbf{x\ y}]$

points in that frame is less than 6, we cannot solve the linear equation and we simply ignore that frame. We can use single value decomposition SVD to solve the above problem $\mathrm{Ax} = 0$, where A is the matrix we construct, and x is the flatten camera projection matrix. Namely, A can be decomposed as $\mathrm{A} = \mathrm{U}\,\Sigma\,V^T$, where U is an m × n matrix, Σ is an n × n matrix with non-negative real numbers on the diagonal known as the singular values of A, and V is an n × n matrix. And the projection matrix P can be reconstructed using the last column of V, which corresponds to the smallest singular values. Using QR decomposition we can get camera intrinsic matrix K and rotation matrix R. And translation matrix t, also related to the camera center, can be found as the null space of the projection matrix. In other words, $\mathrm{t} = -\mathrm{C} = -Q^{-1}b$ where $\mathrm{P} = [Q|b]$ and b is the last column of P. Once we get the rotation matrix and translation matrix, we can compute the movement of the camera between the current frame and the previous frame. If the movement is larger than some threshold, we can consider it as a shot transition candidate. The movement is composed of two parts: difference of translation matrixes, and difference of rotation matrixes. The difference between two translation matrixes is simply calculated through the Euclidean distance of the two vectors as shown in Equation 2. And the different between two rotation matrixes is the angle of difference rotation represented by the rotation matrix $\mathrm{R} = PQ^T$, where P and Q are two rotation matrixes. We can retrieve the angle of difference rotation from the trace of R using *arccos* function as shown in Equation 3. We set the thresholds to be different of translation matrixes at

$$\text{Difference between translation vectors} < \mathrm{t}_{x1}, \mathrm{t}_{y1}, \mathrm{t}_{z1} > \text{and} < \mathrm{t}_{x2}, \mathrm{t}_{y2}, \mathrm{t}_{z2} >$$
$$= \sqrt{(\mathrm{t}_{x1} - \mathrm{t}_{x2})^2 + (\mathrm{t}_{y1} - \mathrm{t}_{y2})^2 + (\mathrm{t}_{z1} - \mathrm{t}_{z2})^2}$$

Equation 2 Calculation for Difference between Translation Matrixes

$$\text{Difference between rotation matrix P and Q} = \arccos\left(\frac{\mathrm{tr}(\mathrm{PQ}^{\mathrm{T}}) - 1}{2}\right)$$
$$\text{where } \mathrm{tr}(\mathrm{R}) = \mathrm{tr}(\mathrm{PQ}^{\mathrm{T}}) = 1 + 2\cos\theta$$

Equation 3 Calculation for Difference between Rotation Matrixes

1 pixel, and difference of rotation matrixes at 8 degrees. We use OR for the two threshold values as we want to eliminate all true negatives and include as many true candidates as possible, so that we can manually get rid of those false positives. In our implementation, we do not take into account the first frame and the last frame for every video sequence.

It turns out that we get in total 8123 pairs of potential shot transition candidates from 404 video sequences out of 419 video sequences of SAIL-VOS. In other words, we only need to look at 12% of the entire dataset to detect shot transitions.

3.2.2 Annotation GUI

The next step is similar to building a shot transition web-based annotator using HTML/CSS and Python for annotating shot transition in our dataset, and similar to the object category annotator described in Section 3.1. The program will read every potential shot transition pair of frames from the previous stage, and the annotator will need to label whether or not the current pair of frames are in a shot transition. We will keep track of all frame pairs that are marked as shot transition in a JSON file for further processing. The interface of the program can be found in Figure 13. The frame pairs will appear side by

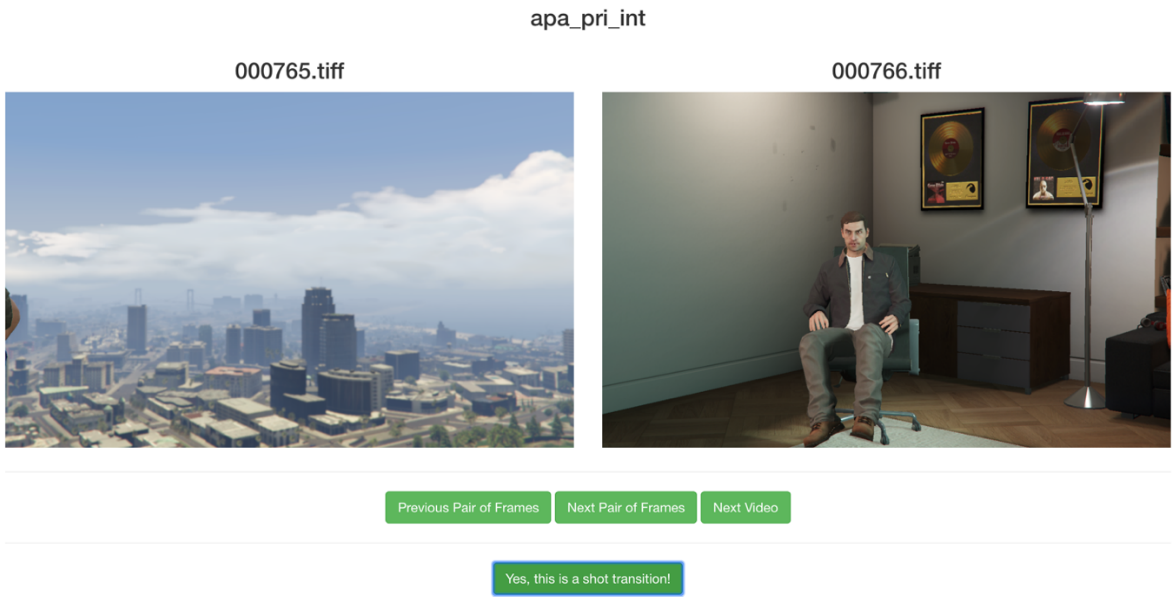


Figure 13 Example of Shot Detection Annotator Interface

side for the sake of easier annotation. The name of the video sequence and the index of the frame will be displayed above the images. Below the images, there are two rows of buttons. On the first row, one can click onto the previous pair of frames, or the next pair of frames, or the next video. The program will be able to notify the annotator if the current pair of frames is the first or last pair of frames, or if the current video is the first or last video. On the second row, one can confirm the current pair of frames as a shot transition by clicking the button "Yes, this is a shot transition!". And the program will take down the current pair of frames in the output file. Logically, if the annotator considers the current pair of frames as a shot transition, he/she needs to click "Yes"; if not, he/she needs to click "Next Pair of Frames" to pass the current pair and check the next pair.

Examples of not shot transition pairs and shot transition pairs can be found in Figure 14 and Figure 15. It is worth mentioning that many pairs of frames are considered as a potential shot transition but actually are not a shot transition simply because objects present at those frames move somewhat faster than expected but can still be thought of as a continuous action from the previous frame to the current frame. Those pairs of frames that are labeled as shot transition are so labeled because either the camera changes focus from one object to another, or the camera makes a huge movement between frames through a large translation or a rotation. In that case, we are unable to infer the current frame based on the previous frame.

