# Analysis of Change Detection - Motion Lab

Melika Najkar

*University of Genoa*

**Abstract** -Object detection is considered one of the most challenging problems in this field of computer vision, as it involves the combination of object classification and object localization within a scene. The study focuses on comparing a sequence of images and identifying the differences between them. Furthermore, motion segmentation is performed to detect moving objects in a sequence of images.

## 1 Introduction

Over the past decade, computer vision has seen great strides across a wide array of tasks including object recognition and detection, semantic segmentation, image captioning, face recognition and many more. The success of these models depends heavily on the availability of computational resources and a key ingredient for learning such complex models – large amounts of human annotated data. However, in many scenarios, unfortunately, it still remains that human labeled data is scarce or worse yet, simply unavailable

Image difference analysis is the process of comparing two or more images to identify the differences between them. This technique is widely used in various applications such as change detection, image registration, and motion segmentation.

Motion segmentation is a crucial step in various computer vision applications, including object tracking, motion analysis, and anomaly detection. By comparing consecutive frames in a video sequence, it is possible to identify regions that have changed, indicating the presence of motion.

## 2 Overview of Method

The image difference analysis is performed using the absolute difference method. The absolute difference between two images is calculated by taking the absolute value of the difference between the corresponding pixels of the two images. The resulting difference image is then analyzed to identify the differences between the two images.

Motion segmentation is performed using the change detection method. In this method, the absolute difference between two consecutive images is calculated, and a binary mask is created by thresholding the difference image. The

Figure 1: Empty scene - reference image

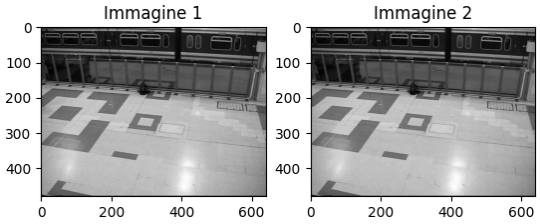


Figure 2: Reference images in grayscale

binary mask is then used to extract the coordinates of the moving objects in the images.

The method involves identifying connected components in each frame, computing their centroids, and then tracking these centroids over time to form trajectories. To enhance the accuracy of these trajectories, a Kalman filter is applied to smooth the observed paths. There are some steps for acquiring the trajectory map:

1.Binary Map Acquisition
2.Connected Component Labeling
3.Centroid Calculation
4.Trajectory Formation
5.Kalman Filtering
6.Visualization

**3 Experimental Setup**

The experiment is conducted using Python and various image processing libraries such as NumPy, SciPy, Matplotlib, and scikit-image. It begins by reading two empty scene which I used them as reference images, that are shown in Fig 1. And converting them to grayscale in Fig 2.

The absolute difference between the images is calculated and visualized to highlight the differences in Fig 3 and Fig 4. The process is then extended to a

