**Activity Detection using Optical Flow**

**Dataset:**

For activity recognition, KTH standard dataset is used. KTH publicly available dataset for action recognition was initiated at the KTH Royal Institute of Technology in 2004. Dataset contains 6 actions/classes: walking, jogging, running, boxing, hand-waving and hand clapping. Each action is carried out by 25 individuals and every individual task is recorded in 4 different environments: outdoor (s1), outdoor with scale variation (s2), outdoor with different clothes (s3), and indoor (s4). So, total video samples of the dataset are 25x4x6 = 600. The resolution of each frame of the video is 120x160 and frame rate is 25fps.

**Folder Structure:**

Dataset is divided into folders visualized below:



Figure : KTH Dataset folder structure

Frames from each video are extracted through code and placed in the perspective folder with the same name and hierarchy visualized in figure 1.

 **Optical Flow:**

Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. This technique is used to describe image motion and direction of motion. It works on consecutive frames (video). Optical flow calculates a velocity for points within the images, and provides and estimation of where points could be in the next image sequence. It is used to track objects in the scenery.

Figure 2: Optical Flow Demonstration in Traffic

**Optica Flow Methodologies:**

Following are the types of optical flow based on the attaention mechanism of the image:

1. **Sparse Optical Flow**

Sparse optical flow provides the flow vectors for selected “interesting features” (such as the edges or corners of an object) in the frame.

1. **Dense Optical Flow**

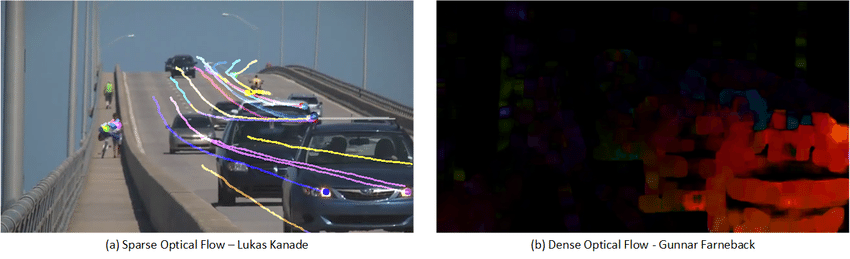
Dense optical flow provides the flow vectors for all pixels in the frame, up to one flow vector per pixel. As expected, Dense optical flow offers higher accuracy but comes at the cost of being computationally expensive and slow.

Figure 3: Sparse optical flow vs Dense optical flow

**Lucas–Kanade** algorithm is the example of sparse optical flow and **Gunnar Farneback’s** is an example of dense optical flow. Python library OpenCV provides both algorithms to use. This project uses Lucas-Kanade algorithm for faster results to meet the real time processing. Below is the description how Lucas-Kanade is integrated with the project.

**Lucas-Kanade:**

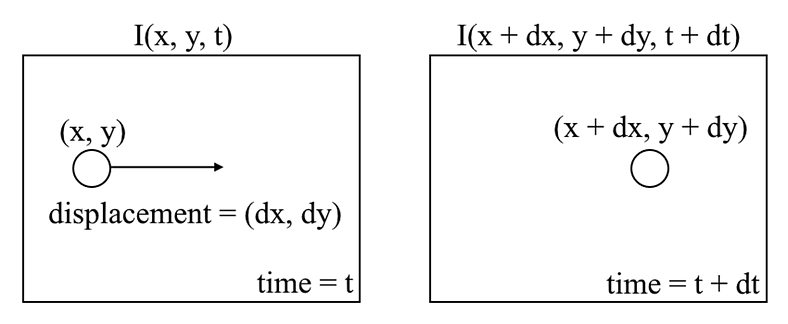
Lucas-Kanade is a popular technique for calculating optical flow under the category of sparse optical flow. This algorithm is based on the Brightness constancy assumption. The key idea here is that pixel level brightness won’t change a lot in just one frame. It assumes that the color of an object does not change between two consecutive frames. So, if we track the pixel as a good point or feature for that frame, the movement of that constant brightness containing pixel would tell us the movement of object in image.

