# Spelke Object Segmentation with Counterfactual World Modeling

Jared Watrous

*CS468 Final Project Report*

**Abstract**

Counterfactual World Modeling [1] (CWM) is a prospective foundation model that attempts to unify many common computer vision tasks, including keypoint prediction, optical flow, and segmentation, and offers a path to extend this framework to more 3D tasks such as novel view synthesis and depth estimation. The latest iteration of this framework, Causal CWM (CCWM), successfully implements these 3D extensions, but loses its original method of predicting object segmentation. In this paper, we propose an inference-time readout for Spelke object segmentation with CCWM and demonstrate its viability on in-the-wild data.

## 1 Background

The concept of *Spelke objects* originates from cognitive scientist Elizabeth Spelke and her colleagues, who revealed that babies early in development learn to group visual elements only when they move together when interacted with [5]. The tendency of two points in space to move together is referred to as *Spelke affinity*, and a collection of points that move together in space can be considered an independent grouping, or Spelke object. Note that this definition is not perfectly well-defined, as Spelke affinity is not a symmetric relationship. For example, an apple sitting on a plate is very likely to move if the plate moves; however, moving the apple itself does not necessarily mean the plate will move with it. For this reason, we consider Spelke segmentation significantly distinct from other definitions of objects such as instance or semantic segmentation. Computer vision work relating to Spelke objects is relatively limited, but we observe that this definition is practically useful for many applications such as robotics, and is relatively convenient to extract using certain formulations of optical flow [2].

## 2 Introduction

The field of computer vision includes many common tasks, such as optical flow estimation, depth extraction, and object segmentation. Often times, these are delegated to specialized models that excel at their particular task. However, especially since the advent of large language models for natural language processing, interest in a general-purpose foundation model for computer vision has grown significantly. In light of this motivation, Counterfactual World Modeling [1] (CWM) arose as a prospective foundation model, and has shown itself to be especially promising in its scalability.

CWM is based on a next-frame-prediction learning objective inspired by biological vision systems. Given the first frame of a video clip, `rgb0`, and some small subset of patches of a subsequent frame, `rgb1`, CWM attempts to predict the entire second frame `rgb1`. Counterfactual prediction, in which we provide false `rgb1` patches to the model, allows us to reconstruct RGB images *as if* the patches were factual. By, for example, moving RGB patches from `rgb0`

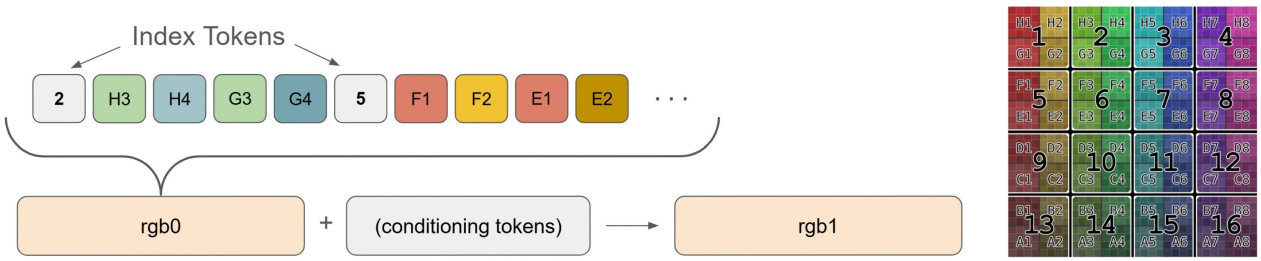



Figure 1: Sequence construction scheme for CCWM. Index tokens indicate to the model which block to decode next, permitting arbitrary rollout conditioning and ordering.

to condition `rgb1`, we can produce images *as if* the object in the counterfactual patch had moved. Furthermore, we can condition patches on the background to counterfactually induce or prevent camera motion.

In a simple ViT-based model, [3] it can be shown that this framework enables many useful computer vision readouts. For example, to extract optical flow, we can perturb `rgb0` at a point and determine the point in `rgb1` that this perturbation most affects. Notice that, because a ViT-based model is differentiable, we can instead use standard autograd tools to compute the partial derivative of output pixels with respect to input pixels, tracking the correspondence between the two frames. Furthermore, by counterfactually moving a single patch and measuring optical flow of the entire image, we can extract rough estimates of Spelke object segmentations at the given query patches. While these properties are useful, this ViT-based model exhibits significant blurriness in its RGB predictions, especially when injecting camera pose information. This is a natural result of using $L_2$ pixel-space loss, which regresses the model to the mean output and encourages blurring uncertain parts of the image. To mitigate this, we switch to a next-token-prediction scheme, inspired by the success of large language models, and we refer to this new model as *Causal Counterfactual World Modeling* (CCWM).

**Tokenization** Typical image tokenizers for vision-language models use a transformer architecture that allows information to flow freely between the tokens, losing spatial locality information. Because spatial locality is vital to the CWM framework, we introduce a novel *local patch quantization* scheme, in which image patches are independently tokenized using a lookup-free quantizer [6], with no information from its neighboring patches. To reconstruct an image from tokens, we use a fully convolutional decoder. In practice, we use a patch size of 4 pixels.

**Sequence Construction** To construct a sequence from a grid of tokens, we group the image into 2-by-2 blocks of tokens. Each block is assigned an "index token," a token that indicates to the model which block in the image the following RGB tokens correspond to. With this formulation, we can reorder the blocks in the sequence with any arbitrary permutation, allowing us to inject arbitrary blocks as conditioning and choose what order to decode `rgb1` blocks in. We can also tokenize any other form of conditioning we choose, such as camera pose, and insert it between `rgb0` and `rgb1` in the sequence.

**Current Models** Currently, we have two different models trained. Our first model conditions on `rgb0` and a relative camera pose `C`, and predicts an *optical flow* image `F`. We define optical flow images similarly to RGB images, with index tokens that allow arbitrary ordering, but using

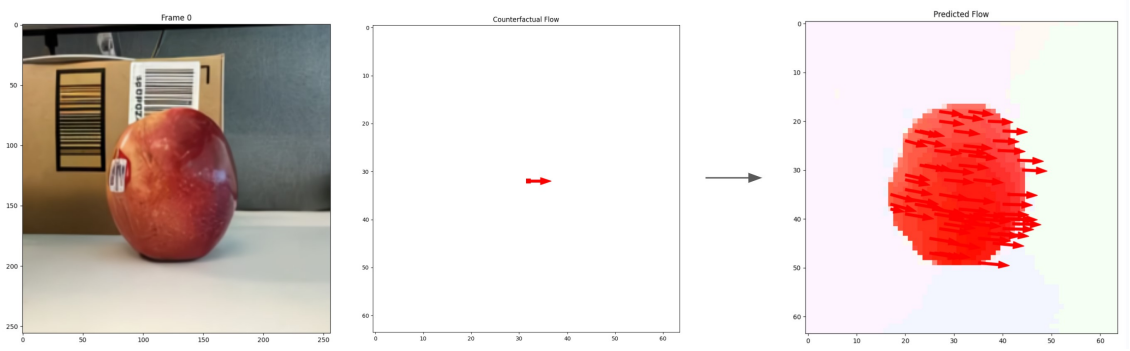



Figure 2: Sparse-to-dense optical flow prediction. By conditioning on camera motion (no motion in this sample) and a counterfactual flow patch, we can use the `rgb0 + C` CCWM model to predict a dense flow image.

