Video Object Segmentation

Shir Hamawie & Nadav Shoham

June 20, 2023

1 Introduction

Video object segmentation is a task in computer vision that involves extracting the foreground objects of interest from a video sequence. One way to perform this task is by a semi-supervised video object segmentation method. This method leverages only one manual annotation in the initial frame of the video, known as the ground-truth mask, and aims to propagate this annotation to subsequent frames in an unsupervised manner.
In this assignment, we aim to design and implement that, based on Or Dinari's ScStream code as a building block. The ScStream code provides a foundation for processing video streams efficiently.
We will use a video of a walking 3D dinosaur as our test case, here is the first frame:

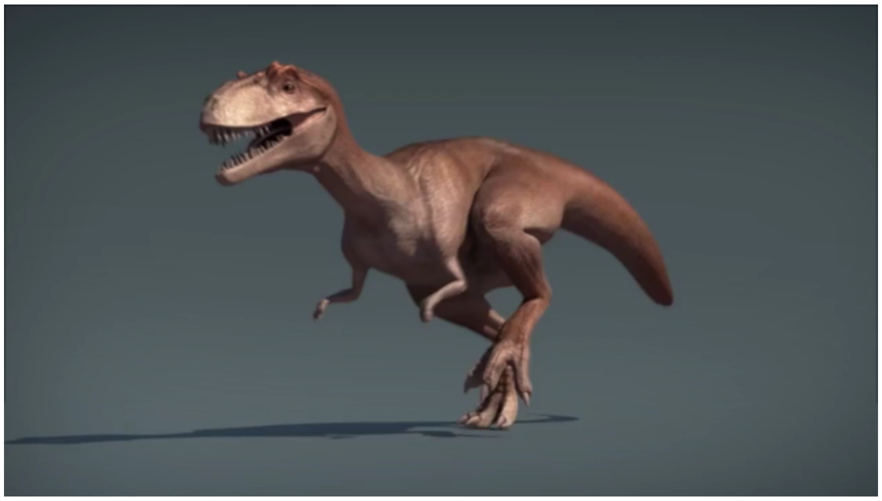

Figure 1: First frame of the video

The remainder of this report is organized as follows:

- Section 2 - provides a brief overview of Or Dinari's ScStream.
- Section 3 - we describe the methodology and architecture in detail.
- Section 4 - presents the critical points that most influenced the application.
- Section 5 - conclusion.

2 Background

2.1 DP-Means

DP-Means is a clustering algorithm based on the Dirichlet Process Mixture Model (DPMM). It aims to automatically determine the number of clusters in a dataset without requiring a predefined number of clusters. However, DP-Means can be computationally expensive, especially for large datasets, as it involves calculating pairwise distances between data points.

2.2 PDC-DP-Mean

PDC-DP-Means is a method proposed by Or Dinari and Freifeld in 2022 that builds upon the DP-Means algorithm. PDC-DP-Means is designed to address the limitations of traditional DP-Means in terms of scalability and computational efficiency. PDC-DP-Means introduces parallelism and delayed cluster creation to improve the scalability and efficiency of DP-Means. By leveraging parallel computing techniques, PDC-DP-Means can distribute the computational load across multiple threads or machines, enabling faster processing of large datasets. This parallelization significantly reduces the runtime of the algorithm. Additionally, PDC-DP-Means incorporates delayed cluster creation, which allows for more efficient memory usage. Instead of creating clusters for all data points at once, PDC-DP-Means delays the creation of clusters until they are needed. This approach reduces the memory requirements of the algorithm, making it more suitable for handling large-scale datasets. The combination of parallelism and delayed cluster creation in PDC-DP-Means results in a highly scalable and efficient algorithm for clustering. It enables the algorithm to handle large datasets with improved computational speed and reduced memory usage compared to traditional DP-Means.

2.3 ScStream

ScStream is a method proposed by Or Dinari and Freifeld in 2022 that builds upon the principles of DPMM. It is specifically designed for clustering streaming data, where data arrives in a continuous and sequential manner. ScStream leverages the sampling-based approach of DPMM to perform clustering on streaming data efficiently. The key advantage of ScStream is its ability to handle any exponential family, making it suitable for a wide range of data types. It offers a fast implementation that can work in either multi-threaded mode or multi-machine multi-process mode, enabling scalability and parallelization.

3 Methodology

Our proposed method for semi-supervised video object segmentation consists of several key steps, which we will outline in detail. The overall workflow follows a sequential process at each iteration, starting with foreground and background extraction, followed by Sc-Stream model training, and finally the creation of Gaussian data mixtures to be used in the next iteration. The central method of our implementation is the **segment** method in video_object_segmentation.py.