Figure 4: Optical flow tracking mechanism

At code level implementation, first important points or features are highlighted with the OpenCV library function cv2.goodFeaturesToTrack that takes the image in gray scale and some feature parameters like maximum corner points in the image, quality of the corner point, etc. This function calculates the corner quality measure at every source image pixel using the cornerMinEigenVal or cornerHarris. The corners with minimal eigenvalue less then quality level is simply rejected. The remaining corners are sorted by the quality measure in the descending order. Function throws away each corner for which there is a stronger corner at a distance less than maximum distance specified in the feature parameter. The feature points highlighting can be visualized in figure 5 and 6.

Figure 5: Actual Frame

Figure 6: Good Points of the frame

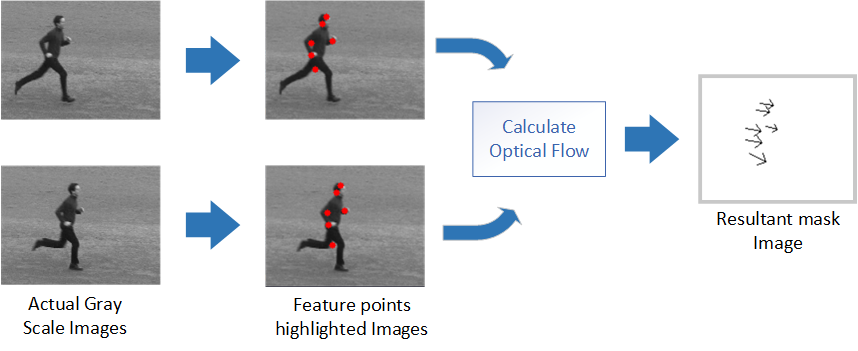
Now cv2.calcOpticalFlowPyrLK function is applied with gray scale consecutive frames and first image good points with additional parameters like window size (in which good point has to be search), maximum level (maximum level of pyramid), etc. Then cv2.calcOpticalFlowPyrLK will return moved good features in second image. Now we have good features for both current frame and previous frame. We can draw directional arrows on mask image with the function cv2.arrowedLine that will show the direction of motion in the frames. This whole process can be visualized in figure 7.

Figure 7: Optical flow calculation steps

Then divide the resultant mask image into 4 segments. If arrows cutting any segment line, then that arrow line is split among those segments. For better visualization, every segment assigned its unique color. Like first, second, third and forth segments are assigned with color red, green, blue and black respectively. This can be visualized in figure 8.

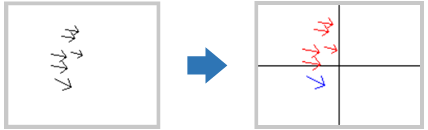
 Now every segment has feature arrows representing direction of motion and amount of motion (magnitude of x and y directions). Important point here is that If any segment has no feature arrow that represent no movement in that segment.

Figure 8: Feature Segmentation

Now normalize every segment of image, add all x directional resultant force and y directional resultant force. For every segment we have single x and y values representing x resultant force magnitude and direction and same for y. Also, we can find the angle between these directional forces.

= tan ()

Resultantly we have three features of each segment (x-directional force, y-directional force, angle). And each image has 4 segments. So, every result of two image optical flow has 4 x 3 = 12 features. These features are then stored in csv file for further model processing.

Standard KTH dataset has 6 actions or classes e.g. walking, running, jogging, boxing, hand clapping and hand waving. 3 of them (walking, running, and jogging) could be further divided as walking-left and walking-right. The resultant x directional force will determine this. If x is a positive (+) number then the moving object is moving in right direction like person walking left to right. Also, the negative x determines leftward direction. So, the dataset for classification here has 9 classes with distribution displayed in figure 9.

