Université de Paris
UFR Mathématiques et Informatique

Study and modeling of spatial relations between objects in image sequences or videos

Master 1 Computer Vision and Intelligent Machine

Yaghmoracen BELMIR
Supervised by: Mr. Laurent WENDLING, Mr. Camille KURTZ

Academic year 2022-2023

ABSTRACT

This project focuses on the analysis of spatial and temporal relationships among objects in videos by employing the Force Banner concept and integrating it into a learning process. Our project follows several key steps, including video reading and frame separation, object detection and segmentation, object tracking over time, computation of Force Histograms between pairs of objects (masks) that will represent a descriptor for each frame, and construction of the Force Banner using the obtained descriptors. The project's methodology will be tested and compared on various types of videos, including synthetic sequences and real-world datasets comprising action scenarios and drone footage. The progress will also involve integrating the developed approach into a learning process, potentially utilizing convolutional neural networks (CNNs) for enhanced prediction of complex spatiotemporal situations. By leveraging the proposed methodology, this project aims to contribute to the analysis of object dynamics in videos by a prediction of their spatiotemporal patterns, with potential applications in domains such as action recognition and video understanding.

Keywords: Computer Vision. Spatial relationships. Temporal relationships. Object analysis. Video processing. Object detection. Object segmentation. Object tracking. Force Histogram. Force Banner. Spatiotemporal patterns. Deep learning. Convolutional Neural Networks (CNNs). Action recognition. Video understanding. Machine learning. Image analysis. Real-world datasets.

Contents

List of Figures

List of Tables

General Introduction

In this project, we aimed to address the problem of analyzing the spatial configuration and temporal evolution of objects in videos. The objective was to develop a solution that could accurately track and analyze the relationships between objects over time, specifically focusing on the perception of human actions.

The existing state of the art in computer vision and object tracking provided valuable insights, but there were specific challenges we aimed to overcome. These challenges included the need for robust object detection, reliable object tracking, and the development of effective spatial-temporal representations.

To tackle these challenges, our solution involved several steps. First, we performed frame extraction from the video, creating a sequence of individual frames. Next, we applied object detection using the YOLOv8 model to identify and localize the desired objects, specifically focusing on detecting persons in our case. We then implemented object tracking using the SORT algorithm to maintain consistent tracks of the selected objects across frames.

To further enhance our analysis, we leveraged the concept of force histograms, which provided a comprehensive representation of spatial relationships between object pairs. This involved computing the force histograms for each pair of objects in the frames and constructing a "banner" matrix to capture the temporal evolution of these spatial relationships.
Additionally, we incorporated deep learning techniques, training a Convolutional Neural Network (CNN) on the generated banner data to learn and predict complex spatial-temporal situations, such as human actions.

Through these steps, we aimed to develop a robust and effective solution for analyzing the spatial configuration and temporal evolution of objects in videos, with the ultimate goal of improving the understanding and prediction of complex spatiotemporal scenarios.

Chapter 1

Definitions and Related Works

Introduction

In this first chapter, we will start with brief definitions of the main keywords relevant to our work, then we will discuss some of the existing solutions that are related to this project.

1.1 Definitions

1.1.1 Object Detection

It is the task of identifying and localizing specific objects within images or videos. It involves detecting the presence of objects and accurately outlining their bounding boxes. This process employs machine learning techniques, such as deep learning and convolutional neural networks, to learn and classify objects into predefined categories.

1.1.2 Object Segmentation

It is the process of identifying and isolating individual objects or regions of interest within an image or video. It aims to separate objects from the background or other objects present in the scene.

1.1.3 Object Tracking

It consists of following and estimating the positions and trajectories of objects over time in a video sequence. It involves associating object instances across consecutive frames while handling challenges such as occlusions, appearance changes, and motion variations.

1.1.4 Force Histogram

Force Histogram is a representation that captures the spatial relationships between pairs of objects by quantifying the forces acting between them. The forces can denote attraction or repulsion,

depending on the nature of the interaction. The histogram summarizes the distribution of these forces, providing a compact and informative descriptor of object interactions.

1.1.5 Convolutional Neural Networks (CNNs)

CNNs are deep learning models specifically designed to process visual data. CNNs consist of multiple layers, including convolutional layers that extract hierarchical features from input images or video frames. They have demonstrated remarkable performance in various computer vision tasks, including object recognition, detection, and segmentation.

1.1.6 Spatiotemporal Patterns

Spatiotemporal patterns are patterns that involve both spatial and temporal dimensions. In the context of video analysis, spatiotemporal patterns refer to the relationships and changes in object positions, shapes, and interactions over time. Recognizing and understanding these patterns is crucial for tasks such as action recognition and event detection.

1.2 Related Works

1.2.1 Image Segmentation and Object Detection

We will briefly present some of the latest and mostly referenced works in Image Segmentation and Object Detection.

