Mini Project Report

on

A Study on Feature Descriptors for Action Recognition

Submitted by

Riddhish Ganesh Mahjan (21bcs094)

Sahil Kirti (21bcs097)

Samarth Shinde (21bcs110)

Shivam Kumar (21bcs112)

Under the guidance of

Dr. Vivekraj V.K

Assistant Professor (CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

28/04/2024

Certificate

This is to certify that the project, entitled **A Study on Feature Descriptors for Action Recognition**, is a bonafide record of the Mini Project coursework presented by the students whose names are given below during Academic Year 2024 in partial fulfilment of the requirements of the degree of Bachelor of Technology in Computer Science and Engineering.

Roll No	Names of Students
21BCS094	Riddhish Mahajan
21BCS097	Sahli Kirti
21BCS110	Samarth Shinde
21BCS112	Shivam kumar

Dr Dibyajoyti Guha
(Project Examiner)

Dr. Vivekraj VK
(Project Supervisor)

Contents

List of Figures

List of Tables

1 Introduction

In contemporary computer vision research, recognizing human movement is crucial, offering insights into interactive environments and facilitating applications such as surveillance and human-computer interaction. Traditional approaches often focus on intricate motion measurements and 3D reconstructions, which pose challenges in handling diverse viewing conditions and dynamic environments. In response, our mini project presents a novel framework centered on action labeling rather than detailed motion measurement, aiming for robust recognition amidst varying viewing conditions.

1.1 Addressing Evolving Challenges

Our approach diverges from conventional 3D reconstruction methods, prioritizing a view-based framework that accentuates movement appearance. By leveraging temporal templates and simplified representations, our method aims to encapsulate the essence of human movement, facilitating robust recognition across dynamic scenarios. This departure from traditional approaches underscores our commitment to addressing the evolving challenges in computer vision research.

1.2 Introduction to Temporal Templates and Spatio-Temporal Features

Central to our approach are two key temporal templates: the Binary Motion-Energy Image (MEI) and the Motion-History Image (MHI). The MEI provides a binary representation of motion, simplifying movement depiction for tasks such as action recognition, while the MHI offers a scalar-valued portrayal of motion history over time, encapsulating the temporal evolution of movement patterns. Additionally, we introduce spatio-temporal interest points (STIPs), extending spatial interest points into the video domain.

1.3 Exploring Recognition Techniques

In our exploration of recognition techniques, we observe a shift towards appearance-based models focusing solely on the two-dimensional appearance of actions. These models, diverging

from traditional 3D reconstruction approaches, offer promising avenues for action recognition in dynamic environments. By balancing the strengths and limitations of model-driven and action recognition approaches, we aim to design effective systems capable of accurately interpreting human movement in real-world scenarios.

Action recognition is a fundamental problem in computer vision, crucial for applications ranging from surveillance to human-computer interaction. Our study proposes novel approaches for action recognition, combining dense sampling with feature tracking to model videos using dense trajectories. Unlike previous methods, dense trajectories efficiently capture motion information by densely sampling feature points and tracking them using displacement information from dense optical flow fields.

Furthermore, we introduce a novel descriptor based on motion boundary histograms (MBH), robust to camera motion, and consistently outperforming other state-of-the-art descriptors like HOG and HOF on challenging datasets. These dense trajectories coupled with MBH descriptors achieve significant improvements over previous methods, demonstrating their effectiveness on datasets like the UCF sport action recognition dataset.

In summary, our mini project presents a comprehensive approach to recognizing human movement, integrating temporal templates, spatio-temporal features, and advanced recognition techniques. This approach not only addresses the challenges posed by dynamic environments but also offers promising avenues for real-world applications in video processing and understanding.

2 Related Work

Human action recognition is a challenging yet essential task in computer vision with numerous real-world applications ranging from surveillance to human-computer interaction. Over the years, researchers have explored various methodologies to tackle this problem.

2.1 Human Recognition Temporal Templates

In the realm of computer vision and human movement analysis, a plethora of research efforts have contributed to the understanding and recognition of dynamic actions. This section presents an overview of existing methodologies and advancements in the field, highlighting their contributions and limitations.

Model-Driven Approaches: Traditional model-driven approaches have long been prevalent in human movement analysis, focusing on detailed 3D reconstructions to capture intricate motion information. These methods, while effective in capturing fine-grained movement dynamics, often face challenges such as self-occlusion and complex backgrounds. Despite these limitations, model-driven approaches remain valuable in providing detailed 3D information for precise motion analysis.

Action Recognition Methods: In contrast to model-driven approaches, action recognition methods prioritize the identification of actions based on their visual appearance rather than detailed motion measurement. These methods offer a more straightforward means of action identification, utilizing view-based representations and normalized object images. While some approaches leverage grayscale images for hand gesture recognition, others employ body silhouettes and contours for broader action recognition. Despite attempts to mitigate variability between individuals, challenges persist, including variations in contour information due to background and clothing complexities.

Appearance-Based Models: A significant divergence from traditional 3D reconstruction approaches is the emergence of appearance-based models for human action recognition. These models focus solely on the two-dimensional appearance of actions, utilizing grayscale images,

body silhouettes, and contours for action recognition. While these approaches offer promising avenues for action recognition in dynamic environments, challenges such as variability in silhouette regions and background complexities pose significant hurdles.

Motion-Based Recognition: In motion-based recognition, techniques prioritize analyzing movement patterns directly rather than static body poses. Examples include tracking walking motions by analyzing repetitive patterns and low-level motion characteristics across the entire body. Others concentrate on facial expressions, tracking motions of interest regions such as the mouth using optical flow. Despite challenges in accurately recovering motion parameters, these techniques offer valuable insights into movement dynamics and expression recognition.

2.2 Spatio-temporal features

Action recognition research has been a vibrant area within computer vision, exploring various methodologies and techniques to decipher human actions from video data. Early approaches primarily focused on hand-crafted features, meticulously designed to capture distinct characteristics of human movement. These features, including the likes of Histogram of Oriented Gradients (HOG)[30] and Histogram of Optical Flow (HOF)[32], offered valuable insights into the appearance and motion patterns associated with actions. However, these methods often required significant domain expertise and struggled to generalize well to diverse datasets or complex scenarios.

The advent of deep learning, particularly Convolutional Neural Networks (CNNs), revolutionized action recognition by enabling the automatic learning of spatial features from raw video data. However, a challenge persisted in capturing temporal information, pivotal for understanding action dynamics. To address this, Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, were adopted, demonstrating proficiency in learning long-range temporal dependencies in video sequences

Subsequent advances in deep learning led to the design of specialized networks tailored for action recognition. For example, two-stream networks employed distinct pathways to process spatial

and temporal information concurrently, effectively integrating appearance and motion cues for improved recognition performance. Additionally, 3D convolutional networks were developed to learn spatio-temporal features directly from video volumes, achieving remarkable results on benchmark datasets

Despite these advancements, challenges persist in action recognition, particularly in real-world scenarios featuring complex backgrounds, occlusions, and varying camera viewpoints. Moreover, the computational cost associated with deep learning models may hinder their deployment in resource-constrained environments. Therefore, contemporary research is focused on developing robust and efficient algorithms, exploring novel feature representations, and integrating contextual information to deepen the understanding of human actions.

2.3 Action Recognition using Dense Trajectory

Traditional approaches for action recognition have relied on local space-time features detected at sparse interest points. Laptev and Lindeberg [16] introduced space-time interest points by extending the Harris detector to video. Other interest point detectors have been developed based on Gabor filters [17] [18], determinants of the spatio-temporal Hessian matrix [19], and other methods. Local descriptors encode the appearance and motion information around these interest points using representations like histograms of gradients and optical flow.

More recent methods have explored trajectories of interest points tracked across multiple frames. Messing et al. [20] tracked Harris3D interest points [16] using the KLT tracker [21] and represented trajectories as sequences of quantized optical flow vectors. Matikainen et al. [22] introduced trajectories from a standard KLT tracker and used affine transformations on the cluster centers as descriptors.

While interest point detectors and KLT tracking provide sparse trajectories, some approaches have investigated denser sampling strategies. Wang et al [23]. showed that dense sampling at regular positions outperforms interest point detectors for action recognition. Sun et al [24]. extracted trajectories by matching SIFT descriptors [25] between consecutive frames with a unique match constraint.

In this work, we build upon the success of dense sampling but propose a more efficient solution to extract dense trajectories directly from optical flow fields computed over the entire video sequence.

2.4 Action Recognition by Improved Dense Trajectory

Several approaches neglect considering camera motion when extracting feature trajectories for action recognition. Uemura et al. [1] propose a method that combines feature matching with image segmentation to estimate the primary camera motion, then separating feature tracks from the background. Wu et al. [2] employ a low-rank assumption to differentiate feature trajectories into segments induced by the camera and those induced by objects. Park et al. [3] introduce weak stabilization to eliminate both camera and object-centric motion using coarse-scale optical flow, which is particularly beneficial for pedestrian detection and pose estimation in videos. Jain et al. [4] present a technique to break down visual motion into dominant and residual motions, aiding in trajectory extraction and descriptor computation.

In efforts to enhance dense trajectories, Vig et al. [5] suggest employing saliency-mapping algorithms to trim background features. This strategy leads to a condensed video representation and enhances the accuracy of action recognition. Meanwhile, Jiang et al. [6] opt to cluster dense trajectories, utilizing the cluster centers as anchor points to model their relationships.

3 Data and Methods

We introduce the UCF Sports Action dataset and the methodologies used for human action recognition.

The UCF Sports Action dataset offers a diverse collection of sports activities captured in real-world settings, serving as a benchmark for action recognition tasks.

Methods for Action Recognition

1. **Temporal Templates**: Basic approach focusing on pixel intensity changes over frames.
2. **Spatio-Temporal Interest Points**: Identifies significant changes in space and time but may face scalability issues.
3. **Dense Trajectories**: Tracks dense points of interest, capturing fine-grained motion cues effectively.
4. **Improved Dense Trajectories**: Enhances dense trajectory methods with additional features for better performance.