a different quantizer. Our second model conditions on `rgb0` and optical flow `F` to predict `rgb1`. This configuration permits many of the same readouts as the original ViT-based model while also introducing novel view synthesis. While this model is no longer differentiable, we have reformulated optical flow such that it no longer relies on the Jacobian matrix of the model. However, our flow interface having changed, we must revisit how to bootstrap optical flow to product Spelke object segmentation readouts. This project addresses the question, *how do we use CCWM to extract Spelke object segmentations?*

# 3 Method

We propose a method for Spelke object segmentation with CCWM using a simple inference-time readout. We first recall that, in a simple RGB-only model, we can arbitrarily choose to condition which `rgb1` patches to condition our rollout on. We observe that, using our current `rgb0 + C` to `F` model (camera pose to flow), we can similarly choose which flow patches to condition on. That is, because flow sequences are constructed similarly to images, we can condition the rollout on arbitrary counterfactual flow patches of our choosing. This effectively treats the model as a *camera-pose-conditioned sparse-to-dense optical flow predictor*. We show that this sparse-to-dense flow predictor can be used to predict Spelke segmentations.

**Readout Algorithm** We start by choosing some view-space direction; without loss of generality, we choose the direction "right". We apply the following conditions to the model: (1) in-plane camera translation to the right some small amount, e.g. 0.1 meters, and (2) at our query point, inject a single optical flow patch to the right some small number of pixels, e.g. 8. After computing sparse to dense flow with these conditions, the scene "should" have flow pointing to the left (opposite the camera motion), except for the queried object, which "should" have flow pointing right (same as the injected flow). We can repeat multiple times with various motion directions and seeds, and determine whether a given patch should be included in the segmentation using the following patch-wise score:

$$S_p = \frac{1}{N} \sum_{i=1}^{N} \text{sign}(F_p^{(i)} \cdot C^i)$$

Here, $F_p^{(i)}$ is the predicted flow of patch $p$ in sample $i$, and $C^{(i)}$ is the counterfactual camera motion direction for sample $i$ projected onto the camera plane. We include a patch $p$ in the segmentation if its score is positive, i.e. if $S_p > 0$.

# 4 Experiments

We conduct rollout experiments on a subset of the COCO dataset [4] with query points chosen uniformly random at least 10 patches (40 pixels) away from the image border. After iteration over many combinations of motion directions and seed counts, we settle on 4 seeds for each motion direction, and 2 motion directions: in-plane left, and in-plane right. We observe other directions, such as up and down, can have detrimental effects on performance, as they often would induce physically implausible counterfactual motions. For example, giving downward flow to an object on a table could imply the object moves through the table, which contradicts the model's prior information on physical interactions. Similarly, an upwards flow could imply the object is defying gravity, further confusing the model. In most cases, leftward and rightward flow are significantly more plausible, although this is not always necessarily the case (e.g. an object placed against a wall.) See the Appendix for sample Spelke segmentations and the optical flow predictions used to make them.

**Evaluation** Quantitative evaluation in this setting is especially difficult for multiple reasons: (1) As described before, Spelke object segmentation is significantly different from other definitions of segmentation, and limited computer vision work on the subject means there are essentially no datasets available for proper evaluation. (2) Also as described before, Spelke affinity is not always well-defined, and thus any hand-designed Spelke object segmentations are inherently subjective and may not correspond with an otherwise "valid" readout provided by the model. (3) We observe a large amount of variance in the model's predictions, possibly due to either the ill-posedness of the task or the subpar quality of our current model, and hence any quantitative evaluation may not be reliable. For these reasons, we choose not to perform a quantitative evaluation, and rather focus on qualitative evaluation of the algorithm's performance in a computational neuroscience context.

# 5 Conclusions

In this paper, we present a method for extracting Spelke object segmentation from Causal CWM models using sparse-to-dense optical flow prediction. Our experiments exhibit acceptable qualitative performance on a number of samples from the COCO dataset [4]. We observe some failure cases in which points are segmented even if they intuitively have high Spelke affinity, such as a tie segmented separately from the shirt it's attached to. Occasionally, stronger failure cases arise, especially images with complex visual patterns that likely confuse the model. The latter may be partially due to the local patch quantization scheme, in which tokens only contain local visual information with no information about the global context of the image.

We also observe large variance in the model's dense flow predictions and, consequently, its segmentations. This variance originates from the randomized nature of the rollout algorithm, including autoregressive token sampling and uniformly randomized flow patch ordering. In future work, we hope to explore ways to improve this rollout algorithm to improve its consistency, such as rolling out patches in order of decreasing logit entropy (similar to keypoint

extraction.) Furthermore, the model used in these experiments is a 100-million-parameter transformer, which is significantly smaller than current state-of-the-art language models. As we train larger models, it's possible that some of these inconsistencies will resolve with scale.

# 6 Contributions

This project is based on current work at Stanford's NeuroAI Lab advised by Professor Daniel L. K. Yamins. This work on Spelke object segmentation was performed by Jared Watrous with helpful consultation from colleagues at the lab. Many thanks to all of those involved in the Causal CWM project, including Klemen Kotar, Honglin Chen, Wanhee Lee, Rahul Venkatesh, and Daniel L. K. Yamins.

# References

[1] Daniel M. Bear, Kevin Feigelis, Honglin Chen, Wanhee Lee, Rahul Venkatesh, Klemen Kotar, Alex Durango, and Daniel L. K. Yamins. Unifying (machine) vision via counterfactual world modeling, 2023.

[2] Honglin Chen, Rahul Venkatesh, Yoni Friedman, Jiajun Wu, Joshua B. Tenenbaum, Daniel L. K. Yamins, and Daniel M. Bear. Unsupervised segmentation in real-world images via spelke object inference, 2022.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