Object Detection

Multiple solutions were developed to answer the problem of object detection in images. Whether the method is based on algorithms and extraction of image features, or deep learning models, this task remains a challenge in the visual computing community. Here are some models that achieved good results on benchmarks:

YOLO (You Only Look Once) YOLO is an object detection algorithm known for its real-time processing speed and simplicity. It divides the input image into a grid and predicts bounding boxes and class probabilities for objects within each grid cell in a single pass. YOLO achieves high detection accuracy and fast inference times, making it suitable for real-time applications [1].

Fast R-CNN Fast R-CNN is a two-stage object detection framework that introduced the concept of region proposal networks (RPN). It generates potential object proposals using an RPN and then classifies those proposals using a convolutional neural network (CNN). Fast R-CNN achieves accurate object detection but is computationally more expensive compared to single-shot detectors like YOLO [2].

RetinaNet RetinaNet is a single-stage object detection model designed to address the problem of class imbalance in object detection. It introduces a focal loss function that assigns higher weights to hard examples during training. This helps to mitigate the dominance of the background class and improves the accuracy of object detection at various scales. RetinaNet achieves a good balance between accuracy and efficiency [3].

Image segmentation

As for image segmentation, solutions mostly rely on deep learning models:

YOLO (You Only Look Once) Although primarily known for object detection, YOLO can also be adapted for image segmentation tasks. YOLO-based segmentation models leverage the architecture and principles of YOLO to perform dense pixel-wise prediction. By modifying the final layer of YOLO, it becomes capable of producing segmentation maps instead of bounding boxes. This approach combines the benefits of YOLO, such as real-time processing and single-pass inference, with the ability to segment objects in an image [1].

U-Net U-Net is a popular architecture for image segmentation that follows an encoder-decoder structure. The encoder extracts feature maps at different levels of abstraction, while the decoder upsamples and combines these features to produce segmentation maps. U-Net incorporates skip connections between corresponding encoder and decoder layers to preserve fine-grained details. It has been widely used in medical imaging and other segmentation tasks where precise boundary delineation is crucial [4].

Mask R-CNN Mask R-CNN is a two-stage framework that extends the Faster R-CNN object detection approach to include pixel-level segmentation. It generates object proposals using an RPN and then performs bounding box refinement and instance-level mask prediction. Mask R-CNN introduces an additional mask branch in the network, enabling precise segmentation of objects. It achieves accurate object detection and high-quality segmentation simultaneously, making it a popular choice for various applications [5].

1.2.2 Object Tracking

SORT (Simple Online and Real-time Tracking) SORT is a widely adopted algorithm for object tracking. It is designed to perform online and real-time tracking of multiple objects in video sequences. SORT utilizes the Kalman filter framework to predict the object's state and estimate its future position. It associates detections with existing tracks based on a distance metric, such as the intersection over union (IoU) or the Mahalanobis distance. SORT excels in handling occlusions and maintaining track identity in complex scenarios, making it suitable for real-world tracking applications [6].

Online Multi-Object Tracking with TrackletNet TrackletNet is an online multi-object tracking algorithm that combines deep learning and graph optimization techniques. It leverages a deep neural network to generate high-quality tracklets, and short track segments and employs a graph optimization algorithm to link tracklets into complete object tracks. TrackletNet is capable of handling occlusions, appearance changes, and track switches, making it effective in complex tracking scenarios [7].

Multiple Hypothesis Tracking (MHT) Multiple Hypothesis Tracking is a framework for tracking multiple objects in cluttered environments. MHT maintains multiple hypotheses about object tracks and dynamically updates them based on new observations. By considering multiple possible associations of measurements to tracks, MHT can handle occlusions, missed detections, and false alarms effectively, resulting in robust tracking performance [8].

Force Banner Force Banner is a visual representation constructed by assembling the Force Histograms computed for different pairs of objects across multiple frames in a video. The Force Banner provides a concise and informative summary of the spatial relationships and dynamics among objects throughout the video as cited in the article by R. Deléarde, C. Kurtz, and L. Wendling titled "Description and recognition of complex spatial configurations of object pairs with Force Banner 2D features" which presents the Force Banner 2D feature, an extension of the force histogram descriptor. This feature successfully describes and recognizes complex spatial configurations between object pairs[9]. Its usage builds upon the Force descriptor introduced in the article by P. Matsakis and L. Wendling, which represents the relative position between areal objects by computing their interactions in all directions. Both articles provide valuable insights into spatial configuration analysis, offering innovative approaches for representing and recognizing intricate spatial relationships between objects. The Force Banner 2D feature extends the capabilities of the Force descriptor, demonstrating its effectiveness in capturing and understanding complex spatial arrangements [10].

Conclusion

After providing the above definitions, and giving a summary of some of the state-of-the-art methods that are related to our project, we will proceed to the next chapter where the problem and the proposed solution will be described.

Chapter 2

Proposed Solution

Introduction

In this chapter, we will provide a detailed explanation of the steps involved in constructing the proposed approach. However, before delving into those specifics, we will first describe the problem in the following section.