We evaluate these methods on the UCF Sports Action dataset to understand their effectiveness in recognizing human actions across various sports activities.

3.1 Temporal Template

The workflow for action recognition using temporal templates involves frame extraction, background subtraction to isolate moving objects, and deriving Motion Energy Images (MEI) and Motion History Images (MHI). These images capture dynamic information over time, enabling the calculation of motion similarities and construction of temporal templates. Action recognition is then performed based on these templates, allowing for effective analysis and classification of actions in video sequences. This workflow highlights the significance of temporal dynamics in understanding and recognizing actions. In Figure [1] workflow for temporal templates technique is shown.

Figure 1. Workflow: Temporal Templates

3.1.1 Motion Energy Image(MEI)

The Binary Motion-Energy Image (MEI) serves as a visual representation of motion within an image sequence, offering a simplified yet informative snapshot of dynamic events. In the MEI, each pixel is assigned a binary value, where black indicates the absence of motion and white denotes motion activity. This binary nature of the MEI facilitates clear identification of areas with movement, allowing for efficient analysis and interpretation of dynamic scenes.

Key Features and Characteristics:
Binary Representation: The MEI adopts a binary representation scheme, simplifying the depiction of motion by categorizing each pixel as either static or dynamic. This binary nature enhances the clarity and interpretability of the image, enabling straightforward identification of motion activity.

Visualization of Motion: By focusing solely on the presence or absence of motion, the MEI provides a clear visualization of motion dynamics within an image sequence. Areas with significant motion are highlighted in white, while static regions remain black, allowing for easy identification and analysis of dynamic events.

Informative Snapshot of Dynamic Events: The MEI offers a concise and informative snapshot of dynamic events, capturing the temporal evolution of motion patterns over time. This snapshot encapsulates the essence of movement within the image sequence, facilitating tasks such as action recognition and motion analysis in computer vision applications.In Figure [2b] extracted Motion Energy Image is shown for running action.

3.1.2 Motion-History Image(MHI)

The Motion-History Image (MHI) offers a scalar-valued portrayal of motion history over time within an image sequence. By encoding the recency of motion occurrences through pixel intensities, the MHI provides valuable insights into the temporal evolution of movement patterns, facilitating the analysis and interpretation of dynamic scenes.

Key Features and Characteristics:
Temporal Representation: Unlike the Binary Motion-Energy Image (MEI), which provides a binary snapshot of motion, the MHI offers a scalar-valued representation of motion history over time. Intensity levels within the MHI correspond to the recency of motion occurrences, with brighter regions indicating more recent activity and darker areas representing older or stationary motion.

Temporal Evolution of Movement: The MHI encapsulates the temporal evolution of movement patterns within an image sequence, offering a comprehensive overview of dynamic events over time. By highlighting the recency of motion occurrences, the MHI enables the analysis of motion trajectories and the detection of temporal patterns in dynamic scenes.

Facilitates Action Dynamics Analysis: With its temporal representation of motion history, the MHI serves as a powerful tool for analyzing action dynamics and temporal segmentation tasks. By examining the distribution of pixel intensities over time, systems can identify key action phases and discern between different stages of movement within the image sequence.

Action Recognition: In addition to temporal segmentation, the MHI plays a crucial role in action recognition tasks. By capturing the temporal dynamics of movement, the MHI provides valuable insights into action dynamics and facilitates the identification of specific actions based on their temporal characteristics.

Gesture Recognition: The MHI is also utilized in gesture recognition applications, where the temporal evolution of hand movements is of particular interest. By analyzing the intensity distribution within the MHI, systems can recognize and interpret complex hand gestures and

movements in real-time. In Figure No [2c] extracted Motion History Image is shown for running action

(a) Run action (b) MEI for run action (c) MHI for run action

Figure 2. MEI , MHI and Run Action

3.1.3 Steps

Our methodology revolves around the utilization of temporal templates, specifically the Binary Motion-Energy Image (MEI) and the Motion-History Image (MHI), for human movement recognition. These templates serve as foundational elements in our approach, capturing the temporal evolution of movement patterns and facilitating robust recognition across varied viewing conditions.

Data Preprocessing: We initiated our study by preprocessing the video sequences obtained from the UCF Sports dataset to prepare them for further analysis. This crucial step involves segmenting the videos into individual frames and annotating each frame with corresponding action labels. By doing so, we ensured that the data were appropriately labeled and ready for feature extraction.

Feature Extraction: Next, we proceeded with feature extraction from the preprocessed frames. Leveraging our methodology centered around temporal templates, specifically the Binary Motion-Energy Image (MEI) and the Motion-History Image (MHI), we computed these representations to encapsulate the temporal dynamics of movement within the video sequences.

The MEI and MHI representations served as foundational elements in our approach, capturing intricate movement patterns essential for robust recognition across varied viewing conditions.

Model Training: With the extracted MEI and MHI features in hand, we moved on to the training phase of our recognition model. Employing machine learning algorithms, including Support Vector Machines (SVM), cosine similarity, and k-nearest neighbors (KNN), we trained the model to learn discriminative patterns for action recognition. By harnessing the power of these algorithms, our model acquired the capability to distinguish between different types of human movements with satisfactory accuracy.

Model Evaluation: Following model training, we meticulously evaluated the performance of our trained model on the testing subset of the UCF Sports dataset. Leveraging a comprehensive evaluation framework, we computed performance metrics such as accuracy, precision, recall, and F1-score to quantitatively assess the efficacy of our approach in recognizing human movements across diverse sports activities. Additionally, we utilized confusion matrices and accuracy scores to gain deeper insights into the model's performance and identify areas for improvement.
.

3.2 Spatio Temporal Features

3.2.1 Human Detection

Human detection is a crucial preliminary step in our video analysis pipeline. We utilize a pre-trained object detection model sourced from the TensorFlow Detection Model Zoo[28]. This model is adept at identifying and localizing humans as shown in Figure3 within each frame of the video. By leveraging this pre-trained model, we efficiently pinpoint regions of interest, streamlining subsequent analysis processes.

The choice of employing a pre-trained model from the TensorFlow Detection Model Zoo is strategic, as it offers a balance between accuracy and computational efficiency. Leveraging a pre-trained model saves substantial time and resources that would otherwise be expended on training a detector from scratch. Moreover, the model's robustness to various environmental conditions and its ability to generalize across different scenarios make it an ideal candidate for

our application.

By integrating human detection into our pipeline, we establish a foundation for further analysis, enabling subsequent stages to concentrate solely on pertinent human-related activities. This facilitates a more targeted and streamlined approach towards understanding and interpreting the dynamics within the video content.

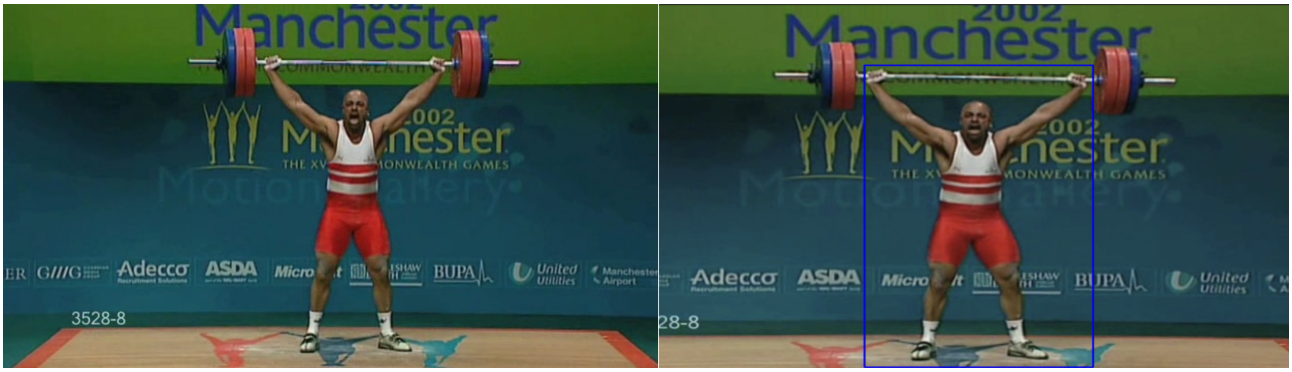

Figure 3. Human Detection

3.2.2 Interest Point Detection

Detecting salient spatio-temporal features within human regions is crucial for understanding motion dynamics. We utilize the Harris 3D detector, an extension of the Harris corner detector, to identify key interest points. These points signify regions with significant changes in appearance and motion, offering valuable insights into human activities.

The Harris 3D detector analyzes variations in intensity and motion across frames, pinpointing areas of notable transformations over time. These interest points as shown in Figure4 serve as anchors for capturing motion patterns and temporal dynamics within the video. By integrating this technique, we enhance feature extraction accuracy, facilitating a deeper understanding of human interactions and behaviors.

Incorporating the Harris 3D detector enriches our system's discriminative capability, enabling it to discern subtle temporal variations and distinguish between different motion behaviors. This integration not only enhances analysis accuracy but also contributes to more insightful interpretations of human activities within the video footage.

Figure 4. Interest Point detection

3.2.3 Feature Extraction

In our feature extraction process, we compute descriptors for each identified interest point to capture both appearance and motion information. Two key descriptors utilized are the Histogram of Oriented Gradients (HOG)[30] and the Histogram of Optical Flow (HOF)[32]. HOG descriptors encode local shape and appearance characteristics, providing valuable insights into the visual attributes of the detected regions.

Concurrently, HOF descriptors capture the direction and magnitude of motion within the vicinity of interest points, offering crucial information about dynamic changes in the video frames. By combining HOG and HOF descriptors, we create comprehensive feature representations that encapsulate both spatial and temporal characteristics, enabling a more holistic understanding of human activities.