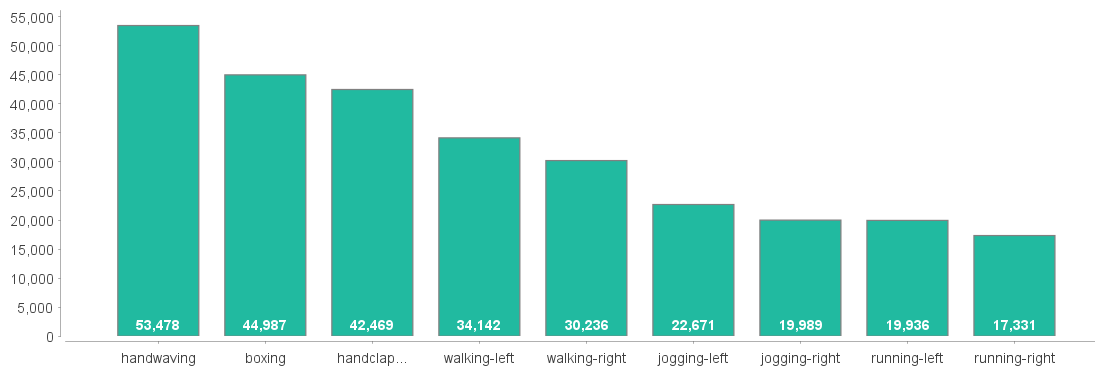


Figure 9: Class distribution of dataset

**Model:**

**Result:**

1. **Action Classification using Optical Flow:**

Action classification is a task for labeling the data stream (video, image, skeleton, etc.) with an appropriate label like walking, watching TV, etc. The main input stream is a live stream or video stream which contains both special and temporal information. The task is to tell which action is performed in the video stream. Optical flow (OF) is the simple and easy way of computer vision (CV) to do this task. Below is the complete methodology diagram which visualizes how optical flow could be used for action classification.