2.1 Problem

The Force Banner approach was used to extend the concept of force histogram to a range of different and complementary forces, capturing various types of interactions between objects such as attraction and repulsion within a single representation. Its effectiveness has been demonstrated in the analysis and interpretation of visual scenes [9, 10].

In this project, the objective is to transform that concept to extend it to the analysis of the evolution of spatial configuration between objects over time, for example in a video.

It will be subsequently integrated into pattern recognition processes to assess its ability to predict complex spatiotemporal situations, such as actions (using neural networks, for example, through learning).

2.2 Proposed Method

To accomplish this project, we considered several steps and aspects:

1. Data Preprocessing
2. Object Tracking
3. Force Histogram
4. Convolutional Neural Network Model

In this section, we will detail the achieved steps.

2.2.1 Data Preprocessing

This step consists of frame extraction and object selection:

- **Frame Extraction:** During the initial stage of our project, frame extraction was conducted on the input video, resulting in the extraction of individual frames. These frames were then stored in the designated "frames" folder for further processing. To ensure proper organization and association, the generated masks for each frame were stored together in the corresponding folder within the "masks" directory as shown in figure 2.1.

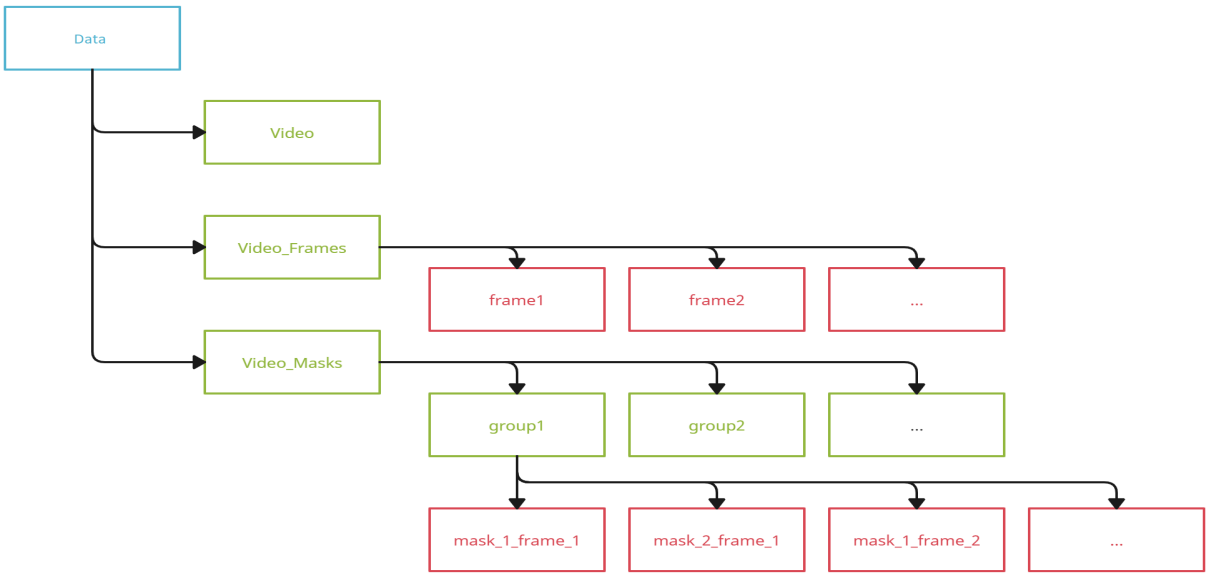

Figure 2.1 – Repertory distribution

- **Object Selection:** In our project, we implemented a program that reads the frames of a video and provides the functionality to pause on a desired frame. Upon pausing, the user has the option to select a specific number of objects from the paused frame to track them throughout the video. The selection can be set to either 2, 3, or a maximum of 4 objects. In our case, we focused on selecting 2 objects.

  To detect the objects, we utilized the YOLOv8 model, which was specifically set to identify persons only. The model was employed to detect every person present in the paused frame, the figure 2.2 shows an example.

  Using the mouse, the user then interactively selects the desired objects to track. Only objects that were recognized as persons by the YOLOv8 model are selectable. Objects that were not detected as persons cannot be selected for tracking in that particular frame.