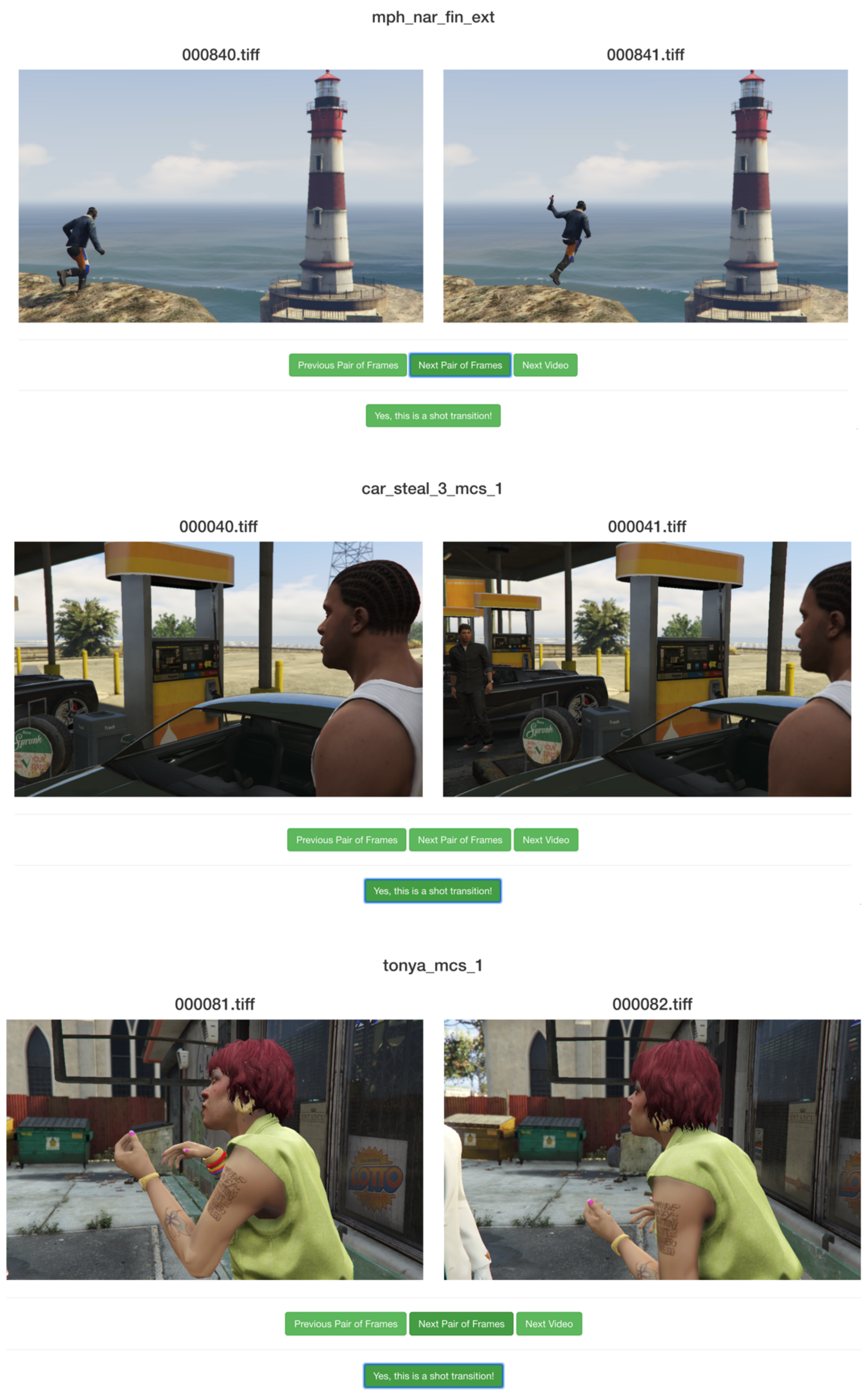
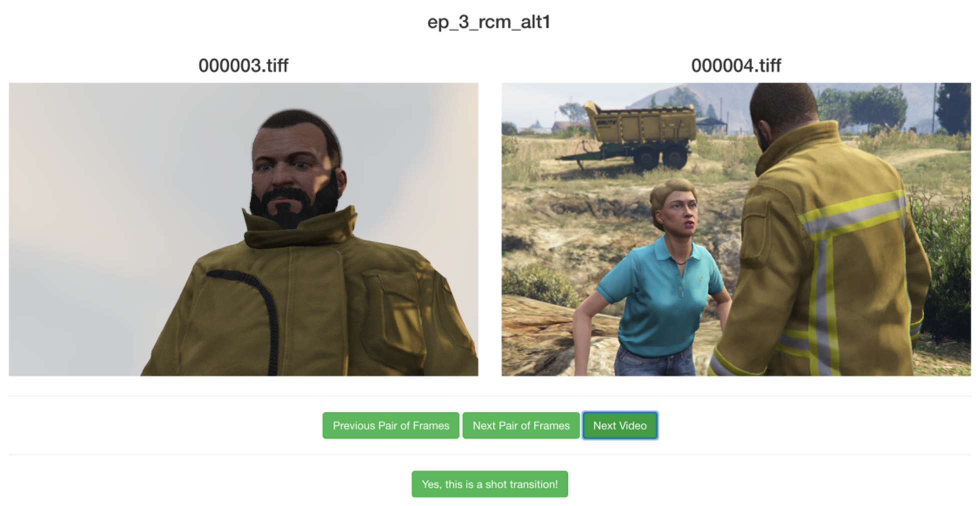
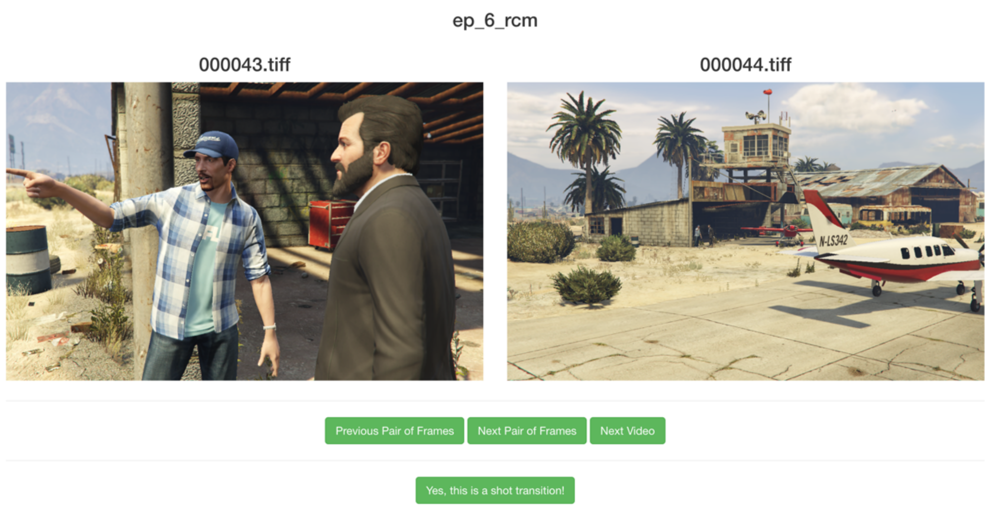


Figure 14 Examples of Pair of Frames Labeled as not a Shot Transition





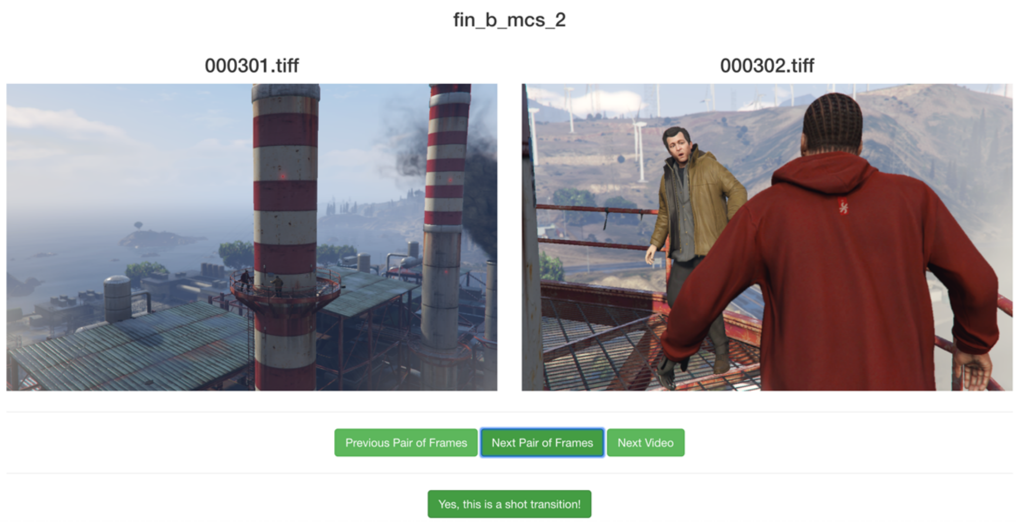


Figure 15 Examples of Pair of Frames Labeled as a Shot Transition

3.2.3 Results and Statistics

Table 1 shows the annotation results for shot detection. Of the video sequences in SAIL-VOS, 92.8% contain at least one shot transition. On average, around 13 shots are present in a single video sequence, and each shot has around 24 frames and lasts around 4.4 seconds.