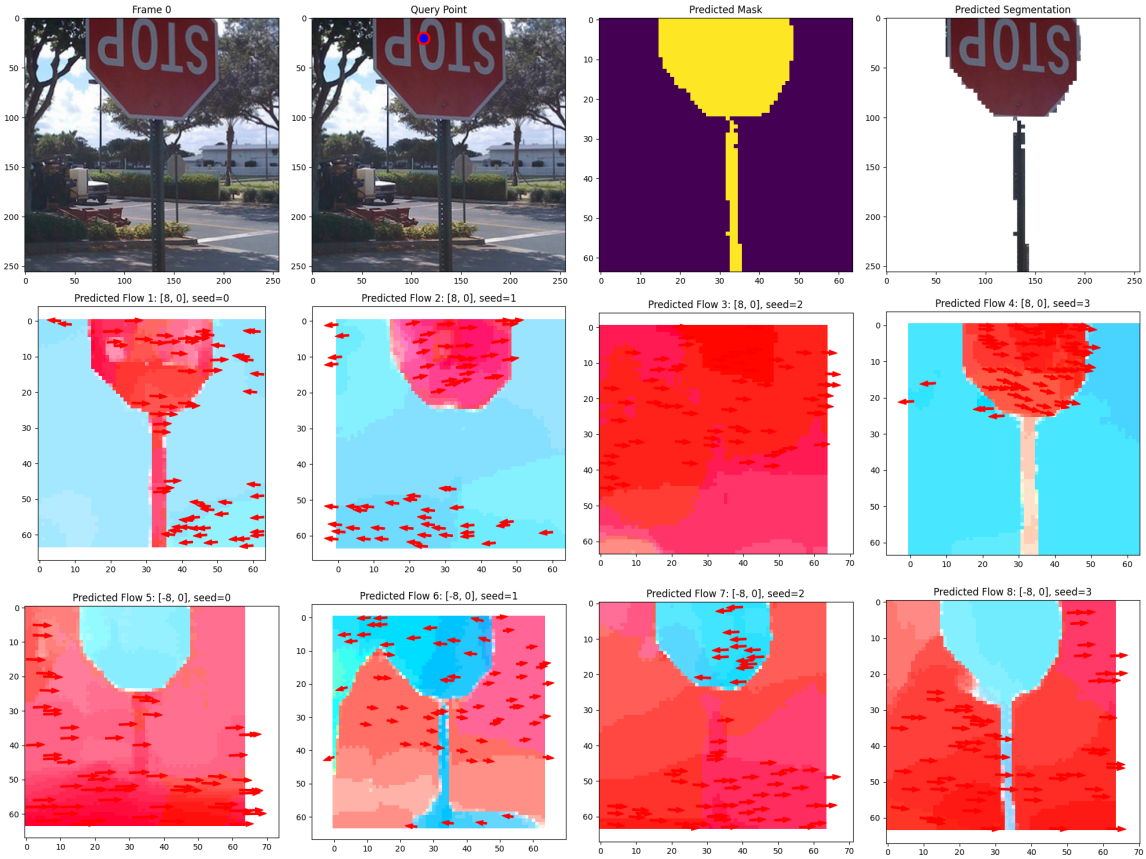
[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

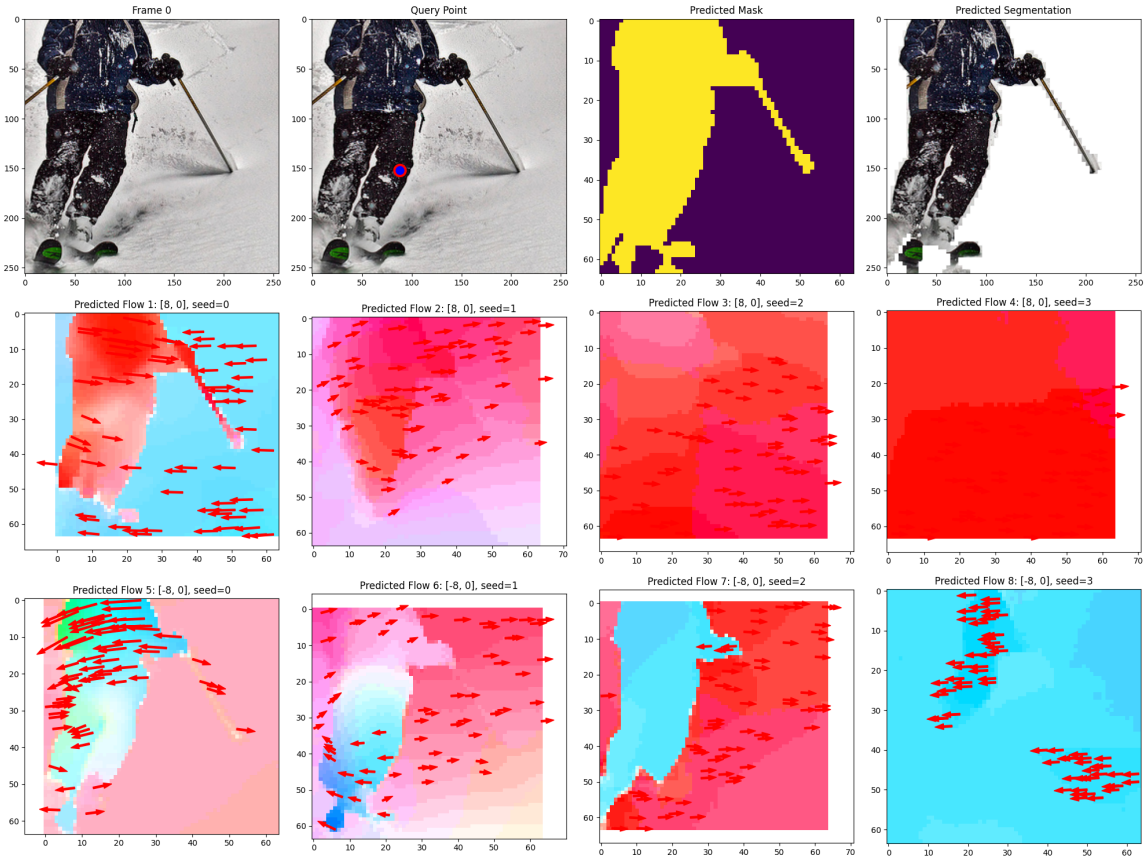
[5] Elizabeth S. Spelke. Principles of object perception. *Cognitive Science*, 14(1):29–56, 1990.

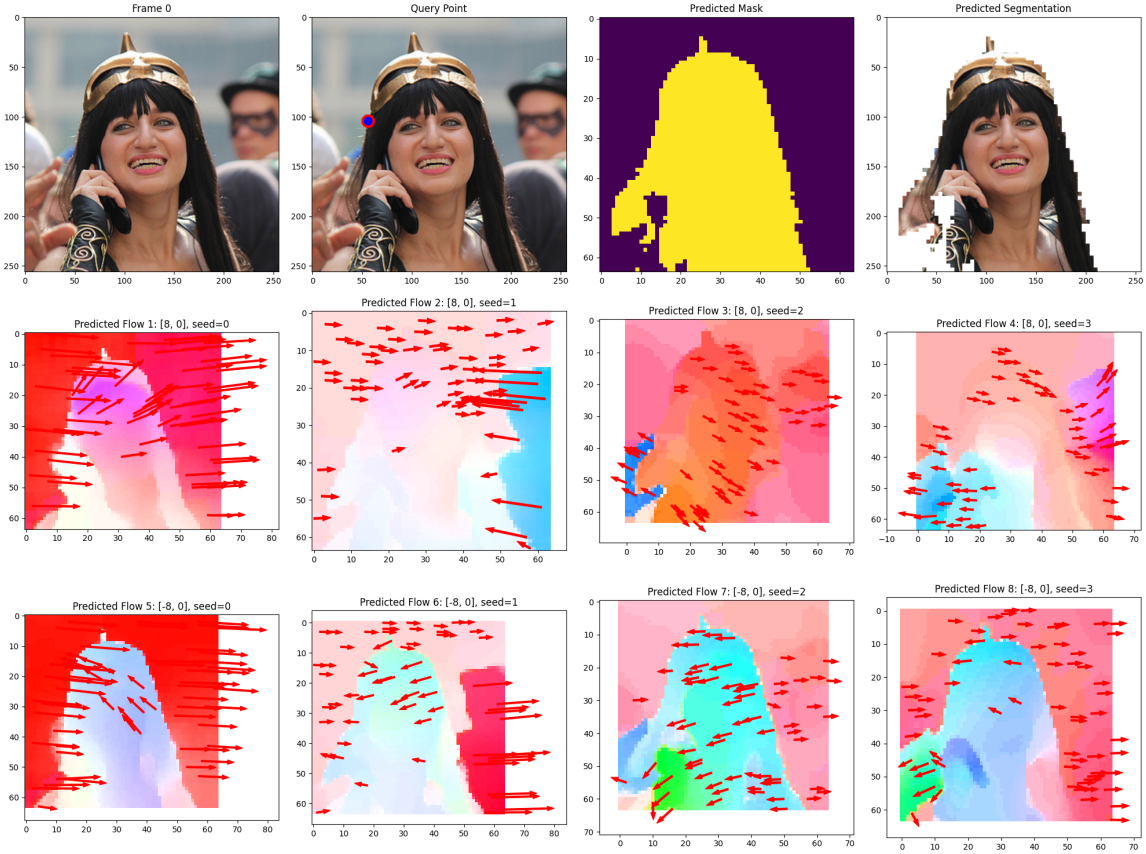
[6] Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion – tokenizer is key to visual generation, 2024.