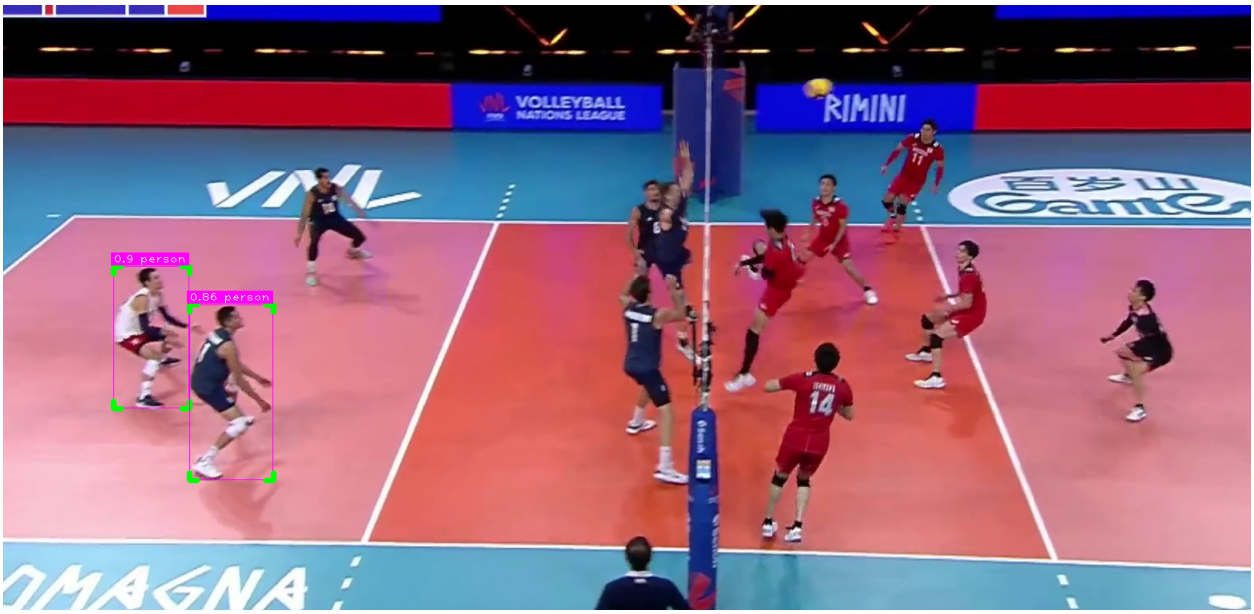

Figure 2.2 – Selection phase

2.2.2 Objects Tracking

After selecting the desired persons to track, the object tracking phase begins. The SORT algorithm is updated to generate unique IDs for the selected persons based on their bounding boxes. By associating the IDs as shown in figure 2.3, it becomes easier to locate and track the same objects across subsequent frames.

In each frame, YOLOv8 is utilized to detect all detectable persons. The SORT algorithm updates the tracking by incorporating the new bounding box information. It assigns new IDs while preserving the IDs of previously tracked objects.

To ensure accuracy, a threshold is applied to limit the maximum allowed distance between the SORT-generated bounding box and the YOLOv8 detected object's bounding box. If the sum of coordinate differences between the bounding boxes is below the threshold, the tracked object's ID is then checked. If it matches one of the objects of interest, the corresponding mask is selected as shown in figure 2.4.

The SORT algorithm is set with a maximum age of 200 frames, meaning if an object is not tracked for more than 200 frames, its ID is freed up for a new object. However, a default black image is used to represent objects that are not detected to ensure a mask is still generated.

Figure 2.3 – ID generated by SORT

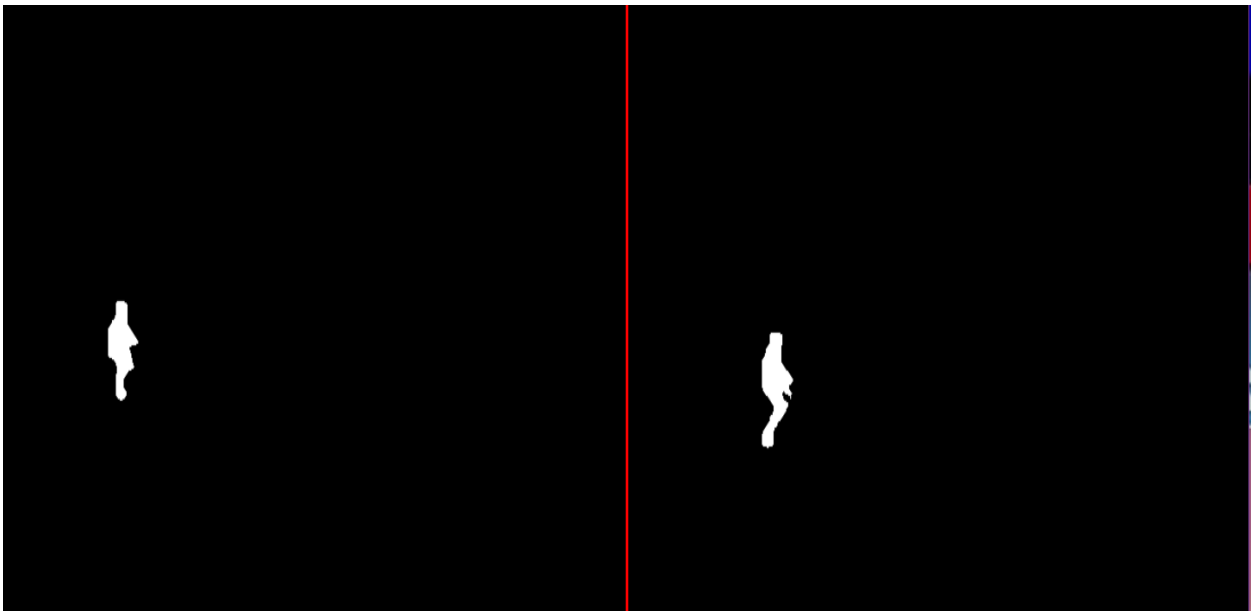

Figure 2.4 – Corresponding masks

2.2.3 Force Histogram

Once all pairs of masks are generated from the video, the next step involves computing the Force Histogram algorithm for each pair. The Force Histogram generates a descriptor, which is an array

