Supplementary Material for STOW: Discrete-Frame Segmentation and Tracking of Unseen Objects for Warehouse Picking Robots

Anonymous Author(s)
Affiliation
Address
email

A Dataset Detail

A.1 Synthetic Data

We build synthetic dataset using high-quality household models from GoogleScanned dataset[1] with two typical settings: a) Shelf and b) Tabletop.

Shelf environment In the context of shelf environments or other bin-based object arrangements, the objects are akin to books and are constrained to be oriented such that their shortest dimension is facing outward. This orientation scheme ensures that each object is guaranteed to have at least one visible face, but also leads to significant occlusion among objects. The camera is positioned at the front side of the bin to capture images of the scene, subject to random perturbations in the location in order to inject noise into the data.

Given that each bin contains a maximum of 3 to 5 objects, segmentation and tracking tasks become trivial if the scene contains fewer than 3 objects. To address this issue, image frames are only generated when the bin is nearly full.

We leverage approximately 900 objects sourced from the Google Scanned dataset, resulting in a training set of approximately 9000 image pairs. The remaining 100 objects are used to generate approximately 1000 image pairs for the test set.

Each image pair may exhibit the introduction of a new object, in addition to existing objects undergoing a flipping operation or relocation with a certain probability.

Tabletop environments Generating datasets of objects placed on table poses requires different settings, given the absence of walls and typically larger surface area, as compared to bin-based object arrangements. As a result, we adopt an alternative strategy for dataset generation. Specifically, each sequence consists of 15 images, with the first 10 images incrementally introducing new objects while shuffling existing objects between frames. No new objects are added in the final 5 frames, though the shuffling of existing objects persists. It is worth noting that due to the random placement of objects on the table, instances of full occlusion may occur in certain frames and subsequently reappear in subsequent frames.

To construct our training and testing datasets, we utilize 900 objects sourced from the Google Scanned dataset, producing 2000 sequences for the training set, with the remaining 100 objects utilized to generate 500 sequences for the test set.

A.2 Real Data

Similar to the synthetic evaluation we split the evaluation into shelf and tabletop environments as these are the most common scenarios encountered in the real world. To evaluate our scenario on

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

Figure 1: Some objects used during the evaluation. Objects vary greatly in shape and with different physical properties, some of them being partially transparent or wrapped in a bag.

challenging real-world scenarios we need a large variety of objects. Figure 1 depicts some of the objects used during the real-world evaluation for the tabletop scenarios. For the former category, shelf environments, we utilize an Azure Kinect RGB-D sensor. For the latter category, tabletop environments, we utilize an Intel Realsense D455 camera. Camera distance ranges from 1 to 1.5 meters. Each time an object is placed on the table or in a new bin a new image is captured. Objects can be rearranged to maximize space utilization as they are placed in the scene. After all the objects are placed in the scene we also displace the objects for a more refined evaluation. Camera images are manually labeled using the interactive segmentation of the object tracking framework XMem [2]. We collected and annotated more than 280 images with more than 150 different objects for the tabletop scenario and 220 images for the shelf scenario.

B Training and Inference Details

B.1 Loss

We keep the loss function that Mask2Former used for classification and mask prediction, which means binary cross entropy and dice loss for mask prediction, and softmax cross entropy loss for classification.

For object embedding head, we also used two losses: contrastive loss and softmax loss (or, say n-pair loss and InfoNCE loss).

Contrastive Loss We use contrastive loss modified from DCN[3] with hard-negative scaling from [4].

$$\mathcal{L}_{\text{matches}}(Q) = \frac{1}{N_{\text{matches}}} \sum_{N_{\text{matches}}} D(q_{t1}^{o_i}, q_{t2}^{o_i})^2 \tag{1}$$

$$\mathcal{L}_{\text{non-matches}}(Q) = \frac{1}{N_{\text{hard-neg}}} \sum_{N_{\text{non-matches}}} (0, M - D(q_{t1}^{o_i}, q_{t2}^{o_j})_{i \neq j}) \tag{2}$$

$$\mathcal{L}(Q) = \mathcal{L}_{\text{matches}}(Q) + \mathcal{L}_{\text{non-matches}}(Q) \tag{3}$$

where

$$N_{\text{hard-negatives}} = \sum_{N_{\text{non-matches}}} \mathbb{1}(M - D(q_{t1}^{o_i}, q_{t2}^{o_i}) > 0) \tag{4}$$