This approach enables us to extract rich and discriminative features that effectively capture the nuances of human motion and appearance. By leveraging both HOG and HOF descriptors, our feature extraction process equips our system with the necessary information to discern complex actions and behaviors within the video content.

3.2.4 Feature Encoding and Classification

Following feature extraction, the HOG and HOF descriptors are combined and potentially encoded using advanced techniques Fisher Vectors. FVs capture higher-order statistics beyond mean and covariance, providing a more detailed characterization of the data distribution.This encoding step aims to condense the extracted features into compact representations that preserve essential information about both appearance and motion characteristics. By employing encoding techniques, we enhance the efficiency and effectiveness of subsequent classification tasks.

Subsequently, a classifier such as Support Vector Machine (SVM) is trained on the encoded features to recognize different actions within the video content. These classifiers leverage the encoded feature representations to learn discriminative patterns associated with various human actions. By training the classifier on labeled data, our system becomes capable of accurately identifying and categorizing different actions.

3.3 Dense Trajectory

What is Trajectory? In computer vision, a trajectory is the course of a moving object over time, and can be represented as a matrix of x, y, or uv coordinates on the ground.

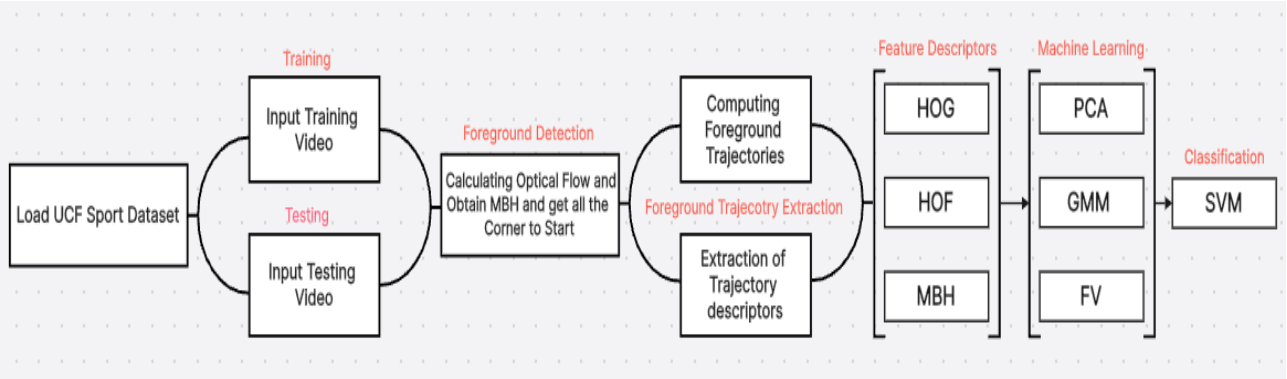

Figure 5. Workflow: Dense Trajectory

3.3.1 Dense Sampling of Feature Points

We have a video frame represented as an image. We want to sample feature points from this frame densely. we have choosen a sampling step size $W = 5$ pixels. This means we will place a feature point on a grid with every 5th pixel location being sampled, as shown in below Figure [6].

Figure 6. Dense sampling

The '*' represents a sampled feature point. We repeat this dense sampling at multiple spatial scales (e.g., 8 scales with a scaling factor of $\frac{1}{\sqrt{2}}$ between each scale) to capture features at different resolutions.

3.3.2 Feature Point Tracking

Figure 7. Trajectories

For each sampled feature point $P_t = (x_t, y_t)$ in frame t, we want to track its position in the next frame $t + 1$. We do this by using a dense optical flow field $\omega = (u_t, v_t)$, which represents the motion vectors between frames t and $t + 1$.

Imagine a point P_t at location $(100, 200)$ in frame t. The optical flow vector ω at this location is $(5, 3)$, indicating a horizontal movement of 5 pixels and a vertical movement of 3 pixels. To find the new location P_{t+1} in frame $t + 1$, we apply median filtering on the optical flow field around $(100, 200)$ and add the filtered motion vector to the original location:

$$P_{t+1} = (100, 200) + (M \times \omega)|(100, 200) = (100, 200) + (5, 3) = (105, 203)$$

We repeat this process for all sampled points, concatenating their positions across frames to form trajectories: $(P_t, P_{t+1}, P_{t+2}, ...)$. We limit the trajectory length to L frames (e.g., $L = 15$) to avoid drifting and prune static or erratic trajectories.Visualization of trajectories is shown in [7].

3.3.3 Trajectory Descriptor

To describe the shape of a trajectory S, we represent it as a sequence of displacement vectors $\Delta P_t = (P_{t+1} - P_t)$. For example, if a trajectory has points $(100, 200)$, $(105, 203)$, $(110, 206)$, the displacement vectors would be:

$$\Delta P_t = (5, 3), \Delta P_{t+1} = (5, 3)$$

We then normalize this sequence by the sum of the magnitudes of the displacement vectors:

$$S' = ((5, 3), (5, 3)) \div \left(\sqrt{5^2 + 3^2} + \sqrt{5^2 + 3^2}\right) = ((5, 3), (5, 3)) \div \left(\sqrt{34} + \sqrt{34}\right)$$

$$S' = ((0.43, 0.26), (0.43, 0.26))$$

This normalized trajectory descriptor S' encodes the shape and motion pattern of the trajectory.

3.3.4 Trajectory-aligned Descriptors

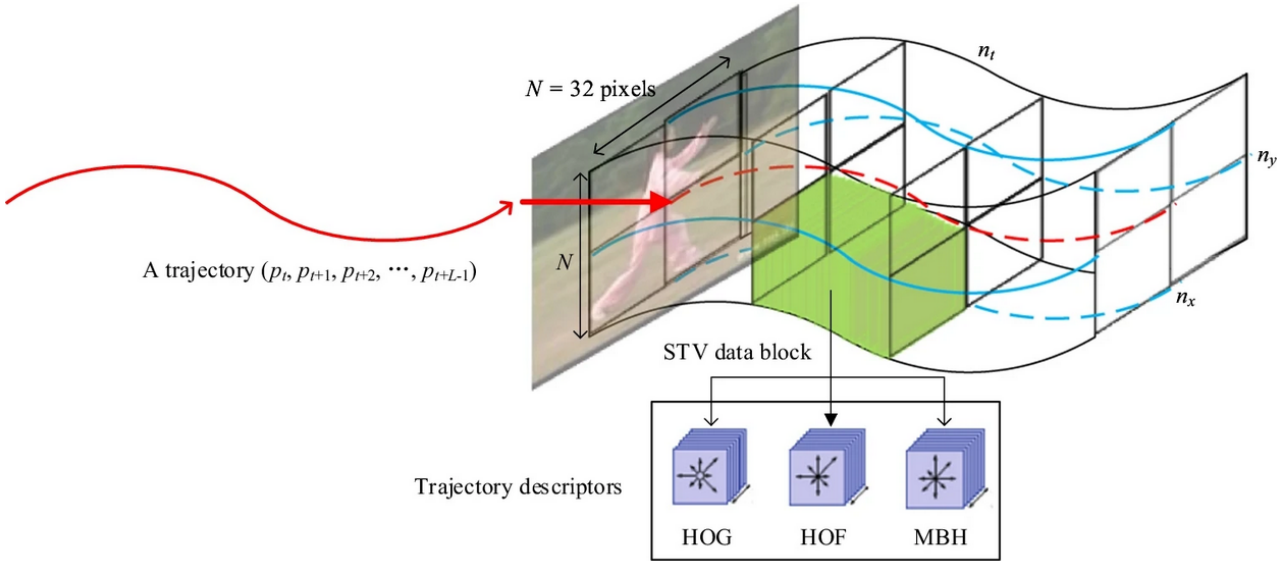

Figure 8. Detail view of dense trajectory descriptor.The left red curve is a trajectory that is constructed by 15 tracked points

For each trajectory, we compute descriptors (Figure [8]) within a space-time volume around it. we have define trajectory has length $L = 15$ frames, and we choose a volume size of $N = 32$ pixels.

We divide this volume into a spatio-temporal grid of size $n_\sigma \times n_\sigma \times n_\tau$, e.g., $2 \times 2 \times 3$. Within each cell of the grid, we compute different descriptors that are describe below:

HOG (Histogram of Oriented Gradients)

HOG captures the appearance information by computing histograms of gradient orientations within the space-time volume surrounding each trajectory. The following steps are involved:

a) Extract the gradient volume (`t.g_volume`) around the trajectory, which contains the gradients computed in the x and y directions for each frame.

b) Split the gradient volume into spatio-temporal grid cells ($n_\sigma \times n_\sigma \times n_\tau$).

c) For each cell, compute the gradient orientations and magnitudes using `cv2.cartToPolar()`.

d) Quantize the orientations into bins (e.g., 8 bins for 0-360 degrees) and accumulate the magnitudes into the corresponding bin for each pixel in the cell.

e) Normalize the histogram and concatenate the histograms from all cells to form the final HOG descriptor.

f) Length of HOG descriptor is 96

HOF (Histogram of Optical Flow)

HOF captures the motion information by computing histograms of optical flow orientations and magnitudes within the space-time volume surrounding each trajectory. The steps are similar to HOG, but instead of using gradients, it uses the optical flow field (`t.of_volume`).

a) Extract the optical flow volume around the trajectory.

b) Split the volume into spatio-temporal grid cells.

c) For each cell, compute the optical flow orientations and magnitudes using `cv2.cartToPolar()`.

d) Quantize the orientations into bins and accumulate the magnitudes into the corresponding bin for each pixel in the cell.

e) Normalize the histogram and concatenate the histograms from all cells to form the final HOF descriptor.

f) Length of HOG descriptor is 108 (because of 9 bins are used)

MBH (Motion Boundary Histogram)

MBH captures the relative motion patterns by computing histograms of spatial derivatives of the optical flow field. It is robust to camera motion and highlights the foreground motion. The steps involved are:

a) Compute the spatial derivatives (x and y) of the optical flow field using `cv2.Sobel()` to obtain the MBHx and MBHy volumes (`t.mbhx_volume` and `t.mbhy_volume`).

b) Split the MBHx and MBHy volumes into spatio-temporal grid cells.

c) For each cell, compute the orientations and magnitudes of the MBHx and MBHy components using `cv2.cartToPolar()`.

d) Quantize the orientations into bins and accumulate the magnitudes into the corresponding bin for each pixel in the cell.

e) Normalize the histograms and concatenate the MBHx and MBHy histograms from all cells to form the final MBH descriptor.

f) Length of MBHx and MBHy descriptor are 96 each respectively

3.3.5 Classification with Feature Encoding

Feature encoding, or constructing a compact representation of the extracted features, is an important step before feeding the data to a classifier for action recognition (or any other classification task).To classify a video, we employ a series of feature encoding and classification techniques for robust and efficient action recognition.