Table 1 Statistics Table for Shot Detection

Statistics	393 video sequences containing shot transitions	The entire dataset (419 video sequences)
Number of Frames	128,229	131,335
Time Duration (s)	23,449.805	24,008.496
Number of Shots	5,330	5,356
Number of Shots per Video	13.562	12.783
Number of Frames per Shot	24.941	24.517
Time Duration per Video (s)	59.669	57.300
Time Duration per Shot (s)	4.400	4.482

4. Semantic Amodal Instance-level Video Segmentation

4.1 COCOA Dataset

The COCOA dataset [13] is a large densely annotated semantic amodal segmentation dataset based on 5000 images from the MS-COCO dataset [10]. To achieve this, annotators are required to annotate both modal and amodal masks for all semantically meaningful objects in an image, as well as their corresponding depth orderings. Figure 16 and Figure 17 show an example of the COCOA dataset. The COCOA dataset with complete annotations and useful APIs allows us to explore video amodal segmentation at the very beginning. However, since the authors of the COCOA dataset did not release their model used to evaluate baselines, we cannot know the details of their video object segmentation model. We can only reproduce the results using their pretrained model. Table 2 shows the reproduced evaluation results, where average recall (AR) is adopted as the metric.

Table 2 Amodal Segmentation Evaluation

		AR1	AR10	AR100	AR1000	AR_none	AR_partial	AR_heavy
Amodal mask	Both stuff and things	0.0352	0.1353	0.2923	0.4344	0.4701	0.4600	0.3639
	Things only	0.0452	0.1647	0.3313	0.4576	0.4786	0.4974	0.3751
	Stuff only	0.0084	0.0540	0.1811	0.3672	0.4142	0.3675	0.3468

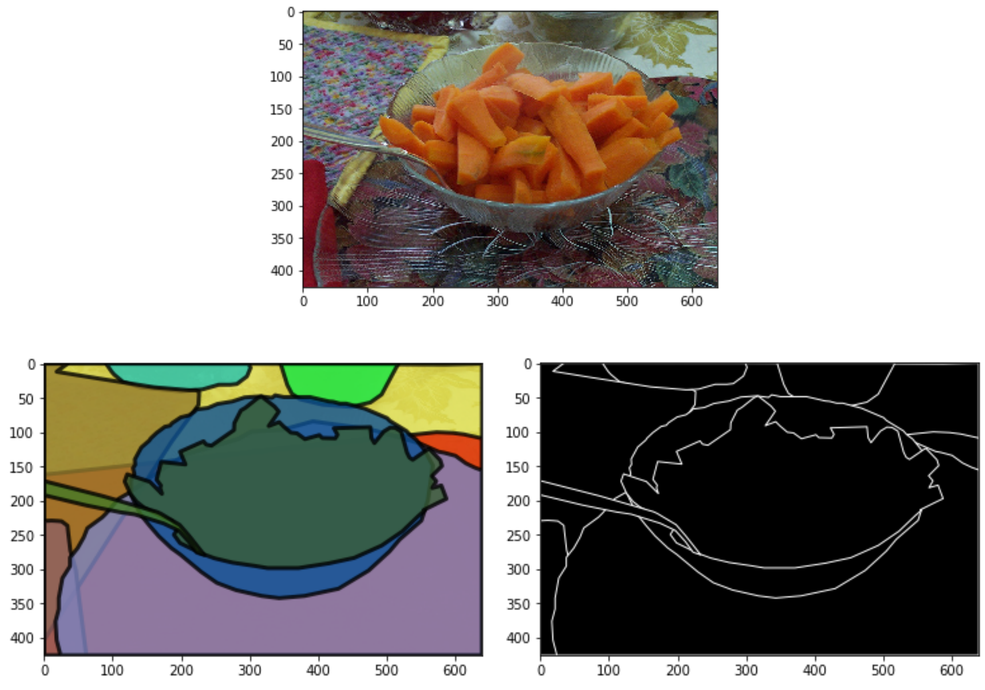


Figure 16 Example Image of COCOA Dataset. **First Row**: Original COCO Image. **Second Row**: Annotated Image with Depth Ordering Effect (left), Edge Map (right).

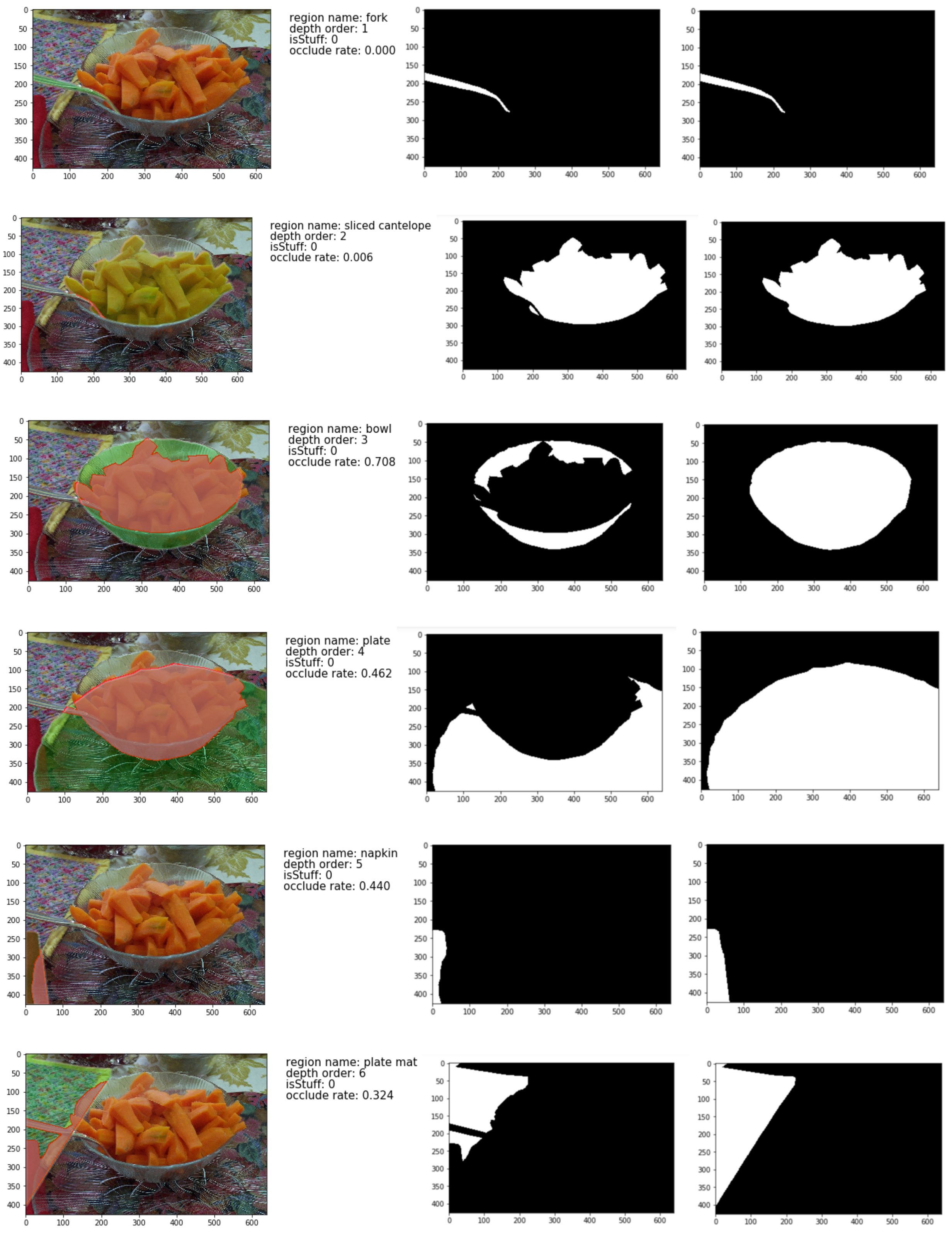


Figure 17 Annotated Instances Ordered by Depth Ordering. For Each Row, **Left**: K-th Object Highlighted in Green with Invisible Part Highlighted in Red, **Middle**: Modal Mask (Visible Mask), **Right**: Amodal Mask (Invisible + Visible Mask).

