UNIVERSITY MOHAMED BOUDIAF MSILA

FACULTY OF MATHEMATHICS AND COMPUTER SCIENCE
DEPARTEMENT OF COMPUTER SCIENCE

This thesis is submitted in partial fulfillment for the degree of

Master in Computer Science

by:

Anis Djaidja

Architectural Optimization of Computer Vision Surveillance: A Hybrid Approach for Human Detection and Profiling

Under the supervision of

Raouf Ouanis Lakehal Ayat

Linda Belabdelouahab Fernini

Composition of the jury

Adel MOUSSAOUI	Univeristy of Msila	President
Raouf Ouanis Lakehal Ayat	Univeristy of Msila	Supervisor
Linda Belabdelouahab Fernini	Univeristy of Msila	Co-Supervisor
Abdessettar GHEMOUGUI	Univeristy of Msila	Examiner

June, 2024

This work wouldn't have been accomplished without the guidance and grace of God the Most Merciful. I am dedicating it to my father who taught me how to embrace the hardships of life and made me the man I am today. To my loving mother who attended to me in my hardest moments and prayed for my success. To the warm family I emerged from, to my friends who cared about my achievements, and in memory of the beloved who still lives in my heart . . .

Acknowledgements

First of all, I thank God Almighty, who gave me strength, patience, and courage to fulfill this goal. Praise be to God, by whose grace good deeds are accomplished.

While working on this project, I knew that being grateful is no less difficult than writing this thesis, acknowledging all the people who helped me make this work a success as it should be.

Then, I sincerely thank my supervisors, Dr. Raouf Ouanis Lakehal Ayat, and Professor Linda Belabdelouahab Fernini, for agreeing to supervise this work with the intent of seeing me succeed, and for their valuable time, support, advice, and guidance throughout this journey.

Also, I refer to the Communities and Journals of related fields and disciplines, for their efforts, the sincerity of the content they publish publicly helped me in my research.

Finally, I express my deepest gratitude to my family, my father, my mother and my uncle, for always being by my side in my most difficult moments, for their love, support, and encouragement.

To all, thank you.

Contents

List of Figures

List of Abbreviations

HOG	Histogram of oriented gradients
YOLO	You Only Live Once (object detection framework)
HPE	Human Pose Estimator
HAR	Human Activity Recognition
SWA	Sliding window Algorithm
IPC	Inter-Process Communication
DL	Deep learning
ML	Machine learning
MPSTA	Multi-Person Skeleton Tracking Alternator Algorithm
MARCRP	Multi Analysis and Result Caching for Robust Profiling Algorithm
IoC	Inversion of Control
DI	Dependency Injection

General Introduction

Providing security and safety to individuals is a major concern of any society today due to the constantly increasing actions causing threats, HAR can be defined as the challenge of recognizing when a person is engaging in certain activities. Hence, HAR attempts to identify the activity being performed by an individual, alongside when the activity is taking place. This is exceptionally more challenging considering that human errors are hard to measure and predict, analysis of deliberate human actions and accidents alike seems to leave the current HAR state of the art not as advanced as for example object detection.

The primary aim of HAR is to accurately describe human actions and their interactions from a previously unseen data sequence acquired by sensors. The ability to recognize, understand, and predict complex human actions enables the construction of many important applications such as intelligent surveillance systems [2].

In this work, we aim to leverage a selection of computer vision tools in a proposal of a new architecture of a hybrid surveillance system to supplement the security/safety sectors in autonomous monitoring and collecting personnel's behaviors over extended periods.

Problem statement

Communal spaces, Government buildings, Hospitals, Shops, Industrial zones and workshops all share the factor of human presence. Consequently, providing security and safety through surveillance systems in these sectors is more challenging consid- ering that human errors are hard to measure and predict, and thus, it is hard to classify legitimate human activities from anomalous ones.

In this goal, there are a lot of types of systems already made, but there is a trend in which we found that, the more effective the system is, the more likely

its complexity is higher, and with it, its operability cost and time required to develop/customize it.

Furthermore, complex computer vision lacks robustness, a key element in any system to be able to run for extended times and with less manual maintenance required, such as when a certain component fails, whether to detect, analyze, or simply stops working, the integrity of the running surveillance should remain.

Contribution goals

After recognizing the ultimate importance of robustness as well as operability and complexity, as described in the problem statement. We aim to contribute to the State-of-the-art Computer Vision-based Surveillance systems in :

- A novel architectural optimization of AI-driven human surveillance systems augmenting stability and reducing resource overhead when it comes to deployment of such a system. Featuring our MPSTA Algorithm,
- A robust approach: component-based multi-analysis HAR through real-time profiling with hybrid techniques over video surveillance. featuring our MARCPR Algorithm,

Dissertation structure

This dissertation is divided into three main chapters. Chapter 1 provides the necessary background and situates the dissertation within the state-of-the-art.

Chapter 2 introduces our proposed system by providing both high and low-level descriptions of its operational modules, introducing our invented and pre-existing mathematical and computer science concepts utilized in designing the two main Algorithms that enable our Architectural optimization.

Finally, Chapter 3 delves into the technical view on the development of a case-specific implementation of our system, for validation purposes.

Chapter 1

Human Behavior Profiling, Background and Positioning to State-of-the-Art

1.1 Introduction

In this chapter, we provide the necessary background and situate the dissertation within the state-of-the-art.

1.2 Artificial Intelligence

Artificial intelligence, or AI, is technology that enables computers and machines to simulate human intelligence and problem-solving capabilities. [3]. AI has various fundamental application incorporating NLP, healthcare, automotive, gaming, speech recognition, finance, and vision systems [4] (Computer Vision).

1.2.1 Current potential: Superintelligence

Advanced artificial intelligence (AI) or superintelligence promises great disruption in the law, economy, and society. The world is close to reaching an inflection point; the so-called existential threat of superintelligence with the potential of replacing human control and decision-making with its creation [5].

1.3 Computer vision

Computer Vision is a branch of Artificial Intelligence, a field of studying and developing technology that enables computers to process, analyze, and interpret digital images [6]. Looking over the history of computer vision, it is important to note that because of the broad spectrum of potential applications, the trend has been the merge of computer vision with other closely related fields. These include: Image processing (the raw images have to be processed before further analysis). Photogrammetry (cameras used for imaging have to be calibrated. Determining object poses in 3D is important in both computer vision and photogrammetry).Computer graphics (3D modeling is central to both computer vision and computer graphics. Many exciting applications need both computer vision and computer graphics) [7]. Fast forward to modern days, At this moment, the technologies that power computer vision have finally begun to catch up to our dreams of its applications [8] - Medical diagnosis computer vision, self-driving-car and autonomous surveillance systems. In the book Deep Learning to See [9], the author states that The remarkable progress in computer vision over the last few years is, by and large, attributed to deep learning, fueled by the availability of huge sets of labeled data, and paired with the explosive growth of the GPU paradigm.

1.3.1 General context

In this work, we are going to study and improve upon the use of applied computer vision for autonomous surveillance of human behaviours. including detection of activities and profiling over time. Utilizing prominent and novel computer science's software engineering ideas to build and optimize a framework for Architectural Optimization of HPE-based Surveillance.

1.3.2 Surveillance systems

The market's inclination towards rapid adoption of new technologies creates opportunities for surveillance systems to offer more than what is currently available. These systems can be applied to a wide range of areas, such as security, safety, and quality control, which are currently lacking. The human factor plays a central role in these areas, making them essential to target through the detection, analysis, and profiling of human behavior.

1.3.3 Human monitoring

The issue of safety and security in the workplace has garnered considerable attention over the past few decades. This increased concern stems from the mounting evidence that clearly demonstrates the profound consequences of human behavior on the loss of lives and properties. Governments, safety organizations, and managerial laws and regulations are actively involved in overseeing industrial safety. Consequently, it becomes crucial for individuals in positions of responsibility to ensure adherence to safety measures in order to avoid the risk of job loss or facing penalties imposed by these regulations.

1.3.4 Challenges

FIGURE 1.1: People smoking up outside Forth Valley Royal Hospital in Larbertnow now face a fine by law cite the reference of this picture as stipulated earlier [10]

Pinpointing challenges in enforcing safety regulations is made exceptionally harder by the unpredictable nature and diversity of human behaviors. The incorporation of systematic methods such as OHSA training for industry, street laws depending on countries, encouragement of safety reports, and real-time monitoring through traditional CCTV all contribute as substantial additions for employees, lawmen, and experts in the field.

In 2014 NHS Forth Valley introduced a tobacco control officer to patrol its hospital sites after complaints from patients and staff about smokers congregating round entrances to its buildings [10].

Let alone the continuous potential danger in any field which can hinder productivity, harm lives, disturb the collective atmosphere all at once.

FIGURE 1.2: A closeup picture of two workers wearing safety gear (legs not visible) - from the web

1.4 State of The Art

1.4.1 Human Detection for Activity Recognition

Human detection methods such as the YOLO framework [11], HOG image descriptor [12], and other similar techniques are widely accessible. These methods are primarily designed to identify pedestrians, who are typically upright and fully visible.

Limitations

Although Human Detectors can accurately predict the general location of the human body in the footage, they fall short in detecting specific activities or behaviors. This limitation arises from the fact that the results obtained are too imprecise to achieve such a task. The outputs merely consist of four points that annotate a rectangle in the image, providing insufficient information.

1.4.2 Human Pose Estimation : A promising alternative

The latest generation Human Pose Estimation (HPE) is a task in Computer Vision that focuses on identifying the position of a human body in a specific scene. by estimating key points (landmarks) from a detection instance of a human body, which is called region of interest in a given digital image.