Reasons for Feature Encoding

- Dimensionality Reduction: High-dimensional raw features from videos can be computationally expensive and prone to overfitting.
- Handling Variable-Length Data: Feature encoding technique Fisher vector classification with SVMs ensure consistent input to classifiers despite varying video lengths.
- Capturing Higher-Level Semantics: Encoding groups similar low-level features to capture higher-level patterns.
- Efficient Storage and Retrieval: Compact representations reduce storage requirements and enable efficient data retrieval.

Below we have demonstrated how we apply this technique to classify the action recognition for ucf-sports dataset

Dimensionality Reduction with Principal Component Analysis (PCA)

Before proceeding to feature encoding, we apply Principal Component Analysis (PCA) to reduce the dimensionality of our feature space. PCA helps in capturing the most significant variations in the feature data, allowing us to represent it in a lower-dimensional space while retaining as much variance as possible.

We first concatenates all the descriptors (trajectory shape, HOG, HOF, MBHx, MBHy) from the training videos into a single matrix. Then, PCA (k = 64) is performed on this matrix to reduce the dimensionality of the feature space. Here, PCA(64) specifies that the dimensionality should be reduced to 64 components.

The transformed descriptors are stored for each video representation. The reduced dimensionality helps in improving computational efficiency and reducing noise in the feature space.

Feature Encoding with Gaussian Mixture Model (GMM)

Following dimensionality reduction, we utilize a Gaussian Mixture Model (GMM) to represent the distribution of features in our reduced space.The GMM is trained on the reduced descriptors to capture the underlying distribution of the feature data. In our implementation we have use GMM component as 3

GMM is well-suited for modeling complex data distributions by representing them as a weighted sum of multiple Gaussian components. Each component captures a distinct mode of the feature distribution, enabling us to characterize the underlying structure of the data more accurately.

Feature Encoding with Fisher Vector (FV):

After obtaining GMM representations of our features, we compute Fisher Vectors (FVs) of length 387 for each video to encode the statistics of our data distribution. FVs capture higher-order statistics beyond mean and covariance, providing a more detailed characterization of the data distribution. By encoding the gradient of the log-likelihood with respect to the parameters of the GMM, FVs offer a powerful encoding scheme that captures fine-grained information about the data distribution.

Classification with Support Vector Machine (SVM)

Finally, we employ a Support Vector Machine (SVM) for video classification using the encoded features. SVM is a powerful supervised learning algorithm capable of handling high-dimensional data and nonlinear decision boundaries. We utilize the χ^2 kernel (the χ^2 (chi-square) kernel refers to a specific kernel function used in the Support Vector Machine (SVM) for video classification.) to compute the similarity between feature histograms, effectively capturing the relationships between different visual words in our feature space.

By integrating PCA for dimensionality reduction, GMM and FV for feature encoding, and SVM for classification, we aim to achieve robust and discriminative action recognition performance. This comprehensive approach allows us to capture the shape, appearance, and motion information present in videos effectively, leading to improved classification accuracy and generalization capabilities.

3.4 Improved Dense Trajectory

Enhancing Dense Trajectories through Camera Motion Estimation and Robust Homography Estimation

Enhancing dense trajectories involves several key steps. Initially, we outline the primary stages of our camera motion estimation technique and its application in refining dense trajectories. Subsequently, we address the process of eliminating potentially unreliable matches based on human input to achieve a reliable homography estimation.

3.4.1 Camera Motion Estimation

To assess the overall background motion, we make the assumption that two consecutive frames are linked by a homography [7], which generally holds true except in cases involving independently moving entities like humans and vehicles. Our process for estimating the homography begins by establishing correspondences between frames. We employ a dual approach to ensure a comprehensive set of candidate matches. Initially, we extract SURF [8] features and utilize the nearest neighbor rule for matching. SURF features are chosen for their resistance to motion blur, as validated by recent evaluations [10]. Additionally, we sample motion vectors from optical flow to achieve dense frame matches. Employing a polynomial expansion-based optical flow algorithm

[11], we select motion vectors for salient feature points using the good-features-to-track criterion [12], which involves thresholding the smallest eigenvalue of the autocorrelation matrix. These methods complement each other, with SURF being effective for blob-like structures and excelling at corners and edges, resulting in a more balanced distribution of matched points essential for accurate homography estimation.

Following this, we robustly estimate the homography using RANSAC [13], enabling us to rectify the image and eliminate camera motion. The difference in optical flow before and after rectification is demonstrated in Figure 9, with rectification suppressing background camera motion and enhancing the visibility of foreground moving objects.

Figure 9. Difference in optical flow before and after rectification.

Rectifying optical flow to eliminate camera motion offers two primary advantages for dense

trajectories. Firstly, it directly enhances the performance of motion descriptors. Previous studies [14] have demonstrated a significant degradation in the performance of the HOF descriptor in the presence of camera motion. Our experiments indicate that HOF can achieve comparable performance to MBH when correct foreground optical flow is available.

Secondly, we can filter out trajectories generated by camera motion by thresholding the displacement vectors of trajectories in the warped flow field. Trajectories with displacement deemed too small are considered similar to camera motion and are thus discarded. Figure 10 provides examples of removed background trajectories. Our method is effective across various camera motions (e.g., pan, tilt, zoom), retaining only trajectories related to human actions (highlighted in green), which yields results similar to sampling features based on visual saliency maps [15].

Figure 10. Examples of removed background trajectories.

3.4.2 Eliminating Inconsistent Matches Arising from Human Presence

In datasets focusing on action, videos predominantly capture human subjects engaging in various activities, often resulting in scenarios where humans occupy a significant portion of the frame. This poses a challenge for accurately estimating camera motion since human movements frequently diverge from the overall motion pattern. To address this challenge, we propose the utilization of a human detection system to discard feature matches occurring within regions occupied by humans. Detecting humans in action datasets presents difficulties due to the dynamic nature of poses during action sequences and the possibility of obstruction or partial visibility.

We employ a cutting-edge human detection model using Tensorflow Object Detection API. Tensorflow Object Detection API is an open-source library made based on Tensorflow for supporting training and evaluation of Object Detection models.The model we considered here is faster_rcnn_inception_v2_coco model from Tensorflow Detection Model Zoo. Training data for this model is sourced from the COCO dataset for humans and near-frontal upper body images from a specified reference. Examples illustrating the results of human detection are showcased in Figure 11.

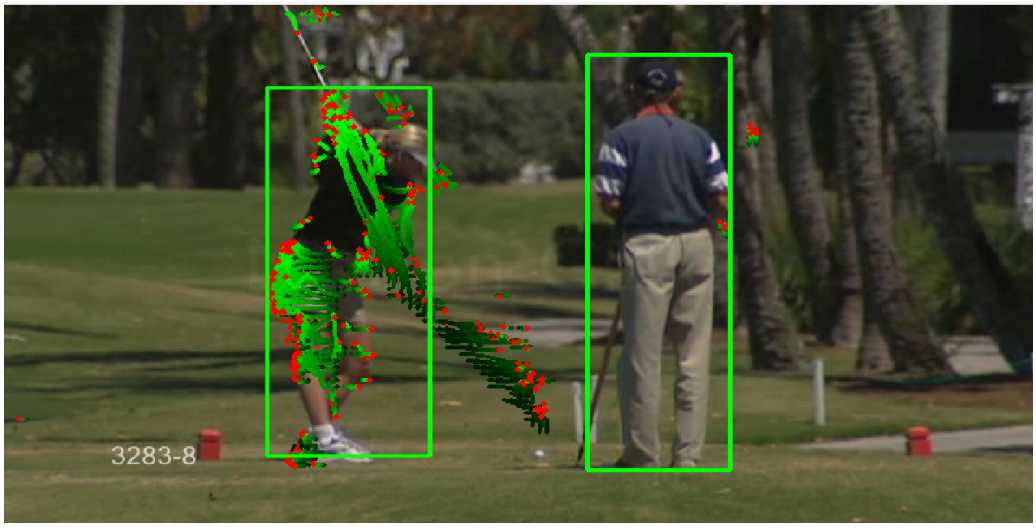

Figure 11. Examples of human detection results.

By utilizing the human detector as a mask during homography estimation, we can effectively

exclude feature matches occurring within human bounding boxes. In the absence of human detection, numerous features associated with moving humans contribute to erroneous inlier matches, leading to inaccuracies in homography estimation and subsequently warped optical flow. Conversely, filtering out matches unrelated to camera motion by employing human bounding boxes enables successful compensation for camera motion.

3.5 The UCF Sports Dataset

The UCF Sports dataset comprises a series of actions sourced from various sports frequently televised on channels like BBC and ESPN. These sequences were gathered from diverse stock footage platforms such as BBC Motion Gallery and GettyImages, totaling 150 clips at a resolution of 720 x 480. It serves as a rich source of actions depicted in varied scenes and perspectives, aimed at fostering research in unconstrained action recognition. Since its inception, researchers have utilized the dataset for tasks including action recognition, localization, and saliency detection, indicating its versatility and utility in advancing related fields.