Here Q denotes all object tokens from images, and $q_t^{o_i}$ denotes the object tokens assigned to object o_i in frame t. M is the margin parameter used to ensure that non-matched pairs have a distance of at least M apart. The distance function D used in this approach is the cosine distance function, as in UCN [5] defined as:

$$D(q^i, q^j) = \frac{1}{2}(1 - r^i \cdot r^j) \tag{5}$$

Here, $r^i = \frac{f(q^i)}{|f(q^i)|}$ is the object embedding of object token i, which is computed by first forwarding the query to a linear layer f and then normalizing it to a unit vector. To expedite the training process,

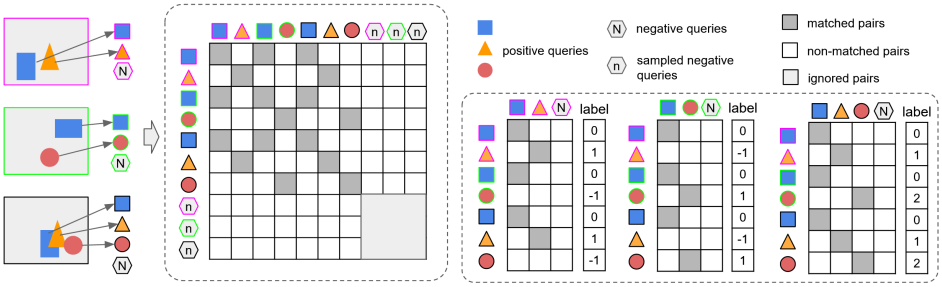

Figure 2: illustrates the tracking loss. In this example, three frames are sampled from a sequence, denoted with different border colors. Object tokens that match the objects in the images are represented by a blue square, an orange triangle, and a red circle. The hexagon denotes background object tokens that do not match to any objects. (a) the contrastive loss is computed between all frames, where matched pairs (dark gray) apply loss using Equation 1, non-matched pairs (white) apply loss using Equation 2, and ignored pairs (light gray) do not contribute to the loss. (b) the n-pair/InfoNCE loss is computed over all positive queries and queries from each frame. Equivalent to using a softmax cross-entropy while setting the label of the index of queries assigned to the same object.

we selectively incorporate a subset of negative queries to contribute to the contrastive loss, thereby enhancing its efficiency.

An illustration of the contrastive loss can be found in Figure 2. Assuming that three frames are sampled from a sequence during training, the contrastive loss will be computed between all frames. In the figure, matched pairs are denoted by dark gray and will apply loss according to Equation 1, while non-matched pairs are denoted by light gray and will apply loss according to Equation 2.

Softmax Loss We also modified the N-pair/InfoNCE loss used in CLIP[6].

$$\mathcal{L}_{\text{softmax}}(t) = -\sum_{k \in Q^+} \sum_{i \in O(t)} \frac{\exp(r_t^k \cdot r_t^i \cdot e^\tau)}{\sum_{j \in Q_t} \exp(r_t^k \cdot r_t^i \cdot e^\tau)} \tag{6}$$

$$\mathcal{L}_{\text{softmax}} = \frac{1}{T} \sum_{t \in 1, \cdots, T} \mathcal{L}_{\text{softmax}}(t) \tag{7}$$

Where Q^+ denotes all positive queries, $O(t)$ denotes all objects in frame t, and Q_t denotes all queries in frame t. This can also be understood as illustrated in Fig. 2-b, where the label of each row corresponds to the index of the query assigned to the same object. If there are n identical objects in the same frame, the softmax loss should be extended by copying all the queries in this frame n times, each time keeping only one query for that object. This allows queries containing the same object to be converted into multiple query sets, each one consisting of only the target object.