There are multiple pre-trained Deep Learning models today that offer this, examples being OpenPose, PoseNet and Google's Mediapipe. Based on BlazePose

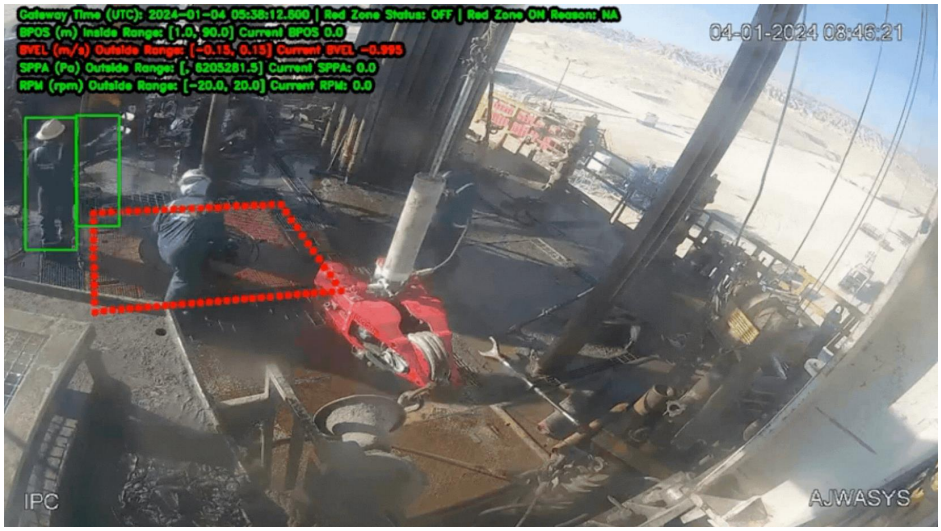

FIGURE 1.3: A preview from a workers' surveillance system made with State of The Art YOLO deep learning model - from the web.

model, Aka Mediapipe pose solution which uses a twostep detection/tracking machine learning pipeline, the pipeline first locates person/pose region-of-interest, the tracker subsequently predicts pose landmarks.

Limitations

Current state-of-the-art approaches rely primarily on powerful desktop environments for inference, whereas Mediapipe's method is able to achieve real-time performance

1.4.3 Related works and literature review

In the HPE scene, there are a couple of developers on the web who claim to have unlocked multi-person detection tracking on blazor pose estimation (Google's media pipe) [13] but have presented no tangible evidence. Google claims in their website of publishing a proposal for multi-person detection for the same HPE but no paper nor implementation has been seen yet too. So we acknowledge based on research that our proposal of multi-person tracking HPE based on Blazor pose, is the first to be revealed as far as we know.

Modular and Hybrid HAR-based surveillance systems have been an active topic of research in the past decade, and their application using skeleton pose is no exception. [14] and [15] both explore the approach of utilizing the sequence

of landmarks extracted from a video in a time series fashion to detect human actions in real-time.

In [16], unlike how in recent years, many authors have published interesting proposals focusing on a particular kind of analysis and based on a single aspect of surveillance, this work [16] has demonstrated the effectiveness of combining multi-analysis methods to achieve a robust surveillance system.

Computation scheduling

The concept of work stealing for multi-processing and multi-threaded scheduling in framework design dates back to the 1980s, but it is a widely reasearched area in computation and it is employed in modern framworks like in the Java fork/join framework [17]. the .NET Task Parallel Library [18]. and the Rust Tokio runtime [19].

1.5 Conclusion

In this chapter we have presented the necessary background of AI and its branch Computer Vision, the context of our research, the techniques available for human detection and a related works discussion exploring similar studies and contributions to what our work is going to address.

Chapter 2

Architectural Optimization of HPE-based Surveillance : A Robust Framework

2.1 Introduction

In the previous Chapter we covered the background concepts, the scope in which our study falls, the challenges and other similar State of the Art works. In this chapter, we will present how we intend to solve the two main difficulties facing our architecture and the components design comprehending all parts of the proposed system.

Namely Single Person Skeleton detection/tracking dilemma, Hybrid action analysis and robust profiling. As well as other minor solutions complementing the robustness element we aim for.

2.2 Architecture overview

This architecture is not entirely new in the state of art of computer vision surveillance but an optimization to its core parts and design which aims to provide a robust autonomous system, we have left the interface of this system unmentioned due to it's insignificance and because it is shown in our previous work. The flow of data and tasks in this system is controlled with Inter-Process Communication measure (check IPC section) to link its dependencies in order to provide a self-contained software abstract enough for engineers to work

with as framework and for other users to use it as a solution with minimal experience and no technical knowledge required.

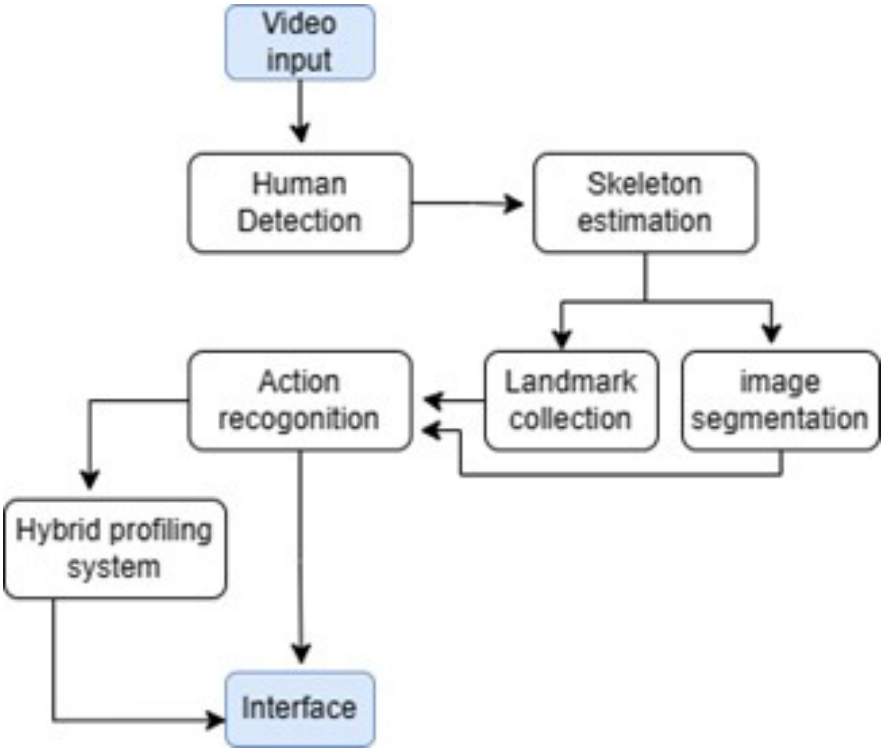

FIGURE 2.1: Our abstract representation of the general architecture of the proposed system.

2.3 MPSTA Algorithm - Multi-Person Skeleton Tracking Alternator

The use of HPE (Human Pose Estimators) models in this system poses a difficulty in term of how many persons can be detected in parallel, Detection-Tracking HPEs are made to offer a practical performance compared to other State-of-the-art, this is our choosing of Google's Blazor Pose Estimation model (Commercially known as Mediapipe solutions) for the detection task, the price for this practicality is limited detection to only one person at a time.

To addressee this, we propose, a multi-process design, inspired from a mathematical concept to bypass the Single-person limitation of the chosen HPE and allow for multi-person detection by fully leveraging the local machine's CPU cores, instantiating an HPE in multiple processes that can each run asynchronously while sharing global and local memory.

MPSTA Algorithm is based on the following ideas which are discussed in this section :

- Based upon any two-step Tracking HPE
- Sliding window algorithm applied in detection distribution over multiple worker processes
- Semi-Continuous detection loop
- Work stealing process scheduling
- Distribution of detection
- Alternation of detection

2.3.1 Sliding window algorithm

Sliding window algorithm, is a computational/mathematical technique that aims to reduce the use of nested loops of the time complexity O(n), and replace it with a single loop of the time complexity O(1), therefore reducing the time complexity. here This is an abstract real-world example that would be observed when applying oil to a connected chain together. Another way to do so is to use a cloth, dip it in oil, and then hold onto one end of the chain with this cloth. Then instead of re-dipping it again and again, just slide the cloth with your hand onto the next section, and next, and so on until the other end [20].

The use of the Sliding Window technique can be done in a very specific scenario, where the size of the window for computation is fixed throughout the nested loop. In this situation, the size of the window is fixed arbitrarily by taking into account the number of cores the local machine has.

2.3.2 MPSTA Algorithm

MPSTA Algorithm is our adaptation of the SWA on detecting and tracking human pose, its main goal is to allow multiple (more than one) fixed size semicontinuous (see next sub-section) tracking of persons in a scene and occasionally switch to track any other persons that are not covered by the fixed number of trackers.

It is composed of two main steps in algorithmic sense, or concerns in design sense. first is a distribution mechanism orchestrated by the a parent process, spawning a skeleton estimator model and handling its configuration and later running state, second is the alternation phase, where each estimator has access to decide a person not worth monitoring currently and marks its location to be obscured globally (the parent scope), effectively allowing it self and other estimators to check out another unseen person.

2.3.3 Semi-Continuity

In mathematical analysis, semicontinuity (or semi-continuity) is a property of extended real-valued functions that is weaker than continuity. An extended real-valued function f is upper (respectively, lower) semicontinuous at a point x_0 if, roughly speaking, the function values for arguments near x_0 are not much higher (respectively, lower) than $f(x\_0)$ [21].

2.3.4 Work scheduling : stealing threads

Moreover, the scheduler should also try to maintain related threads on the same processor, if possible, so that communication between them can be minimized [22].

In Work stealing scheduling, a processor (or a thread) with a computation task (most likely continious or semi-continious) has a queue to hold incoming work items for processing, the process parent may also spawn other processors in parralelle when there is a lot of work or simply because it is programmed to schedule more processing, these new processors will behave like each other.

When a processor runs out of work, it looks at the queues of the other processors and "steals" their work items. In effect, work stealing distributes the scheduling work over idle processors, and as long as all processors have work to do, no scheduling overhead occurs [23].

2.3.5 Distribution

Each detection process starts independently from a single provider process, which simply shares frames from a video stream to whatever process asks for it at any given time, in this case being the 1st detection process, which runs pose detection. depending on the result, it then stores the resulting landmarks as a global entity, later to be profiled, conceals the detected person withing

the frame, and finally passes the modified frame to the next detection process, acting as it's provider.

The first detection process immediately asks for a new frame to process it. Every detection process is the same as the 1st except for the last one, which does not have to act as a provider, knowing that there are no processes bellow it in the chain of detection. a detection process will always receive a frame where the number of concealed persons equals the number of processes on top of it, in the chain of detection.

The chain of detection repeats every frame, but does not necessarily end in the same amount of time, meaning, this algorithm, may cause a slight delay in each detection instance relative to the video stream timeline.