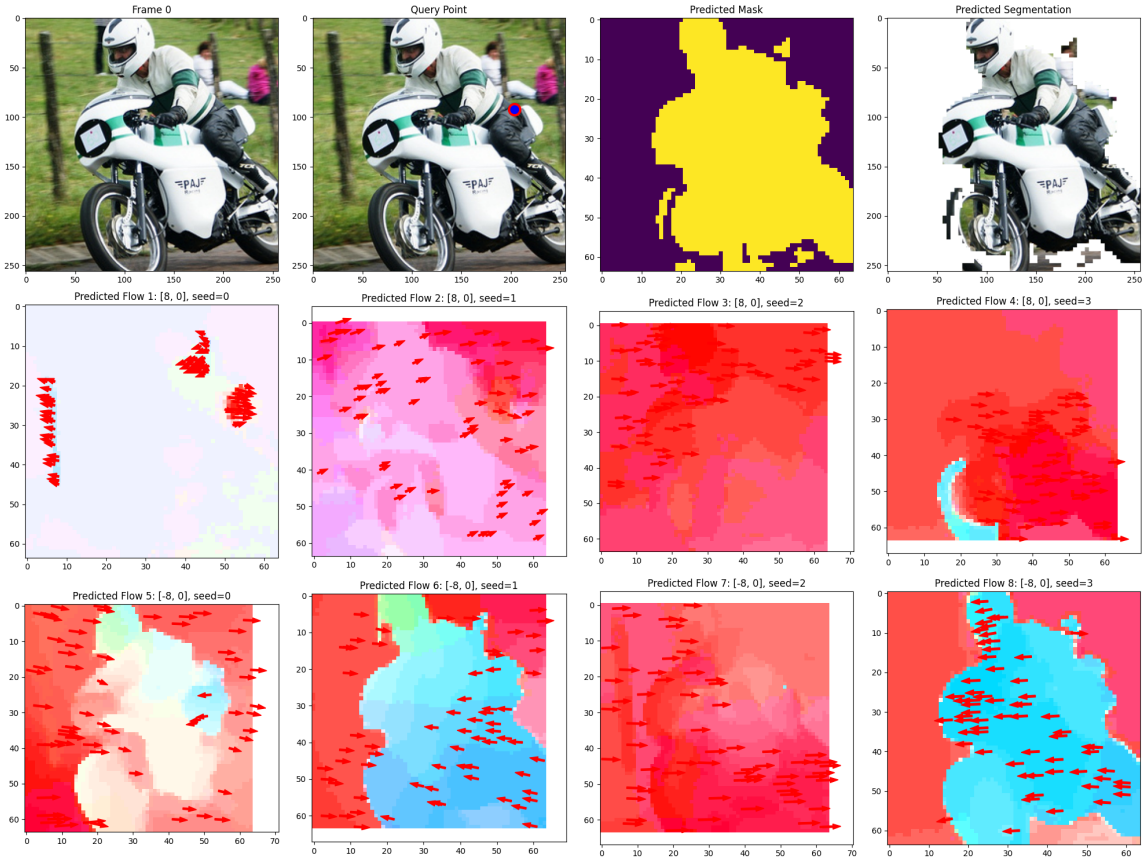
# Appendix: CCWM Spelke Segmentation Examples

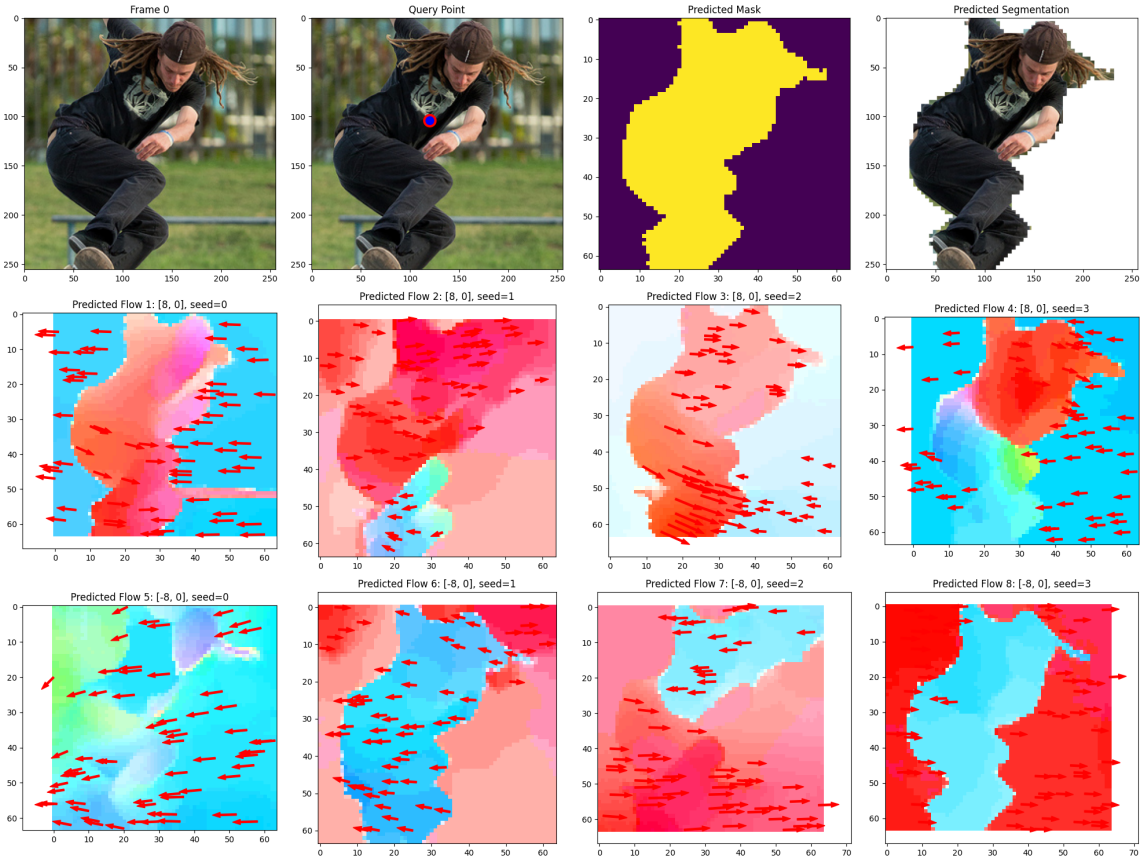
Here we present many samples of Spelke object segmentation using our method described in this paper. In each sample, the top left image is the input `rgb0` to the model, the top middle-left indicates the query point (the blue and red circle), the top middle-right is the predicted segmentation mask, and the top right is the segmentation applied to the image. The bottom two rows are the 8 optical flow predictions used to compute the segmentations; the middle row indicates counterfactual camera and object motion to the right, inducing background motion to the left, and the bottom row reverses these directions. Each column in these two rows indicates a different seed used to roll out the result.

