Occlusion-Aware Object Representations in Unsupervised Video Tracking

Natasha Sharan
August 22nd, 2025

Motivation

- TASK: object tracking throughout videos
  - Supervised training is extremely costly
  - Unsupervised training has many challenges
- FOCUS: Occlusion-Aware Object Tracking
  - Occlusion -> geometric understanding
  - Video data -> temporal understanding
- IDEA: Incorporate spatiotemporal relationships

Hypothesis:

Modelling spatio-temporal relationships allows a model to be more robust towards visual geometric challenges such as occlusion. Understanding the behaviour of objects across time allows the model to maintain a degree of object permanence when objects are momentarily obscured in a frame.

Methodology

- Object Detector Detects initial candidate objects
  - Depth and pose network aid in providing geometrically sound detections
- Scene Graph Construction
  - For each frame, use information from detections, depth and pose to create node and edge embeddings for a scene graph
  - At each frame, update the evolving scene graph and incorporate temporal edges
- GNN Tracker
  - Refine node embeddings to determine which properties are most reliable in the presence of occlusion to maintain object identity across video
  - GNN processes the evolving scene graph at each frame

Training

Iterative pipeline:

Object detection

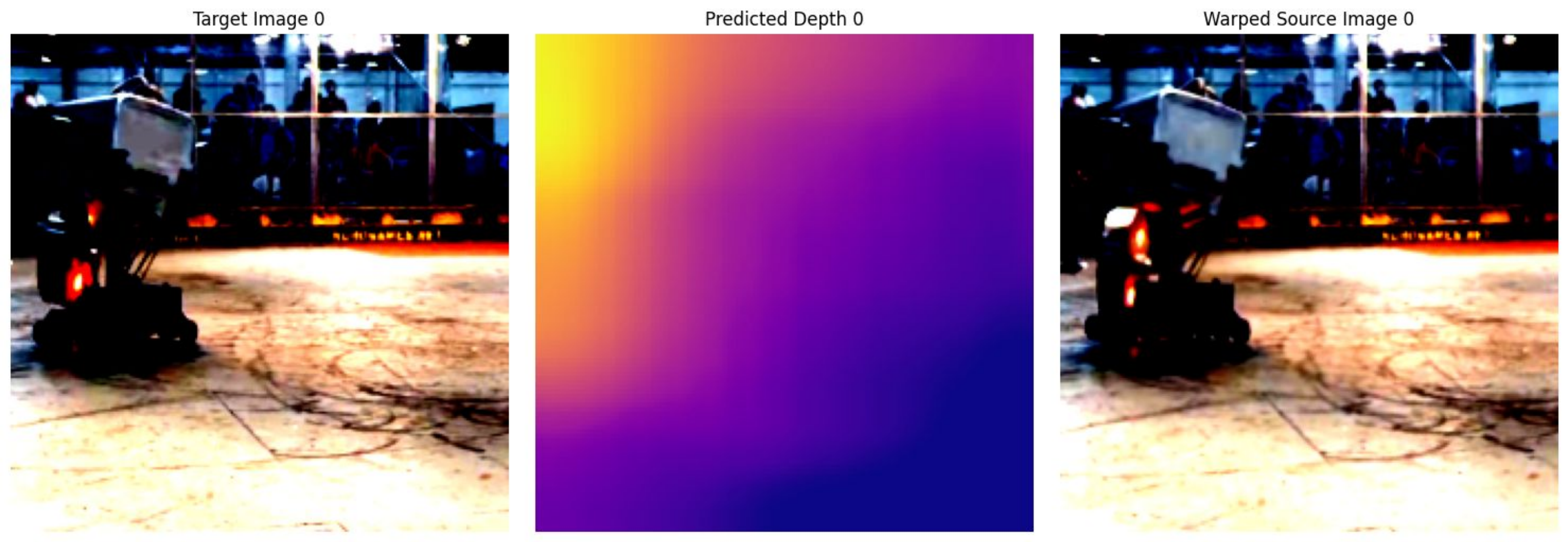
Scene graph construction

GNN tracker

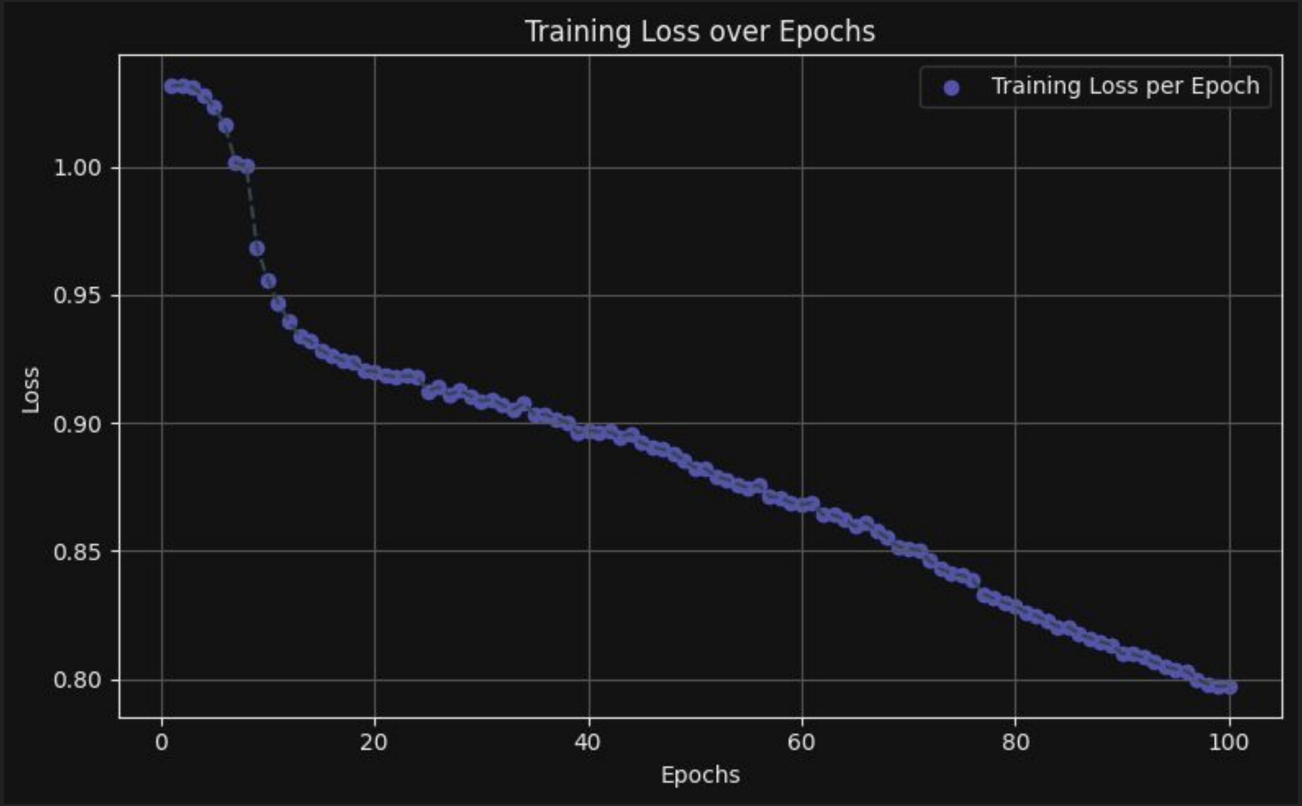
Object detection

Depth Estimation Module

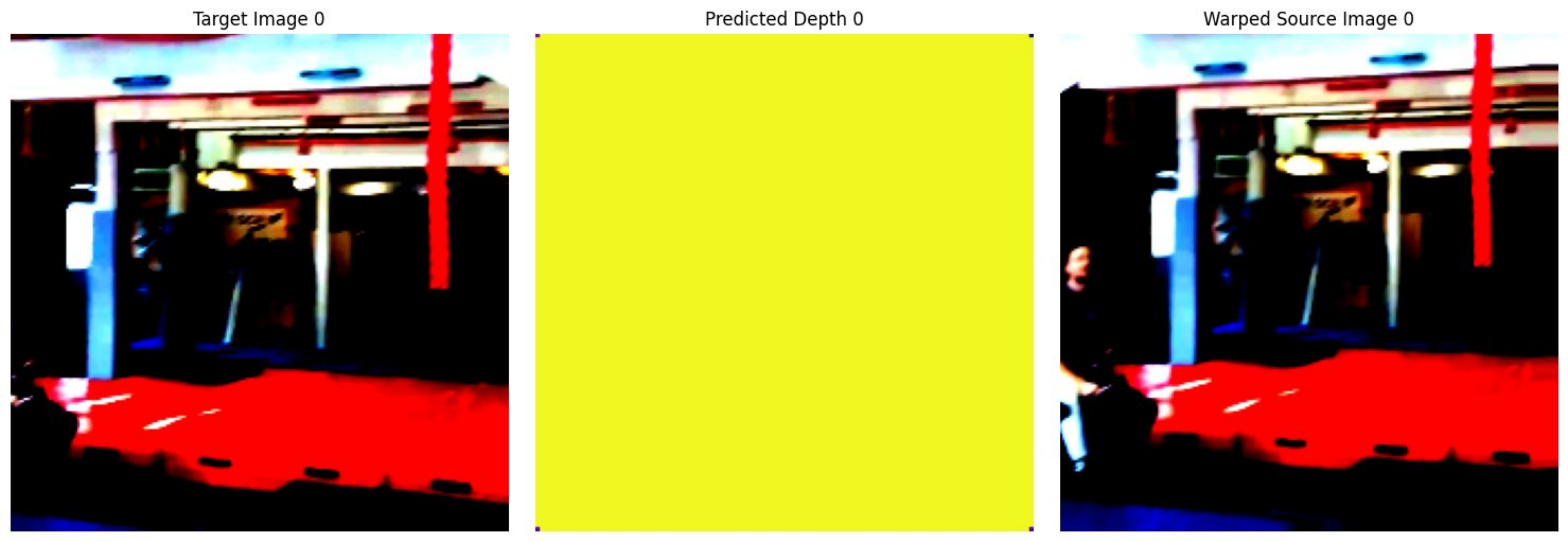
Depth Estimation Module



Depth Estimation Module



Depth Estimation Module

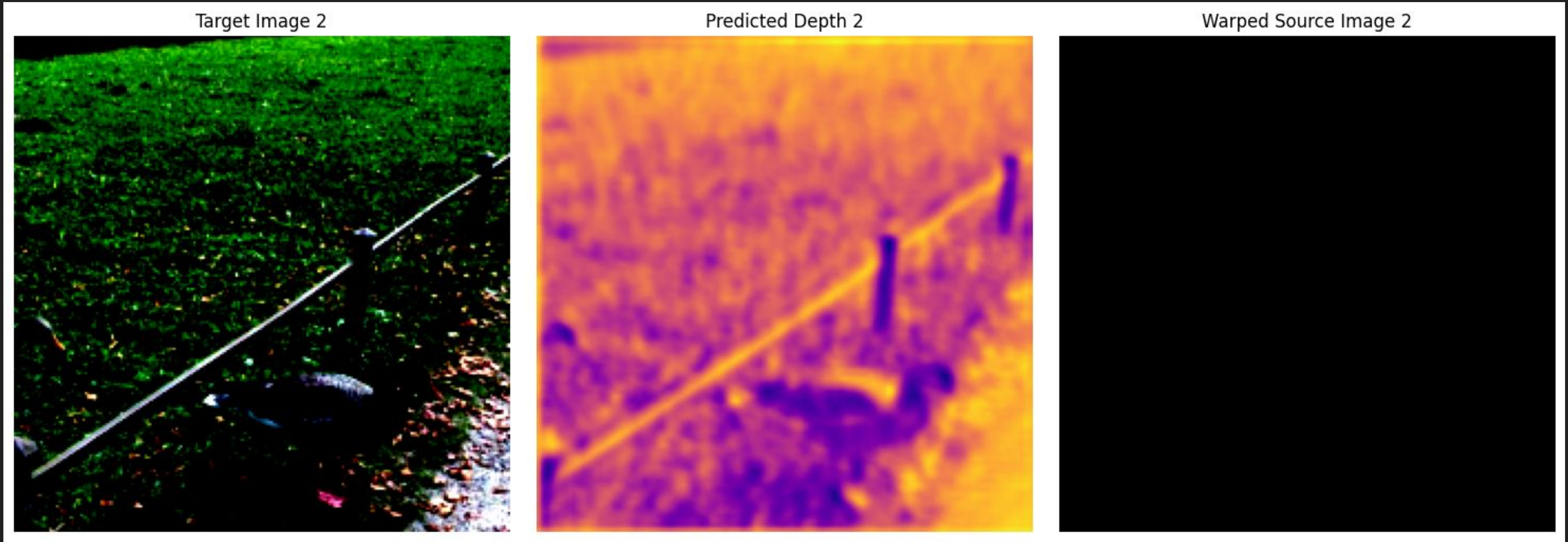


Depth Estimation Module

- Simple model: noisy results
- Incorporate skip connections
- Structural similarity Loss