The detection instance delay (Δ) is calculated by function of the detection process number (β) and its corresponding overhead (initial delay Δ_0)

$$\Delta_\beta = \Delta_0(\beta - 1)$$

The maximum number of persons we can detect in parallel (α) equals the total of detection processes (β) running in the chain of detection.

$$\alpha = \Sigma\beta$$

Mediapipe's two-step pipeline ensures, to some extent, that every detection process detects the most similar person to its last detection by tracking the detection instance until it disappears or becomes obscured from the frame stream, which improves the multi-person accuracy and more so, stability [13].

The following Figure 2.2 explains.

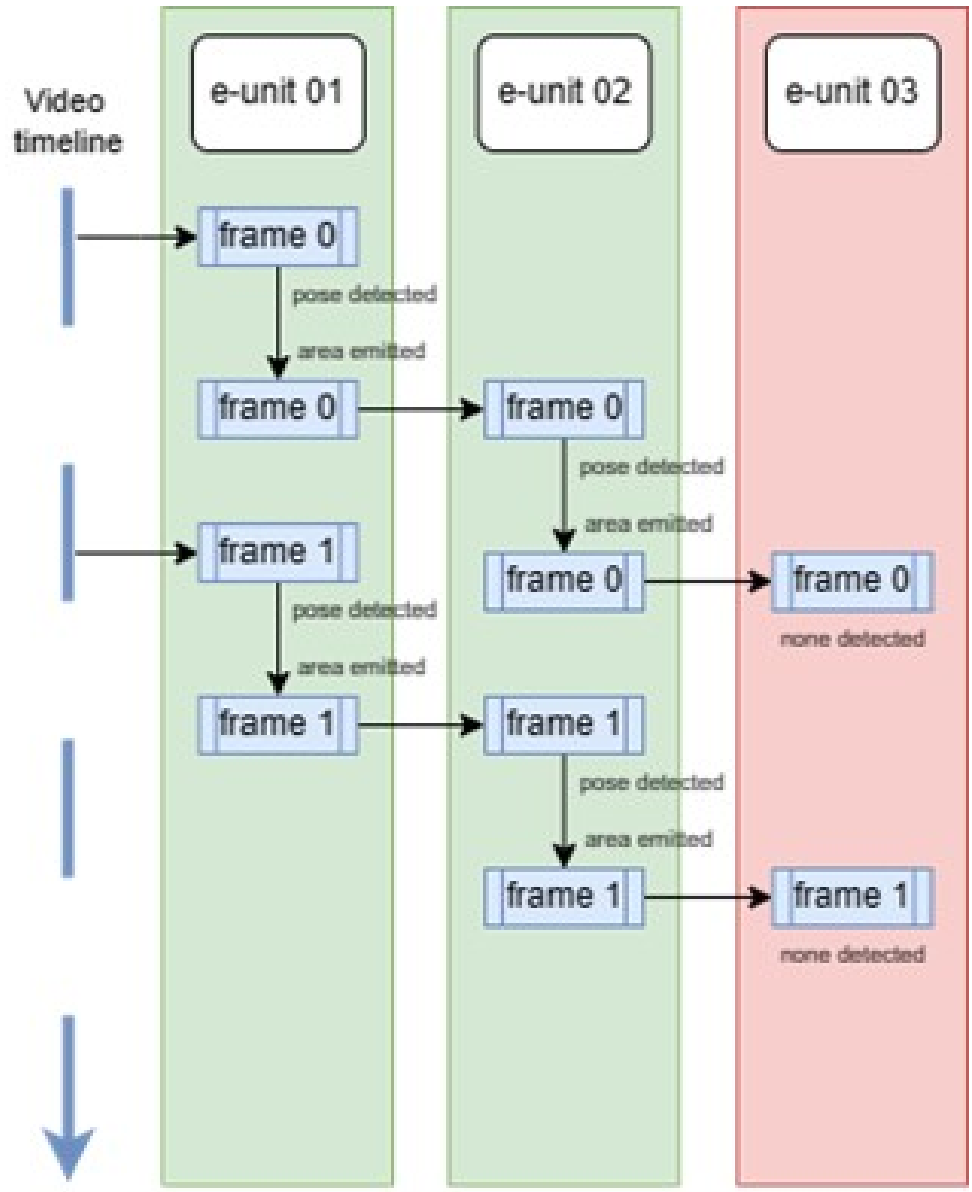

FIGURE 2.2: Our representation of the fixed-sized detection units handling detection in a scene where only two people are visible.

2.3.6 Alternation

After setting up a distributed computation mechanism for the skeleton detection, we further enhance it with an alternation behavior, a fast-paced shift in attention to cycle between visible subjects in the scene.

In a live stream camera video where there is a dozen of subjects, the estimation based on sliding window concept would not be limited to its fixed size, our proposal is to divide the timeline between possible subjects, given that the

video is processed in a respectable speed (e.g. 30 FPS), every skeleton estimation unit holds a reference to the last detected subject and based on it emits its last known area in the subsequent frame to look for another target.

To avoid infinite traverse looking for new targets, the number of times this happens is set to be configured manually, to allow the estimation unit to regress back to the first natural target it detects.The number of possible detections can now be increased linearly by the number of alternations it is allowed (ϕ).

$$\alpha = \Sigma\beta\phi$$

The following figure 2.3 explains.

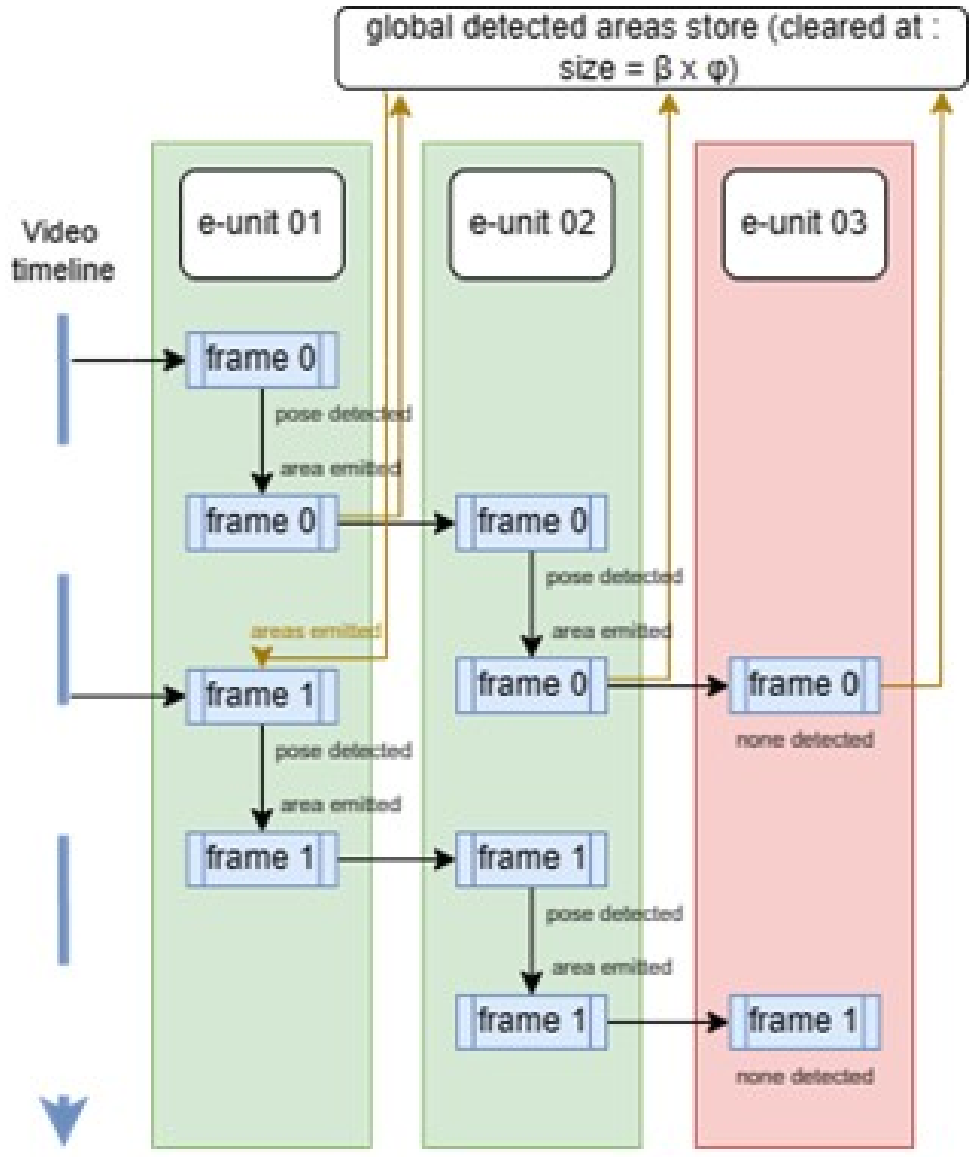

FIGURE 2.3: Our representation of the fixed sized detection units handling detection in a scene where only two people are visible.

2.4 MARCRP Algorithm - Multi Analysis & Result Caching for Robust Profiling

To perform real-time Profiling of a person's behavior, the second part of our system involves a combination of rule-based calculations, estimations, and data classifications, data entry being mainly the extracted pose landmarks, and body parts images snapped from real-time footage. The profiling system relies on Rule-Based calculations, Machine learning, and Deep learning pipelines. This inference runs in parallel with HPE (1st phase) in architecture design.

Parallel profiling also puts to work unsupervised machine learning / deep learning models, although built on a pipeline set by an expert that specifies classes. The actual model takes the inputs whether a set of scenes or image datasets performs human body extractions, and finally runs unsupervised to achieve both training and inference.

2.4.1 MARCRP Algorithm

MARCRP is our algorithm for Multi Analysis & Result Caching for Robust Profiling, its ideas are :

- Assign analysis based on need and fit of the best DL or ML technique (hybrid analysis concept)
- Assign specific data pipeline ends to each group of analyzers based on role and input.
- The use of Inversion of Control (IoC) in orchestration of groups and individual models from a centralized point. Creating an abstract framework layer and a application (runtime) layer.
- Loose use of Dependency Injection for configuration on the framework layer and for model assignment to unlock a powerful way to put the framework to use on custom use cases.

it spawns a fixed set of profilers (working processors) which will run asynchronously in parallel with detection processes, meaning the whole two operations combined would end in the same (n) (of both scopes of operation) number of outputs, final outputs are profiles of detected humans in the targeted scene.

It features the use of Rule-Based Methods that mainly work for classification purposes, these simple classifiers may target basic activities that are clearly expressed by a human body pose, through a mix of supervised Deep Neural Networks (DNN) where the neurons are fed with Euclidean calculations as well as Motion tracking of key points, as shown in figure 3.