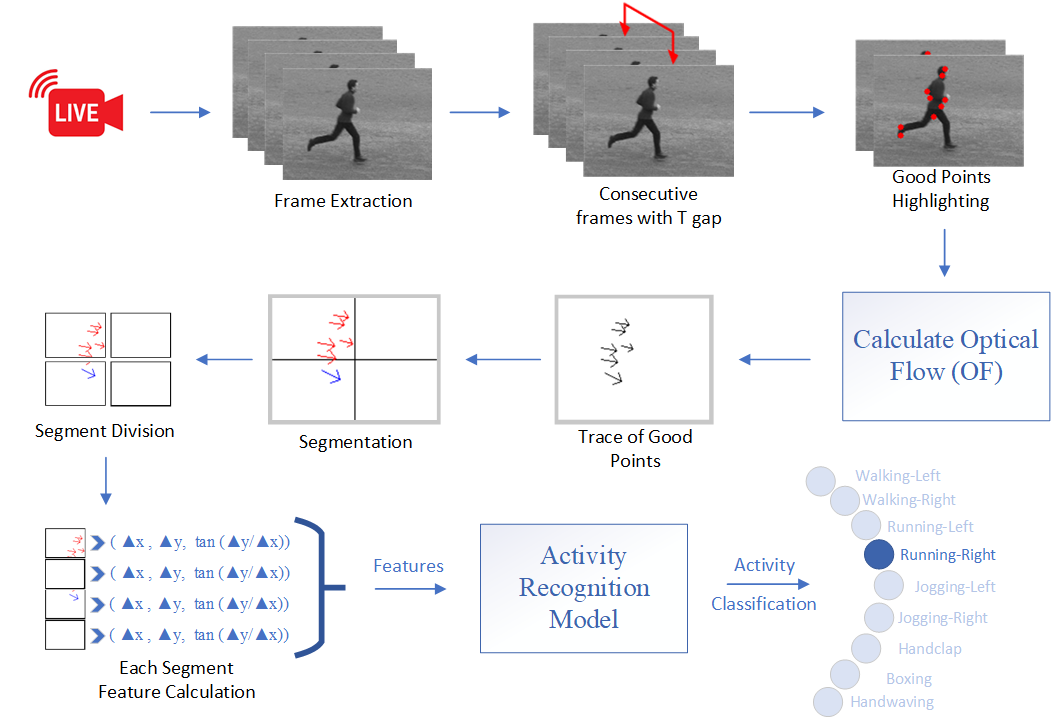


Figure 1: Optical Flow Complete Methodology

Stages of action classification using optical flow (visualized above) are explained below.

1. **Dataset**

For activity recognition using optical flow, the KTH standard dataset is used. KTH is a publicly available dataset for action recognition. It was initiated at the KTH Royal Institute of Technology in 2004. The dataset contains 6 actions/classes: walking, jogging, running, boxing, hand-waving and hand-clapping. Each action is carried out by 25 individuals and every individual task is recorded in 4 different environments: outdoor (s1), outdoor with scale variation (s2), outdoor with different clothes (s3), and indoor (s4). So, total video samples of the dataset are 25x4x6 = 600. The resolution of each frame of the video is 120 x 160 and frame rate is 25fps. Folder structure of the KTH standard dataset is visualized below.