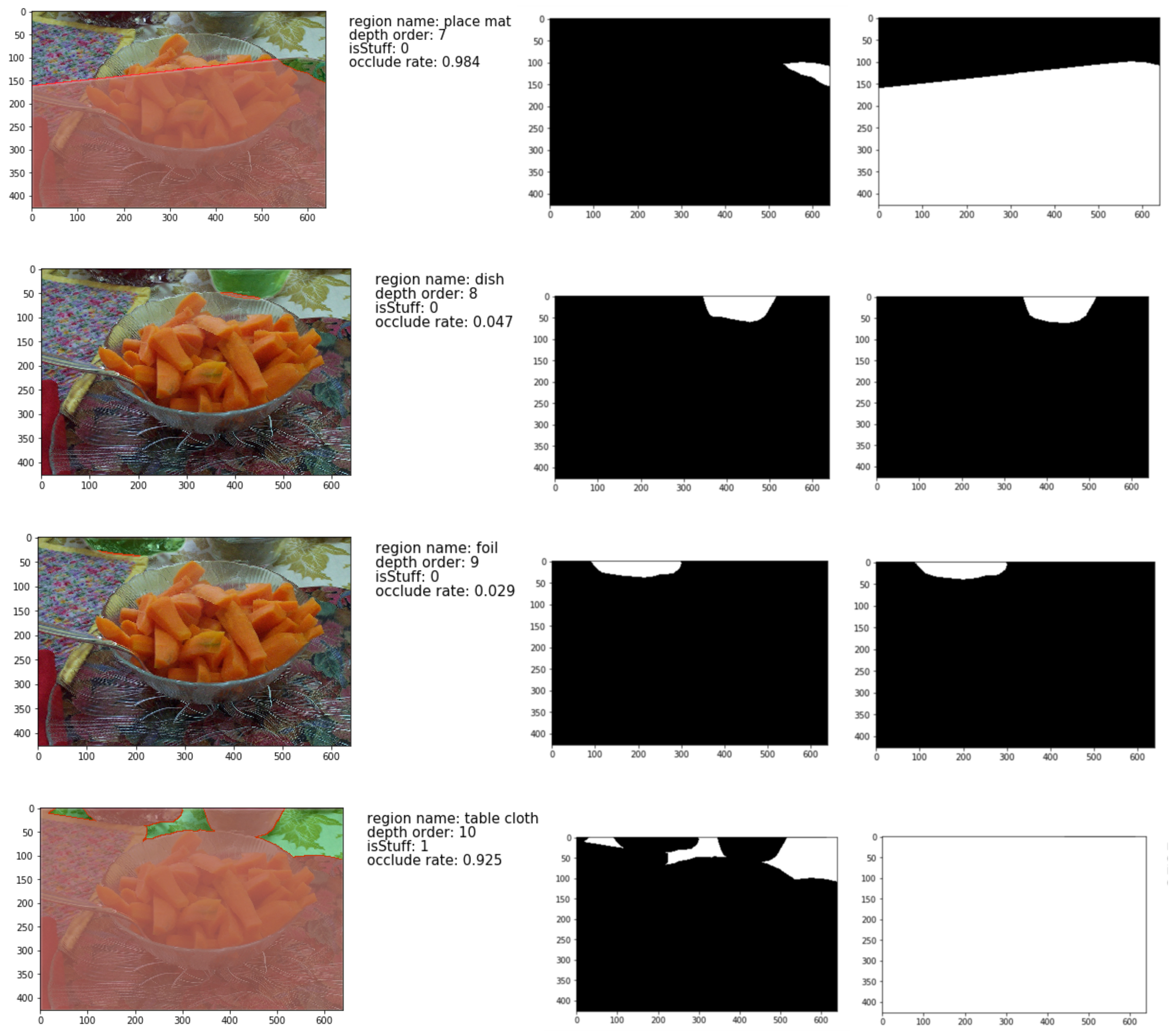


Figure 17 Continued

Compared to our proposed SAIL-VOS, the number of classes in COCOA cannot be counted since the annotator can name whatever category they want as long as it makes sense. There are in total 5000 images from the MS-COCO dataset [10], defining 46,314 objects, which is much smaller than the number of objects in SAIL-VOS. COCOA is split into 2500/1250/1250 images for train/val/test respectively [13].

4.2 DeepLab

DeepLab [16] is one of the state-of-the-art methods that address the task of semantic image segmentation. It makes three main contributions. First, convolution is highlighted with upsampled filters, also known as "atrous convolution", which not only controls which feature responses are computed within deep convolutional neural networks (DCNNs), but also enlarges the effective field of the filters to incorporate large context without increasing the number of parameters or the amount of computation. Second, atrous spatial pyramid pooling (ASPP) is proposed to better segment objects at multiple scales. Lastly, a fully connected conditional random field (CRF) is incorporated with the final DCNN layer to improve the localization of object boundaries. DeepLab reports state-of-the-art on PASCAL VOC 2012 benchmark [9], as well as other challenging tasks. Figure 18 illustrates the DeepLab model in details.

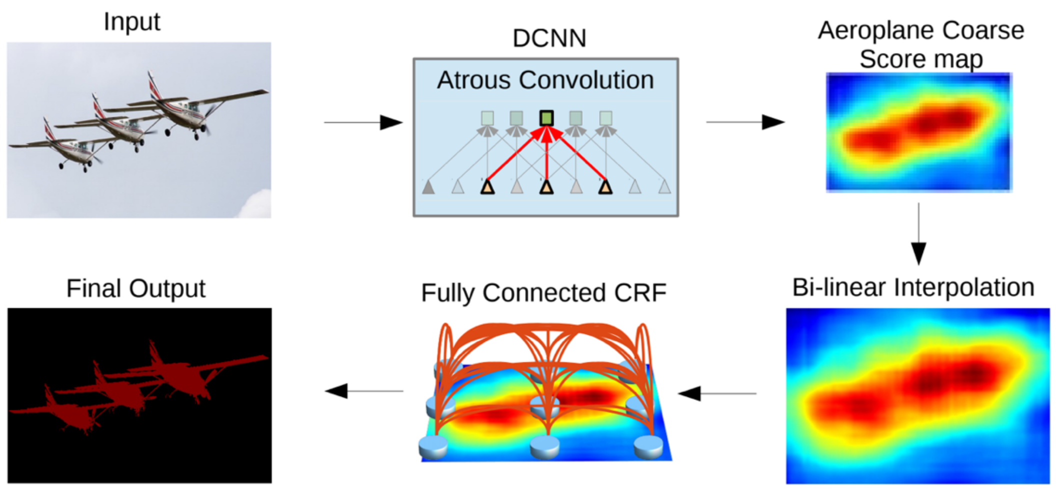


Figure 18 DeepLab Model Illustration. A Deep Convolutional Neural Network such as VGG-16 or ResNet-101 is Employed in a Fully Convolutional Fashion, Using Atrous Convolution to Reduce the Degree of Signal Downsampling (from 32x down 8x). A Bilinear Interpolation Stage Enlarges the Feature Maps to the Original Image Resolution. A Fully Connected CRF is then Applied to Refine the Segmentation Result and Better Capture the Object Boundaries. [16]

Due to the success of DeepLab, we decided to use it to evaluate on COCOA dataset [13]. Our approach is a little different from that of the original DeepLab as we want to finally predict an amodal mask instead of a modal mask for each object. There are two main considerations: on the one hand, instead of letting the model predict all semantically meaningful objects at the same time, we want the model to know which object it should focus on at each time; on the other hand, since we want to guide the model on the object of interest, we add an extra layer of the modal mask for the object onto the input. In this case, the model has a sense what it should look at and focus on in an image, and will predict its corresponding amodal mask. In our implementation, we iterate all the objects in COCOA dataset and generate a training sample, in total 22,163/12,753/11,398 samples for train/val/test respectively. We incorporate the modal mask as the 4th channel of the input data. It is noted that we are using binary modal masks in the range of [0,255] instead of [0,1]. This is due to the fact that [0,1] binary mask shows no progress after training. It makes senses that the RGB image we are using is based on [0,255] range so that the model might easily ignore the binary mask if it is in [0,1] basis.

