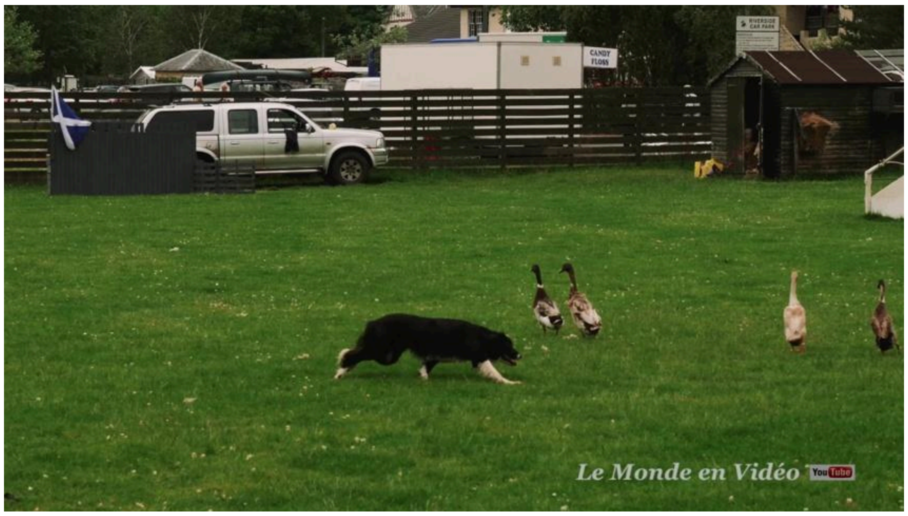
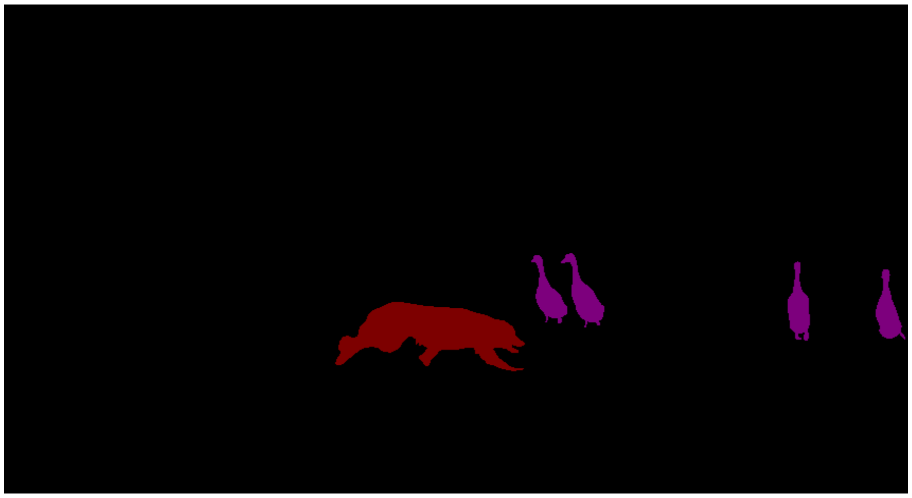
Problem Statement: Instance Segmentation from Semantic Labels

Task: Given a 2D semantic segmentation mask, where each pixel is assigned a unique semantic class label (e.g., “dog”, “goose", “person", "car", "bag"), relabel the mask to produce an instance segmentation mask. In instance segmentation, each connected component of pixels belonging to the same semantic class is assigned a unique instance label.

An Image of a “Dog” chasing “Geese”