Thus the final tracking loss can be represented as

$$\mathcal{L}_{\text{track}} = \lambda_{\text{contra}} \mathcal{L}_{\text{contra}} + \lambda_{\text{softmax}} \mathcal{L}_{\text{softmax}} \tag{8}$$

B.2 Associator

An example of code demonstrating how to associate object tokens from a new frame to the trajectory bank built in previous frames. In implementation, we set $\sigma_{score} = 0.6$ and $\sigma_{\text{match}} = 0.2$ (similarity ranged in $[-1, 1]$).

```
def associate_one_frame(traj_bank, object_tokens_cur_frame, delta_score,
    delta_track):
    object_tokens = [x for x in queries_this_frame if x['score']>
        delta_score]
    num_trackers = len(traj_bank)
```

```
    Nq = len(object_tokens)
    similarity = torch.ones(num_trackers+Nq, num_pred)*delta_track

    # Extract object embedding from current frame's object tokens
    obj_embed = torch.stack([x['obj_embed'] for x in object_tokens])

    # Compute similarity between object embedding of trajectory and
        current frame's object tokens
    for traj_idx, traj in enumerate(traj_bank):
        traj_obj_embed = torch.stack([x['obj_embed'] for x in traj])
        sim = traj_obj_embed @ obj_embed
        similarity[traj_idx] = sim.max(dim=0)[0]

    # Perform Hungarian matching to find bipartite matching which have
        hightest similarity
    traj_indices, obj_token_indices = hungarian_matching(-similarity)

    # Update tracker
    for traj_idx, token_idx in zip(traj_indices, obj_token_indices):
        if traj_idx > num_trackers:
            # if it is not matched with any existing trajectory
            traj_bank.append([object_tokens[token_idx]])
        else:
            traj_bank[traj_idx].append(object_tokens[token_idx])
    return traj_bank
```

B.3 Training Details

We set the maximum number of iterations to 16k, using an initial learning rate of 1e-5, which was then dropped by 0.1 after 14k iterations. The number of classes is all set to 1 as we are aiming to handle unseen objects. For the shelf dataset, we trained our network with a batch size of 32 and leveraged 2 frames from each sequence, while for the table dataset, we set the batch size to 8 and randomly selected 4 frames from each sequence. To enhance the diversity of our dataset, we applied random color jittering and rotation to the input before feeding it to the network. The training process was executed on a single NVIDIA A-40 GPU and took approximately 13 hours.

During the training phase, we excluded the initial predicted object embedding, which was directly generated from the query feature. Additionally, when handling negative queries, we adopted a more selective approach by only considering queries whose IoU with any ground truth was lower than 0.6, rather than regarding all unmatched queries as negatives. This was motivated by the lack of clarity regarding which patches truly represent objects in unseen object settings, in contrast to close-set settings. More details can be found in the supplementary material.

Results can be shown in **??**. The experiments is conducted in the tabletop environment with same setting as in **??**. From **??** we can get following conclusions: 1) from image AP result on synthetic validation set and real test set, MinVIS shows better performance than Mask2Former Video and VITA, implying that the paradigm in Mask2Former Video and VITA that uses one object token to predict object masks in the whole sequence has worse performance than use individual object tokens for each frames. This can be understand as in discrete frames, with large movment and appearance changing between frames, it is challenging for a object tokens handle it; 2) from results

C Failure Case

As illustrated in Figure 5, the reasons for our method's occasional failure can be classified into two categories: *fail to segment* and *fail to track*.

Fail to segment Although the network produces accurate segment predictions in some frames, it may still encounter over-segmentation or under-segmentation issues. This implies that communica-

Figure 3: Result from different methods on the tabletop dataset. Methods ordered from top to bottom: MinVIS, Mask2Former-Video, VITA, and Ours (STOW)

tion between frames is not yet optimal, and there is room for improvement in leveraging information from adjacent frames to enhance segmentation performance.

Fail to track The network may also struggle with tracking when two objects have similar appearances, leading to false positive matches. The bottom-left example in Figure 5 demonstrates a case where objects with indices 3 and 5 are mistakenly switched. Furthermore, when the same object undergoes significant appearance changes (e.g., flipping), it may result in false negatives. The bottom-right example in Figure 5 shows a case where the network fails to recognize that object 1 in the first image is actually the same as object 5 in the second image.

References

[1] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022.

