# Self-Supervised Object Detection from Egocentric Videos Supplementary Material

Peri Akiva[1,2] Jing Huang[2] Kevin J Liang[2] Rama Kovvuri[2] Xingyu Chen[2]
Matt Feiszli[2] Kristin Dana[1] Tal Hassner[2]
[1]Rutgers University [2]Meta AI

## 1. Implementation details

For the patch feature extractor and patch matching network, we use a ResNet-50 (random initialization) and a deformable transformer encoder to generate multi-scale patch features (see Figure 1 for example of patch indexing notation). For the object residual module, the number of clusters is $K = 16$ and feature dimension $D = 64$. For the random affine transformation $\mathbb{T}$, we sample random rotation and translation parameters from the ranges $(-10, 10)$ degrees and $(-15, 15)$ pixels respectively. For training, we use a batch size of 16, learning rate of 0.0001, momentum of 0.9, weight decay of 0.05, and number of negative samples of 200 patches per anchor patch. We first train the patch matching network for 8 epochs (everything else is frozen), after which the patch matching network is frozen (everything else is un-frozen) and used to predict multi-temporal patch matches for the main task objective. The patch feature extractor uses sine positional encoding, multi-scale self and deformable attention modules with 8 heads and 4 scales. The patch feature extractor is trained for 100 epochs or until convergence. During inference, we use $\eta = 50$, $\gamma = 0.25$, $\beta = 10$, and $\psi = 0.05$.

### 1.1. Pseudo-code

The implementation for the object residual module can be found in Pseudo Code 1, and the implementation for the patch matching module can be found in Pseudo Code 2.

## 2. Additional Qualitative Results

Figure 2 presents additional qualitative examples of viewing angle and illumination condition variation on the pre-smoothed cluster maps. Fig. 3, 4, and 5 present additional qualitative results for the Ego4D [1], EgoObjects [3], and COCO [2] validation sets. Figure 6 and Figure 7 present qualitative results of challenging cases and batch-wise smoothing outputs.

## References

[1] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022. 1

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[3] Lorenzo Pellegrini, Chenchen Zhu, Fanyi Xiao, Zhicheng Yan, Antonio Carta, Matthias De Lange, Vincenzo Lomonaco, Roshan Sumbaly, Pau Rodriguez, and David Vazquez. 3rd continual learning workshop challenge on egocentric category and instance level object understanding. *arXiv preprint arXiv:2212.06833*, 2022. 1

