Task 4 - Image Segmentation Using Clustering Methods

Image segmentation is one of the most classic and extensively studied tasks in computer vision, with a wide range of applications—from segmenting tumors in medical images to identifying and separating different elements of a road scene in autonomous driving, such as lanes, vehicles, pedestrians, and traffic signs. Despite this, image segmentation tasks often face substantial challenges related to data. One of the primary issues is the need for high-quality, pixel-level annotations, which are both time-consuming and expensive to produce—especially in domains like medical imaging, where expert knowledge is required. To address these challenges, unsupervised methods have been deployed to automatically generate segmentation masks required for training these data. One such method is clustering, which groups pixels based on their features—such as color, intensity, or spatial information— into different segments. Although this might be a complex problem in cluttered images it is possible for some image datasets. In this section you are going to create segmentation masks for players from the football player segmentation dataset.

0. Understanding Segmentation

The goal of image segmentation is to divide an image into meaningful parts or regions. This is done by assigning a label to each pixel. Note that this is different from classification where each image is assigned to one or multiple labels. This makes segmentation essential for tasks requiring precise localization, such as identifying individual players in a football game or isolating specific objects in complex scenes. Segmentation tasks can be categorized into multiple categories.

Segmentation tasks can be categorized into multiple categories. The simplest one is **semantic segmentation**, which assigns each pixel a class label based on the object or region it belongs to, without distinguishing between different instances of the same class. For example, in a football game image, all pixels belonging to players would be labeled as "player," regardless of which specific player they represent. **Instance Segmentation** is another type of segmentation whose goal goes beyond semantic segmentation by not only labeling pixels by their class but also differentiating between individual instances of the same class. In the football player dataset, instance segmentation would assign unique labels to each player, enabling separation of player 1 from player 2, even though both belong to the "player" class. Finally **Panoptic Segmentation** combines semantic and instance segmentation, labeling every pixel with both a class and, for countable objects, an instance ID. In the context of the football dataset, panoptic segmentation would assign unique IDs to each player while labeling background elements like the ground or crowd as single, separated regions. Focus of this project is mainly on semantic segmentation. You are going to use simple clustering approaches for this assignment. The reason is that it is possible to cluster image pixels where each cluster becomes one segment of the image. With this knowledge you are ready to start the next parts. Good luck!

1. Dataset Loading

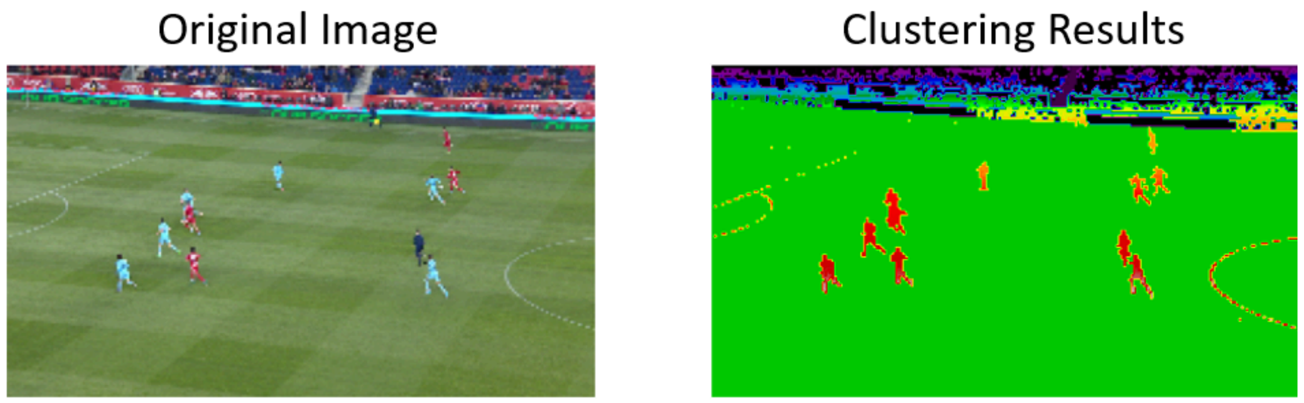
- Load the dataset from the Kaggle link. Because this section contains no training and not the whole dataset is needed. Sample 50 images from the dataset and use them for the rest of this section. This dataset contains two directories. The annotations which contain a json file of all segmentation masks and the images. The images are in size of 1920*1080. Down scale images to 1/8 or 1/16 of the original size or otherwise you will run into performance and memory issues in the upcoming parts.