- Encoder-decoder model

Depth Estimation Module

ssim + L1 loss -> depth trained, pose collapsed



Depth Estimation Module

ssim + L1 loss -> depth trained, pose collapsed

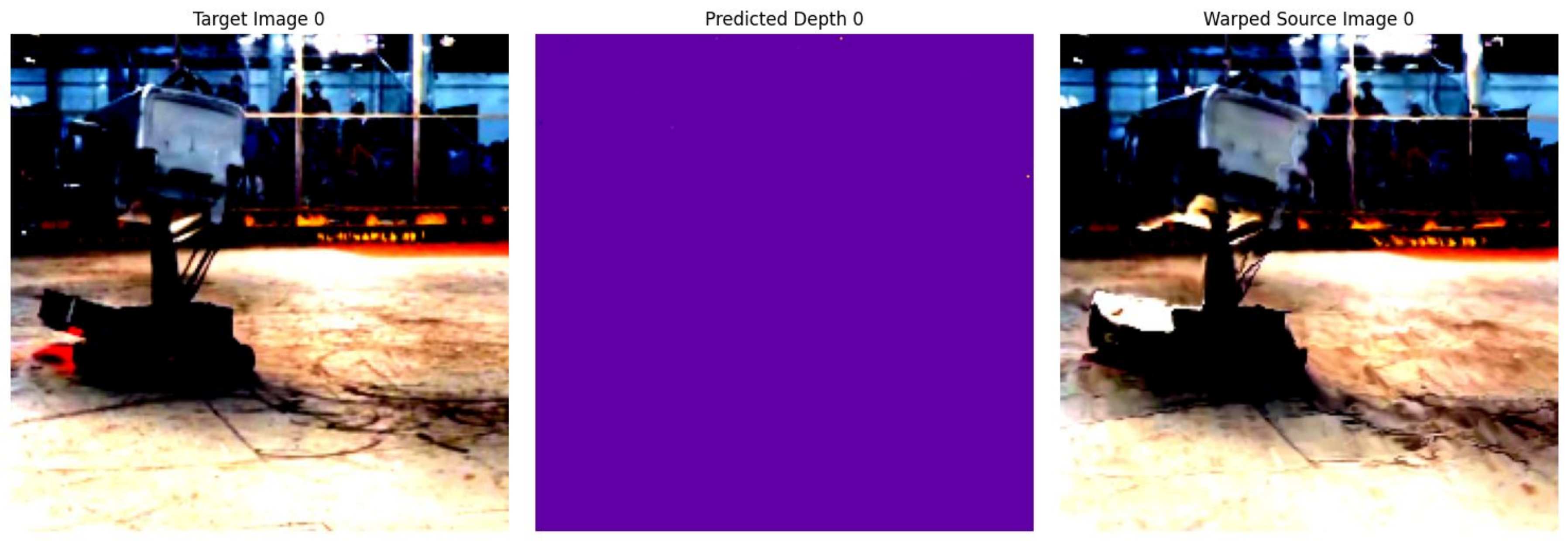
```
# photometric loss: guide warped image reconstruction with ssim and l1
def photometric_loss_ssim_l1(tgt_img, warped_img):
    # compute both losses
    ssim_loss = ssim(tgt_img, warped_img).mean(1, True)  # mean over channels
    l1_loss = torch.abs(tgt_img - warped_img).mean(1, True)

    # weighted average
    alpha = 0.85
    return (alpha * ssim_loss + (1 - alpha) * l1_loss).mean()
```