To achieve this, the profiling system acts as a separate process after it receives raw data from the previous extraction phase. It then bundles the data compatible with the set of RBA inputs, and channels them into a Collector instance which is unique to the Profiler instance, (in other words, every Profiler has its own Collector), the Collector then runs a set of pre-processing, stores a copy of the data as history for the purpose of motion tracking, then returns both the current data and the history to Profiler to apply Rule-Based Assumption.

2.4.2 Inversion of control

Inversion of control is a well established design pattern where by creating a central parent runtime, we call it Pool parent process, it acts as a framework that in theory calls implementations provided by the application, which is in practice the rest of the system. This is beneficial to our profiling system which needs what we call a loose control over its callbacks, but ultimately more beneficial with the use of Dependency Injection, an extension of IoC.

The core of the system will be running in a non monolithic manner but controlled in its calls and callbacks by the parent process. We can now for example specify a set of combination of SVM classifier models and DL recognition for a single image analysis task simply by Path based string pointing to where the models are stored or by calling an interface of calculation handler classes in the Pool Parent Class.

2.4.3 Dependency injection

Dependency Injection is a set of software design principles and patterns that enables development of loosely coupled code [24], Thus, facilitating the implementation of more mature architectures.

The advantage of loose coupling is the same in software design as it is in the physical socket and plug model: Once the infrastructure is in place, it can be used by anyone and adapted to changing needs and unforeseen requirements

without requiring large changes to the application code base and infrastructure. This means that ideally, a new requirement should only necessitate the addition of a new class, with no changes to other already-existing classes of the system. [24].

There are three main styles of dependency injection. referred to in the topic of Inversion of Control as : type 1 IoC (interface injection), type 2 IoC (setter injection) and type 3 IoC (constructor injection) [1].

Other than loosely coupled code, when designing software with Dependency Injection in mind, it becomes second nature to make modularity oriented design decisions. which serves in the goal of a robust scene analysis system, by injecting what's appropriate from a set of calculators, graphs, trained machine learning models and rule based classifiers into various analysis components for processing a single pipeline of imagery for the best collaborative results, a requirement in the robustness aspect of our proposed system. (see chapter 3 section 4)

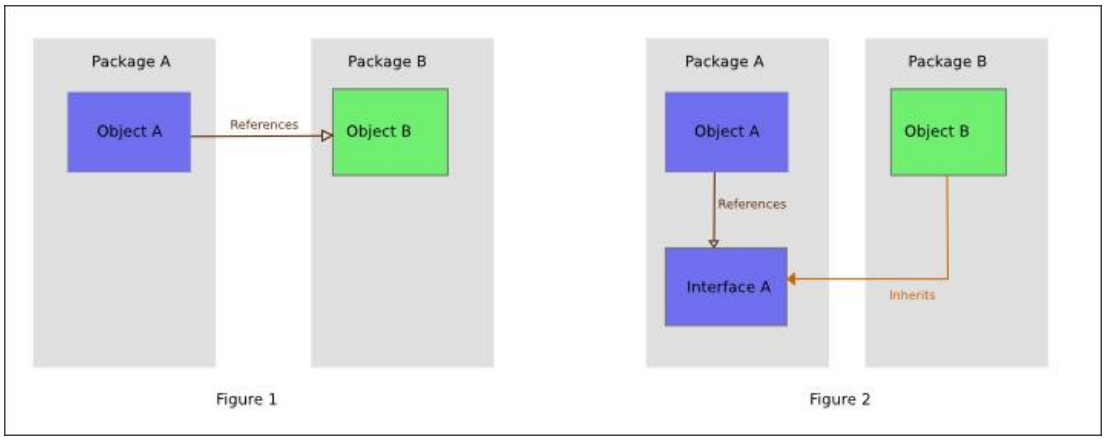

FIGURE 2.4: Dependency Injection of Type 1 IoC (Interface Injection) [1]

2.5 Conclusion

In this chapter we showcased the general architecture of our proposed framework, and provided design description of our MPSTA and MARCRP Algorithms, explaining not only their sequence of processing individually but also how would they be situated and interacted with in our system. In the next chapter we provide a discussion about an experimental implementation in the goal of validating our framework's claims.

Chapter 3

Implementation for validation purposes

3.1 Introduction

The previous chapter was about the design and architecture of our system and its algorithms. In this chapter, we provide a detailed discussion on the system's implementation for validation purposes, acknowledging that a comprehensive description of such a system would require an entire book.

3.2 Experimental Implementation

This section gives an overview of the specific configuration and experimental-conditions used for our experimental Implementation of the profiling system.

3.2.1 Hardware environment

Machine 01: used for development

- **Operating system:** Windows 11
- **CPU:** Intel Core I3 9100f
- **Graphics Card:** Nvidia Geforce Gtx 1660 ti 6GB
- **Memory:** 16 GB RAM

Machine 03: used for performance benchmarking in low resources enviroment

- **Operating system:** Windows 11
- **CPU:** Intel Core I5 11055G7
- **Graphics Card:** Intel IRIS XE 4GB
- **Memory:** 08 GB RAM

3.2.2 Software environment

- **Visual Studio 2022:** Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to enhance the software development process. [25]
- **Visual Studio Code:** Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE. [26]
- **Desktop.ini Editor 1.1:** A simple GUI program for managing ini configuration files.
- **MediaPipe:** The MediaPipe Pose Landmarker task lets you detect landmarks of human bodies in an image or video. You can use this task to identify key body locations, analyze posture, and categorize movements. This task uses machine learning (ML) models that work with single images or video. The task outputs body pose landmarks in image coordinates and in 3-dimensional world coordinates. [13]
- **TensorFlow:** TensorFlow is a python library that makes it easy to create ML models that can run in any environment. Learn how to use the intuitive APIs through interactive code samples. [27]
- **Keras:** Keras is an API designed for human beings, not machines. Keras follows best practices for *reducing cognitive load*: it offers consistent & simple APIs, it minimizes the number of user actions required for common

use cases, and it provides clear & actionable error messages. Keras also gives the highest priority to crafting great documentation and developer guides. [28]

- **Open CV2:** OpenCV is the world's biggest computer vision library It's open source, contains over 2500 algorithms and is operated by the non-profit Open Source Vision Foundation. [8]
- **Configparser:** A simple script program for parsing config strings into config files
- **Numpy:** NumPy is an open-source Python library that facilitates efficient numerical operations on large quantities of data. [29]
- **Pandas:** pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. [30]
- **System.IO.Pipes:** C# low level library
- **System.Diagnostics:** C# library

3.2.3 Programming languages

- **Python:** With its core focus on readability, Python is a high-level interpreted programming language, which is widely recognized for being easy to learn, yet still able to harness the power of systems-level programming languages when necessary. Aside from the benefits of the language itself, the community around the available tools and libraries make Python particularly attractive for workloads in data science, machine learning, and scientific computing. [31]
- **C#:** Microsoft's .NET platform and the C# programming language were formally introduced circa 2002 and have quickly become a mainstay of modern-day software development. The .NET platform enables a large number of programming languages (including C#, VB.NET, and F#) to interact with each other. A program written in C# can be referenced by another program written in VB.NET. More on this interoperability later in this chapter.[32] . C# is a static strongly typed JIT compiled language

3.3 Detection scheduling

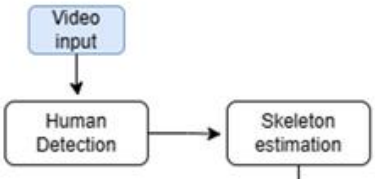

FIGURE 3.1: A cropped section of Our first phase of the system architecture

The first phase of the system promises to provide skeleton-based representations of persons detected in the scenes, so we implement a pose detection model.

3.3.1 Scheduling the detection phase

A provider class is instantiated from the main thread with a focus on implementing dependency injection into it, which allows it to efficiently run an additional layer on the current scene and attach the results before feeding it to the Estimator.

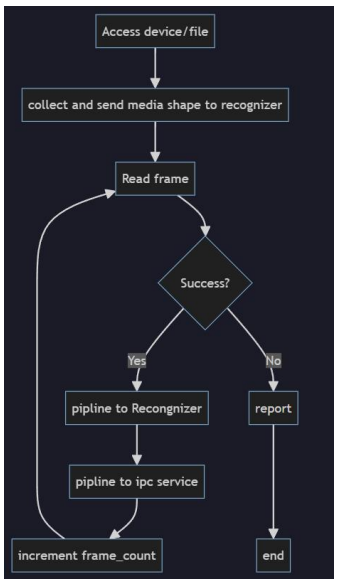

FIGURE 3.2: UML Flowchart describing Provider class behavior

Feeding the Estimator means passing the scene and pre-processing results (if there are any) into the parent Human Detector class that is implemented as a process pool, instantiating and initiating a fixed-size array of estimators each with a Mediapipe pose instance.

3.3.2 Estimators scheduling mechanism

The collection of estimators is first configured and then loosely linked, in a rather static manner. The first estimator input inputPipe and outputPipe are the ends of a low-level Pipe, usually used in multi-processing where memory is not shared. The input is retrieved to feed incoming scenes directly into it, by assigning it in runtime as the parent's input, which is what is referenced in the scene provider as the target pipe push into, with send(), and the recipient estimator does recv(), its output is then linked to the next estimator's input, the rest are inter-chained accordingly.