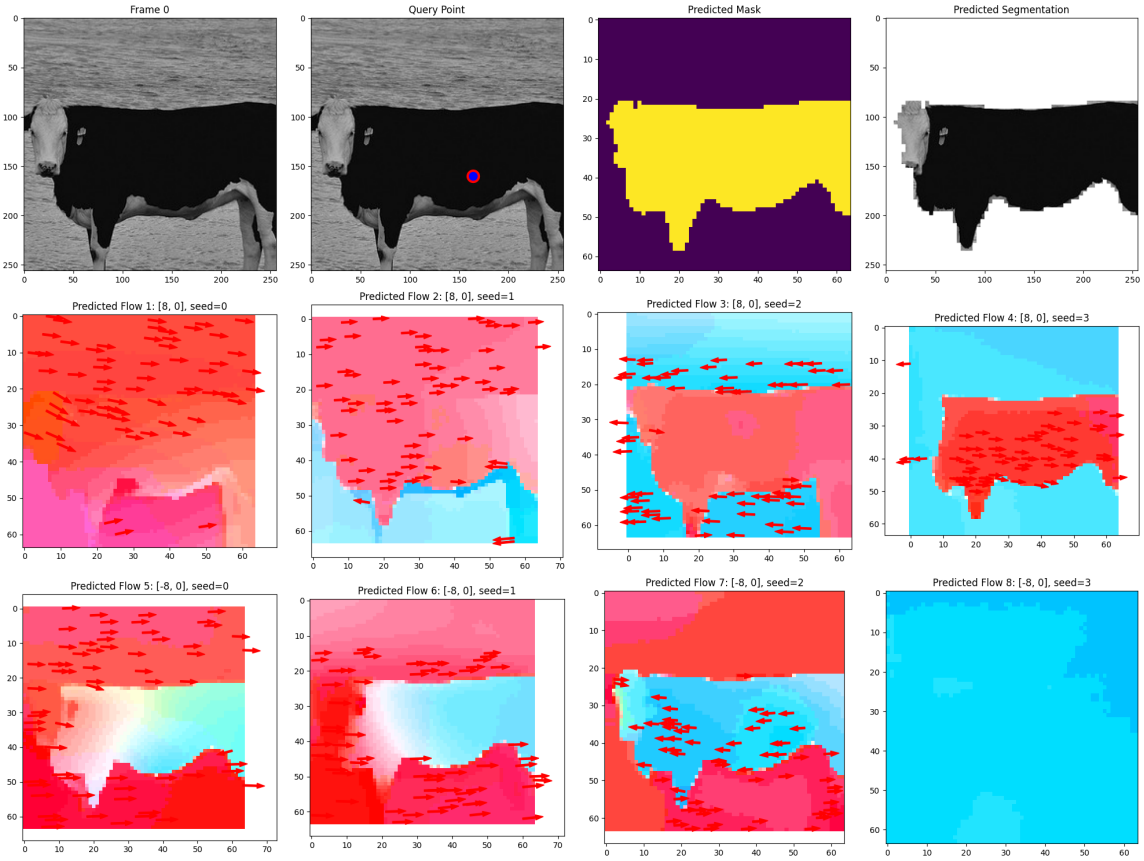

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3

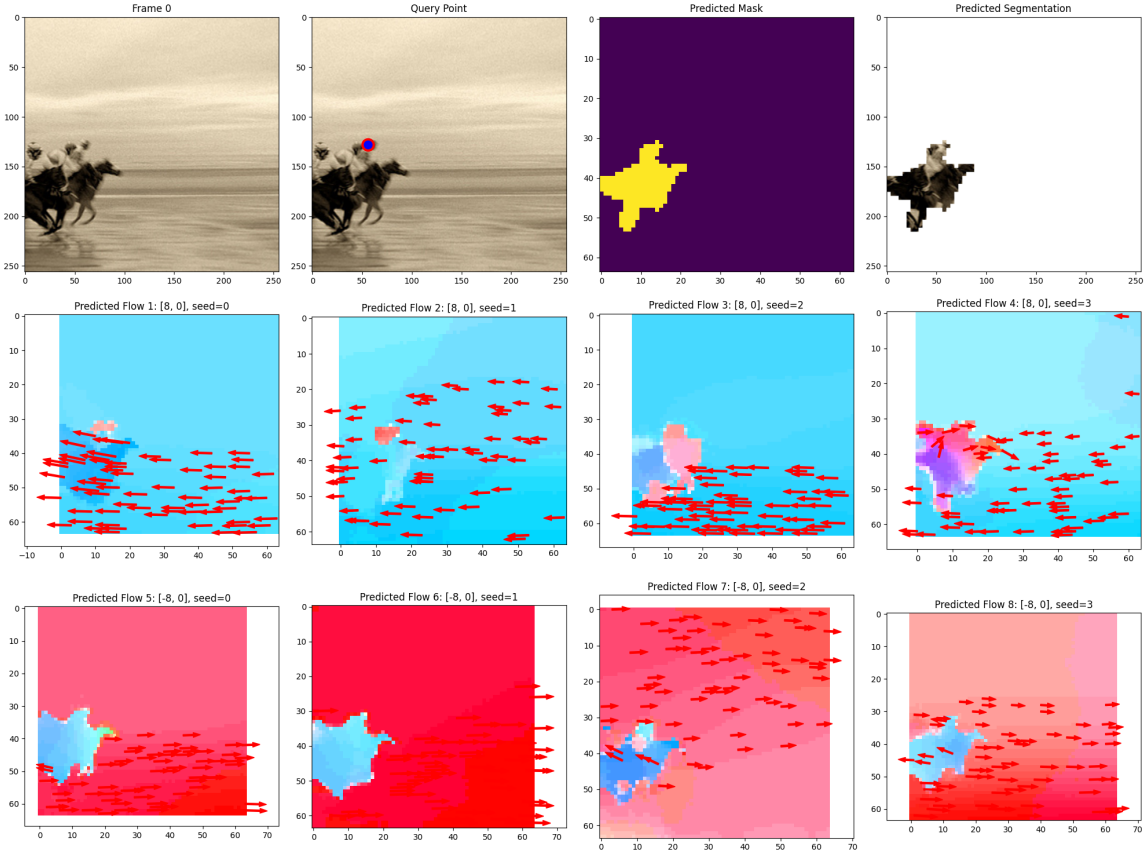

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3

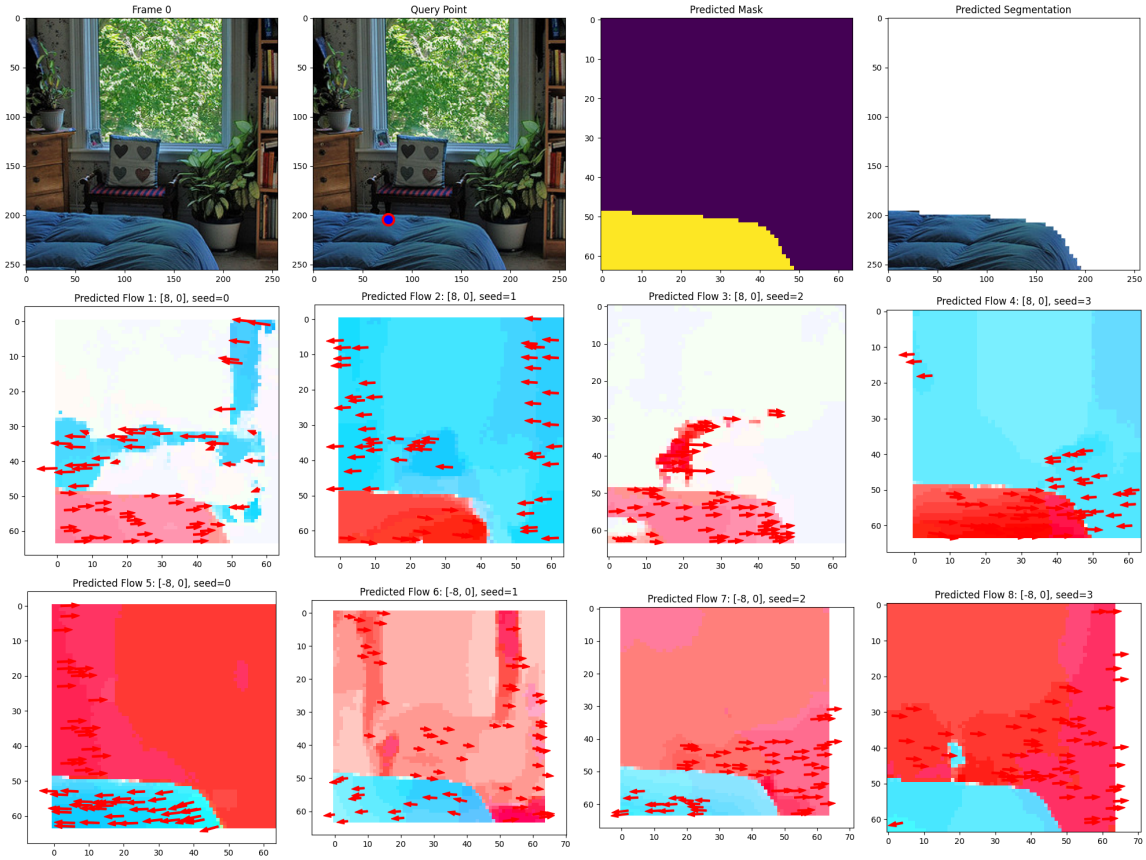

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3

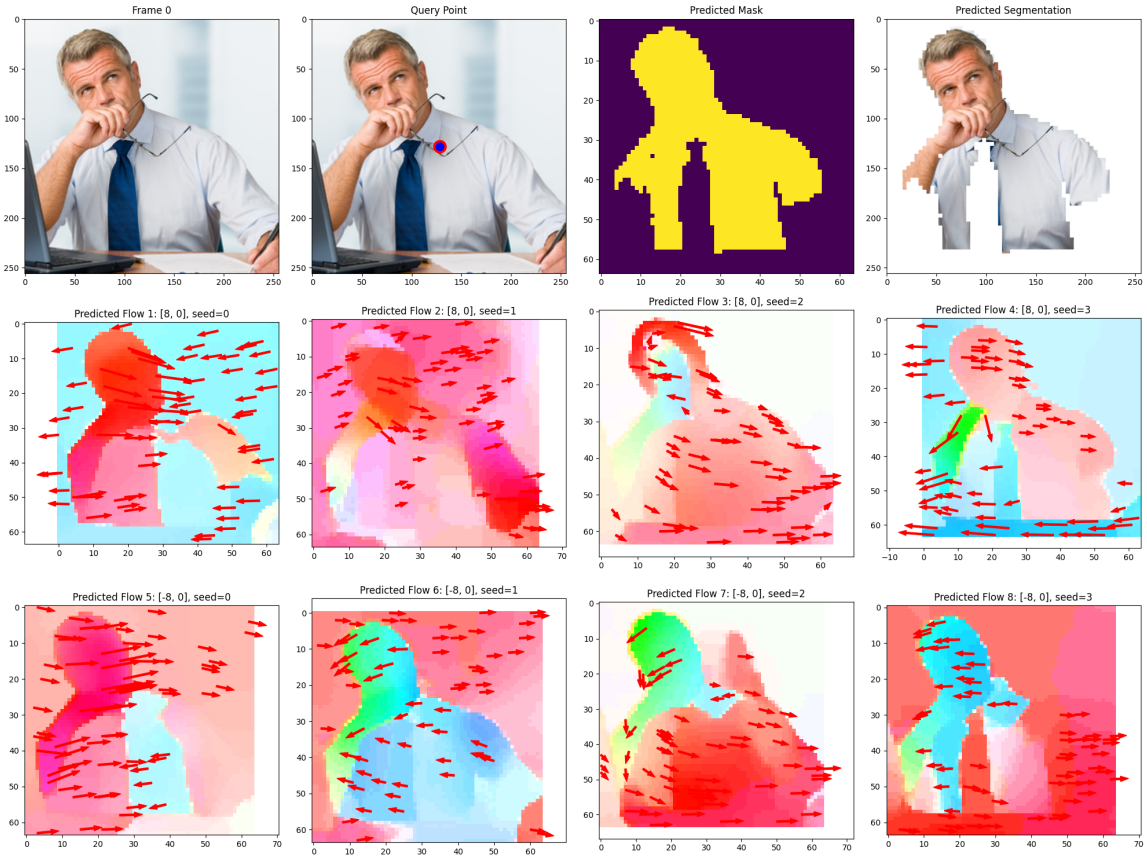