Table 1
Summary of UCF Sports Data

Property	**Value**
Actions	10
Clips	150
Mean clip length	6.39s
Min clip length	2.20s
Max clip length	14.40s
Total duration	958s
Frame rate	10 fps
Resolution	720×480
Min num. of clips per class	6
Max num. of clips per class	22

3.5.1 Statistic of UCF Sports Dataset

The following Figure 12 shows the distribution of the number of clips per action as the number of clips in each class is not the same.

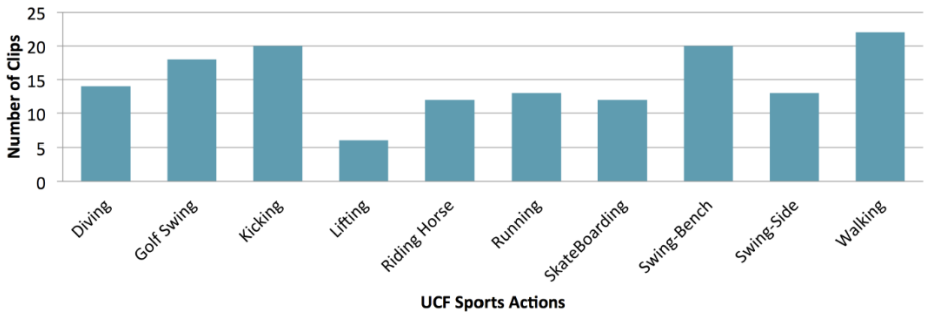

Figure 12. Number of clips per action class.

The Figure 13 shows two things for different types of actions: the total time of all clips (in blue) and the average length of each clip (in green). Some actions, like kicking, are usually short, while others like walking or running are longer and happen more regularly. Even though the actions vary in length, they often have similar average clip durations. So, just looking at how long one clip is wouldn't tell us exactly what action is happening.

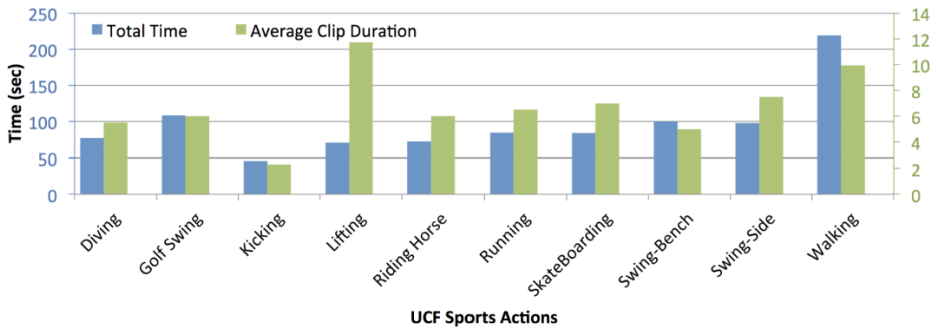

Figure 13. The total time of video clips for each action class is shown in blue. Average length of clips for each action is shown in green.

3.5.2 Statistic of UCF Sports Dataset after Preprocessing

In the process of computing our results, videos with fewer than 15 frames were excluded to ensure accurate feature descriptor calculation. As the minimum number of frames required for computing the descriptor is 15, because we are tracking a single keypoint for next 15 frames for descriptor computation. Hence videos having less than 15 frames are dropped. Initially, the UCF Sports Action Dataset contained 150 videos. However, after preprocessing, 133 videos remained. These were then divided into a training set consisting of 106 videos and a testing set containing 27 videos.

The following figures shows the distribution of the number of clips per action as the number of clips in each class after preprocessing for training and testing split.See Figure [14].

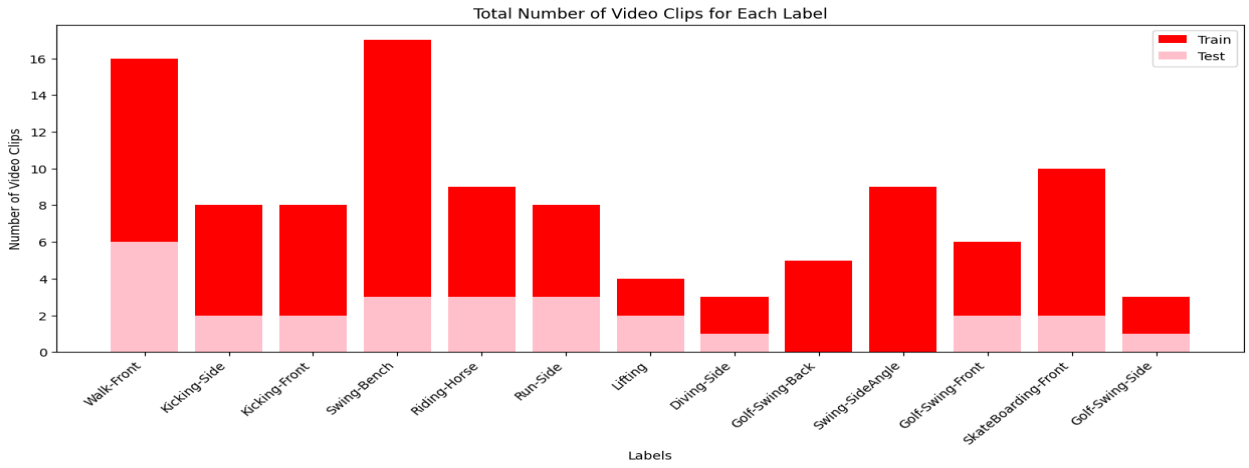

Figure 14. Total number of video clips for each label.

The following figures shows the duration of clips per action as the Duration of clips in each class after preprocessing for training and testing split.See Figure 15 .

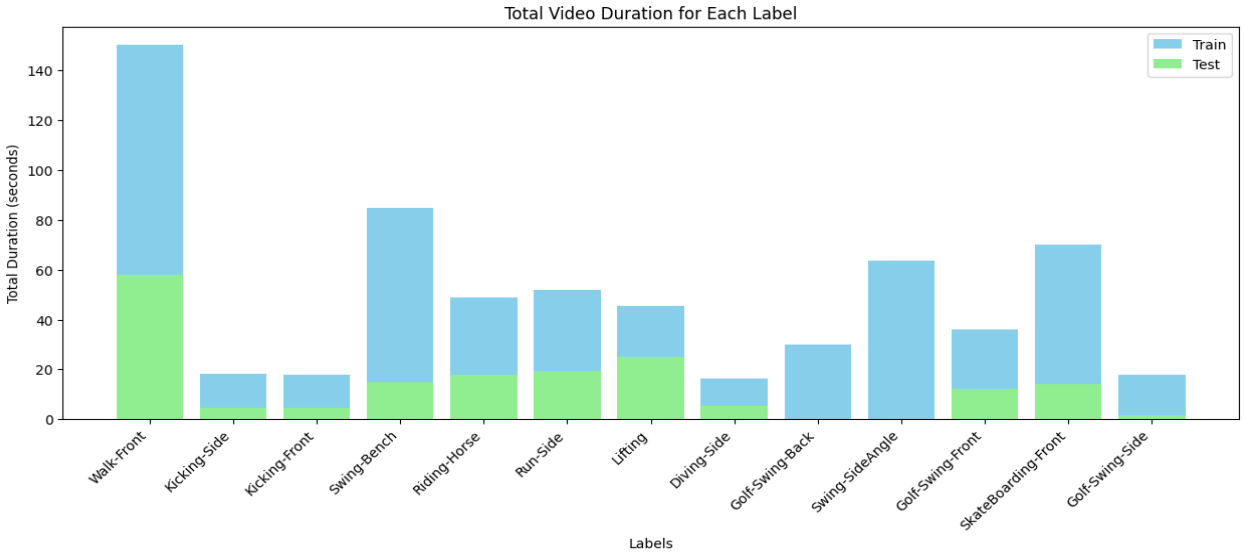

Figure 15. Combined duration for train and test.

4 Results and Discussions

Our experiments on the UCF Sports Action dataset revealed varying performance across methodologies. While temporal templates provided basic insights, they struggled with complex motion dynamics. Spatio-temporal interest points showed moderate accuracy but faced scalability issues. Dense trajectories excelled, capturing fine-grained motion cues effectively. However, improved dense trajectories surpassed them all by integrating additional features, highlighting the importance of enriching trajectory-based methods. Our analysis emphasized the necessity of incorporating both spatial and temporal information for accurate action recognition.

4.1 Temporal Template

In this report, we conducted an evaluation of various machine learning models to determine their performance on a given dataset. Our objective was to identify the most effective model for the task at hand based on its accuracy and suitability.

First, we explored the K-Nearest Neighbors (KNN) algorithm, a straightforward classification method that assigns labels to data points based on the majority class among their k nearest neighbors. Through a grid search process using cross-validation, we determined that the optimal value of k was found to be 5. Subsequently, the KNN model achieved an accuracy of 47.37% on the testing set.In Figure [16] confusion matrix for knn is given.