Pose gradients vanished in ssim computations

Depth Estimation Module

L1 loss -> depth collapsed, pose trained



Depth Estimation Module

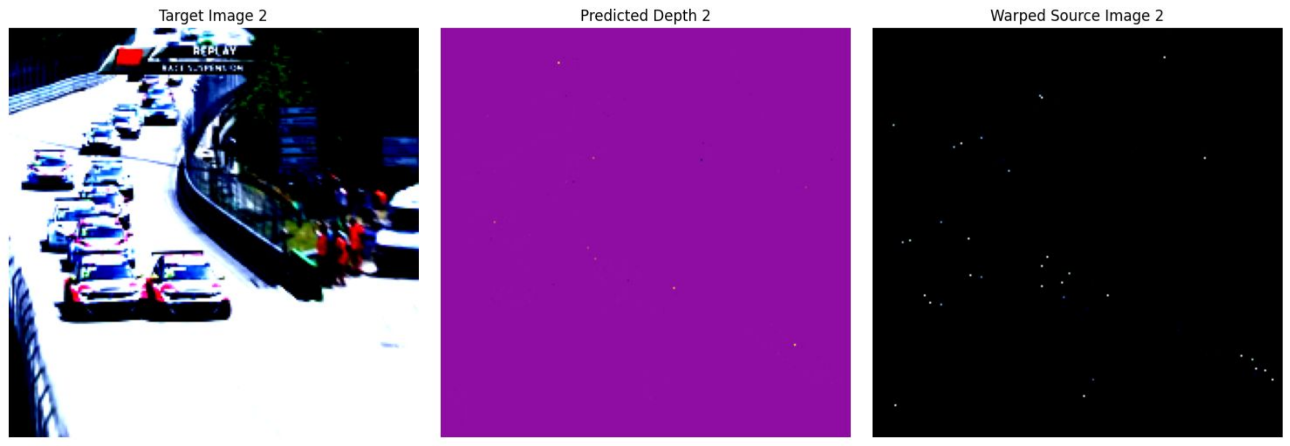
L1 loss -> depth collapsed, pose trained

```python
# photometric loss: guide accuracy of warped image reconstruction
def photometric_loss_l1(tgt_img, warped_img):
    # loss is computed as the average difference between the target image and warped image
    loss = torch.abs(tgt_img - warped_img).mean()
    return loss
```