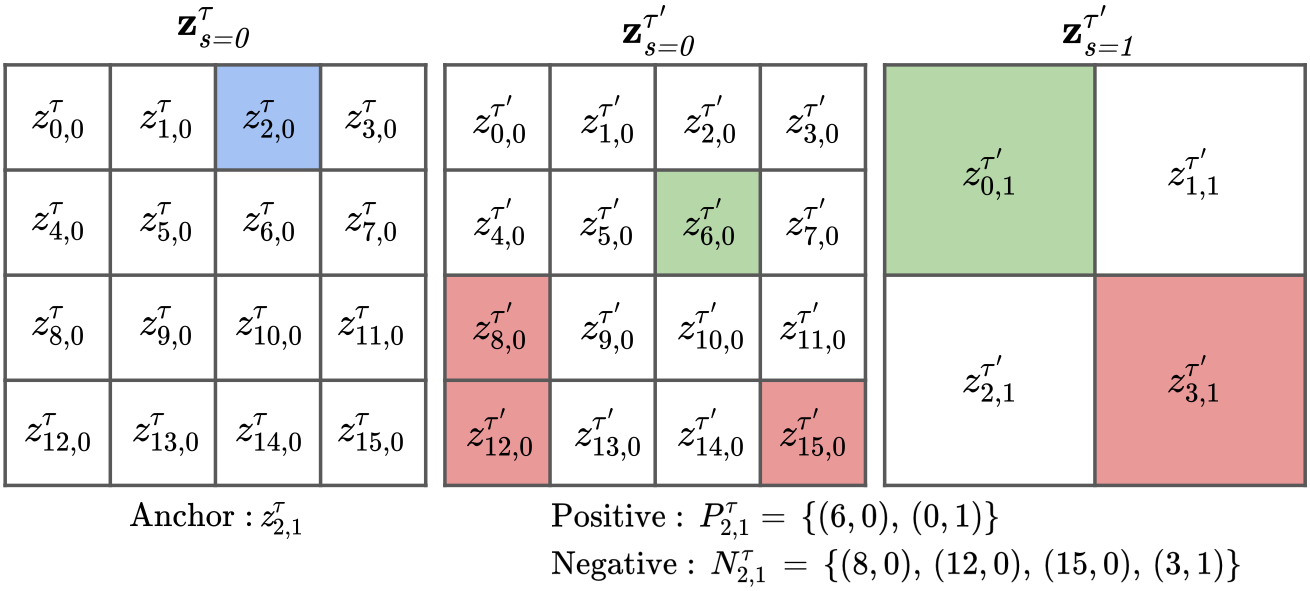


Figure 1: **Patch indexing notation example.** Anchor $z^{\tau}_{2,1}$ is matched with 2 positive patches highlighted in green, $z^{\tau'}_{6,0}$ and $z^{\tau'}_{1,1}$, and 4 negative patches highlighted in red, $z^{\tau'}_{8,0}$, $z^{\tau'}_{12,0}$, $z^{\tau'}_{15,0}$, and $z^{\tau'}_{3,1}$.

```python
class ObjectResidualModule(nn.Module):
    def __init__(self, D: int, K: int) -> None:
        """Object Residual Module.
        Args:
            D (int): number of channels to encode
            K (int): number of clusters
        """
        self.D, self.K = D, K
        self.clusters = nn.Parameter(K, D, requires_grad=True))
        self.scale = nn.Parameter(K, requires_grad=True)

    def normalize(self, x):
        """find weights of cluster centers using l2 distance
        Args:
            x (torch.Tensor): input features
        Returns:
            torch.Tensor: cluster-wise weights
        """
        num_clusters, in_channels = self.clusters.size()
        batch_size = x.size(0)
        scale = self.scale.view((1, 1, K))

        x = x.expand((batch_size, x.size(1), K, D))
        clusters = self.clusters.view((1, 1, K, D))

        norm = scale*(x - clusters).pow(2).sum(dim=3)
        return norm

    def accumelate(self, theta, x):
        """accumelation of weighted residuals
        Args:
            theta (torch.Tensor): learned cluster weights
            x (torch.Tensor): input features
        Returns:
            torch.Tensor: accumelated residuals
        """
        clusters = self.clusters.view((1, 1, self.K, self.D))
        batch_size = x.size(0)

        x = x.expand((batch_size, x.size(1), self.K, self.D))
        out = (theta*(x - clusters)).sum(dim=1)
        return out

    def forward(self, x):
        """forward function for Surface Encoder with
        Args:
            x (torch.Tensor): input features
        Returns:
            torch.Tensor: residual features
        """
        theta = F.softmax(self.l2_norm(x, self.clusters, self.scale), dim=2)
        residuals = self.accumelate(theta, x, self.clusters)
        return residuals
```

Listing 1: **Object Residual Module Pseudo-Code**

```
# x: input image

def patch_matching_train(x):

    rand_affine_params = affine_transformation_param_generator()
    x_transformed = affine_transformation(x, rand_affine_params)

    z1 = patch_matching_network(x)
    z2 = patch_matching_network(x_transformed)

    z1_transformed = affine_transformation(z1, rand_affine_params)


    # Both sets of patches are spatially aligned
    z1 = rearrange_as_patchs(z1)
    z2 = rearrange_as_patchs(z2)

    loss = InfoNCE(z1, z2)

    return loss
```