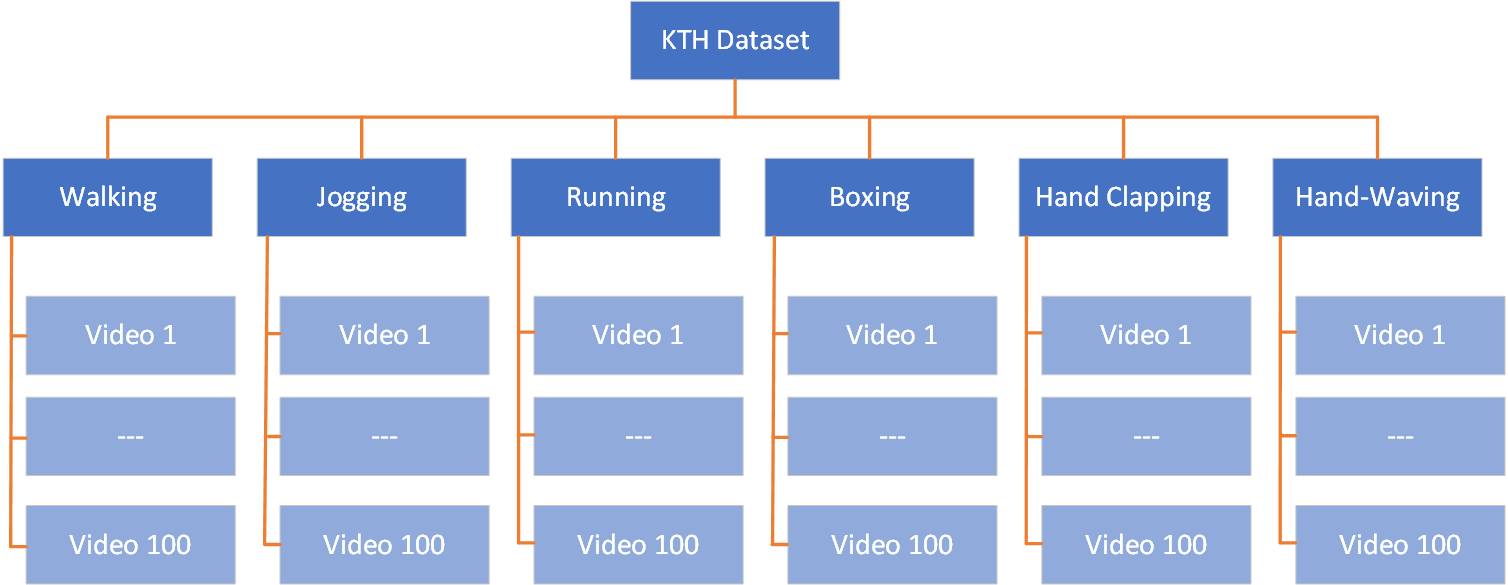
1. **Input Process**

Figure 2: KTH Dataset folder structure

KTH standard dataset (used in this methodology) is a video-based dataset with extension .avi. For the calculation of optical flow, videos are converted to image sequences with a .png extension. A video of 15 seconds will approximately generate 15 x 25 = 375 images. Each image has a dimension of 120 x 160 with an RGB color sequence. The adopted convention organizes these images within a folder with the same name as the video from images are extracted. The total size of KTH videos is 1.08 GB on the disk and extracted images have the size of 9.08 GB on the disk.

1. **Input preprocess**

Before calculating optical flow, images are pre-processed. The pre-processing step involves the good point highlighting on the images. Good points are the points that the optical flow traces to determine the speed and direction of motion in video. These points are also called features. For feature highlighting, cornerMinEigenVal or cornerHarris algorithms are used. These algorithms determine the quality measure at every source image pixel. The corners with minimal eigenvalue less than the quality level are simply rejected. Python cv2 library allows feature highlighting with the function named cv2.goodFeaturesToTrack. This function accepts the grayscale images and returns feature points. Highlighted points can be visualized with the filled red circle in the below illustration.



Figure 3: Actual Frame

Figure 4: Good Points of the frame

1. **Optical Flow**

Figure 5: Optical Flow Demonstration in Traffic

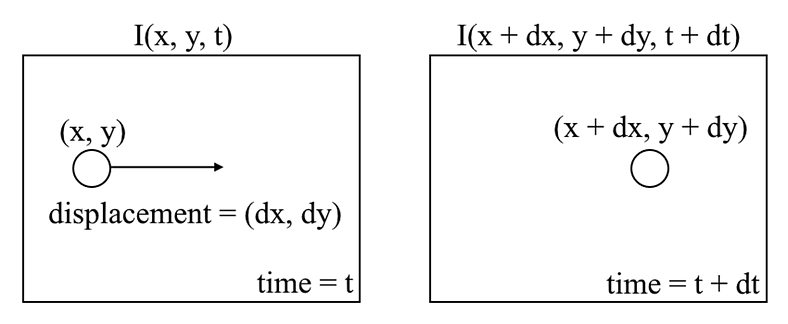
Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between and observer and a scene. Optical flow is a technique used to describe image motion. It is usually applied to a series of images that have a small-time step between them, for example, video frames. Optical flow calculates a velocity for points within the images, and provides an estimation of where points could be in the next image sequence. Optical flow works on the assumption of **Brightness Constancy**. The key idea here is that pixel level brightness won’t change a lot in few consecutive frames. It assumes that the color of an object does not change between two consecutive frames. So, if we track the pixel as a good point or feature for that frame, the movement of that constant brightness containing pixel would tell us the movement of object in image.

Figure 6: Optical flow tracking mechanism

1. **Post Processing of Optical Flow**

Optical flow determines new good points in the current frame considering features of previous frame. This process fulfills the requirement for vectors to make a mask image for motion velocity and direction determination. Input image shape is 120 x 160, so resultant mask image created after optical flow has the same shape i.e. 120 x 160. Segmentation is the process of dividing mask image into sections. Here mask image is divided into 4 equal segments. For better visualization, each segment assigned with unique color i.e. red, green, blue and black for segment 1, 2, 3 and 4 respectively. Compare to mask or input image shape, segment shape reduces 60 x 80 (120 / 2 = 60 and 160 / 2 = 80).