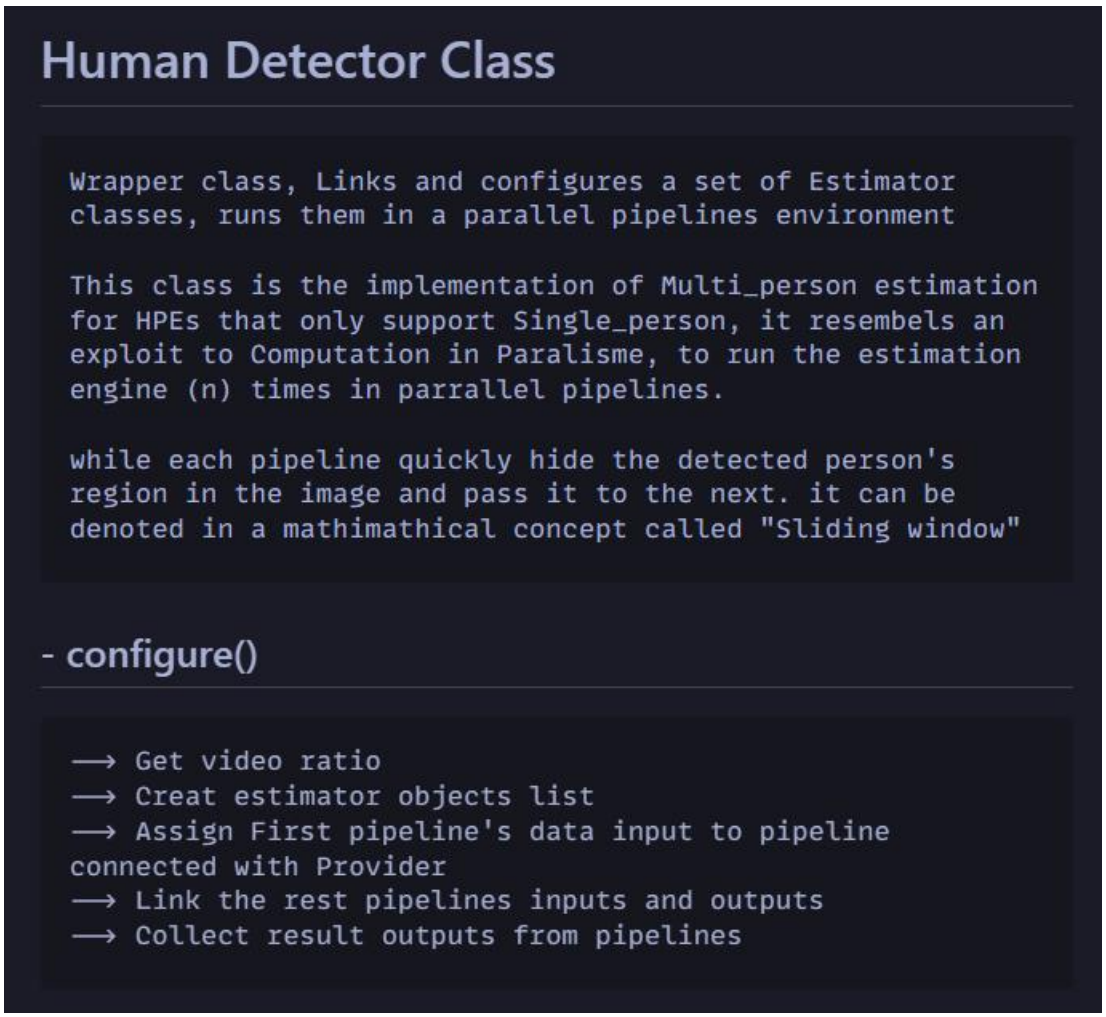

FIGURE 3.3: Pseudo procedural annotations describing Human Detector configuration

3.3.3 Estimation unit

Each Estimator has a state describing if it is busy or not, an image segmentation helper, a results pipe, a single pose estimation instance, and a custom state describing if it's the last unit in the chain.

Upon initialization, each estimator takes the arguments of its ID, input dimensions, and optionally a custom results drawer for fast data transfer enabling real-time results drawing.

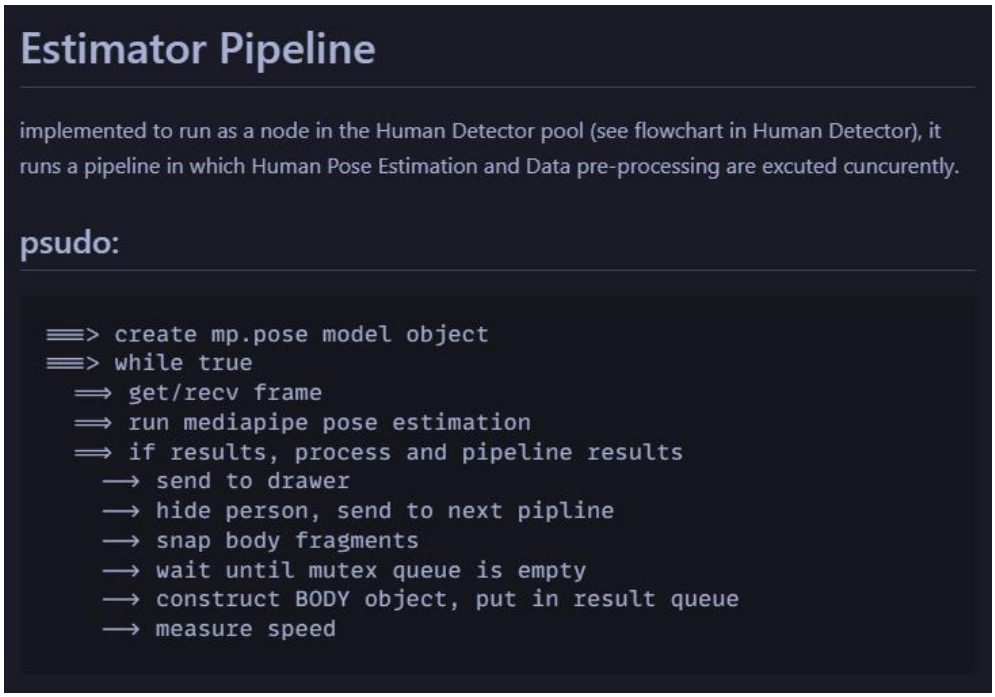

FIGURE 3.4: Pseudo procedural annotations describing how an Estimation unit works

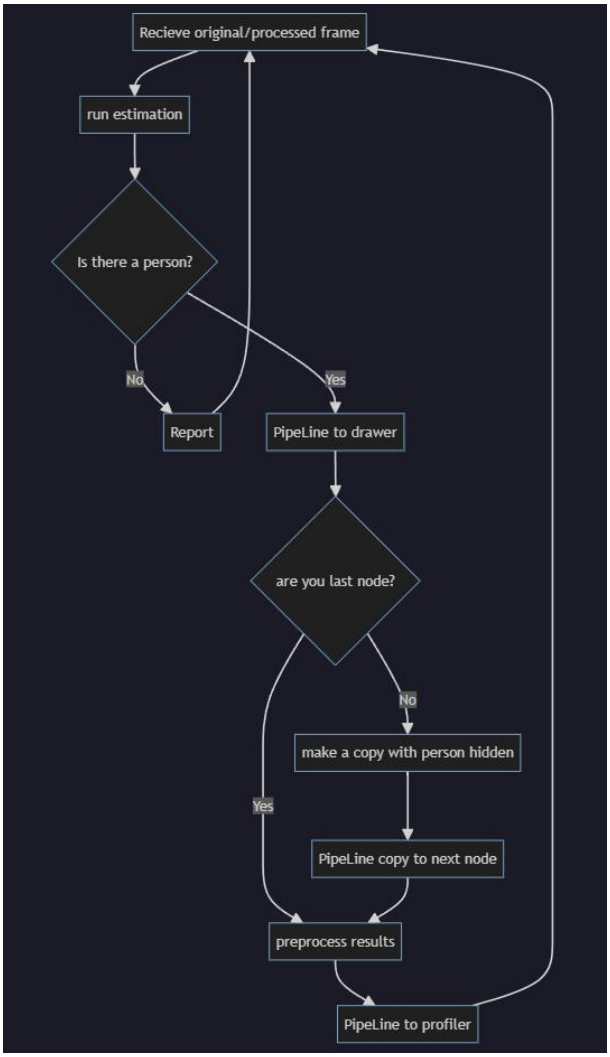

FIGURE 3.5: UML Flowchart describing MPSTA Algorithm Estimation unit behaves in distribution (not including alternation phase)

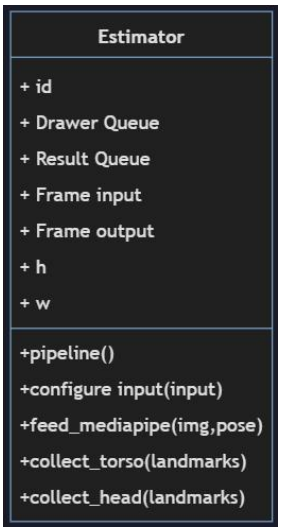

FIGURE 3.6: Estimation unit UML class model

3.3.4 Synchronization

When in action, estimators act as a chain of computation, adhering to the sliding window concept discussed in the previous section. After receiving a scene and a targets list (the last processed person excluded).

Due to the complete separation of estimators into processes, now each one can cache results and choose its target based on cached previous estimations, sticking to the same person for as long as possible.

3.3.5 Detection examples

In the following 3.7 and 3.8 figures, we can observe screen captures of human detection in real time using the implemented MPSTA Algorithm, the custom annotations are our own Drawer class which is working in another process in the framework (see next implementation sections), the digits above each person's skeleton is it's profile index (profileId), a demonstration of the success of our MPSTA algorithm in maintaining detection semi continuity while keeping track of the detection instance.

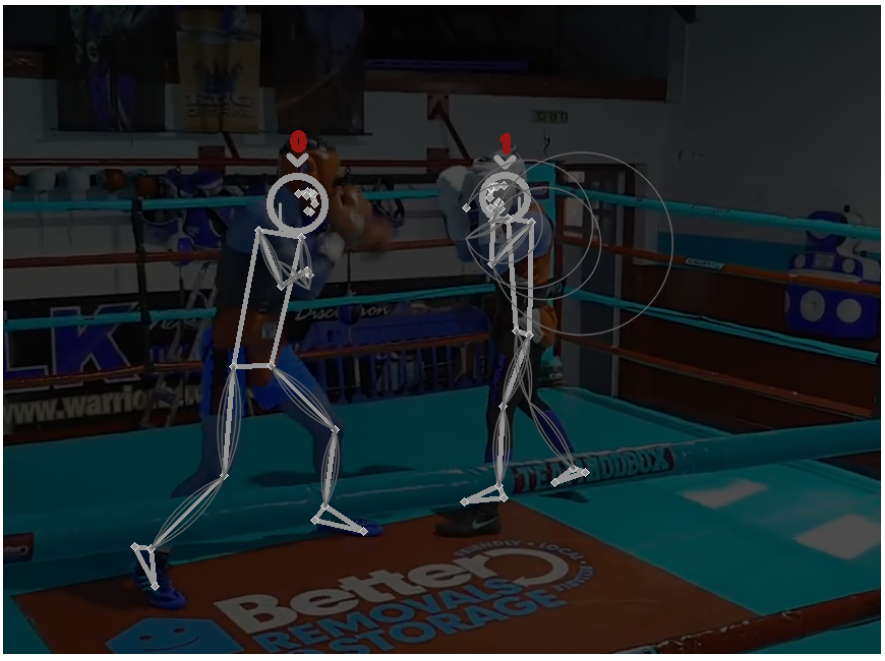

FIGURE 3.7: Our multi-person detection tracker after the implementation

FIGURE 3.8: Our multi-person detection tracker after the implementation

Figure 3.9 shows monitored performance of an average 14 fps, this was run in our second machine with a laptop non-cooled Core i5 cpu, very limited resources for any surveillance system.