composed of 180 elements (the number of elements can be adjusted based on the parameters). This descriptor captures important spatial relationship information between the objects in the pair.

These descriptors from subsequent frames are then stacked together into a single matrix called the “banner”. The banner matrix represents a comprehensive spatial configuration across multiple frames. To preserve and utilize this information in the next step, the banner matrix is stored as an image. This image serves as valuable data for the training phase of our model, enabling the model to learn and analyze complex spatiotemporal patterns.

2.2.4 Convolutional Neural Network Model

For the training phase, it is important to note that the size of the banners varies depending on the duration of the videos. To ensure consistency, a reshaping process is applied to all banners, making them have the same shape.

The data is divided into three classes: Class 1 represents pairs of players from Team 1, Class 2 represents pairs of players from Team 2, and Class 3 represents pairs of rivals, where one person is from Team 1 and the other is from Team 2. Each class is homogeneously separated into three sets: 80% for training, 10% for validation, and the last 10% for testing. This split is consistently applied to each class.

Consequently, the corresponding sets are grouped together, such as training sets with training sets and test sets with test sets. This organized data is then used to train a Convolutional Neural Network (CNN) in a supervised manner, as the labels are provided for each data sample.

The chosen architecture consisted of multiple convolutional layers, max pooling layers for down-sampling, and fully connected layers for classification. The input image was fed into the network, followed by two sets of convolutional layers with increasing filter sizes, and max pooling layers to reduce spatial dimensions. This was followed by additional convolutional layers and another max pooling layer. The feature maps were then flattened and passed through a fully connected layer with ReLU activation. Finally, the output layer with a softmax activation function provided the classification predictions, as shown below (figure 2.5). The architecture’s design allowed for the extraction of relevant features at different levels of abstraction, enabling effective classification of the input data.

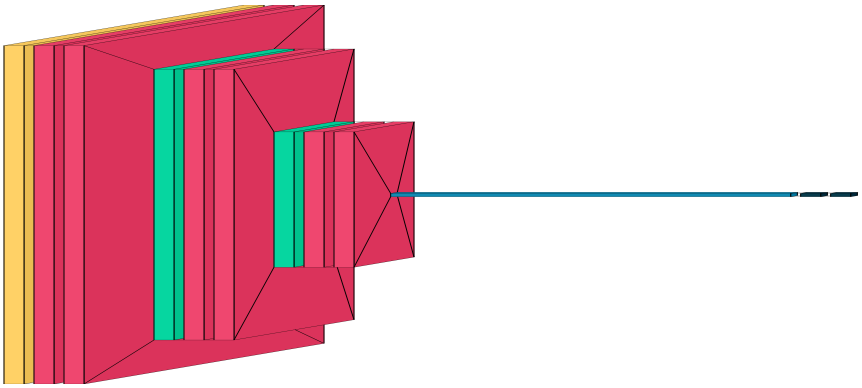

Figure 2.5 – CNN architecture

To summarize, the following algorithm outlines the sequence of steps involved in the processing pipeline:

Algorithm 1: Processing Pipeline

```
1:  procedure MAIN(video)
2:      main(video)
3:      object_detection_to_masks(frames)
4:      banner(masks)
5:      CNN(banners)
6:      output: prediction of complex spatial thing
7:  end procedure
8:  procedure MAIN(video)
9:      /* Create frames from the video */
10: end procedure
11: procedure OBJECT_DETECTION_TO_MASKS(frames)
12:     /* Read the frames */
13:     selectedObjects ← trackObjects(frames)
14:     masks ← createMasks(selectedObjects)
15:     storeMasks(masks)
16: end procedure
17: procedure BANNER(masks)
18:     /* Read the masks */
19:     fHistograms ← computeFHistogram(masks)
20:     banners ← generateBanner(fHistograms)
21:     storeBanners(banners)
22: end procedure
23: procedure CNN(banners)
24:     /* Read the banners */
25:     model ← trainModel(banners)
26:     output ← predictComplexSpatialThing(model)
27: end procedure
```

Conclusion

Throughout this chapter, we have outlined the key steps involved in building our approach, from data acquisition to feature extraction and temporal representation. Additionally, the training process of our deep neural network model has been discussed.

The next chapter will showcase the experimental study and the results obtained from it. The methodology, dataset, evaluation metrics, and experimental setup will be explained, followed by a detailed analysis and discussion of the achieved results.