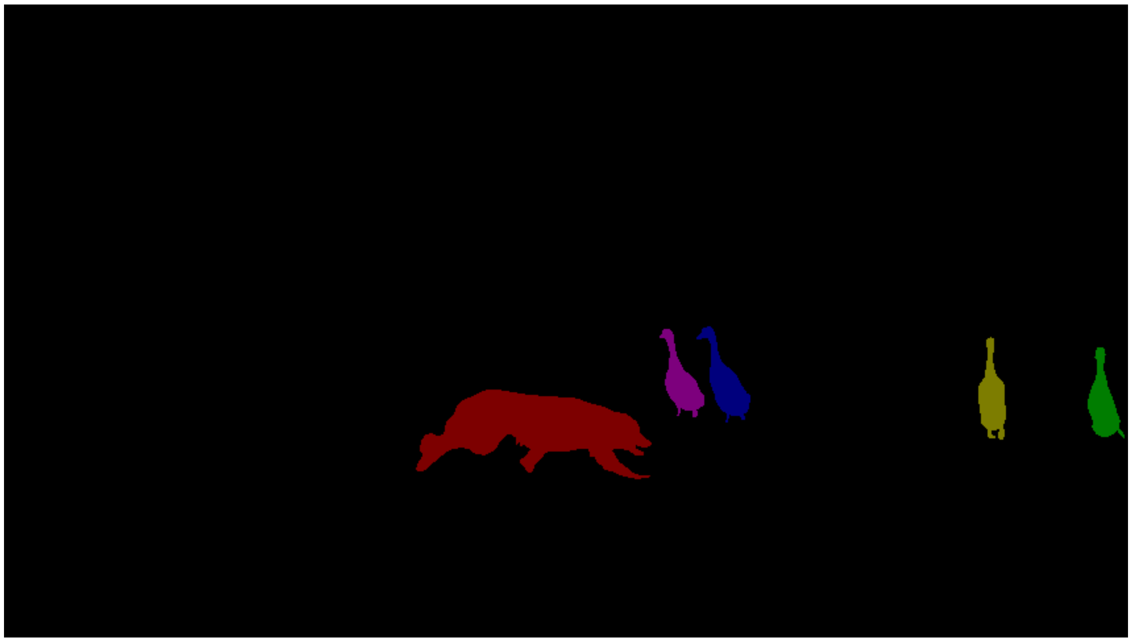


Semantic Segmentation Mask



- Dog semantic class labelled in Red Pixels
- Goose semantic class labelled in Magenta pixels

Instance Segmentation Mask



- First Dog Instance in Red Pixels
- First Goose Instance in magenta pixels
- Second Goose Instance in blue pixels
- Third Goose Instance in yellow pixels
- Fourth Goose Instance in green pixels

Note that specific colours do not matter as long as all of them are distinct.

Definitions:

- **Semantic class:** A category assigned to a pixel in a semantic segmentation mask, representing the object it belongs to. (e.g., "person", "car", "sky").
- **Instance:** A specific occurrence of a semantic class within an image. For example, in an image with two people, there are two person instances.

Input: A semantic segmentation mask: A 2D image where each pixel is assigned a semantic class label.

Output: An instance segmentation mask: A 2D image where each pixel is assigned an instance label. Pixels with the same instance label belong to the same connected component within the same semantic class.

Constraints:

1. Program must be written in python over Google colab. After finishing the code, download python notebook as .ipynb (File>Download>.ipynb) and save it as <Rollnumber>.ipynb. (example 20EC91J02.ipynb)
2. Use OpenCV, Numpy and Matplotlib to load the image or save the result, create a numpy array and display an image. A sample code has been provided to load, create a numpy array and plot an Image. **Apart from the functions listed in sample code, You are not allowed to use any built-in function from these or any other libraries.**
3. Do not Hardcode. Your code should automatically accommodate any number of semantic labels in semantic segmentation masks. Your code will be tested on a held-out test image.
4. **Graph Formulation:** Consider each pixel as a node in the graph. Sum the value of the red, green, blue channel of every pixel to get its corresponding label (example provided in sample code). You must use connected components over graphs discussed in class to get the connected components.
5. **8-way pixel connectivity:** node (i.e. pixel) connectivity should be determined using 8-way pixel connectivity. Two pixels, P1 and P2, are considered connected, i.e. an edge exists between the pixel P1 and P2, if they have the same semantic class label and their Euclidean distance is less than or equal to $\sqrt{2}$ (thus P1 should be a neighbour of P2 in either of North, South, East, West, North-East, North-West,South-East or South-West direction. let $P1_x$ and $P1_y$ be the x- and y-coordinate of P1 respectively, and label(P1) is the label of P1, then P1 and P2 are connected if:

   label(P1) == label(P2) AND $\sqrt{\left(P1_x - P2_x\right)^2 + \left(P1_y - P2_y\right)^2} \leq \sqrt{2}$

Example:

Semantic Segmentation Mask: (0: background, 1: person, 2: car)	**Instance Segmentation Mask:** (0: background, 1: First person instance, 2: First car instance, 3: second person instance)
[[0, 0, 1, 1, 0], [0, 1, 1, 1, 0], [2, 2, 0, 0, 1], [2, 2, 0, 0, 0], [2, 2, 0, 0, 0], [2, 2, 0, 0, 1]]	[[0, 0, 1, 1, 0], [0, 1, 1, 1, 0], [2, 2, 0, 0, 1], [2, 2, 0, 0, 0], [2, 2, 0, 0, 0], [2, 2, 0, 0, 3]]