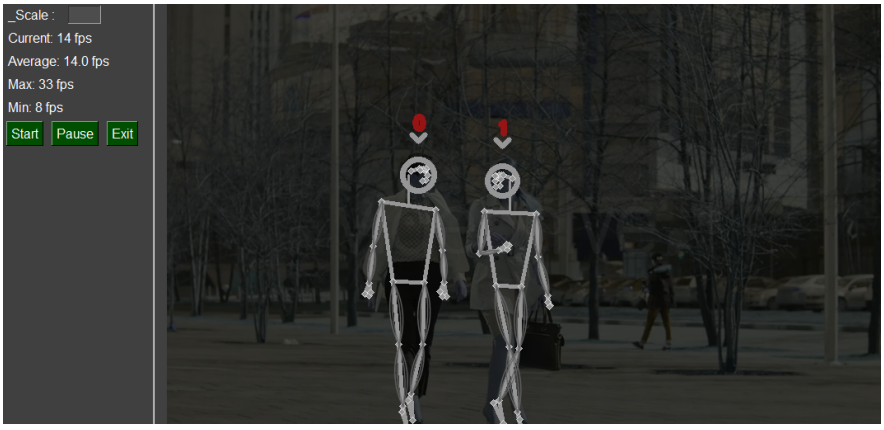

FIGURE 3.9: Our multi-person detection tracker after the implementation

3.4 Profiler and Collector

3.4.1 Body data object

The data object communicated throughout the analysis phase which represents what has been estimated and captured, is defined and initially received from the estimation phase, it encapsulates the body_id which is inherently the

estimator's index, pose and pose_3d which are the data collections containing the landmarks, and the segmented images which are a data object containing the image data and its frame count: head_region, torso_region, legs_region, finally vRatio describes the ratio of the detected skeleton's boundaries.

FIGURE 3.10: Body data object UML class model

3.4.2 Analysis distribution

Upon processing the incoming data object, the profiler channels the data_collection to the collector unit, in which the raw data is pre-processed and additional basic calculations, to minimize the overhead.

Similarly, the segmentation images are channeled to the unit responsible for invoking deep learning models, it runs the available models against the images, depending if an image exists or not, which is the subject of its visibility during the detection.

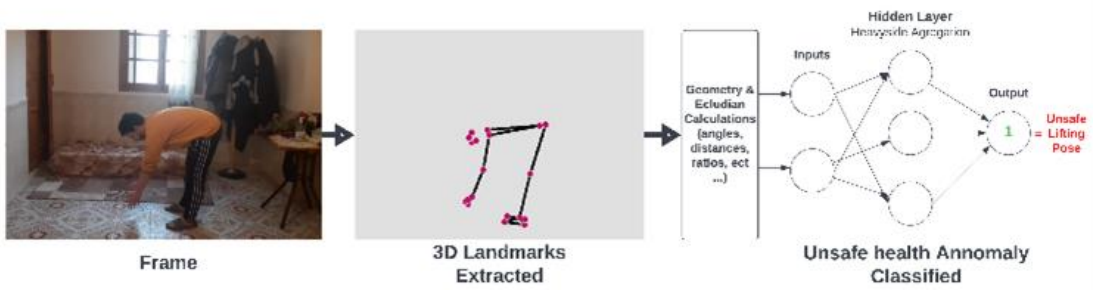

FIGURE 3.11: Our Supervised Neural Network pipeline example

FIGURE 3.12: Our Semi-Supervised Deep Learning pipeline example

3.4.3 Scheduling and Profiling

The analysis's main task doesn't wait for results from the image-based recognition but indeed relies on the collector unit to supply new data before stepping ahead.

As the Collector unit prepares the data and calculates attributes, it is then cached as packets in an array as a time series, managed as a FIFO queue. Representing what has been effectively collected from each scene from a single estimator.

Profiling then utilizes the Analysis unit to deal with this time series of packets, by running calculations involving attributes from across the provided timeline.

The profiler scheduler decides when to add into account results from image analysis depending on its availability and its relevance (if available and relevant, calculate with image recognition results as additional weights). These results are similarly stored as packets in a queue but independently managed since the profiler doesn't depend on them. The image recognizer has an additional queue for incoming images since it is not in sync with the main computation unit, and because the image batches are packets providing the frame count from which they are from, it decides to release the entire queue if it the analysis Frame_count exceeds recognition's Frame_count + queue.size(), effectively skipping to the most recent scene, ensuring the result's relevance doesn't become obsolete over time.

The training and testing process of an AI model to target more complicated behaviors is done and integrated into the architecture.

3.5 Modularity

Because of the use of a dependency injection design pattern around the main computation units, detection and analysis both are modular, this unlocks the possibility of injecting an extra object detection model before the skeleton estimation for a slightly different approach.

To showcase the powerful modular design of the system, we experimented with providing the scene in a bundle with pre-detected boxes of all the people in the scene, this need to alter the estimation scheduling mechanism, instead of passing the same scene between estimators. The Recognizer unit would decide on a fixed-size array of object areas and then pass them accordingly to each estimator, eliminating the need for synchronization entirely. this is the alternation phase of our MPSTA algorithm and we will not include it in the implementation nor validation.

The Figure 3.13 bellow shows a code snippet of the main execution point of our framework, in which an engineer could call modular components with specific configurations:

```
"""
H B P    P R O C E S S   P O O L
"""
import Kernel
import Gui
import os
import multiprocessing

# __ ________  P A T H S  ________ __

def initialize():
    """Initializer for Core, Creats Objects Pools and Pipeline Components,
    then runs them according to the general architecture

    Returns:
        List [Provider Prompt Queue, List : [DRAWER , RECOGNIZER , PROFILER]]
    """
    # __ ________  M O D U L E S  ________ __
    CONFIG = Kernel.Config_manager('/config.ini')
    _scale = 2 #CONFIG.read_scale()
    PROVIDER = Kernel.Provider()
    DRAWER = Kernel.Drawer(_scale)
    RECOGNIZER = Kernel.Human_Detector(_scale,DRAWER)
    PRFLR = Kernel.Profiler(_scale,RECOGNIZER,DRAWER)
    # ___________________________________

    PROVIDER.run(RECOGNIZER.subPipe, DRAWER.sceneQueue, "C:\\Users\\pc\\Desktop\\Repos\\1  HUM
    RECOGNIZER.pool()
    PRFLR.pool()
    DRAWER.run()
    os.system('cls')
    return PROVIDER.prompt, PROVIDER.exit_key, [DRAWER,RECOGNIZER,None]
```

FIGURE 3.13: Our framework's main execution point (Main Pool)

3.6 Flow-control based communication (IPC)

To provide a performant means of communication for a large number of processes running in parallel in the system, a low-level inter-process communication measure is implemented to control the flow of data between units and to bring core functionalities to a higher level of the system's application interface or a monitoring shell. It is built like semaphores, low-level data pipelines, that can leverage the full machine's memory speed and capacity, as well as the preprocessing mechanism to be compatible with both low and high-level platforms.

As shown in figure 2.10, this module is used whenever a variable amount of data needs to travel across the two platforms, for example, the video stream captured by Core being displayed in real-time in the Application interface to the user.

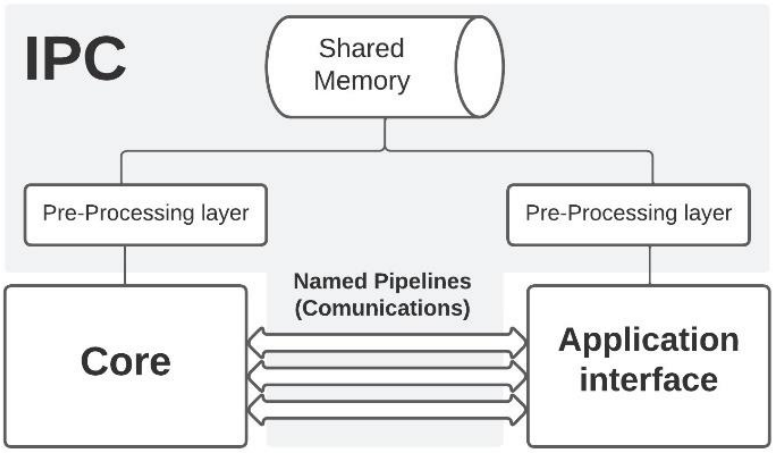

FIGURE 3.14: Our Inter-Process-Communication Design

3.7 Sub-services

In this sub-section, we describe the sub-services supporting our system, which can be an independent runtime putting stealer threads to work on the main memory allocated by the main system.

• Profile Drawer - Draws and provides visual annotation of the extracted landmarks directly to IPC Bridges.

• Video Recorder API - Available at runtime to record/store video stream, receives requests from any process, and is observed by the same requesting process.

• Config Manager - Works and is available to all phases and throughout the runtime to access the configuration database.

• Runtime API - Has access to all processes in the main pool, it is an interface to control the engine's STD outputs and Terminate processes at the end of runtime.

• Knowledge base manager - Lower-level Service class responsible for the AI Model Training process.