3.1 First Frame

3.1.1 Initial Foreground and Background Extraction

The first step of our method involves extracting the foreground and background regions from the initial frame of the video. To accomplish this, we employ the **GrabCut** algorithm, a popular technique for interactive foreground extraction in images. The GrabCut algorithm combines color and spatial information to iteratively refine the foreground and background segmentation. It requires an initial bounding box or a rough mask specifying the location of the foreground object. In our case, this a great option to define the ground-truth mask of the object in the first video frame, since it is a semi-supervised method. Our implementation of GrabCut uses the OpenCV library and can be found in the grab_cut.py file. Example result:

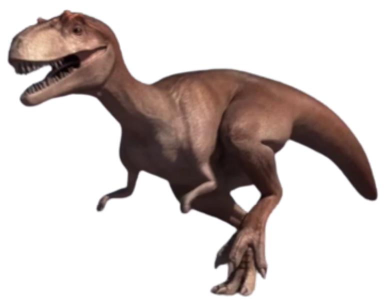

Figure 2: Foreground

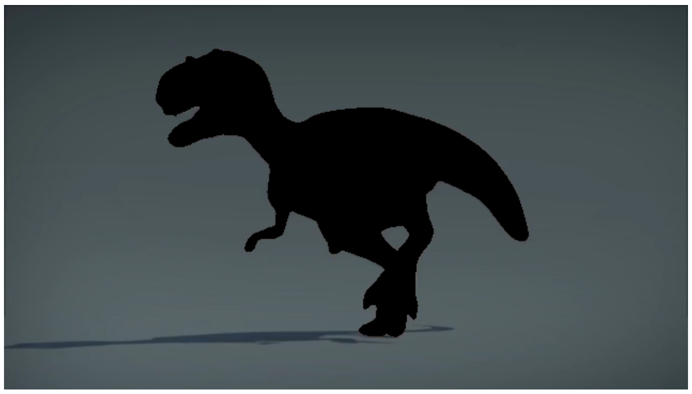

Figure 3: Background

3.1.2 Models Initialization

Once the foreground and background regions of the current frame have been extracted, we proceed to initialize our models. Our goal is to be able to classify data to foreground and background by their characteristics. Therefore, creating a model for each of them will create two separate and different gaussians that we can later extract from the models and use as our classifiers. When initializing our models with `fit_init` we have do some hyper-parameter tuning that we will discuss in the next section.

3.2 Segmentation Loop

3.2.1 Creation of Gaussian Data Mixtures

As previously said the gaussians of the foreground and background models are our classifiers, helping us determine how to separate a new frame into 2 parts. we can build them by extracting the means and covariances of the clusters from each model and initialize a **GaussianMixture** instance for both of them. The GaussianMixture class is defined in gaussian_mixture.py and uses `Scipy.stats.multivariate_normal` to represent each gaussian. The class supports two pdf options, pdf of the mixture using weights and sum of all gaussians pdfs, and max pdf that returns the gaussian pdf with the largest value.

```python
class GaussianMixture:
    def __init__(self, means, covariances, weights=None):
        self.means = means
        self.covariances = covariances
        self.weights = weights if weights else [1/len(means)]*len(means)
        self.gaussians = [multivariate_normal(mean, covariance) for mean,
    covariance in zip(means, covariances)]

    def pdf(self, x):
        pdfs = [weight * gaussian.pdf(x) for weight, gaussian in zip(self.
    weights, self.gaussians)]
        return sum(pdfs)

    def max_pdf(self, x):
        pdfs = [gaussian.pdf(x) for gaussian in self.gaussians]
        return max(pdfs)
```

3.2.2 Foreground and Background Extraction

Now that we can no longer supervise the separation of the foreground and background we need to define a decision rule that exploits the given gaussianMixtures and classifies each pixel in the new frame into foreground and background. This is the most time-consuming part of our loop as we need to go over every pixel and calculate its pdfs. Further discussion about the decision rule will be in the next section.

3.2.3 Model adjustment

Given we have a new foreground and background we should utilize them to update our models, given it is very likely that our object changed its position from the previous frame. The updated models will provide new gaussian mixtures that should classify the next frame better than the previous mixtures. The adjustment stage will also be discussed in the next section.

3.3 Building The Segmented Video

At each iteration we produce foreground and background images for each frame, which we can use to build a segmented video. To highlight the foreground object, we can apply a color mask to the foreground image. The color mask is simply a binary mask with the same dimensions as the foreground image, where the foreground pixels are set to green and the background pixels are set to black. Then using the OpenCV library, we can apply the color mask to the original frame to highlight the foreground object. This is implemented in the **classify_frame** method in video_object_segmentation.py.

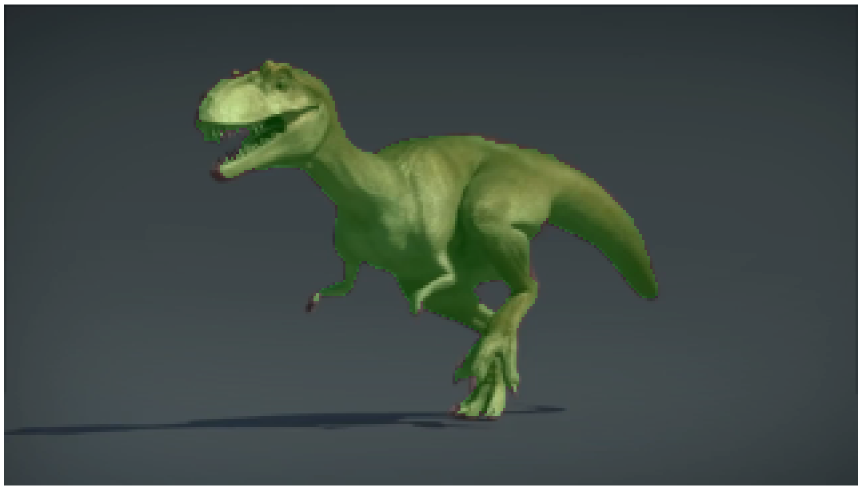

Figure 4: Segmented first frame

4 Critical points

4.1 Including X and Y

4.2 Decision Rule

4.3 Defining The Prior

4.4 Model Adjustment

5 Conclusion