Chapter 3

Experiments and Results

Introduction

This chapter focuses on the experimental study conducted to evaluate the proposed approach. It outlines the methodology, dataset, and experimental setup. The chapter presents the obtained results, providing insights into the effectiveness and performance of the approach. The analysis of these results contributes to a better understanding of the approach's capabilities and potential applications.

3.1 Working Environment

One limitation we encountered with the SORT tracking method is that YOLOv8 may occasionally fail to detect a tracked person in certain frames, resulting in SORT losing track of that object. However, we observed that the SORT algorithm is robust enough to rediscover the object once it is detected again by YOLOv8 in subsequent frames.

We also faced two additional constraints in our working environment. Firstly, the provided code was designed to work only on Linux distributions. To overcome this limitation, we set up a virtual machine and imported our data through Google Drive. This allowed us to compute the force histogram and perform other necessary computations.

The second constraint was the limited availability of GPU resources. We utilized YOLOv8 with extra-large weights to enhance the object detection performance, but due to the lack of GPU resources, we had to rely on our CPU for SORT computations. Our system specifications include an Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz, 2.59 GHz, with 8GB RAM. On average, it took approximately 4 seconds to process one frame of video (around 10 minutes for 250 frames) under Windows distribution.

However, during the training of the CNN model, we took advantage of the computing resources provided by Google Colab, which includes access to a GPU (Graphics Processing Unit) runtime. GPUs are well-suited for accelerating deep learning tasks, such as training neural networks, due to their parallel processing capabilities. By leveraging the GPU runtime, we were able to significantly reduce the training time and efficiently train our CNN model.

3.2 Data Collection

The dataset used in this project consists of a video from YouTube, specifically the 2021 FIVB Volleyball Men's Nations League match between the Japanese and USA teams. The video has a duration of 10 minutes and captures the gameplay between these two teams [11].

To create the dataset, the video was downloaded and processed to extract relevant sections where the camera angle provided a clear view of both teams. These selected segments ensured that both teams were prominently visible in the small-cut videos.

Subsequently, for the CNN data, each cut sequence from the original video was analyzed to identify and select all possible detectable pairs for each class.

3.3 Results

Data generation for the CNN

For our experiments, we worked with a raw video that was divided into four trimmed sections. Each trim represented a specific sequence of the video. For each trim, we generated two associated folders. The first folder contained the extracted frames from the trim, while the second folder consisted of three classes of subfolders as shown in the table below 3.1.

Within each class, there were multiple variations of subfolders, with an average of ten subfolders in each variation. These subfolders contained pairs of the selected masks. On average, each subfolder of masks contained around 800 pictures, one example about the 1^{st} used trim is shown in the table 3.2.

Video	Trim video name	Folder name	Class folder name
Raw video	Trim_1	Frames	-
		Masks	A
			B
			C
	...	...	...

Table 3.1 – Folder splitting for the video

Trim video name	Class folder name	Nb. of associated folders	Nb. of images per class
Trim_1	A	10	7448
	B	12	9034
	C	8	5992

Table 3.2 – Class folders and the number of masks generated for each.

It is worth noting that there were additional possible trims available in the original video. However, for our experiments, considering the significant amount of time required for mask generation, we limited the analysis to a selected set of trims (4 trims).

Having three classes in our dataset corresponds to three distinct labels for our supervised learning model. The labels are as follows: Class A represents players playing for Team A, Class B represents players playing for Team B, and Class C represents players who are rivals. Accordingly, we created three separate folders (Class A, Class B, and Class C) dedicated to storing the resulting force histogram images. After generating all the required masks from each trim, we proceeded to process the folders trim by trim. For each trim folder, we iterated through its subfolders while checking their corresponding classes. If a subfolder belonged to Class A, we stored the generated force histogram inside the Class A folder. The same process was followed for subfolders belonging to Class B and Class C, ensuring that the force histogram images were stored in their respective folders.