• Reporter-Bridge - Lower-Level Service class responsible for wrapping profile reports into alerts and readable information

3.8 Example behavior profiling implementations

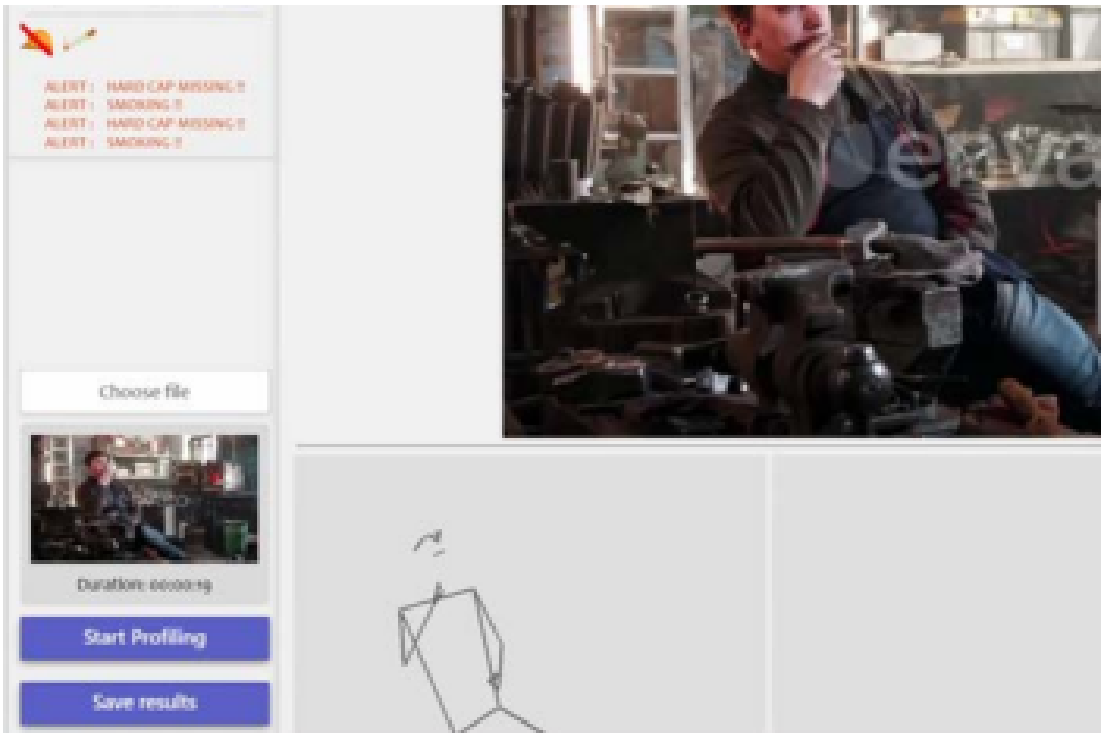

FIGURE 3.15: Our implementation of a smoking action/behavior profiler

FIGURE 3.16: Our previous implementation of a hard-cap wearing action/behavior profiler

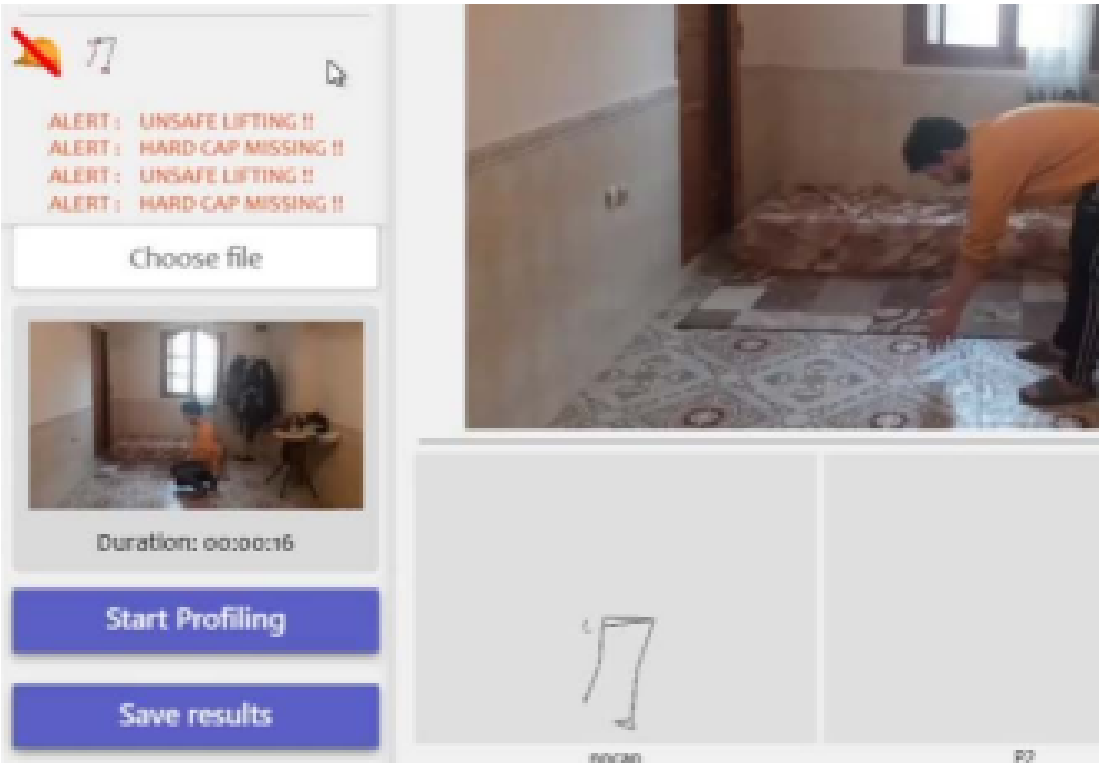

FIGURE 3.17: Our implementation of an unsafe object lifting pose action/behavior profiler

3.9 Conclusion

In this chapter we implemented a tangible framework representing our framework and we are ready for two validation experiments presented in the next chapter.

Chapter 4

Validation

4.1 Introduction

This section discusses the validation of the main contributions of our work.

It is clear that Our work doesn't break the HAR benchmarks when it comes to accuracy. In fact, only the unsupervised DNN leveraged in behavior profiling yields competitive results because of the advantage of pre-processed inputs. As Figure 4 shows.

The combination of DNN and calculation-based estimations, in powerful scheduling, synchronization, and modularity, though, defines the benefits of our approach.

4.2 Helmet detection, a hybrid simulation

To benchmark one aspect of our system, namely the object detection unit, we benchmark use [33] dataset, which has been previously experimented by its authors to prove its advantage over other safety helmet datasets as it contains fewer images with better labels and more classes, making helmet detection more accurate. The dataset has been tested by its authors on various state-of-the-art YOLO models which work to detect the helmet's position in a frame.

In our case, with the pre-processed condition the frame reaches our classification model, it is safe to say it would be a much easier task, hence the utilization of classifications instead of detection. With that in mind, we trained a simple Tensorflow image classifier, on the # training images of the benchmark dataset

and tested it against # of testing images, the benchmark results using darknet found peaked at in terms of accuracy.

Due to classification occurring inside our system's scheduling, we took the same testing part as the benchmark dataset and extracted the annotation labels manually.

The testing dataset was fed to our system as a series of frames to see how many matching cases were detected. This is rather a generalization of the benchmark and results were as follows: the annotated images contained 855 cases of persons without a safety helmet and 662 persons with a helmet, our classifier outputted 507 cases of persons without helmets and 495 with helmets at the end of the simulation run.

The results indicate two things, a high accuracy in helmet classification considering we also compared the individual portion frames and their annotations with output results and found a matching rate of 0.84 on average. It also indicates a slight failure in the number of detected persons to begin with, mainly due to many of the dataset images being cropped or not as closely or far away taken as a dedicated camera for surveillance purposes would capture.

Considering our embedded classifier is trained on a set of 300 images only, and a stable fixed speed (30 FPS) we see this as an effective gain in the direction we set this work to target.

4.3 Profiling Aggressive behaviour, simulation

Another powerful application possible to achieve in our system is behavior analysis. In this section, we will target Human aggressive behavior.

Monitoring aggressive behavior for the purpose of detecting to detect possible danger, is a difficult task. One way to handle it is a multi-factor combination, this can be easily done in our system by attaching the required modules to computation units, in this example we chose the following combination:

FIGURE 4.1: Aggressive behavior detection (annotation separated from the scene)

1. k-nearest neighbor algorithm, trained to classify Human interaction (Passive / Aggressive / No-interaction signs).

2. Keras sequential Classification Model, this model is delegated to the image-based recognition module in the analysis unit, so it's asynchronous to only occasionally provide results like the helmet detection example, it's trained on body parts of aggressive ppl (Agro/not Agro).

3. Body Motion monitoring, Motion Score extracted and calculated from storing landmarks over time (High, Moderate, Low). Motion=((x1-x2)*s) . Where s is a value chosen by the expert calibrating the profiler during experiments to express what is high from low motion, x is the point in the body where we are basing the motion calculation on.

4. A final layer of collective analysis, proximity calculation between bodies delegated to the main analysis unit to run calculations between currently collected body objects, and finally a violence score averaging, combining all the previously obtained results.

FIGURE 4.2: Aggressive behavior detection (annotation separated from the scene)

4.4 Conclusion

In this final chapter, despite the rarity of validation ways for similar studies, we have come up with two ways to showcase both our Algorithms and framework in action, the first addressed a helmet detection model training based on a dataset used by a non-HPE person and helmet detection, proving that training with a much smaller dataset and testing it on the used dataset yielded very close scores.

The second way, although less comparative to related works, proved our claims of a robust profiling multi-analysis approach utilizing multiple models and techniques, DL, ML, and Rule-based to achieve a surveillance analysis of aggressive behavior in humans.

General Conclusion

In this thesis, we presented an architectural approach for enhancing computer vision human surveillance systems. The detection of human-related anomalies, activities, and characteristics is an important element in controlling human error and incidents related to human behavior in workplaces. In this work, we proposed a new approach to leveraging Computer vision. The human Behaviour Profiler framework represents our proposed AI architecture that competes with State-of-The-Art technology.

To enhance the operability of the human detection system we looked at the current state of the art techniques and chose HPEs for the task for the benefits they offer. this choice meant we faced a bigger operability issue in performance, so we proposed the MPSTA Algorithm, leveraging tracking HPE feature of tracking a detection instance rather than continuously re-running detection every frame, combining it with our proposed steps of semi-continuous Distribution of de- tection and Alternation over sequence. We ended up with a performant and reliable extractor of human skeletons from real-time videos or camera feeds.

In every HAR-oriented system, the main goal is to not only extract human locations but also classify, detect, and understand their behavior and actions. From this notion we looked for a way to not only detect but also run profiling since we treat sequences of monitored people, MARPCRP Algorithm was our proposed method for multi-analysis to ensure the robustness lacking in the related works and fitting best DL and ML techniques to what they excel best from the wide variety of human actions and traits. As well as short-term result caching similar to FCNN algorithms and long-term result cashing for traits profiling and generating reports about what was being monitored.