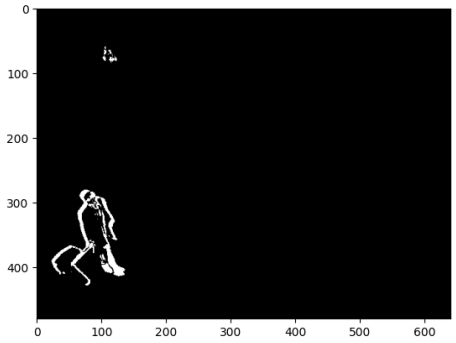


Figure 3: Binary change detection mask of moving object in two frames in a row

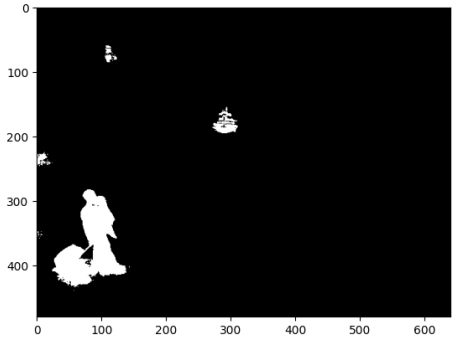


Figure 4: Change detection by comparing first frame and reference image

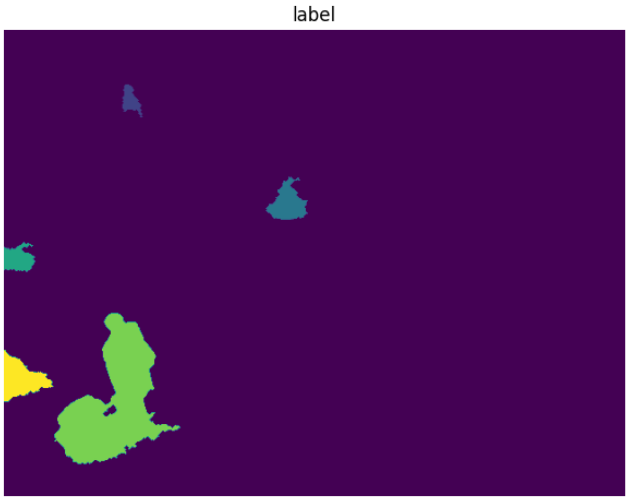


Figure 5: Labeling the connected components

directory containing multiple image files, where the absolute difference between pairs of consecutive images is computed and compared using a specified threshold.

For instance, two frames are compared to detect changes, and the result is displayed as a binary image. Additionally, the coordinates of a target pixel identified as having changed are printed for reference.

Histograms of pixel differences between images are plotted to analyze the distribution of changes. Morphological operations such as closing and removing small objects from the binary change detection mask are applied to refine the segmentation results. Connected components are labeled and visualized to identify distinct regions of motion in the images which are shown in Fig 5.

The motion tracking starts by initializing the tracking process with the centroids of the connected components identified in the first frame. This method provides a robust approach to tracking objects over time in a sequence of binary maps, with the Kalman filter helping to produce smooth and reliable trajectories despite potential noise and occlusions in the data.

For each binary map, the connected components are labeled and then the centroids of these connected components are calculated which are shown in Fig 6.

Starting from the second frame, centroids are matched with those from the previous frame based on their proximity. A threshold is used to determine whether a centroid in the current frame is close enough to a centroid in the previous frame to be considered the same object.

A Kalman filter is applied to the trajectories to smooth the positions over time. The filter uses an initial state that includes the first position and an