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3

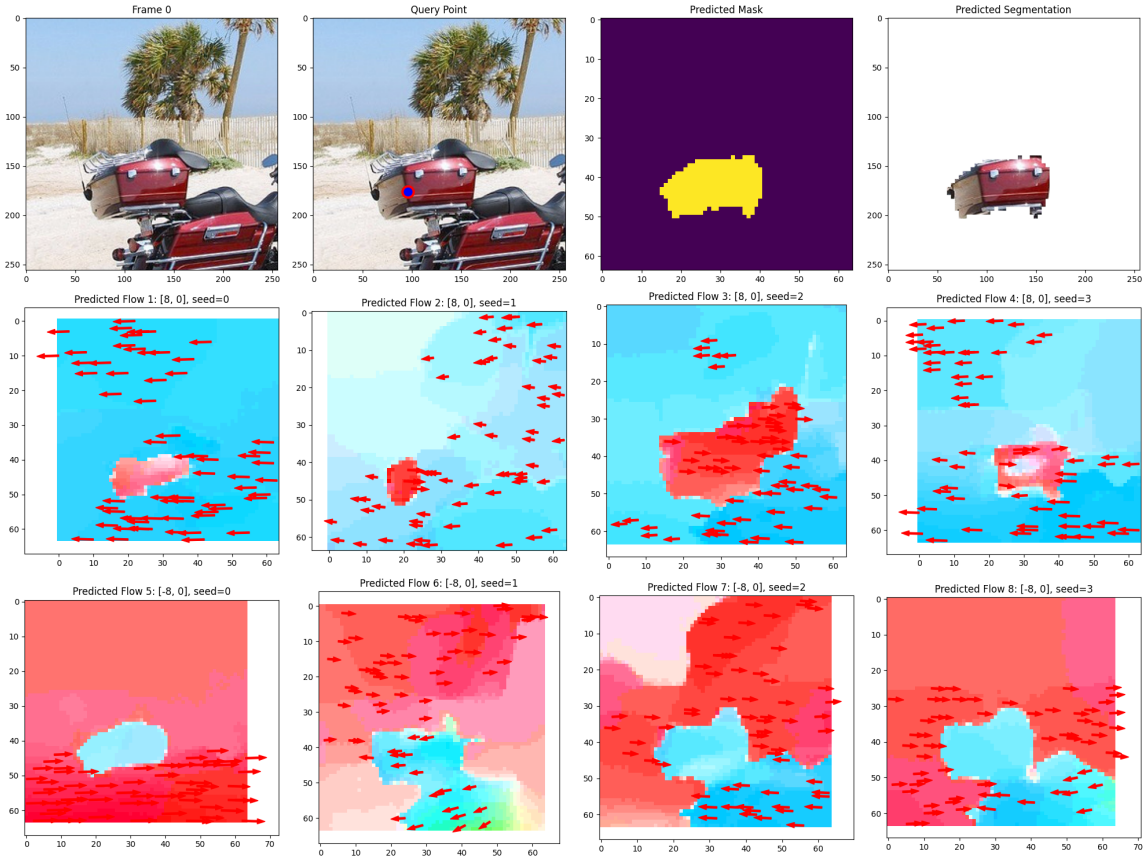

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3

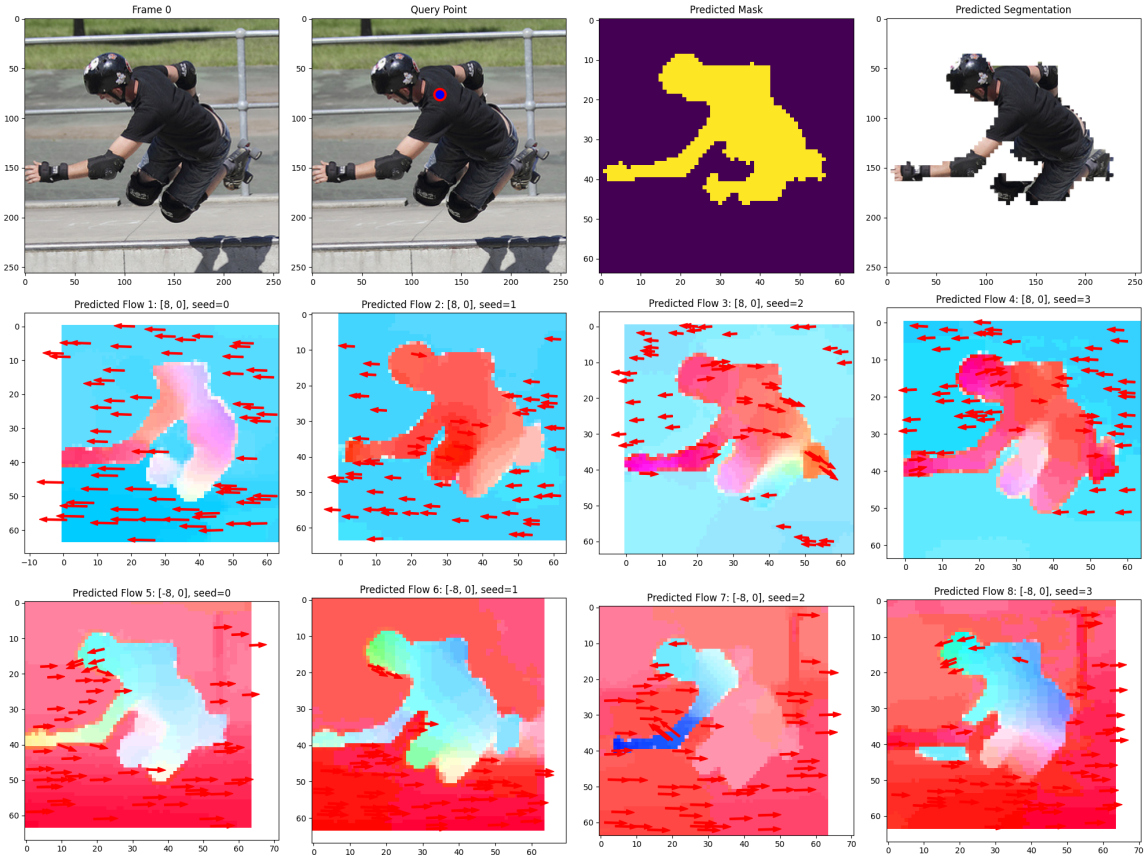

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3

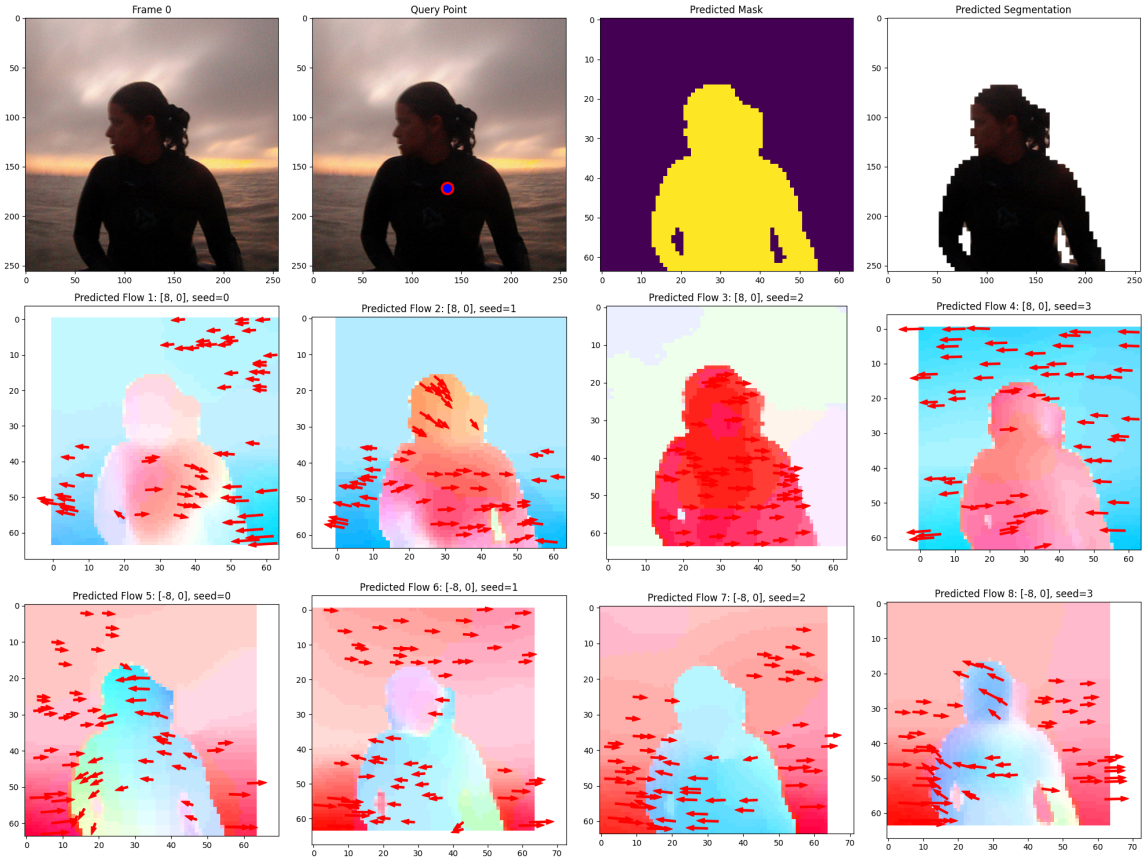

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3

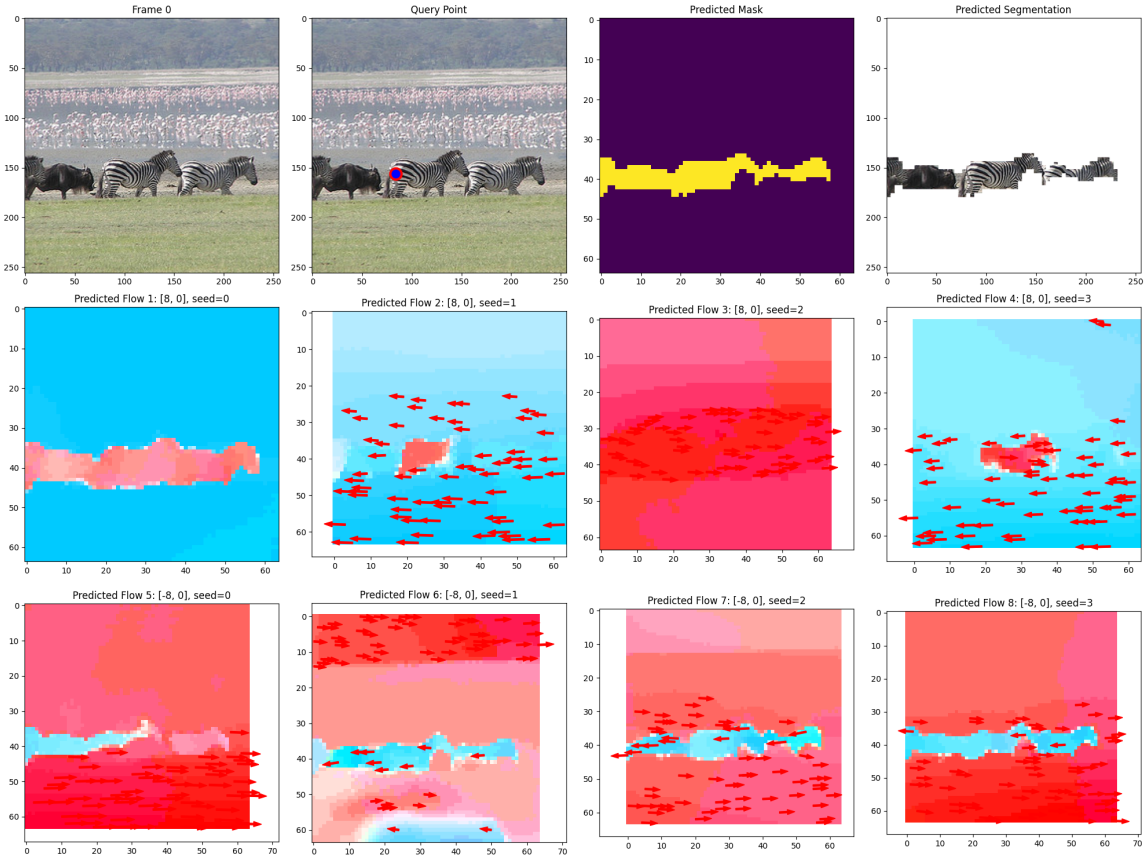