Depth relied on good warped images generated by pose

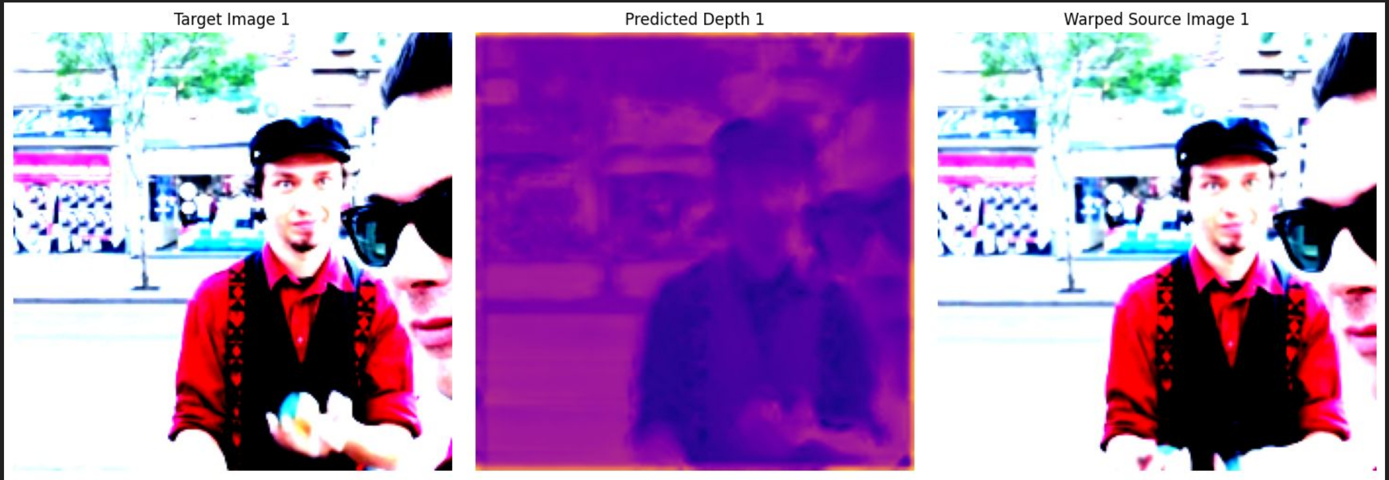
Depth Estimation Module

- Attempted Solution:
  balance training both depth and pose
- Results were not reliable



Depth Estimation Module

- Actual Solution:
  Pretrain both and fine tune together



Depth Estimation Module

- Loss Balance Notebook:
  https://colab.research.google.com/drive/1Vu4TMYIYKWB36zoMRJzvpjFjwkJbzsQ3?usp=sharing
- Fine Tuning Notebook:
  https://colab.research.google.com/drive/172QrXUewwVL_3w4yo_Oe3tjYZKBNOZdH?usp=sharing

Object Detection Module

- Slot Attention:
  - Decompose image into 10 slots
  - Iteratively refine slot embeddings with attention computations
- Mask Decoder:
  - Project slots into feature space with features extracted from DINO backbone (dim=768)
  - Produce soft masks for each slot to segment the image into object regions
- Object Detector:
  - Integrates feature map from DINO, depth cues, and soft masks
  - Depth + appearance features allows for spatial understanding
  - Bounding boxes computed from masks

Scene Graph Construction Module

- Scene Graph Constructor Step:
    - Transform object level features to graph
    - Construct node and edge embeddings
    - Estimate a soft adjacency matrix between objects
    - Differentiable step for backpropagation
- Evolving Scene Graph:
    - At each frame, scene graph constructor produces information to update evolving scene graph
    - GRCell integrates past embeddings to the current embeddings
    - Constructs temporal edges between nodes in consecutive frames

GNN Tracker

- Graph Neural Network refines node and edge features and learns which cues are reliable for maintaining object identity across frames
- Using message passing, object embeddings are iteratively refined through relationships with other objects
- Can leverage geometric and temporal cues for robustness to occlusion

Limitations

- Depth and pose trained through unsupervised methods, not robust to visual challenges
  - Future improvement: import pretrained depth and pose networks, still an unsupervised pipeline
- Object detection training jointly with the GNN from scratch propagates noise
  - Possible solution: import pretrained object detection for candidate object detections and fine tune alongside the GNN
  - Alternatively, pretrain the object detection for some epochs until results are reasonable, and then train with GNN
- Training an object tracker could be possible unsupervised, but would require some disjoint pre-training steps that would also use a lot of resources (compute) to train