2. Creating Features (20 points)

Feature creation is the most important part of the clustering. Pixel features should be crafted in such a way that meaningful regions which include players get allocated into one or multiple clusters. Like the example in the course lectures, it is possible to do the clustering only using pixel colors. But note that this simple feature will not likely result in great results. You should try different more complex feature selection methods and report the results for each of them. For example one such example can be concatenating the pixel colors with the pixel positions in the image. Report and evaluate all feature selection methods you explored.

3. Cluster pixels (30 points)

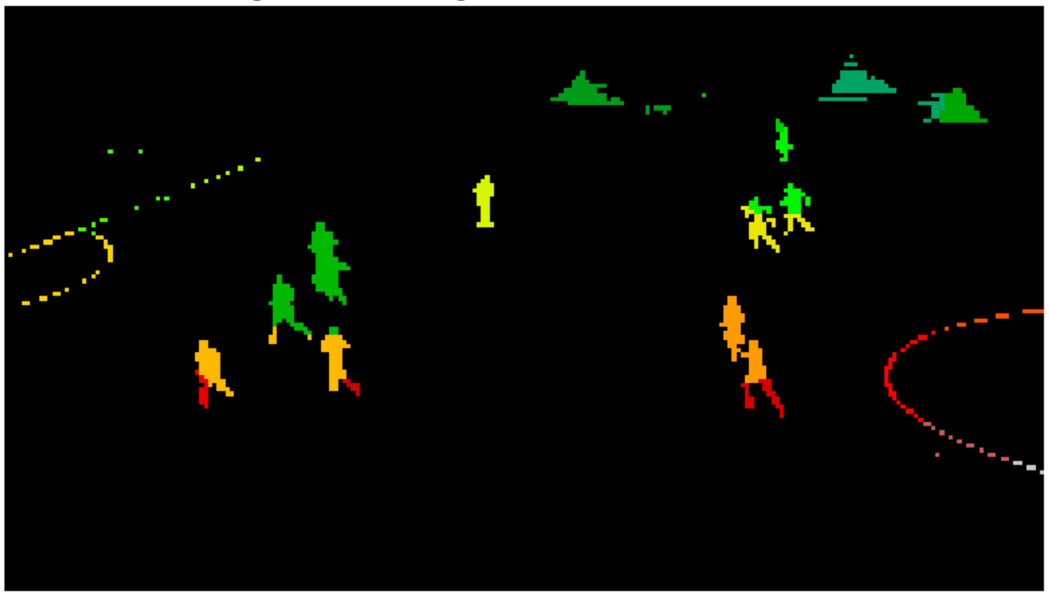
Many different clustering methods from simplest ones like K-Means to more advanced approaches such as density-based clustering (e.g., DBSCAN) and hierarchical methods (e.g., Agglomerative clustering). Use these methods or any similar clustering approach that you find more suitable. The most important task of this part is tuning the clustering parameters. For example if you use K-Means, the selection of K can be challenging especially in this task where it might be difficult to exactly determine the K based on prior knowledge. Use metrics like Silhouette score, inertia, and any suitable metric you find. Output of this part should be multiple clusters. It is not necessary that each player is clustered in only one cluster. You can visualise your clusters to see the results of your segmentation. Note that you might see that each player is segmented into multiple clusters which is fine up to this part. But be careful that there might exist a problem if for example parts of the ground and a player parts are in the same cluster. An example of the clustering can be like this:



4. Filtering and Merging(15 points)

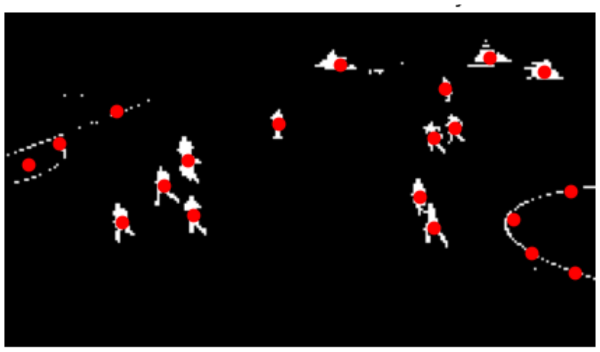
More than the problem mentioned in the last part, there is the problem of segments that do not contain the players, for example the ground. To filter these segments you can set different thresholds on the size of each cluster and apply filters that can drop unrelated clusters. After that you need to merge smaller clusters that are positionally next to each other in the image to create bigger clusters.

After this part you should notice each player is segmented into one or two clusters. An example is shown below.



5. Even More Clustering! (15 points)

In this step, you will generate a binary mask from the clusters obtained in previous parts, where each pixel is assigned a value of 1 (True) for regions corresponding to players and 0 (False) for non-player regions, such as the ground or other background elements. The binary mask should isolate players, ensuring that only their pixels are marked as 1, while all other areas (e.g., the field, crowd, or other objects) are marked as 0. After creating the mask, apply clustering again to identify connected components within the mask and compute their centroids. Finally, visualize the results to display the binary mask and the identified components with their centroids. An example of such a binary mask is shown below. White color corresponds to value 1.



6. More Advanced Features! (10 points bonus)

Use a pretrained convolutional model like ResNet or HRNet to create rich feature representations from the input image. Note that these models decrease the height and width of the model at each step. Down scaling the input image can result in losing details of the image and unacceptable results. I suggest that after generating the features apply DBSCAN and try to tune the parameters so that DBSCAN assigns -1 labels to players pixels.
Explain why DBSCAN shows this behaviour in this part. Finally repeat parts 3 and 4 to get you final player segments.

7. Evaluation (20 points)

To evaluate your segmentations, use the ground-truth segments provided in the JSON file located in the annotation folder of the football player segmentation dataset. For evaluation, create a binary mask from your clustering results (from part 5), where pixels corresponding to players are assigned a value of 1, and all other pixels (e.g., ground, crowd) are assigned 0. Similarly, load the ground-truth annotations from the JSON file and convert them into binary masks in the same format, with player pixels set to 1 and non-player pixels set to 0. The goal is to measure the quality of your segmentation by comparing your predicted binary masks against the ground-truth masks. What we want to measure is the amount of overlap between the output segments and ground truth labels. In this part, we will use the Dice coefficient and Intersection over Union (IoU) as evaluation metrics. Compute and report these metrics on a sample of 50 images from the dataset.

Dice Coefficient:

The Dice coefficient is used to measure the similarity between two sets. In the context of image segmentation, it measures the overlap between the predicted segmentation and the ground truth. It is calculated as twice the area of intersection divided by the sum of the predicted and ground truth areas. A Dice score of 1 indicates perfect agreement, while a score of 0 indicates no overlap.

Intersection over Union (IoU):

Intersection over Union (IoU), is a common evaluation metric for segmentation tasks. It is defined as the area of overlap between the predicted and ground truth masks divided by the area of their union. Unlike the Dice coefficient, IoU gives more penalty to mismatches and is often considered a stricter metric. A higher IoU value shows better segmentation performance, with 1 representing a perfect match and 0 indicating no overlap.

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

Questions

- Provide an example of a segmentation task where each type of segmentation—semantic, instance, and panoptic—is appropriate and sufficient, and explain why each type suits the specific task.
- Explain the difference between the Dice coefficient and Intersection over Union (IoU) as evaluation metrics in image segmentation. In what scenarios might one be preferred over the other?
- Imagine a scenario where we need to cluster a large set of images into multiple categories. Methods such as K-means can become computationally expensive and inefficient when applied directly to high-dimensional image data. One approach to address this issue is to use an autoencoder architecture to reduce the dimensionality of the images before clustering. Explain how autoencoders can be used for this purpose. How does this method help improve the efficiency and effectiveness of clustering algorithms like K-means on large image datasets?