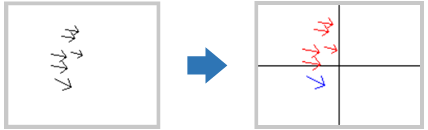


Figure 7: Feature Segmentation

1. **Feature extraction**

Mask Segment has feature arrows representing the direction of motion and amount of motion (magnitude of x and y). An important point here is that if any segment has no feature arrow represents no movement in that segment.

Now normalize every segment of image, add all x directional resultant force and y directional resultant force. For every segment we have single x and y values representing x resultant force magnitude and direction and same for y. Also, we can find the angle between these directional forces.

= tan ()

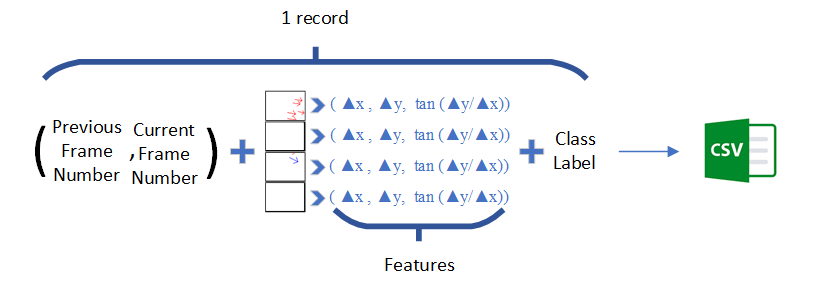
Resultantly we have three features of each segment (x-directional force, y-directional force, angle). And each image has 4 segments. So, every result of two image optical flow has 4 x 3 = 12 features. These features are then stored in csv file for further model processing. Single record saving in CSV file consist of (previous frame count, current frame count), features and then class label. Feature extraction mechanism can be visualized below.

Figure 8: Feature Extraction Format

1. **Action classification**

Machine learning has multiple classification models that process input features to predict output labels or class labels. So, Features are extracted from the KTH dataset successfully and saved in a CSV file. Now to get minimum error, multiple machine learning models can be tested. This brute force approach provides contrast among multiple models and clarifies the model that is best suited for our features. Here below are the results of some classification models applied on features:

Table : Models and their classification errors

|  |  |
| --- | --- |
| Classification Models | Classification Error |
| Naïve Bayes | 0.8073 |
| Generalized Linear Model | 0.8151 |
| Logistic Regression | 0.8071 |
| Decision Tree | 0.8125 |
| Random Forest | 0.8125 |
| Support Vector Machine | 0.8391 |

From above results, Logistic regression is the model best suited having least classification error for out features.

1. **Experimental setup**

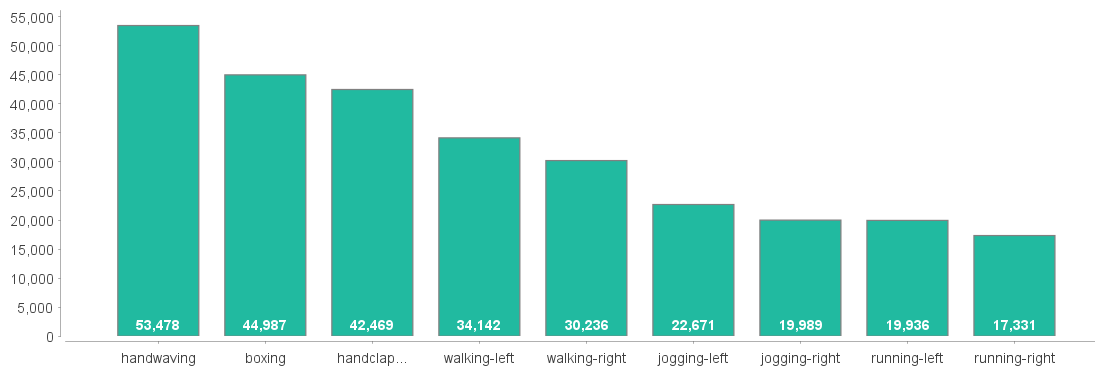
The standard KTH dataset has 6 actions or classes e.g. walking, running, jogging, boxing, hand clapping, and hand waving. 3 of them (walking, running, and jogging) could be further divided as walking-left and walking-right. The resultant x-directional force will determine this. If x is a positive (+) number then the moving object is moving in the right direction like a person walking left to right. Also, the negative x determines the leftward direction. So, the dataset for classification here has 9 classes. Dataset class distribution can be visualized in the figure below.