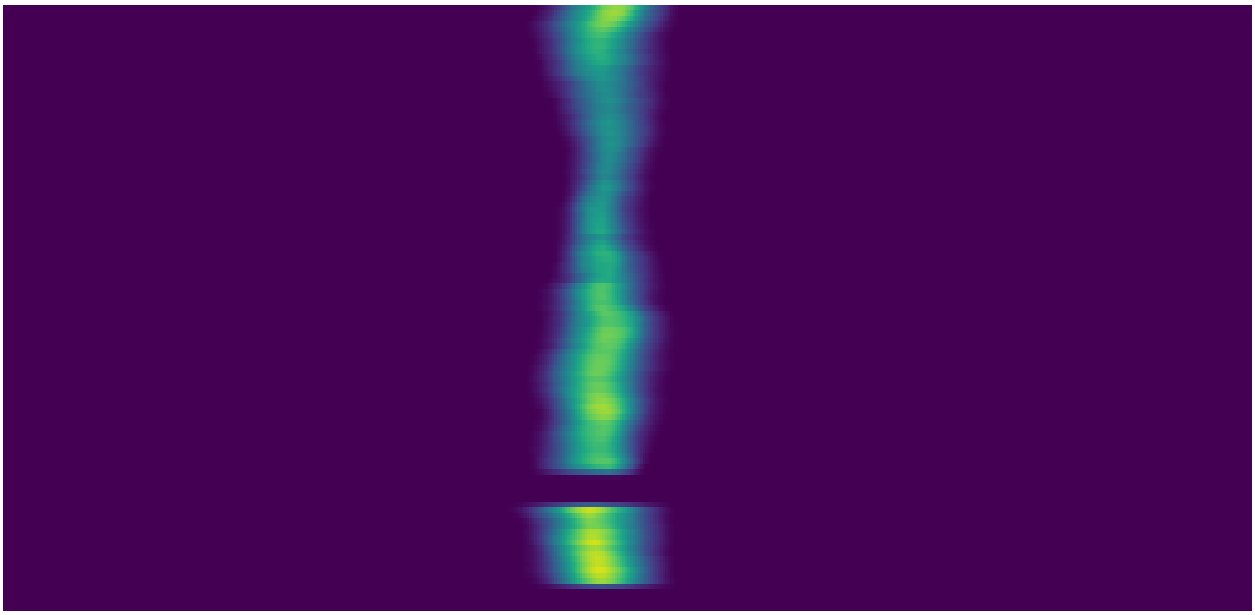

Figure 3.1 – Force histogram of a sample from the class C

Data distribution for the CNN

At the end of the process, we successfully generated a total of 120 subfolders containing approximately 104,986 images for the masks. From these masks, we extracted 120 force banners, corresponding to the different classes (A: 41 folders, B: 38 folders, C: 41 folders). Each class was then split into training (80%), validation (10%), and test (10%) sets. After the split, we reorganized the sets and their respective labels (grouping back all three sub-training sets into one, etc.), ensuring that each image remained associated with its corresponding label.

No preprocessing was applied to the images, as each pixel held a significant value. Similarly, we decided against normalizing the data due to the varied ranges of values present in each image. To ensure uniformity, we reshaped all the images to a consistent size of (180x400) pixels, preparing them for input into our learning model. The reshape size was decided because the length of the force histogram array was set to 180p (representing 180 elements), while the average number of generated pairs per subfolder was approximately 400p.

Model training phase

During this phase, the CNN is fed with generated images from the previous step, and the expected output is the corresponding class to which each image belongs. We used "Adam" Optimizer[1] and the binary cross entropy loss function, which is calculated using the following formula:

$$\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

where:

- N represents the total number of training examples,
- y_i is the true label for the i-th example,
- p_i is the predicted probability of the i-th example belonging to the positive class.

Trained for 10 epochs with a batch size of 64, the figures 3.2 depict the accuracy and loss curves during the training process, We can observe that the curves exhibit a well-converged behavior, indicating successful classification. The convergence of the curves implies that the model has learned to accurately classify the data and minimize the loss function.

[1]The Adam optimizer is a widely used optimization algorithm for training neural networks, including CNN models. It adapts the learning rate for each parameter based on the observed gradients, combining the advantages of AdaGrad and RMSprop. Adam dynamically adjusts the learning rates, leading to faster convergence and better handling of sparse gradients, making it popular for deep learning tasks.