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3

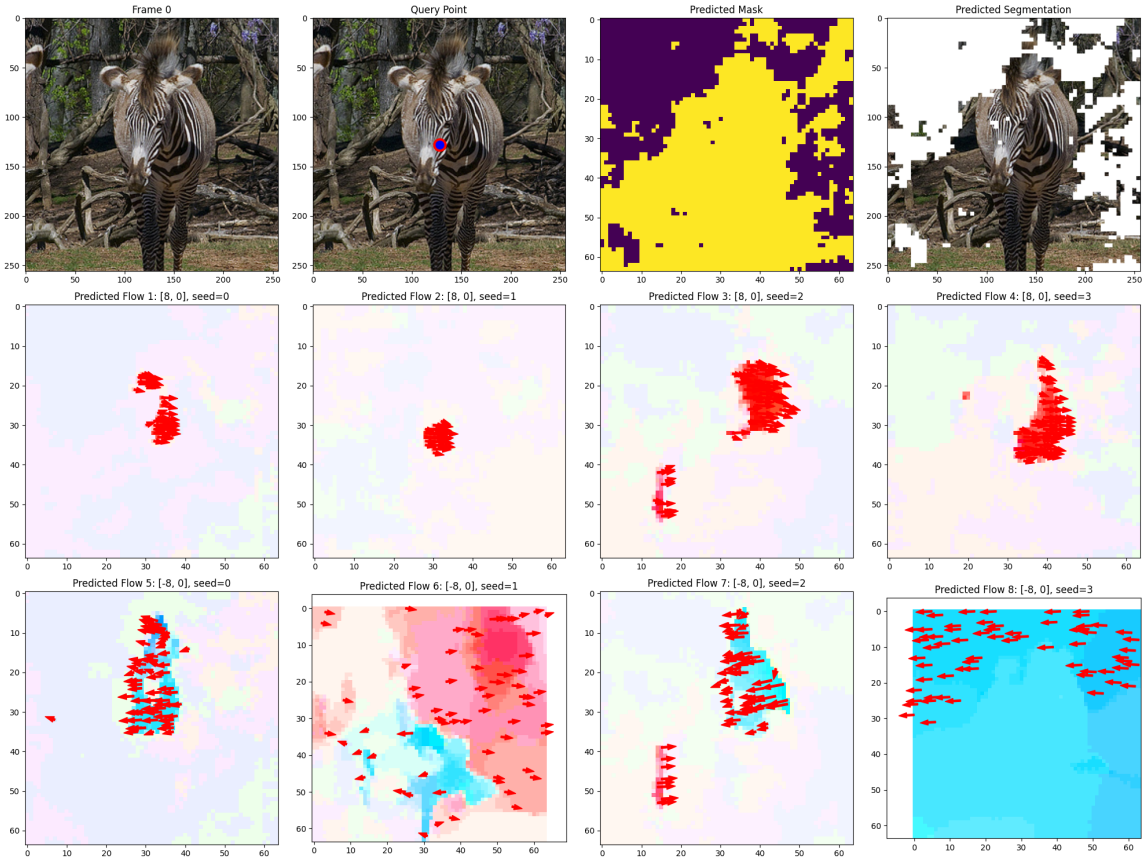

Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3


Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3


Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3


Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3


Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3


Frame 0
Query Point
Predicted Mask
Predicted Segmentation
Predicted Flow 1: [8, 0], seed=0
Predicted Flow 2: [8, 0], seed=1
Predicted Flow 3: [8, 0], seed=2
Predicted Flow 4: [8, 0], seed=3
Predicted Flow 5: [-8, 0], seed=0
Predicted Flow 6: [-8, 0], seed=1
Predicted Flow 7: [-8, 0], seed=2
Predicted Flow 8: [-8, 0], seed=3