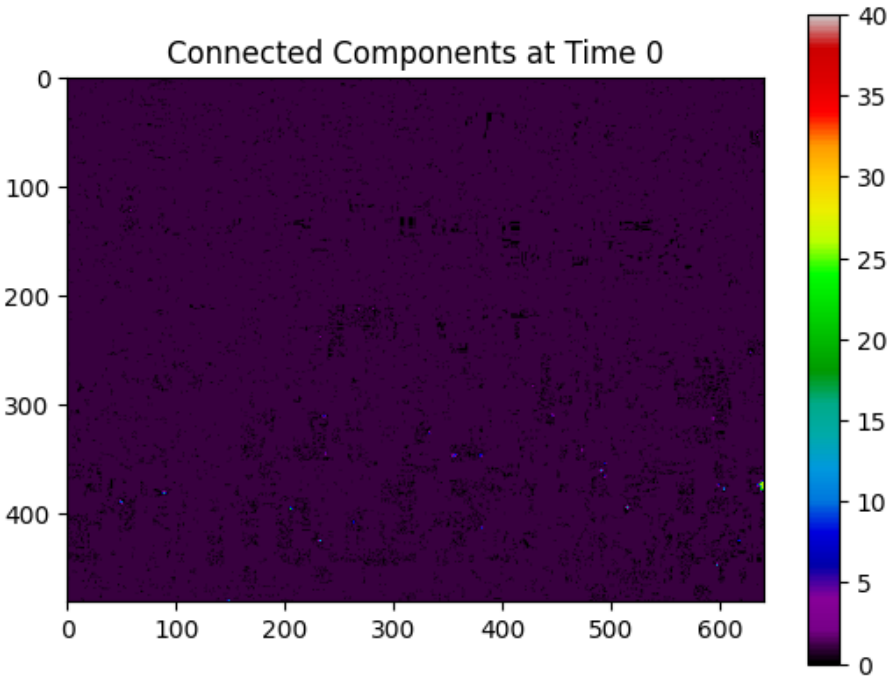


Figure 6:

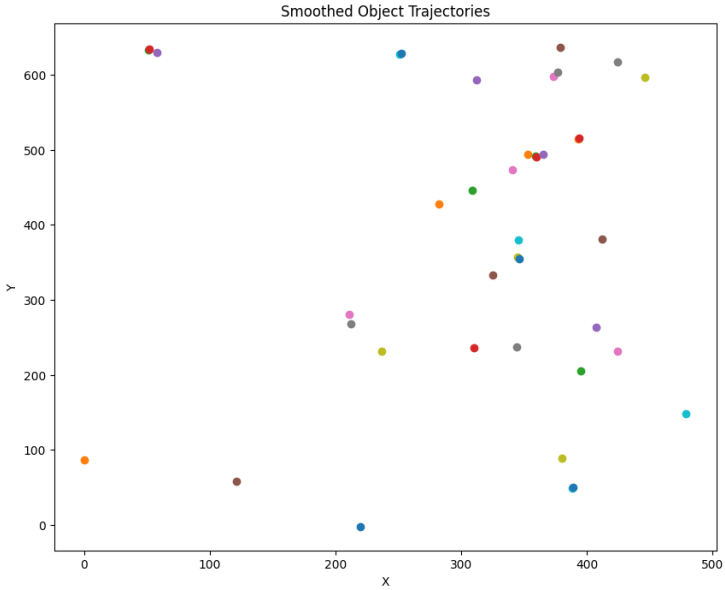


Figure 7: The trajectory

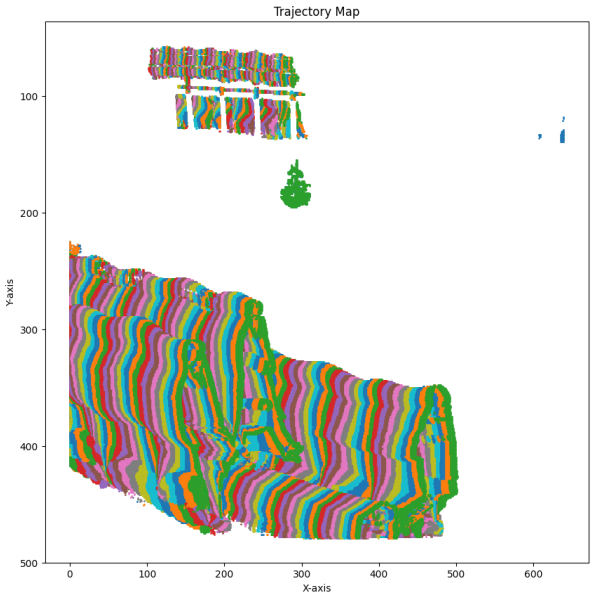


Figure 8: Trajectory of moving object in 70 frames

assumed initial velocity of zero. The filter parameters, including the transition and observation matrices, are defined to model the expected motion dynamics of the objects.

## 4 Conclusion

In this report, I have presented the analysis of image differences and motion segmentation using Python and image processing libraries. The results show that the proposed method can effectively detect changes between images and track the motion of objects in a scene and also there is an obtained trajectory that is referred in Fig 8.

This method provides a robust approach to tracking objects over time in a sequence of binary maps, with the Kalman filter helping to produce smooth and reliable trajectories despite potential noise and occlusions in the data.