The training results from iteration 0 to iteration 20000 can be found in Figure 19. As we can see, the model gradually picks up the object of interest and predicts the amodal mask. Compared with the amodal mask, the model seems to do a dilation or expansion on the modal mask to generate the amodal mask. Example inference results on val and test set can be found in Figure 20. The performance is measured in terms of pixel intersection over union (IoU) averaged across 81 classes (80 classes from MS-COCO dataset [10], plus background). For the val set containing 12,753 objects, the mean IoU is 0.905; and for the test set containing 11,398 objects, the mean IoU is 0.923. The results are very promising since over 0.9 IoU indicates heavy overlap between the prediction and ground truth.

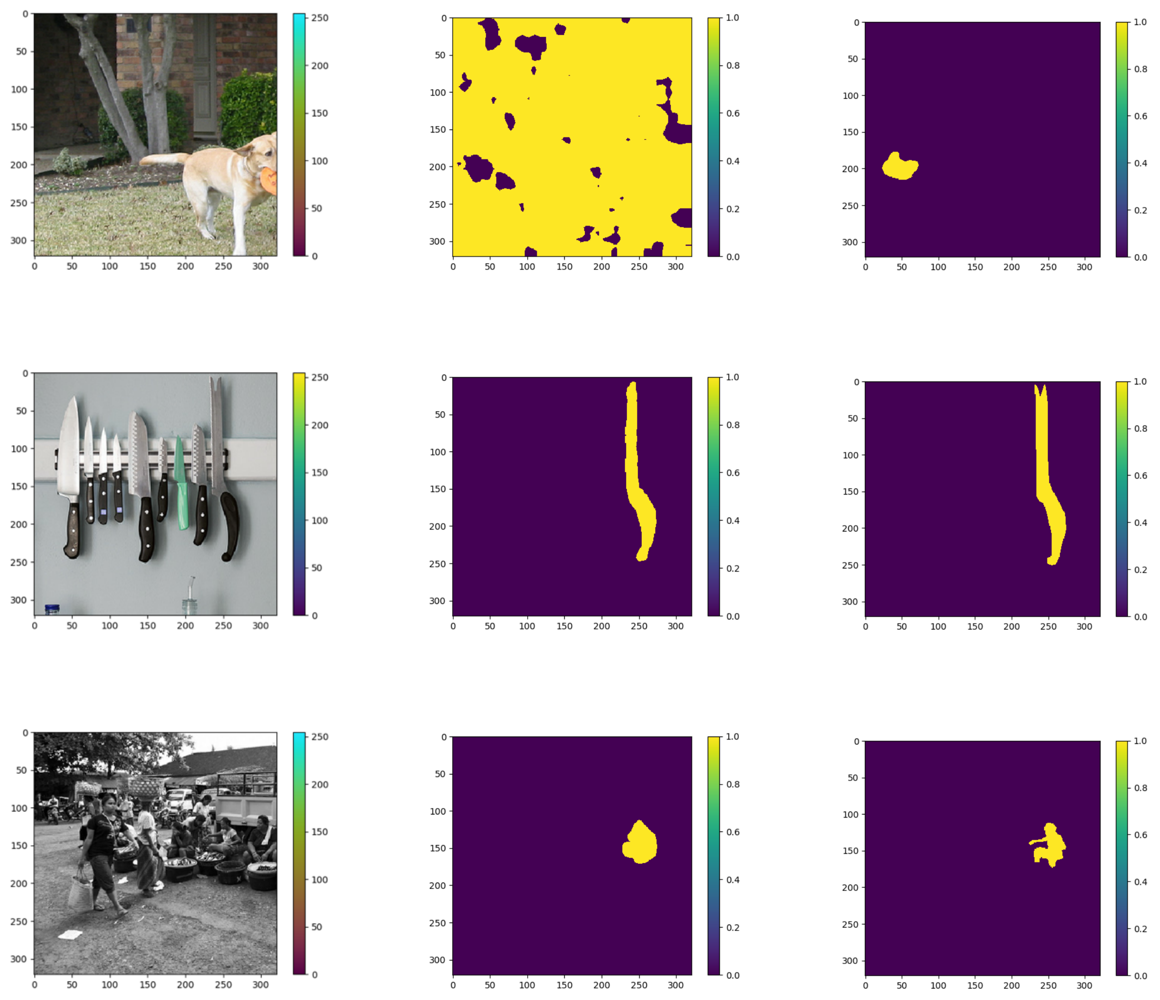


Figure 19 Training Results at Iteration 0, Iteration 5000, Iteration 10000, Iteration 15000, and Iteration 20000 from Top to Bottom. For Each Iteration, **Left**: Training Image, **Middle**: Predicted Amodal Mask, **Right**: Ground Truth Amodal Mask.

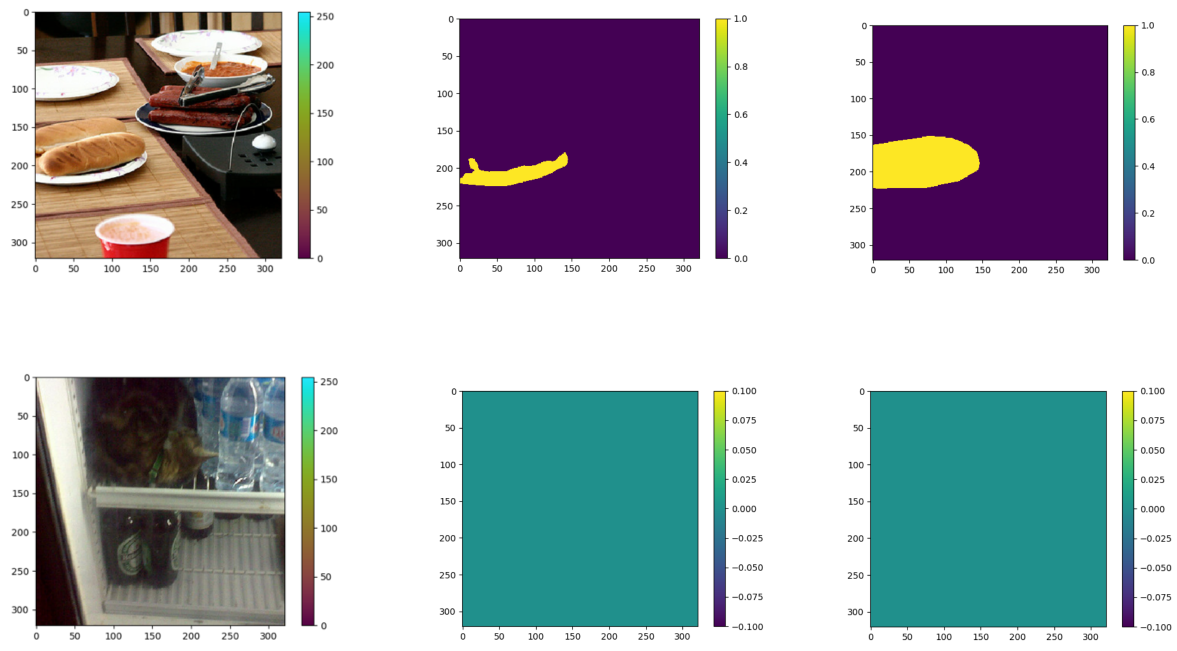


Figure 19 Continued

4.3 Our Model

Inspired by the success of semi-supervised learning in video object segmentation [17], [18], and the preliminary experiment using DeepLab [16], we develop our own model to tackle the video semantic amodal object segmentation task by predicting the amodal mask under the guidance of the RGB image from the current frame, modal mask from the current frame and amodal mask from the previous frame. As we know, a video is a time sequence of images or frames, and the temporal dependence of the video in our model is handled by enabling the network to predict the output of the current frame given the knowledge of the previous frame. In training mode, the input of the model includes three parts: the original RGB image, the perturbed modal mask, and also the perturbed amodal mask of the current frame. The RGB image is necessary as it provides us thorough and fundamental information about background, foreground and all kinds of details of a video frame. Similarly, modal mask, which is part of the amodal mask by definition, will help the network quickly pick up the object of interest. Note that we are not using the amodal mask from the previous frame prediction as in training phase we are not able to guarantee that the prediction from the previous frame is eligible to help the prediction of the current frame. In this case, using perturbation, we want to simulate it as the prediction-like amodal mask from the previous frame. The perturbation we are using is basically affine transformation including scaling, rotation, translation and sheer, and common morphological operations including erosion and dilation. It