We implemented both our algorithms with a notable exception: did not show the implementation of the alternation step of the MPSTA Algorithm and did not explain the implementation of most DL or ML models injected into our

MARPCRP Algorithm. (Will be published soon in a future research paper), this is also for validation purposes, since this work is a study rather than a production project. we showed screen captures of our detection phase and profiling phase with description of sub-components specifically developed for our experimental implementation. The results in experiments of the first phase, estimation of body pose landmarks in the processed scene allow for an easier task for further Deep Neural Networks running in the profiling pipeline by providing an abundance of Euclidean compatible data about the skeleton as well as precisely cut fragments of the same body.

Finally, his work is not only a research theme but can also be implemented in products that serve in profiling behaviors, a powerful, easy-to-use framework for safety regulation in industrial environments, gyms, hospitals, government installations, etc...., surpassing traditional CCTV by nature competing with state-of-the-art AI vision. Our framework succeeds in offering immediate, clear, and useful information regarding multiple persons at the same time, we call them profiles because each detected person is analyzed individually in parallel with the rest of the detection, thus, it is the worker's profile updated in real-time.

Bibliography

[1] M. Fowler, *Inversion of Control Containers and the Dependency Injection pattern*, https://martinfowler.com/articles/injection.html, [Accessed 27-05-2024], 2004.

[2] H. H. Pham, L. Khoudour, A. Crouzil, P. Zegers, and S. A. Velastin, *Video-based human action recognition using deep learning: A review*, 2022. arXiv: 2208.03775 [cs.CV].

[3] *An overview of artificial intelligence: Technologies and applications*, Accessed: 2024-06-06, 2024. [Online]. Available: https://www.ibm.com/topics/artificial-intelligence.

[4] *Major branches of artificial intelligence*, Accessed: 2024-06-10, 2024. [Online]. Available: https://www.analyticssteps.com/blogs/6-major-branches-artificial-intelligence-ai.

[5] L. A. DiMatteo, "Artificial intelligence: The promise of disruption," in *The Cambridge Handbook of Artificial Intelligence: Global Perspectives on Law and Ethics* (Cambridge Law Handbooks), L. A. DiMatteo, C. Poncibò, and M. Cannarsa, Eds., Cambridge Law Handbooks. Cambridge University Press, 2022, 3–17.

[6] S. Shinde, D. Medhane, and O. Castillo, *Applied Computer Vision and Soft Computing with Interpretable AI*, 1st. Chapman and Hall/CRC, 2023. DOI: 10.1201/9781003359456.

[7] T. S. Huang, "Computer vision: Evolution and promise," 1996. [Online]. Available: https://api.semanticscholar.org/CorpusID:67163270.

[8] A. Srvistava, "The future of computer vision: Envisioning a world where machines see and understand," *Forbes*, 2024. [Online]. Available: https://www.forbes.com/sites/abhinai/2024/06/06/the-future-of-computer-vision-envisioning-a-world-where-machines-see-and-understand.

[9] A. Betti, M. Gori, and S. Melacci, *Deep Learning to See: Towards New Foundations of Computer Vision*. Springer International Publishing, 2022, ISBN:

9783030909871. DOI: 10.1007/978-3-030-90987-1. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-90987-1.

[10] J. Buchanan, *Nhs forth valley: Fines for smoking outside hospitals as new law introduced*, [Online; accessed 30-May-2024], 5th Sep 2022. [Online]. Available: https://www.falkirkherald.co.uk/health/nhs-forth-valley-fines-for-smoking-outside-hospitals-as-new-law-introduced-3831422.

[11] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, https://pjreddie.com/darknet/yolov3/, 2018.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," pp. 886–893, 2005. DOI: 10.1109/CVPR.2005.177.

[13] G. (n.d.), *Pose landmark detection guide | mediapipe | google for developers.* https://developers.google.com/mediapipe/solutions/vision/pose_landmarker, [Accessed 5-01-2024], 2022.

[14] F. B. Klein and A. Cangelosi, *Human activity recognition from skeleton poses*, 2019. arXiv: 1908.08928 [cs.CV].

[15] H. Ramirez, S. A. Velastin, P. Aguayo, E. Fabregas, and G. Farias, "Human activity recognition by sequences of skeleton features," *Sensors*, vol. 22, no. 11, 2022, ISSN: 1424-8220. DOI: 10.3390/s22113991. [Online]. Available: https://www.mdpi.com/1424-8220/22/11/3991.

[16] J. Albusac, D. Vallejo, J. Castro-Schez, S. Schez-Sobrino, and C. Gómez Portes, "Multi-analysis surveillance and dynamic distribution of computational resources: Towards extensible, robust, and efficient monitoring of environments," *Expert Systems with Applications*, vol. 175, p. 114 692, Feb. 2021. DOI: 10.1016/j.eswa.2021.114692.

[17] D. Lea, "A java fork/join framework," *ACM 2000 Java Grande Conference*, May 2000. DOI: 10.1145/337449.337465.

[18] D. Leijen, W. Schulte, and S. Burckhardt, "The design of a task parallel library," OOPSLA '09, 227–242, 2009. DOI: 10.1145/1640089.1640106. [Online]. Available: https://doi.org/10.1145/1640089.1640106.

[19] *What is tokio*, [Online; accessed 30-May-2024], 15th Sep 2020. [Online]. Available: https://tokio.rs/tokio/tutorial.

[20] K. Thakral, *Sliding Window Technique*, https://www.geeksforgeeks.org/window-sliding-technique, [Accessed 13-02-2024], 2024.

[21] Wikipedia contributors, *Semi-continuity — Wikipedia, the free encyclopedia*, [Online; accessed 02-Feb-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Semi-continuity&oldid=1219551999.

[22] R. D. Blumofe and C. E. Leiserson, "Scheduling multithreaded computations by work stealing," *J. ACM*, vol. 46, no. 5, 720–748, 1999, ISSN: 0004-5411. DOI: 10.1145/324133.324234. [Online]. Available: https://doi.org/10.1145/324133.324234.

[23] S. Chen, P. Gibbons, M. Kozuch, *et al.*, "Scheduling threads for constructive cache sharing on cmps," *Annual ACM Symposium on Parallelism in Algorithms and Architectures*, Jun. 2007. DOI: 10.1145/1248377.1248396.

[24] M. Seemann and S. van Deursen, *Dependency Injection Principles, Practices, and Patterns*. Manning, 2019, ISBN: 9781638357100. [Online]. Available: https://books.google.dz/books?id=zzczEAAAQBAJ.

[25] *Visual studio 2022*, Accessed: 2024-06-10. [Online]. Available: https://visualstudio.microsoft.com/.

[26] *Visual studio code faq*, Accessed: 2024-06-10. [Online]. Available: https://code.visualstudio.com/docs/supporting/FAQ.

[27] *Why tensorflow*, Accessed: 2024-06-10. [Online]. Available: https://www.tensorflow.org/about.

[28] *Keras website*, Accessed: 2024-06-10. [Online]. Available: https://keras.io/.

[29] *Introduction to pandas and numpy*, Accessed: 2024-06-10. [Online]. Available: https://www.codecademy.com/article/introduction-to-numpy-and-pandas.

[30] *What is pandas*, Accessed: 2024-06-10. [Online]. Available: https://pandas.pydata.org/.

[31] T. S. P. P. Team, *Scholarly*, Accessed: 2024-06-07, 2024. [Online]. Available: https://pypi.org/project/scholarly/.

[32] A. Troelsen and P. Japikse, *Introducing C and .NET 6*. Springer, 2022.

[33] N. N. Jing, *Safety helmet (hardhat) wearing detect dataset(shwd)*, https://github.com/njvisionpower/Safety-Helmet-Wearing-Dataset, [Accessed 18-04-2024], 2019.

Abstract

This thesis presents a Robust Framework featuring two algorithms MPSTA and MARCPR, an innovative approach to optimizing automated human surveillance systems, which are often hindered by their inherent complexity and the rapid evolution of computer vision techniques. We introduce a human-behavior profiling system that addresses constraints architecturally, eliminating the need for synchronism. Our system outperforms continuous detection systems in terms of speed, it tackles each detection challenge using various computer vision techniques, thereby enhancing its efficiency and effectiveness. We propose a novel human behavior profiling approach, a system featuring a hybrid computer-vision kernel with dual execution times distributed architecture along with pattern extraction/employment, pre-processing techniques, and feature engineering. These methods have the potential to enhance profiling performance and accuracy in certain contexts, enabling the detection and analysis of both collective behaviors, such as interactions and conflicts among individuals, and distinct individual behaviors.

Keywords: Human Behavior Profiling System; Surveillance Systems; Computer Vision Techniques; Interconnected Constraints.

الملخص

تقدم هذه الأطروحة إطارعمل متين يضم خوارزميتين MPSTA وMARCPR, و نهجًا مبتكرًا لتحسين أنظمة المراقبة البشرية الآلية، والتي غالبًا ما يعيقها التعقيد المتأصل فيها والتطور السريع لتقنيات الرؤية الحاسوبية. سوف نقدم نظام وصف وتحديد السلوك البشري الذي سيعالج تلك القيود معمارياً، ويقدم تصميما قد يلغي الحاجة إلى التزامن في طرق معالجات البيانات. يتفوق نظامنا على أنظمة الكشف المستمر من حيث السرعة، فهو يعالج كل تحدٍ من تحديات الكشف باستخدام تقنيات الرؤية الحاسوبية المختلفة، وبالتالي تعزيز كفاءته و وفعاليته. بهذا نقترح نهجًا جديدًا لتنميط ووصف السلوك البشري، وهو نهج يتميز بنواة رؤية حاسوبية هجينة ذات أوقات تنفيذ مزدوجة موزعة إلى جانب استخدام الأنماط وتقنيات المعالجة المسبقة و وهندسة السمات. هذه الأساليب لديها القدرة على تعزيز الأداء والدقة في السياقات المدروسة، مما يتيح اكتشاف وتحليل كل من السلوكيات الجماعية، مثل التفاعلات والنزاعات بين الأفراد، والسلوكيات الفردية، مثل التدخين وغيره من التصرفات.

الكلمات المفتاحية: نظام تحديد السلوك البشري؛ أنظمة المراقبة؛ الرؤية الحاسوبية، التقنيات؛ القيود المترابطة.