Figure 9: Class distribution of dataset

1. **Evaluation Matrices**

Precision (P) or detection rate defines the percentage of accurately classified instances to the total labeled instances. It signifies the true positive value and can be used to measure the prediction model. It is illustrated below:

*Precision (P) = TP / (FP + TP)*

Recall (R) or Sensitivity is defined as the proportion of labeled occurrences to all instances. The R measure, which is commonly defined as the true positive figure, denotes the predictions’ model. which is defined by:

*Recall(R) = TP / (F N + TP)*

The F1 score is the harmonic mean of precision and recall, which means that the F1 score will tell you the model’s balanced ability to both capture positive cases (recall) and be accurate with the cases it captures (precision). It is calculated by the formula:

*F1 Score = 2 x (Precision x Recall) / Precision + Recall*

1. **Results**

The selected technique (Logistic Regression) uses to generate results for the features. Features generate the following resultant values for the evaluation matrix:

Table : Results for Evaluation Matrix

|  |  |
| --- | --- |
| Evaluation Matrix | Result |
| Accuracy | 0.297433740008414 |
| Recall | 0.297433740008414 |
| F1 Score | 0.297433740008414 |

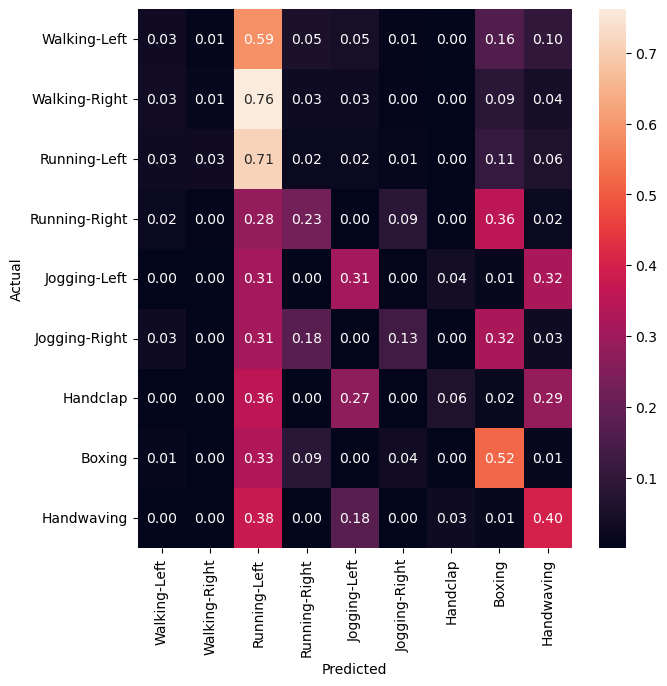
A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm. A confusion matrix diagonal values from left-top to right-bottom defines accurately determined values. The confusion matrix created using logistic regression is visualized below.

Figure 10: Confusion Matrix of Activity Recognition using Optical Flow through Logistic Regression Technique

Above created confusion matrix clearly visualizes greater error rate in classes of Walking and Handclapping as both classes have least diagonal value.

1. **Conclusion**

The utilization of optical flow with logistic regression for human activity recognition showcased some promising results. Result shows good accuracy for diverse actions and worse results for similar-category activities within video sequences. The technique demonstrated low precision, recall, and F1 score determines its capacity and limitation. This limitation creates a solid foundation to ensemble other methods and models for accurately determine complex and similar-category activities.