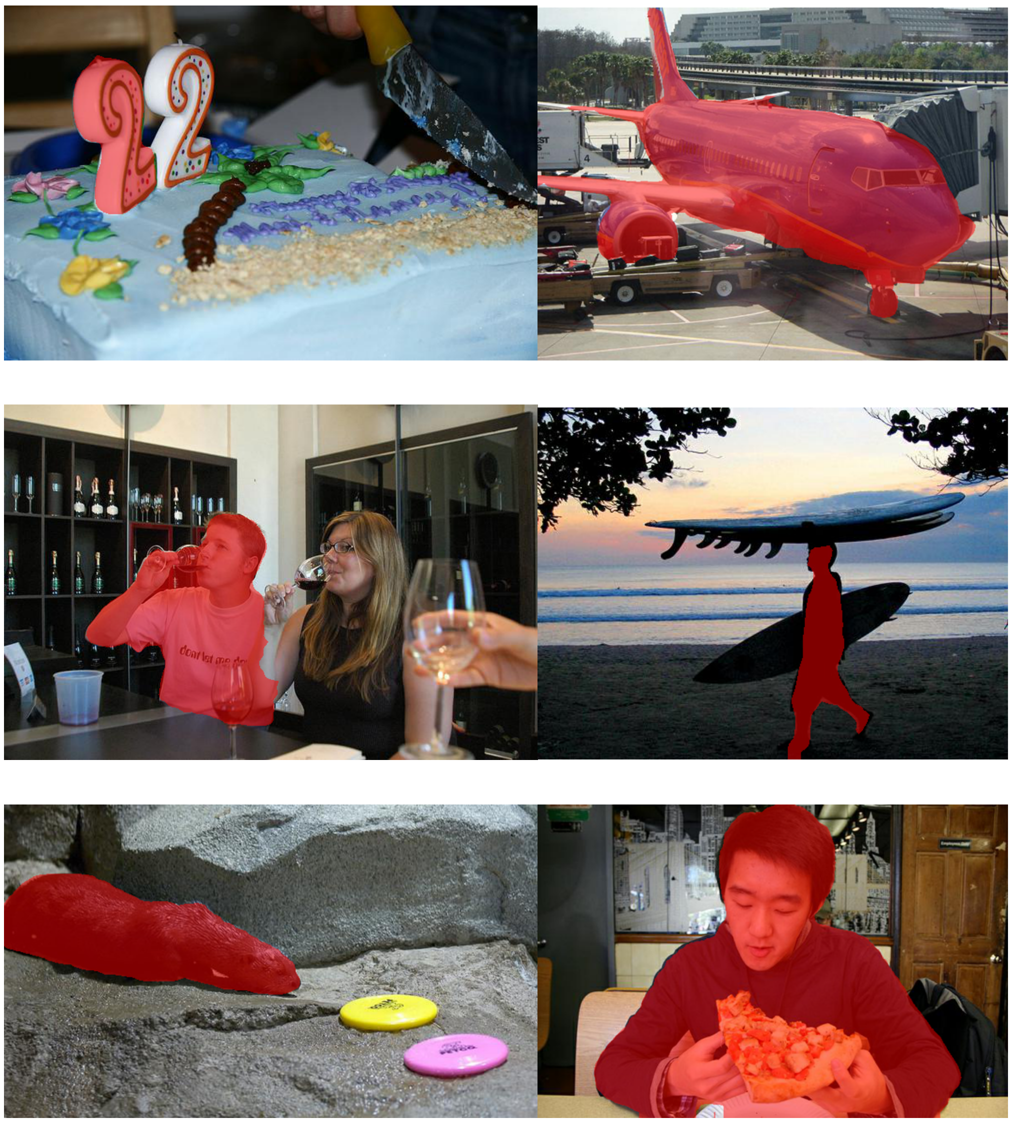


Figure 20 Amodal Segmentation Prediction using DeepLab Model on COCOA Dataset

is known that erosion and dilation are cancelling each other's effect, so we apply erosion to the mask at the probability of 33.3%, apply dilation to the mask at the probability of 33.3%, and apply nothing to the mask at the probability of 33.3%. Besides these traditional data augmentation methods mentioned above, we also try elastic deformation that is generated using a scaled normalized random displacement field in image space in order to correspond to uncontrolled oscillations of the hand muscles, dampened by inertia [40]. Since we still want to keep the shape and basics of the object mask to let the model know which part it should focus on, we perform the perturbation in as trivial a way as possible. However, the trivial perturbation will make nonsense to the model since the model will incline to simply memorize the amodal mask in the input and reproduce it for the output. Thus, the best perturbation we prefer is as trivial as possible so as to not change the mask, and also as distinctive as possible to make the model meaningful and useful. Table 3 shows the perturbation parameters we are using after experiments. Figure 21 shows the perturbation effects before and after on the mask images. Since we are applying random perturbations on the masks, the outputs after perturbation can be thinner or less conspicuous, which is less wanted, and the outputs can be larger and broader, which is very helpful for the prediction. Yet both cases satisfy our expectation to not only maintain the mask but also stimulate the training process.

Table 3 Perturbation Parameters

Affine Transformation	
Scale	Random (0.8, 1.2) in x axis, Random (0.8, 1.2) in y axis
Rotation	Random(-5^o , 5^o)
Translation	Random (-5, 5) in x axis, Random (-5, 5) in y axis
Sheer	Random(-5^o , 5^o)
Morphological Operations	
Erosion	Random size of disk-shape structuring element (3, 9)
Dilation	Random size of disk-shape structuring element (3, 9)
Elastic Deformation	
Sigma	Random (3, 7)
Scale	Random (3, 7)

The network remains the same as the previous experiment on the COCOA dataset [13], except that the channel of the first layer is modified to 5 instead of 4. There are 360644 training samples from our SAIL-VOS dataset, segmented by shots and objects from our annotation, which results in at least 36065 training steps with the batch size of 10 to cover the entire training dataset. It is worth mentioning that the objects that cannot exist for at least half of the duration of a shot are excluded. It makes sense that it is difficult for even humans to predict their location and shape in a frame for those objects that rarely show up, let alone for the network. As default, the learning rate is 0.00025, the momentum is 0.9 and the training steps are set to 60000. The training takes much longer than expected. Possibly it is due to the applied perturbation that costs longer to prepare the data for the network. The training curve is shown in Figure 22. We notice that the loss quickly drops in the first 5000 steps, and bounces to around