[2] H. K. Cheng and A. G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *Computer Vision–ECCV 2022: 17th European Conference,*

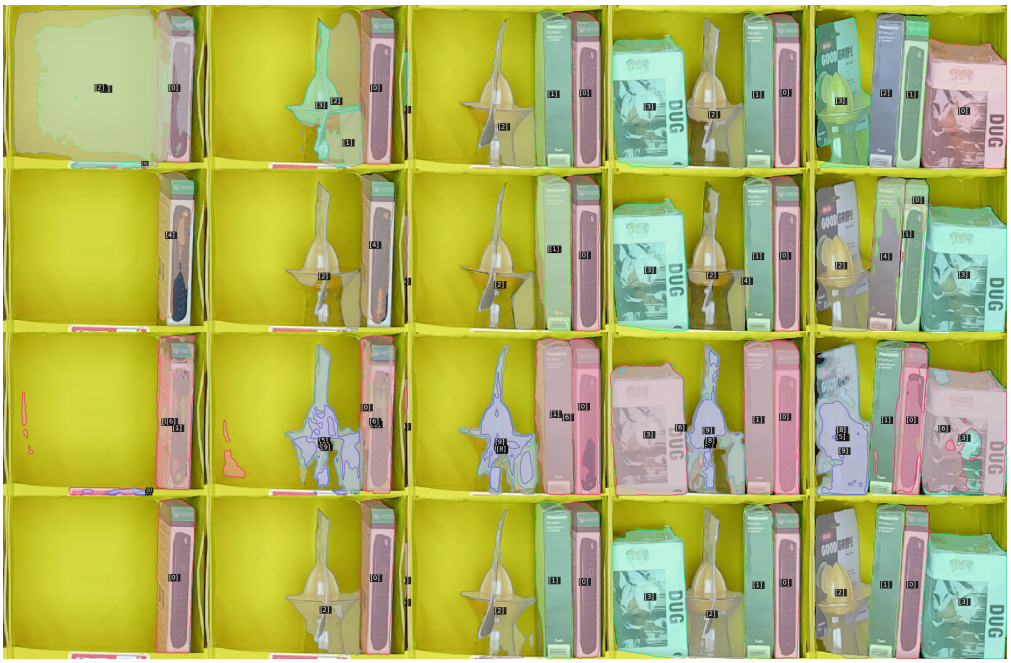

Figure 4: Result from different methods on the bin dataset. Methods ordered from top to bottom: MinVIS, Mask2Former-Video, VITA, and Ours (STOW)

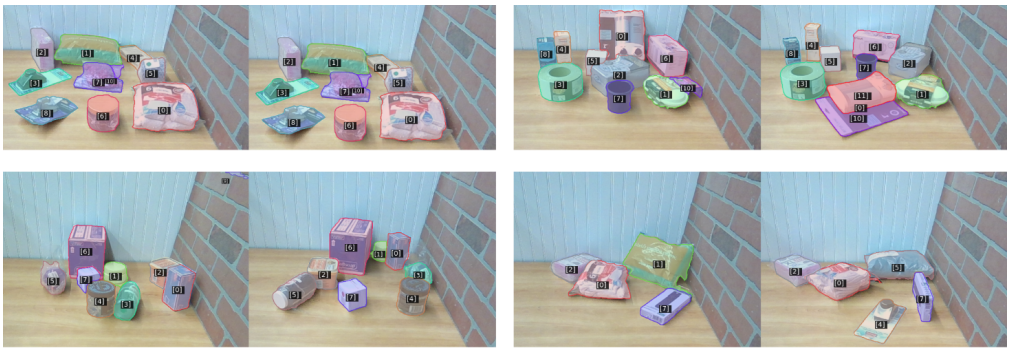

Figure 5: Failure cases on the tabletop dataset. Top left: under segmentation. Top right: over-segmentation. Bottom left: false positive (object mismatch). Bottom right: false negative (treat an existing object as new object)

Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII, pages 640–658. Springer, 2022.

[3] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.

[4] P. R. Florence. *Dense visual learning for robot manipulation*. PhD thesis, Massachusetts Institute of Technology, 2020.

[5] Y. Xiang, C. Xie, A. Mousavian, and D. Fox. Learning rgb-d feature embeddings for unseen object instance segmentation. In *Conference on Robot Learning*, pages 461–470. PMLR, 2021.

[6] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.