Listing 2: **Patch Matching Pseudo-Code**

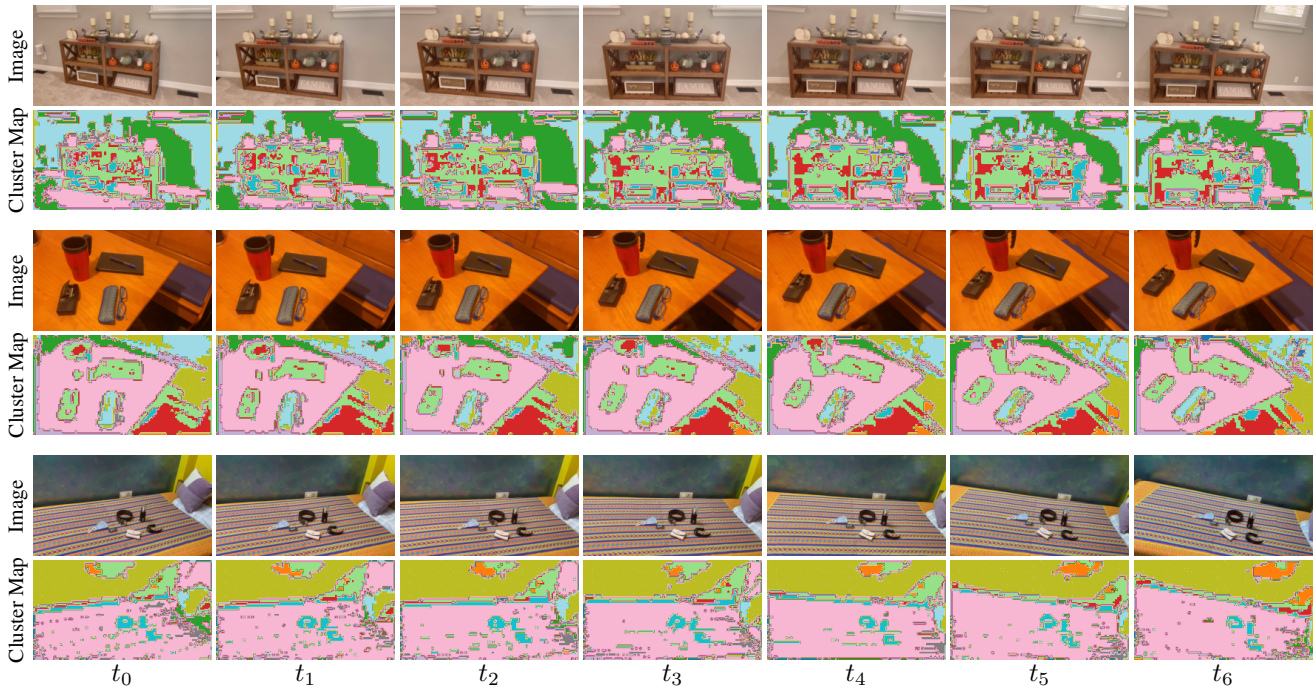


Figure 2: **Additional qualitative examples of viewing angle and illumination conditions variation on the pre-smoothed cluster maps.** Best viewed in color and zoomed.


Figure 3: Additional qualitative results on Ego4D validation sets.