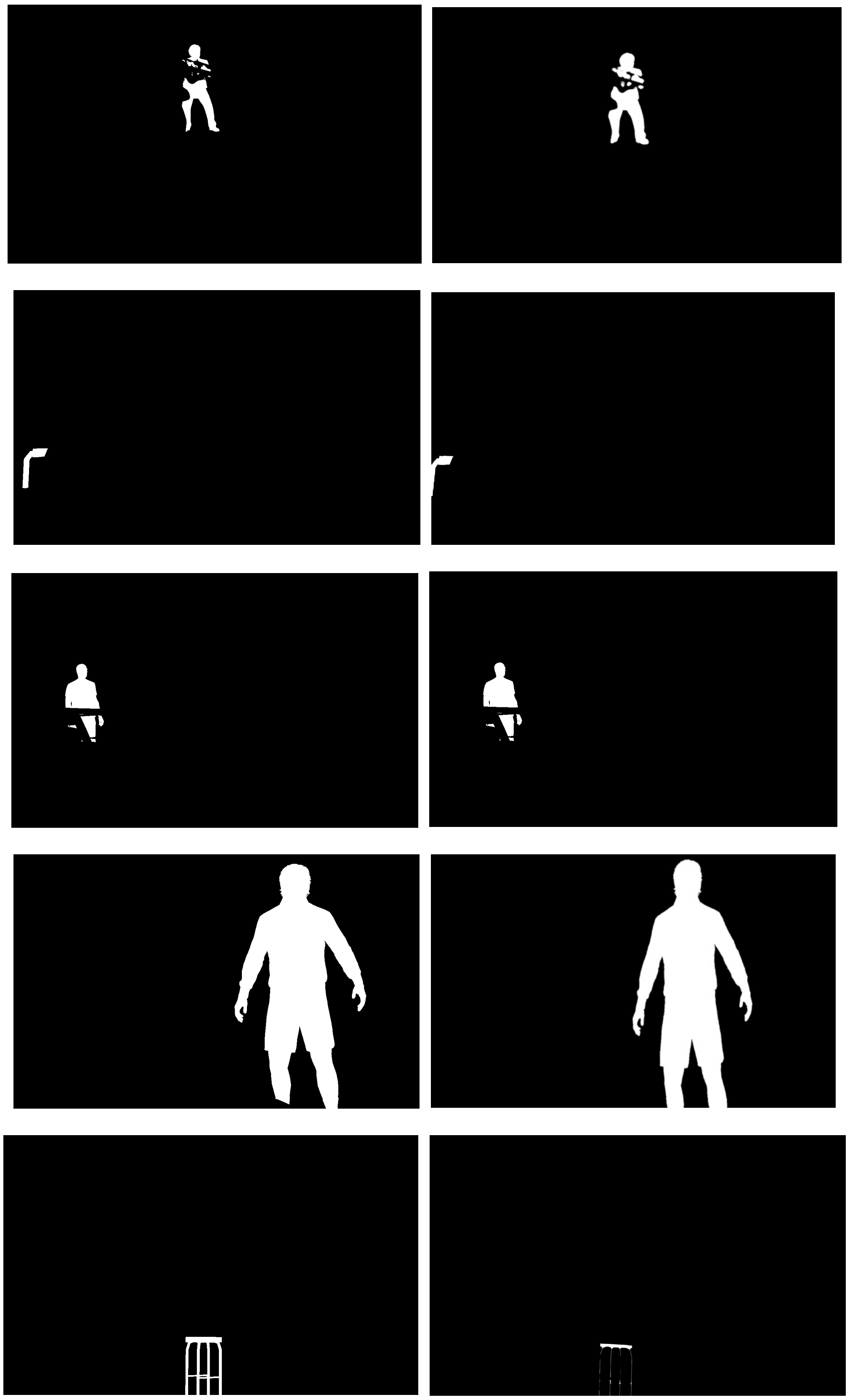


Figure 21 Perturbation Effects Before and After. **Left**: Before Applying Perturbation. **Right**: After Applying Perturbation.

1.2 in the end. The loss function we are using is simply the pixel-wise softmax loss. Figure 23 and Figure 24 show the training results, which prove the effectiveness of our network in turn. From Figure 23, the network starts from predicting noise to picking up something during the first 5000 steps when the training loss decreases quickly. On the other hand, according to Figure 24, when training loss tends to be relatively stable from step 5000, the network seems to know how to make a correct prediction and the output is very close to the ground truth. Since we are using random perturbation for the input masks, it is noticed that the prediction sometimes suffers from the unstable random perturbation masks. But we are hoping that training the network more will mitigate this side effect.

In inference mode, similar to training, the input to the network contains three parts: the original RGB image of the current frame, the perturbed modal mask of the current frame, and the predicted amodal mask from the previous frame instead of the current frame. In this way, the network is implemented in an online mode, which will be left for our future work.

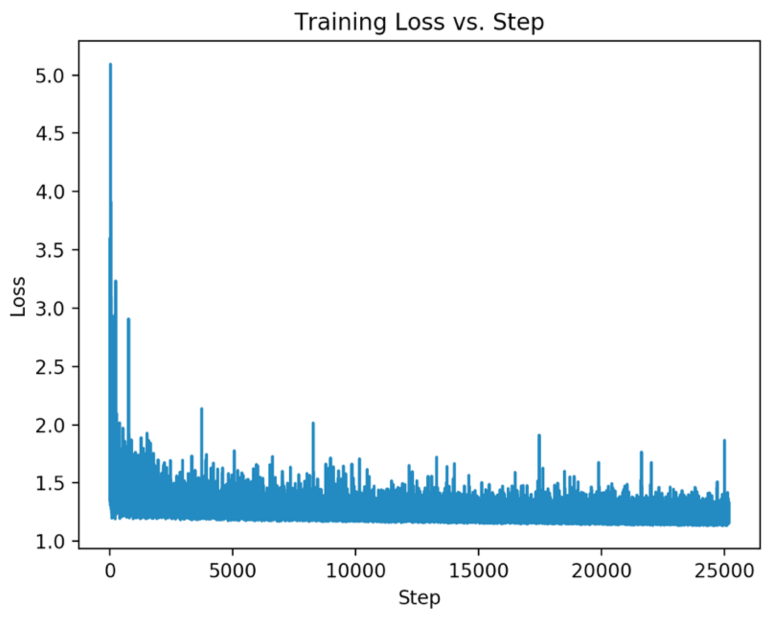


Figure 22 Training Curve

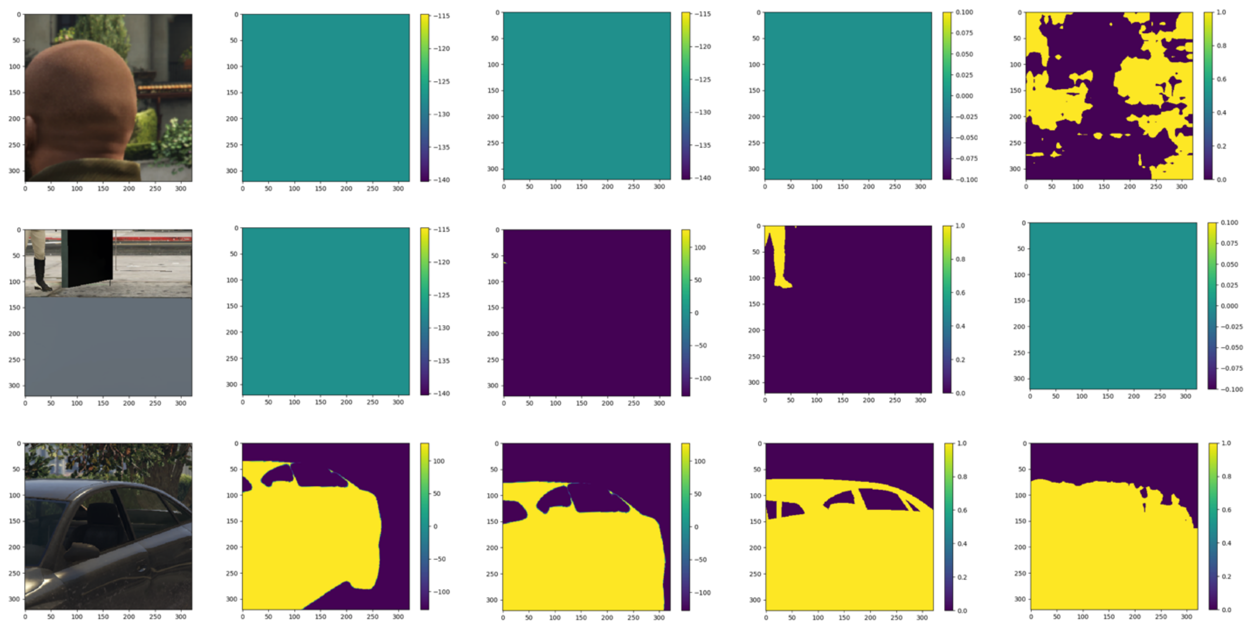


Figure 23 Training Results at Step 0, Step 2000 and Step 4000 from Top to Bottom. From Left to Right, **First**: RGB Image, **Second**: Perturbed Modal Mask, **Third**: Perturbed Amodal Mask, **Fourth**: Ground-truth Amodal Mask, **Fifth**: Predicted Amodal Mask.