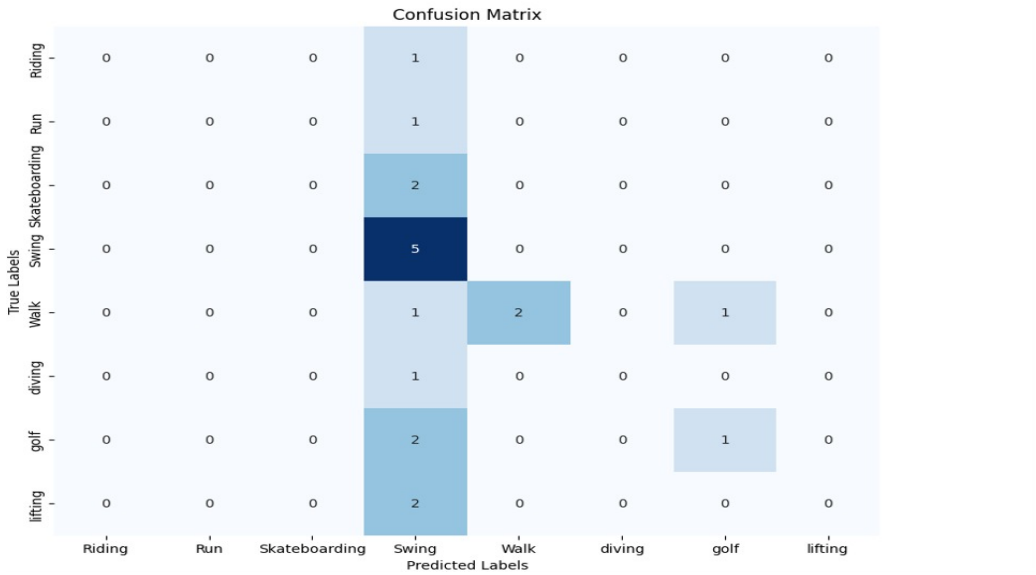

Figure 16. KNN

According to confusion matrix given in Figure [16] The K-Nearest Neighbors (KNN) model achieved moderate performance for action recognition. While skateboarding, running, and walking were recognized with decent accuracy, the model struggled to distinguish between riding, diving, and golf. Consequently, during the nearest neighbor search for these less represented actions, the model is more likely to encounter and misclassify them as swing due to the predominance of swing examples in the dataset. This imbalance underscores the importance of balanced training data to ensure equitable representation of all target classes and enhance the model's ability to generalize across diverse action categories.

Additionally, we examined the Cosine Similarity Classifier, which calculates the cosine similarity between test instances and all training instances. This classifier assigns the label of the most similar training instance to the test instance. Our evaluation revealed an accuracy of 42.11% for this classifier on the testing set.In Figure [17] confusion matrix for cosine similarity is given.

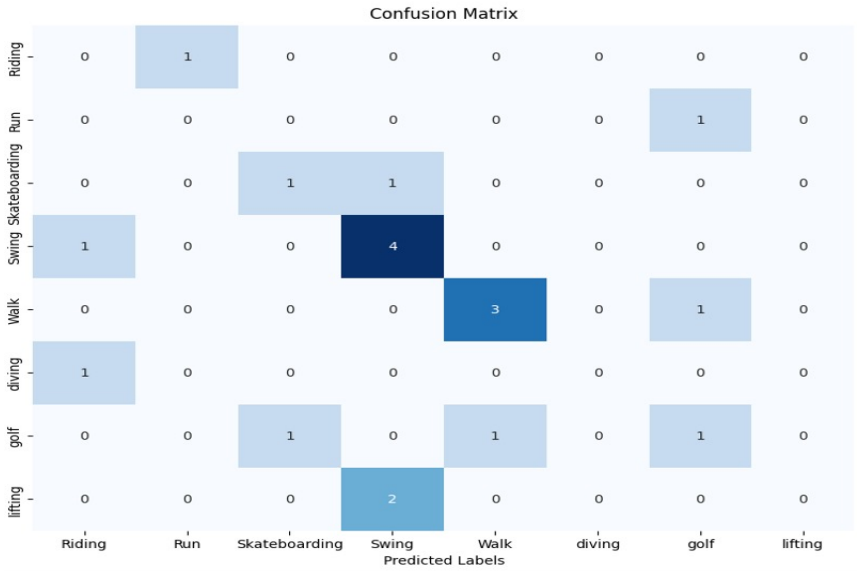

Figure 17. Cosine Similarity

The action recognition model using cosine similarity achieved mixed performance. While actions like skateboarding, walking, and diving have decent recognition rates (high values on the diagonal), riding and golf are frequently confused, as indicated by the significant off-diagonal values in those rows.

Furthermore, we explored the Support Vector Machine (SVM) algorithm, a powerful classification technique that identifies the optimal hyperplane to separate data points into different classes. Employing the Radial Basis Function (RBF) kernel, the SVM classifier achieved an accuracy of 45.11% on the testing set.In Figure [18] confusion matrix for SVM is given.

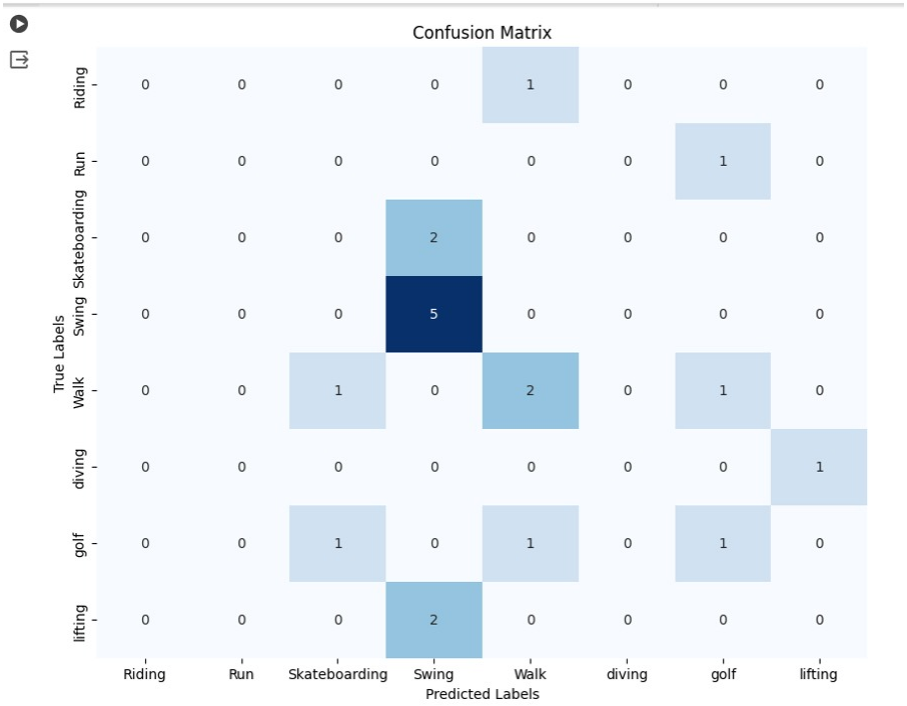

Figure 18. SVM

In conclusion, our evaluation indicates that the KNN algorithm with k=5 stands out as the most accurate model for our dataset, achieving an accuracy of 47.37% on the testing set. However, it's important to consider various factors such as dataset size, feature complexity, and computational resources when selecting the most suitable model. Future research may involve exploring additional machine learning models and optimization techniques to further enhance classification performance.

Comparative analysis with existing approaches underscored the advantages of our methodology. Unlike traditional model-driven approaches, which often face challenges in complex environments and computational complexity, our method prioritizes action labeling and temporal dynamics, offering a practical and efficient solution for real-world applications. Visual inspection of the MEI and MHI representations further highlighted the temporal evolution of movement across different sports activities, emphasizing the simplicity and interpretability of these temporal templates for human movement analysis.

4.2 Spatio Temporal Features

The proposed action recognition method, utilizing spatio-temporal features derived from interest points, has undergone thorough evaluation on the UCF Sports Action dataset. The results obtained demonstrate the effectiveness of the approach, particularly highlighting the benefits of combining HOG and HOF descriptors for capturing both appearance and motion information. This fusion of features leads to improved classification accuracy compared to using either descriptor individually, showcasing the complementary nature of appearance and motion cues in representing actions.

The utilization of spatio-temporal interest points plays a crucial role in focusing the analysis on informative regions within the video sequences. By identifying points with significant variations in both space and time, the method effectively pinpoints areas of dynamic activity, which are more likely to contain discriminative information about the actions being performed. This focused approach not only enhances the overall recognition accuracy but also contributes to computational efficiency by reducing the amount of data that needs to be processed.

Furthermore, exploring the potential of deep learning architectures offers exciting possibilities for future research. Recurrent neural networks, particularly LSTMs, could be employed to learn temporal dependencies directly from the sequence of extracted features, capturing the dynamics of actions more effectively. Alternatively, two-stream networks could be implemented to process appearance and motion information in parallel, leveraging the strengths of both HOG and HOF descriptors while learning higher-level representations from the data.

Figure 19. Confusion matrix for the classification results.

While the model demonstrates strong performance in recognizing several action classes, there's room for improvement, particularly for actions with similar motion patterns or appearances. Investigating the misclassified cases and exploring techniques to better capture subtle differences between actions would be beneficial. The confusion matrix as shown in Figure [19] reveals strong performance in recognizing actions like "Walk-Front," "Golf-Swing," and various kicking and diving actions, indicating the model's effectiveness in capturing distinct motion patterns. However, there's room for improvement in classifying actions with similar appearances or movements, such as "Riding-Horse," "Run-Side," and "Skateboarding-Front."

4.3 Dense Trajectory

The UCF Sports dataset serves as a good benchmark to evaluate the performance of the dense trajectory features on realistic action videos captured in uncontrolled environments.

Based on the results on the UCF sports dataset, the proposed method of dense trajectories

with a combined descriptor (Trajectory+HOG+HOF+MBH) achieves an average accuracy of **57%**. This demonstrates the effectiveness of the dense trajectory approach and the motion boundary histogram (MBH) descriptor in capturing the relevant motion information for action recognition, even in complex and realistic sports video datasets.