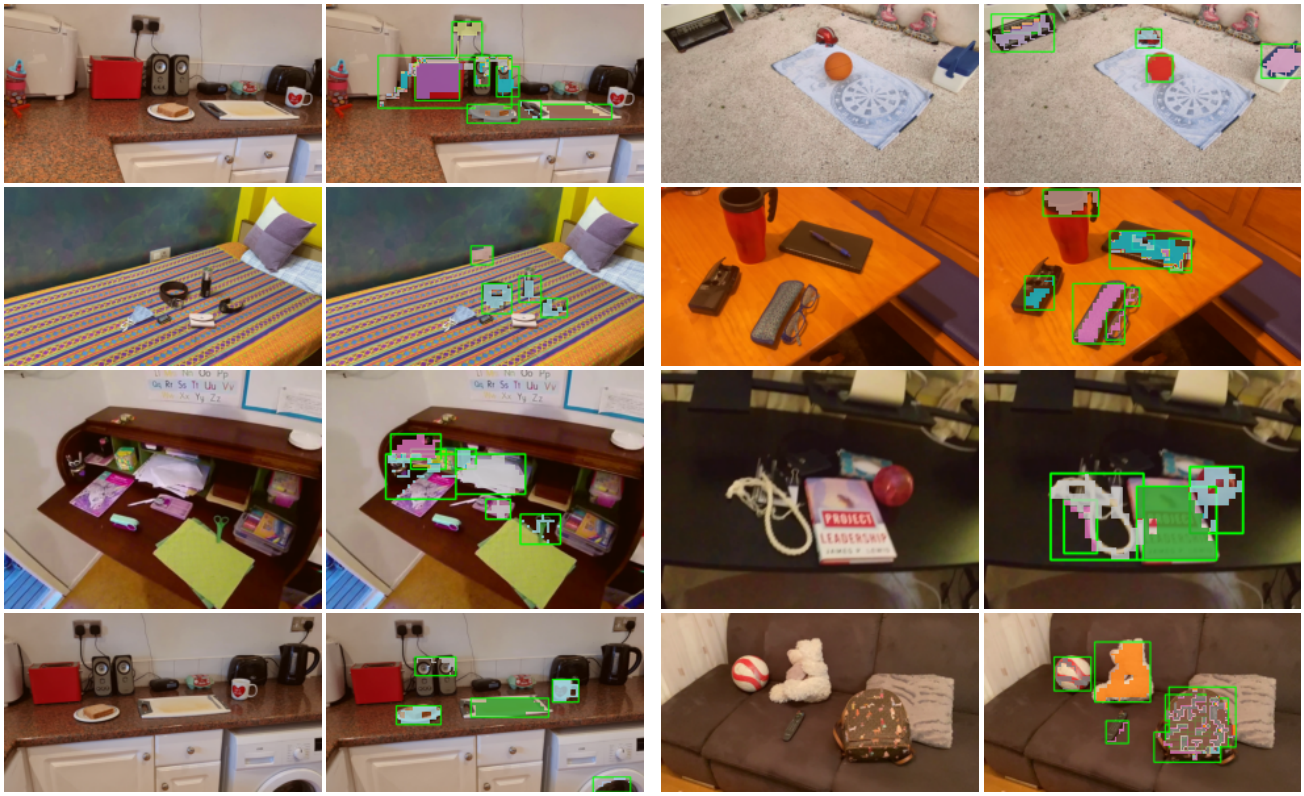


Figure 4: Additional qualitative results on EgoObjects validation sets.


Figure 5: Additional qualitative results on the COCO validation sets.

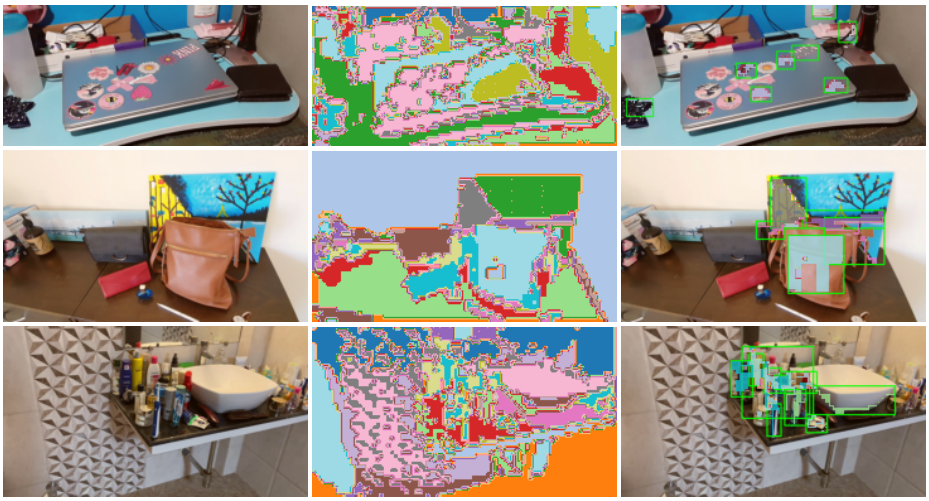


Figure 6: Qualitative examples of the challenging examples.


Figure 7: Qualitative examples of the smoothing operation.