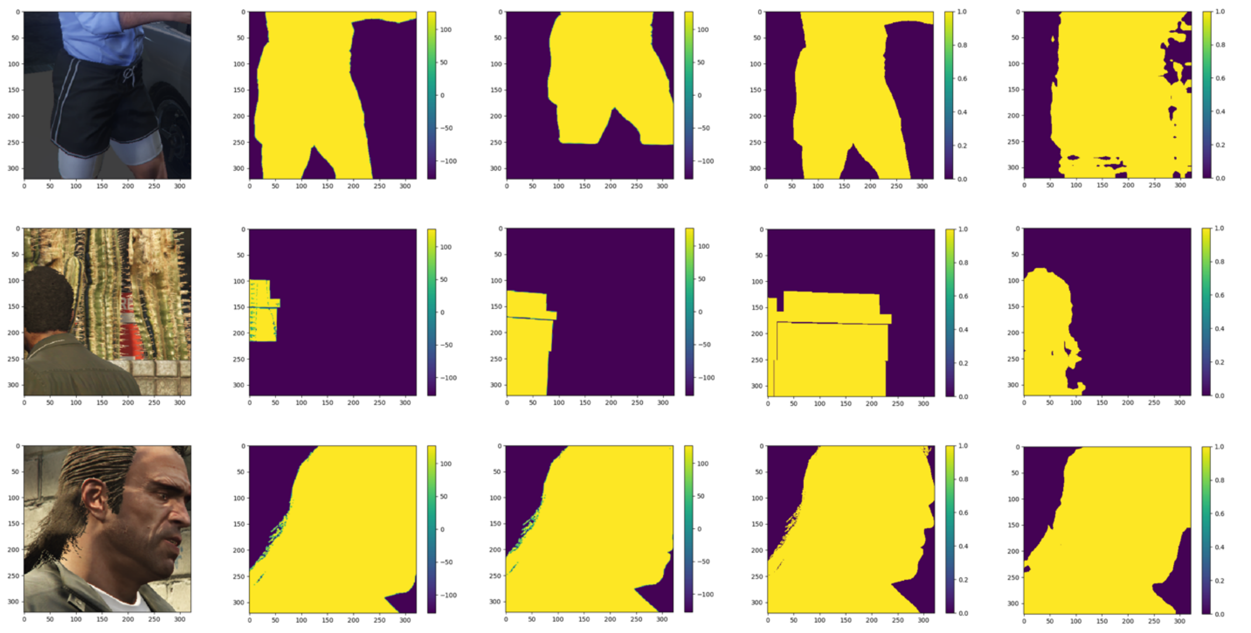


Figure 24 Training Results at Step 9000, Step 15000 and Step 25000 from Top to Bottom. From Left to Right, **First**: RGB Image, **Second**: Perturbed Modal Mask, **Third**: Perturbed Amodal Mask, **Fourth**: Ground-truth Amodal Mask, **Fifth**: Predicted Amodal Mask.

5. Conclusion

We support the new proposed semantic amodal instance-level video segmentation SAIL-VOS dataset by providing data annotation of object category and shot change with convenient tools. Thanks to the rich information in our SAIL-VOS dataset — frame-wise images with densely annotated, pixel-accurate modal and amodal masks with semantic labels — we further propose a new method of video semantic amodal segmentation with the goal of successfully applying DeepLab model on COCOA dataset. The new model is able to handle temporal dependence of a video shot and predict amodal mask of the current frame with the guidance of RGB image and modal mask of the current frame, and the amodal mask predicted from the previous frame. Promising training results are observed on the SAIL-VOS dataset using our new model. Thus, we are confident that our new model will eventually achieve better results on the video semantic amodal segmentation task. We hope this new dataset and one of its applications on video semantic amodal segmentation can stimulate new research in various directions.

References

[1] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, Lake Tahoe, 2012.

[2] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.

[3] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, 2014.

[4] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *International Conference on Learning Representations*, Scottsdale, 2013.

[5] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, Montreal, 2015.

[6] J. Shotton, J. Winn, C. Rother and A. Criminisi, "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *European Conference on Computer Vision*, Graz, 2006.

[7] P. H. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *31st International Conference on Machine Learning (ICML)*, Beijing, 2014.

[8] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, 2015.

[9] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *International Journal of Computer Vision,* vol. 88, no. 2, pp. 303-338, 2010.

[10] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollár, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*, Zurich, 2014.

[11] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Eighth IEEE International Conference on Computer Vision*, 2001.

[12] M. Maire, S. X. Yu and P. Perona, "Hierarchical scene annotation," in *British Machine Vision Conference*, Bristol, 2013.

[13] Y. Zhu, Y. Tian, D. Mexatas and P. Dollár, "Semantic amodal segmentation," 2016. [Online]. Available: https://arxiv.org/pdf/1509.01329.pdf.

[14] P. Alessandro, C. Leistner, J. Civera, C. Schmid and V. Ferrari, "Learning object class detectors from weakly annotated video," in *IEEE Conference on Computer Vision and Pattern Recognition*, Rhode Island, 2012.

[15] P. Federico, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.

[16] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," 2016. [Online]. Available: https://arxiv.org/pdf/1606.00915.pdf.

[17] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers and L. V. Gool, "One-shot video object segmentation," in *Computer Vision and Pattern Recognition*, Hawaii, 2017.

[18] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele and A. Sorkine-Hornung, "Learning video object segmentation from static images," in *Computer Vision and Pattern Recognition*, Hawaii, 2017.

[19] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," in *European Conference on Computer Vision*, Graz, 2006.

[20] C. Dorin and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 603-619, 2002.

[21] R. Carsten, V. Kolmogorov and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics (TOG),* vol. 23, no. 3, pp. 309-314, 2004.

[22] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers and A. W. M. Smeulders, "Segmentation as selective search for object recognition," in *The 13th International Conference on Computer Vision*, Barcelona, 2011.

[23] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, no. 11, pp. 2274-2282, 2012.

[24] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez and L. V. Gool, "Convolutional oriented boundaries," in *European Conference on Computer Vision*, Amsterdam, 2016.

[25] I. Kokkinos, "Pushing the boundaries of boundary detection using deep learning," in *International Conference on Learning Representations*, San Diego, 2015.

[26] B. Gedas, J. Shi and L. Torresani, "Semantic segmentation with boundary neural fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.

[27] T. Brox and . J. Malik, "Object segmentation by long term analysis of point trajectories," in *European Conference on Computer Vision*, Crete, 2010.

[28] S. Xie and Z. Tu, "Holistically-nested edge detection," in *IEEE International Conference on Computer Vision*, Las Condes, 2015.

[29] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.

[30] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan and T. Darrell, "BDD100K: A diverse driving video database with scalable annotation tooling," [Online]. Available: https://arxiv.org/pdf/1805.04687.pdf.

[31] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, Rhode Island, 2012.

[32] F. Xiao and Y. J. Lee, "Track and segment: An iterative unsupervised approach for video object proposals," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.

[33] N. Marki, F. Perazzi, O. Wang and A. Sorkine-Hornung, "Bilateral space video segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.

[34] S. E. Palmer, *Vision science: Photons to phenomenology*, Cambridge: MIT Press, 1999.

[35] R. Guo and D. Hoiem, "Beyond the line of sight: labeling the underlying surfaces," in *European Conference on Computer Vision*, Florence, 2012.

[36] S. Gupta, P. Arbeláez and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *IEEE Conference on Computer Vision and Pattern Recognition*, Portland, 2013.

[37] K. Li and J. Malik, "Amodal instance segmentation," in *European Conference on Computer Vision*, Amsterdam, 2016.

[38] D. Hoiem, "Projective geometry and camera models," University of Illinois, 2011.

[39] K. Hata and S. Savarese, "CS231A course notes 1: Camera models," Stanford University, 2017.

[40] P. Y. Simard, D. Steinkraus and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Seventh International Conference on Document Analysis and Recognition*, Edinburgh, 2003.