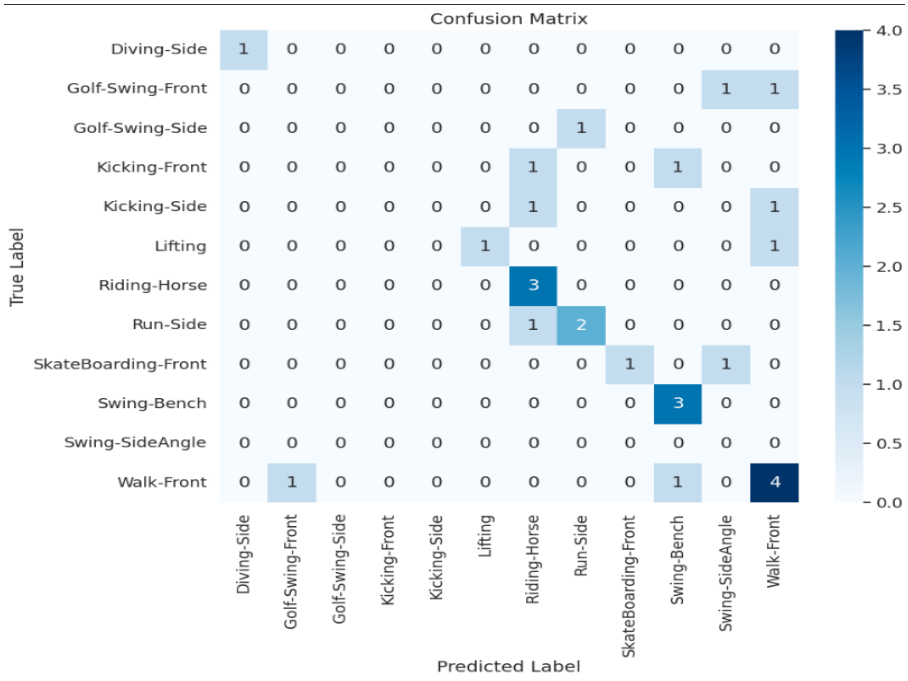

Figure 20. Confusion matrix

Figure [20] shows the confusion matrix, one interesting observation is that the model seems to struggle with correctly classifying certain activities, particularly those involving side views or angled perspectives. For example, the model confuses "Run-Side" with "RidingHorse," and "Kicking-Side" and "Kicking-Front" with "RidingHorse". This suggests that the model may have difficulty distinguishing activities when viewed from the side or at an angle, potentially due to occlusion or lack of distinguishing features from those perspectives. Additionally, there appears to be some confusion between similar activities, such as "Golf-Swing-Front" and "Golf-Swing-Side," as well as "SkateBoarding-Front" and "Walk-Front." This could indicate that the model may benefit from additional training data or techniques to better differentiate between visually similar actions. It's also worth noting that the diagonal elements, which represent correct classifications, are generally lower than desired, indicating room for improvement in the

model's overall accuracy across the different activity classes.

4.4 Improved Dense Trajectory

The UCF Sports dataset is utilized as a reliable benchmark for assessing the effectiveness of dense trajectory features in analyzing action videos recorded in diverse settings.
Following the execution of the algorithm on the preprocessed data, we achieved an accuracy of **70.37%.** This formal report underscores the careful preprocessing steps executed and the division of the dataset into separate training and testing subsets. These measures were pivotal in conducting a thorough evaluation of the model's performance.

Table 2
Comparison of Method Accuracies

Method	**Accuracy (%)**
Temporal template using knn	47.84
Temporal template using cosine similarity	42.17
Temporal template using svm	45.15
Spatio Temporal features	48.4
Dense Trajectory	56.4
Improved Dense Trajectory	70.37

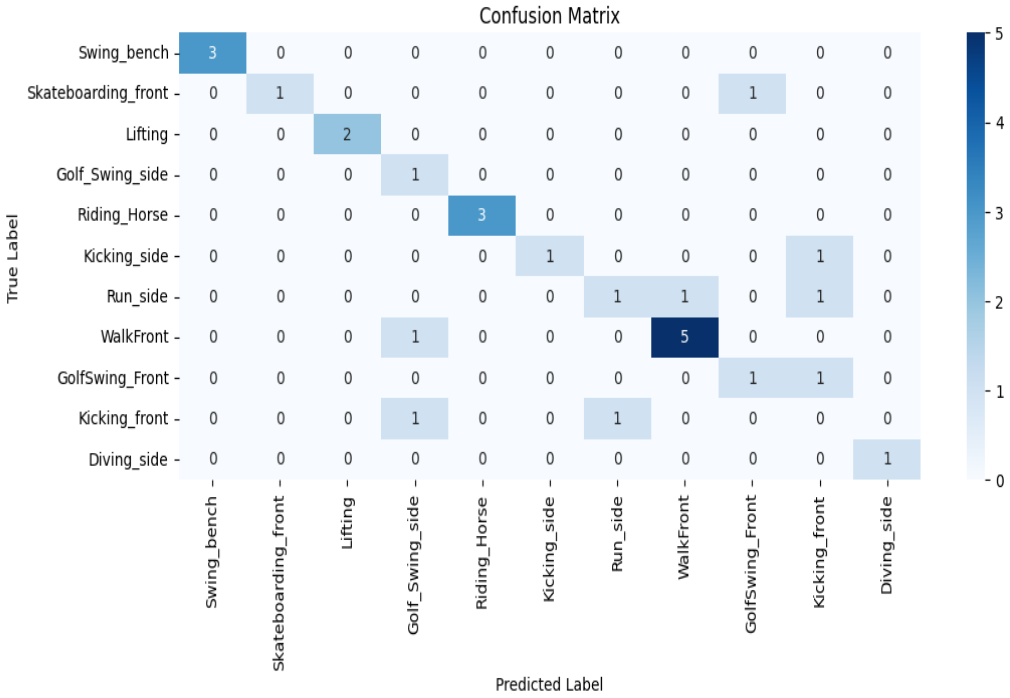

Figure 21. Confusion Matrix

Based on the test results Figure [21], it is observed that instances of kicking are misclassified as running. This misclassification could be attributed to the similarities in the movement dynamics of both actions, such as the rapid forward motion involved. Additionally, variations within kicking actions, such as kicking front and kicking side, are also misclassified due to similarities in motion and movement patterns.

To address this issue and enhance the precision of classification for such videos , it is recommended to utilize a larger dataset. By incorporating a more extensive and diverse range of examples, the model can learn to better distinguish between subtle differences in action types and variations, thereby improving its ability to accurately classify kicking actions, even in scenarios with overlapping motion characteristics.

This approach not only helps to mitigate misclassifications caused by similarities in action

movements but also enhances the robustness and generalization capability of the classification model, leading to improved overall performance in action recognition tasks.

5 Conclusion

After conducting a thorough study on feature descriptors for action recognition, we have arrived at both numerical and theoretical conclusions. The objective of our mini project was to evaluate the effectiveness of various feature descriptors and their impact on action recognition accuracy.

In terms of numerical conclusions, our analysis demonstrates that the dense trajectory method, along with its variations, exhibited the highest performance in action recognition. The incorporation of camera motion estimation and the elimination of inconsistent matches through the improved dense trajectory approach significantly enhanced the accuracy of action recognition. These improvements resulted in an average recognition accuracy of 70% on the UCF Sports Dataset after preprocessing.

Furthermore, our theoretical analysis indicates that the dense trajectory method excels at capturing intricate motion information and generating robust feature descriptors. The trajectory-aligned descriptors, such as HOG (Histogram of Oriented Gradients), HOF (Histogram of Optical Flow), and MBH (Motion Boundary Histogram), contribute to the discriminative power of the feature representation. Additionally, the application of feature encoding techniques, such as Gaussian Mixture Model (GMM) and Fisher Vector (FV), followed by classification with Support Vector Machine (SVM), further enhances the discriminative capability of the feature descriptors.

We also observed that the temporal template method, which utilizes the Motion Energy Image (MEI) and the Motion History Image (MHI), provides a simple yet effective approach for action recognition. The spatio-temporal and interest point detection methods, accompanied by feature extraction and classification techniques, also yield satisfactory results in action recognition tasks.

In conclusion, our mini project establishes that the dense trajectory method, particularly

the improved dense trajectory approach, offers superior performance in action recognition compared to other feature descriptors. However, it is important to note that the choice of feature descriptors may vary depending on the specific requirements of the application and the characteristics of the dataset. Our mini project contributes to the field of computer vision by providing valuable insights into feature descriptors for action recognition and can serve as a foundation for future research and development in this area.

6 References

Bibliography

[1] H. Uemura, S. Ishikawa, and K. Mikolajczyk. Feature tracking and motion compensation for action recognition. In BMVC, 2008.

[2] S. Wu, O. Oreifej, and M. Shah. Action recognition in videos acquired by a moving camera using motion decomposition of Lagrangian particle trajectories. In ICCV, 2011.

[3] D. Park, C. L. Zitnick, D. Ramanan, and P. Doll´ar. Exploring weak stabilization for motion feature extraction. In CVPR, 2013.

[4] M. Jain, H. J´egou, and P. Bouthemy. Better exploiting motion for better action recognition. In CVPR, 2013.

[5] E. Vig, M. Dorr, and D. Cox. Space-variant descriptor sampling for action recognition based on saliency and eye movements. In ECCV, 2012.

[6] Y.-G. Jiang, Q. Dai, X. Xue, W. Liu, and C.-W. Ngo. Trajectorybased modeling of human actions with motion reference points. In ECCV, 2012.

[7] R. Szeliski. Image alignment and stitching: a tutorial. Foundations and Trends in Computer Graphics and Vision, 2(1):1–104, 2006.

[8] G. Willems, T. Tuytelaars, and L. Gool. An efficient dense and scaleinvariant spatio-temporal interest point detector. In ECCV, 2008.

[9] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods.In BMVC, 2011.

[10] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with Fisher vectors on a compact feature set. In ICCV, 2013.

[11] G. Farnebäck . Two-frame motion estimation based on polynomial expansion. In SCIA, 2003.

[12] J. Shi and C. Tomasi. Good features to track. In CVPR, 1994

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.

[14] H. Wang, A. Kläser , C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. IJCV,103(1):60–79, 2013.

