Introduction

The topic of our project is clustering of streaming-data – meaning, the data arrives in batches, and our mission is to divide it into clusters.

We chose the Video Object Segmentation Project. Our task was to track an object in an input video, while using the frames of the video as the streaming-data and dividing the pixels in each frame into clusters of object and background.

To accomplish this task, we used Or Dinari's ScStream code as the foundation of our project. By using this Clustering Streaming Data algorithm, we created an object model and a background model using the first frame. We segmented the object throughout the entire video using these models.

To demonstrate the use and performance of our project, we used a video showing a deer walking in a snowy area. We tracked the deer as the object in our video.

Our project includes the following files:

- ***'input_video'*** : The input video.
- ***'first_frame'***: The very first frame of the video, out of which we created the first models to describe the object and the background.
- ***'mask'*** : The binary mask we made, which was used to divide the first frame of the video into object and background.
- ***'object'*** and ***'background'***: The results of applying the binary mask on the first frame.
- ***'masking.py'***: A Python program which applies the mask on the first frame.
- ***'object_segmentation.py'***: The main python program of our project.
- ***'final_output'***: The output video which was created after running our project.
- ***'comparing_video'***: displaying the input and output videos simultaneously side by side.

Program Description

To segment the object appearing in the input video, our project performs the following steps:

'masking.py':

1. Extracts the first frame of the video and applies the binary mask to it. This resulted in 2 separate images, one which shows the object surrounded by a black background, and one which shows only the background with the object colored in black.

   'object_segmentation.py':

2. For each of those 2 images, performs a transformation to a matrix, such that every row represents the data of one pixel and is of the form:

   $$[B, G, R, \frac{x}{100}, \frac{y}{100}]$$

   Where x, y represent the location of the pixel, and B, G, R represent the color channels of the pixel.

   We divided the values of x, y by 100 in order to add a relative weight to their value and improve the results.

   This was done in the *'pre_process_image'* function.
3. Runs Or Dinari's parallel streaming clustering code, using the *'fit_init'* function, from which we received 2 Gaussian Mixture-Models describing the object and the background.

   This was done in the *'cluster_first_frame'* function.
4. Extracts the mean vectors and covariance matrices for each Gaussian distribution in the object model and in the background model.

   This was done in the *'extract_model_means_vars'* function.
5. Iterates over each frame in the video, and:
   - Iterates over every pixel, transforms it into an array of the form $[B, G, R, \frac{x}{100}, \frac{y}{100}]$, and determines whether the pixel is more likely to be a part of the object or the background.

     This is accomplished by checking the pdf value of this pixel in all the Gaussian distributions of the object and background, and choosing the one which yielded the highest probability, while considering the distance between the pixel location and the average location (x, y values) of the mean vector.
   - Adds the data of the pixel to a new data matrix for the object or the background, depending on the probability we got.
   - Creates an image where all the pixels that were classified as part of the object are colored.

- Creates a new frame by adding the object colored image to the current frame. This resulted in a new frame, which shows the current frame in addition to coloring the object.
- Saves the newly created frame into the output video.
- Runs Or Dinari's *'fit_partial'* function on the models of the object and background, each with the new data of the corresponding pixels of the current frame. This updates the model to describe the clustering of object and background in the current frame.
  This was done in the *'video_segmentation'* function.

6. The result of this process is the output video, which shows the input video while coloring the object in a specific color throughout the entire video.

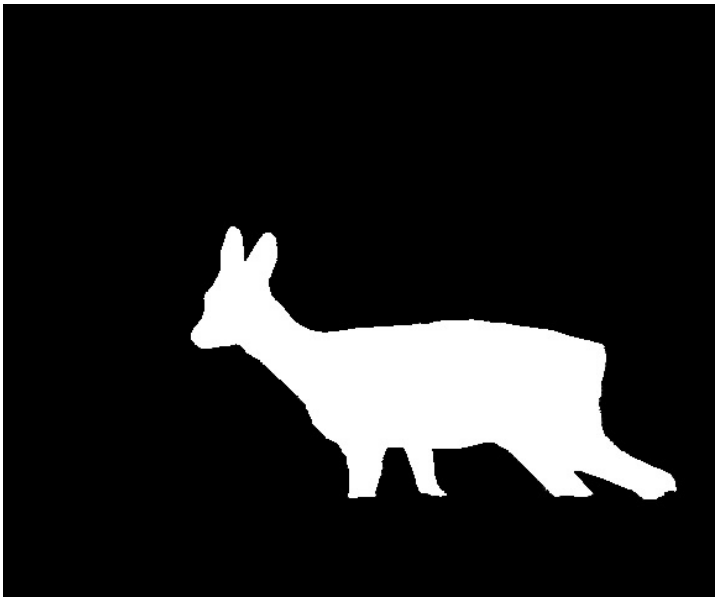
Results

To show the performance and results of our project, we have included the following findings:

- As mentioned, the output of our entire project is the video named ***'final_output'***. We also added a video named ***'comparing_video'***, displaying the input and output videos simultaneously side by side.
- Below are images showing:
    - The first frame of our video.
    - The binary mask which was used to separate the object and background in the first frame.
    - The image of the object extracted by the mask.
    - The image of the background extracted by the mask.
    - An image showing the clustering done by Or Dinari's code (specifically, the clustering done by the models that were the output of the *'fit_init'* function) on the images of the object and background. In these images, every cluster is painted in a different color to show the different clusters.

‘first-frame’

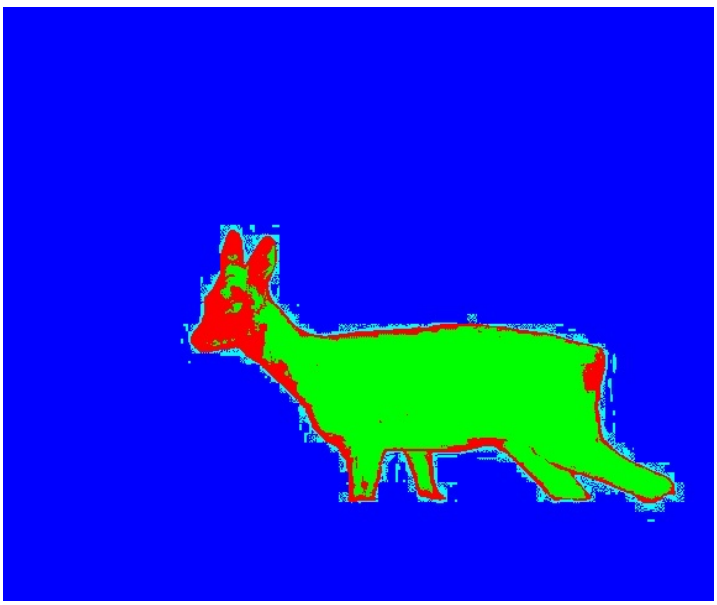
‘mask’



‘object’

‘background’

‘object_colored_by_labels’



‘background_colored_by_labels’