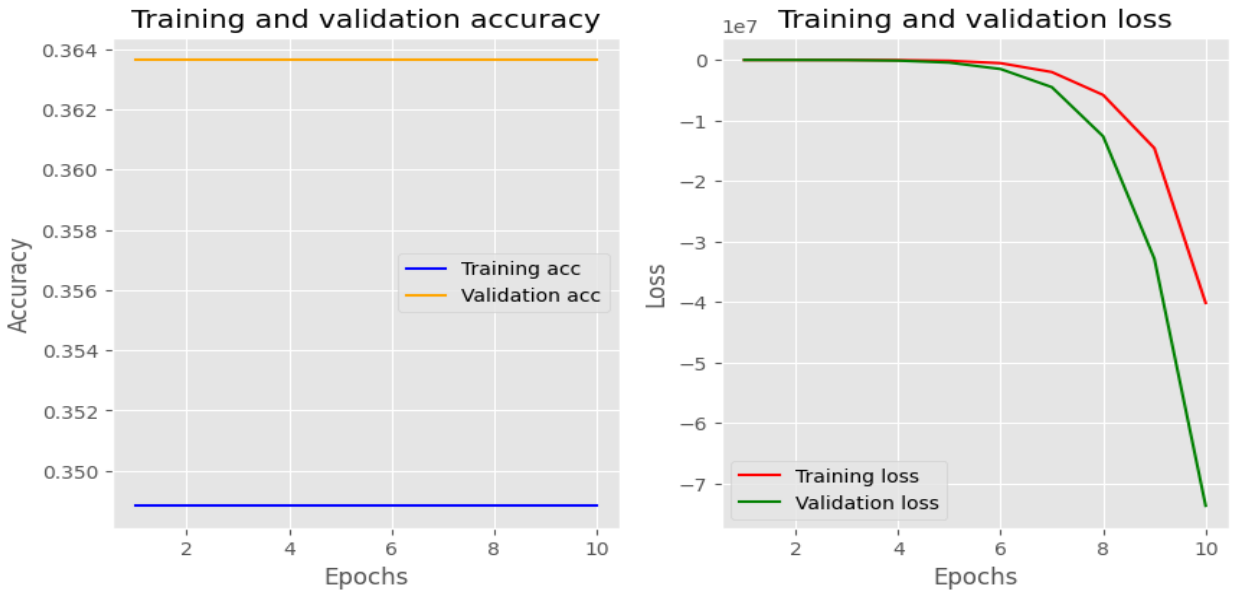

Figure 3.2 – Accuracy and loss curves

Model evaluation phase

The test set is used for evaluation, and the model's predictions are compared against the ground truth labels.

Metrics such as accuracy, precision, recall, and F1 score are used to measure its performance. This step helps determine the model's accuracy and suitability for real-world applications, identifying areas for improvement if necessary.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Population}} \tag{3.1}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.2}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.3}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

The following table shows a summary of the results:

Metric	Accuracy	Precision	Recall	F1
Result	0.33	0.11	0.33	0.17

Table 3.3 – Results of the metrics used for the evaluation

We observed from the values of these metrics that the classification performance was not optimal. However, it is important to note that with a larger and more diverse dataset, the model would have the potential to improve its results. Increasing the amount of training data can help the model learn more representative patterns and generalize better to unseen examples, potentially leading to better classification accuracy.

Conclusion

In conclusion, this chapter has presented the experimental study and results obtained from evaluating the proposed approach. By carefully designing the methodology, selecting appropriate datasets, and defining evaluation metrics, we were able to assess the performance and effectiveness of the approach.

Conclusion and Future Perspectives

In conclusion, this project focused on tackling the challenge of analyzing the spatial configuration and temporal evolution of objects in real-world videos, particularly in the context of human actions. We presented a solution that involved frame extraction, object detection using YOLOv8, object tracking using SORT, and the computation of force histograms to capture spatial relationships between object pairs. We also utilized deep learning techniques, training a CNN on the generated data to predict complex spatiotemporal scenarios. Despite the challenges, our solution provided valuable insights into the spatial configuration and temporal evolution of objects in videos. It opens up possibilities for further research and development in the field of computer vision and object tracking. The combination of spatial-temporal analysis and deep learning techniques holds promise for advancing our understanding and prediction of complex spatiotemporal scenarios. As for future perspectives, we proposed several improvements such as:

- Enhanced Object Selection: Provide users with the option to choose between selecting detected objects or manually selecting regions of interest. This interactive feature allows users to manually define the object's region by interacting with the window and selecting the four corners that form the region of interest. Additionally, consider incorporating a second type of tracking specifically for the manually selected regions and also to ensure accurate and consistent tracking even if the SORT algorithm loses track. In this case, prioritize the SORT algorithm, and only utilize the second tracker when the desired ID is not found.

- Direct Video Processing: Enable the program to directly process videos without the need for pre-framing. This enhancement would allow users to input raw videos directly into the system, eliminating the step of frame extraction.

- YOLOv8 Model Expansion: Retrain the YOLOv8 model on datasets that include classes not currently included in its predictions. By expanding the training dataset and incorporating additional classes, the model's detection capabilities can be improved and extended to cover a wider range of objects or entities.

- CNN Training with Expanded Data: Retrain the CNN using a larger dataset that includes more diverse classes. By expanding the training data, the CNN can learn more robust and generalized features, leading to improved performance in object tracking and classification tasks.

These future perspectives aim to enhance the flexibility, accuracy, and capability of the project, allowing for more user interaction, broader object detection, and improved tracking and classification performance.

Bibliography

[1] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. 2023.

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[3] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Retinanet: Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, 2017.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, 2017.

[6] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and real-time tracking. In *Proceedings of the IEEE International Conference on Image Processing*, 2016.

[7] Xin Li, Wanli Ouyang, Ping Luo, Xiaogang Wang, and Chen Change Loy. Trackletnet: A deep learning framework for online multi-object tracking. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[8] Ronald Mahler. Multiple hypothesis tracking for multiple target tracking. In *IEEE Aerospace and Electronic Systems Magazine*, 2007.

[9] Robin Deléarde, Camille Kurtz, and Laurent Wendling. Description and recognition of complex spatial configurations of object pairs with force banner 2d features. *Pattern Recognition*, 123:108410, 2022.

[10] P. Matsakis and L. Wendling. A new way to represent the relative position between areal objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(7):634–643, 1999.

[11] 2021 fivb volleyball men's nations league dataset. YouTube Video, June 2021. Accessed on: May 2023.