[15] E. Vig, M. Dorr, and D. Cox. Space-variant descriptor sampling for action recognition based on saliency and eye movements. In ECCV, 2012.

[16] Laptev and T. Lindeberg. Space-time interest points. In ICCV, 2003.

[17] M. Bregonzio, S. Gong, and T. Xiang. Recognising action as clouds of space-time interest points. In CVPR, 2009.

[18] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In VS-PETS, 2005.

[19] G. Willems, T. Tuytelaars, and L. V. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In ECCV, 2008.

[20] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In ICCV, 2009.

[21] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In International Joint Conference on Artificial Intelligence, 1981.

[22] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In ICCV workshop on Video-oriented Object and Event Classification, 2009.

[23] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In BMVC, 2009.

[24] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In CVPR, 2009.

[25] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: label transfer via dense scene alignment. In CVPR, 2009.

[26] Wang, Ullah, Klaser, Laptev, Schmid: Feature for Action Recognition

[27] https://medium.com/@madhawavidanapathirana/real-time-human-detection-in-computer-vis

[28] https://gist.github.com/madhawav/1546a4b99c8313f06c0b2d7d7b4a09e2

[29] https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_harris.html

[30] https://learnopencv.com/histogram-of-oriented-gradients/

[31] https://www.youtube.com/watch?v=gtSlrb9cvpE&t=77s

[32] https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html

7 Appendix

7.1 Temporal Template

This appendix provides instructions on how to use the machine learning project repository for feature extraction and model training.

Cloning the Repository

To get started, clone the repository to your local machine using the following command:

```
git clone https://github.com/samarth4669/miniproject.git
```

Installing Dependencies

Navigate to the project directory and install the required dependencies using pip:

```
pip install -r requirements.txt
```

Dataset Preparation

Download the dataset from the provided link (`https://www.crcv.ucf.edu/data/UCF_Sports_Action.php`) and place it in the appropriate directory within the project.

Feature Extraction

Run the `featureextraction.py` script with the dataset path as a command-line argument to extract features:

```
python featureextraction.py /path/to/your/dataset
```

This script will save the extracted features in an NPZ file named `features.npz`.

Machine Learning Model Usage

1. Import necessary libraries and load features file by inputting path to your feature file in appropriate place.Using command line run the program

```
python mlmodel.py functionname
```

   where function name is name of model you want to use.for KNN is knnmodel,for Cosine similarity it is cosinemodel, for svm it is svmmodel.

   .

File Structure

- `featureextraction.py`: Script for extracting features from datasets.
- `mlmodel.py`: Script for training machine learning models and making predictions.
- `requirements.txt`: List of Python dependencies for the project.

7.2 Spatio Temporal Features

You can find the implementation at : Github Code

System Requirements

- Python 3.6 or higher
- TensorFlow 1.5 (or above)
- OpenCV 3.0 (or above)
- Numpy

Installation Guide

To set up your environment for the project, follow these steps:

1. Install Required Libraries: Ensure Python, TensorFlow, OpenCV, and Numpy are installed on your system. You can install them using pip:

```
pip install tensorflow
pip install opencv-python
pip install numpy
```

2. Set Up Project Directory: Create a directory specifically for this project to keep all related files organized.

3. Download and Set Up TensorFlow:

   - Download `faster_rcnn_inception_v2_coco.tar.gz` from the TensorFlow Detection Model Zoo.
   - Extract the contents into your project directory.

Configuration

Before running the scripts, configure the paths in your scripts:

- Model Path: Path to the TensorFlow model, typically ending with `frozen_inference_graph.pb`.
- Video Path: Directory path where your input videos are stored.
- CSV Path: Path where descriptors will be saved in a CSV file.

Procedure

1. Prepare Input Data: Ensure that your videos are placed in the specified Video Path. Edit `descriptor_saving(final).py` with paths pointing to your model refer this Medium article, video, and output CSV paths.

2. Run the Descriptor Saving Script:

```
python3 descriptor_saving(final).py
```

This will process each video and save the extracted descriptors into a CSV file named after the input video.

3. Handling Multiple Videos: If you need to process multiple videos, you can use `multiple.py` to automate the processing of multiple files in a directory.

Training the Machine Learning Model

1. Prepare Training and Test Sets: Ensure you have `train.txt` and `test.txt` files that list the video paths and their corresponding labels for training and testing.

2. Train Your Model by running this command:

```
run training.ipynb
```

Output

The descriptor data (HOG and HOF features) for each video is saved in the designated CSV file, facilitating easy use in machine learning models for action recognition.

7.3 Dense Trajectories

Code Implementation on Github: Click here

This project presents a computer vision system for computing video descriptors employing a combination of Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), and Motion Boundary Histograms (MBH) techniques. The main innovation lies in the extraction and description of dense trajectories from videos, providing a robust representation of motion patterns compared to previous trajectory approaches. Additionally, the MBH descriptor effectively distinguishes relevant foreground motion from background camera motion in videos. The implementation is in Python 3.10.12.

Implementation of code is based on the paper **"Dense trajectories and motion boundary descriptors for action recognition"** by Wang et al (2013).

Dataset Preparation

Download the dataset from the provided link (`https://www.crcv.ucf.edu/data/UCF_Sports_Action.php`) and place it in the appropriate directory within the project.

Execution Requirements

To execute this project, follow these steps:

1. **Install Dependencies using requirement.txt file:**

```
Python (3.10.12)
OpenCV (4.9.0)
NumPy (1.26.4)
Scikit-Learn (1.4.1.post1)
Pandas (2.2.1)
Matplotlib (3.8.3)
Seaborn (0.13.2)
```

2. **Ensure Hardware Requirements:**

```
More than 4GB of RAM
At least 30GB of disk space
```

File Structure

- First you should load UCF sport dataset into your local system
- Then you should write a code to extract all the avi video file in one folder named `extracted`.

- Based on this extracted folder you should split avi files into train and test splits with labels in `train.txt` and `test.txt` files in the `data` folder.
- Folder structure must look like this:

```
data
  |--- extracted
  |       |--- first.avi
  |       |--- second.avi
  |       |--- third.avi
  |            .
  |            .
  |       |--- last.avi
  |--- train.txt
  |--- test.txt
```

Input Parameters

- **Video Path:** Path to the video file for which descriptors need to be computed.

Usage Example

To run the code, follow these steps:

1. Navigate to the `DenseTrajectory` directory using the command:

```
cd DenseTrajectory
```

2. Execute the `run.py` file using the command:

```
python3 run.py
```

3. **Note:** Before running `run.py`, ensure `already computed descriptors = False` in the main function to compute your own descriptors based on input videos. The UCF-sport dataset has been used for descriptor computation.

4. If finding difficulties in understanding the code, debug the code using `demo.ipynb` file.

Output

The system processes the video and computes descriptors, which are then saved in the `DenseTrajectory/out` directory.

Note

- The `train.txt` file should contain paths to training videos along with their corresponding labels for model training.
- The `test.txt` file should contain paths to testing videos along with their corresponding labels for model evaluation.

7.4 Improved Dense Trajectories

You can find the implementation at : Github Code

Project2: Descriptor Extraction System (Improved Dense Trajectories)

Overview:

Project2 is a computer vision system designed for computing video descriptors using a combination of Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), Motion Boundary Histograms (MBH), Human Detector, and Camera Motion Estimation techniques. It is implemented in C++ using the OpenCV libraries and built using Visual Studio. The system generates an executable file (.exe) upon successful compilation, which is executed from the command line with two input arguments.

Execution Requirements:

To execute Project2, follow these steps:

1. Build the project from the provided source code, linking the required OpenCV libraries.
2. Execute the generated .exe file from the command line.

3. Provide two arguments during execution:

    - Video Path: The path to the video for which descriptors are to be extracted.
    - Bounding Box File Path: The path to the file containing precomputed bounding boxes for each frame of the video.

Input Parameters:

- Video Path: Path to the video file for which descriptors are to be computed.
- Bounding Box File Path: Path to the file containing precomputed bounding boxes for each frame of the video. (This can be generated using a Human Detector TensorFlow model.)

Note: The bounding box file is essential for accurately computing descriptors for human-related activities.

Usage Example:

```
Project2.exe <Video_Path> <Bounding_Box_File_Path>
```

Example: `Project2.exe video.mp4 bounding_boxes.txt`

Output:

The system processes the video and computes descriptors based on the provided bound.

Human Detection and Descriptor Extraction Pipeline

Overview:

This pipeline integrates the "Project2" video descriptor extraction system with the TensorFlow Object Detection API for real-time human detection. It allows for the extraction of descriptors from videos using "Project2", followed by training a machine learning model using the extracted descriptors.

Setup Instructions

1. Build "Project2" Source Code:
    - Build the "Project2" source code to generate the executable file (Project2.exe).
    - Ensure that the necessary OpenCV libraries are linked correctly.
2. Install TensorFlow:
    - Install TensorFlow in Python. You can follow the official installation instructions provided by TensorFlow.

      ```
      pip install tensorflow
      ```

3. Install TensorFlow Object Detection API:
    - Follow the instructions in this Medium article to install the TensorFlow Object Detection API.
4. Update Paths in Code:
    - Make sure that all paths in the code are correctly set, including:
        - Path to your dataset.
        - Path to your Project2.exe file.
        - Path to your extracted descriptors, if applicable.
5. Execute run.py:
    - Open run.py and set `already_computed_descriptors=False` in the main function.
    - Execute run.py to start the descriptor extraction process.
    - Wait for the descriptors to be extracted from the videos.
6. Train ML Model:
    - Ensure that you have train.txt and test.txt files containing paths to videos and their corresponding labels for training and testing, respectively.

- Train your machine learning model using the extracted descriptors.

Note

- The train.txt file should contain paths to training videos along with their corresponding labels for model training.
- The test.txt file should contain paths to testing videos along with their corresponding labels for model evaluation.