D4.5 Video Surveillance Techniques: Final Release

Work Package:	WP4
Lead partner:	2-CERTH
Author(s):	Panagiotis Giannakeris, Stavros Paspalakis, Alexandros Petropoulos, Konstantinos Ioannidis, Ilias Gialampoukidis (CERTH), Antonis Chronakis (ACCELLI)
Due date:	Month 20
Version number:	1.0 **Status:** RELEASE
Dissemination level:	Public

Project Number:	883284 **Project Acronym:** 7SHIELD
Project Title:	Safety and Security Standards of Space Systems, ground Segments and Satellite data assets, via prevention, detection, response and mitigation of physical and cyber threats
Start date:	September 1st, 2020
Duration:	30 months
Call identifier:	H2020-SU-INFRA-2019
Topic:	SU-INFRA01-2018-2019-2020 Prevention, detection, response and mitigation of combined physical and cyber threats to critical infrastructure in Europe
Instrument:	IA

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883284

Revision History

Revision	Date	Who	Description
0.1	18/03/2022	CERTH	First release of the template. ToC and assignments finalisation.
0.2	19/04/2022	CERTH	Added Object detection contribution
0.3	19/04/2022	CERTH	Added contribution for face recognition
0.4	27/04/2022	CERTH	Addressed comments of 1st reviewer
0.5	27/04/2022	CERTH	Addressed comments of 2nd reviewer
0.6	29/04/2022	CERTH	Final proofreading. Fixed acronyms, references, captions, document styles.
1.0	29/04/2022	CERTH	Final release

Quality Control

Role	Date	Who	Approved/Comment
Internal review	27/04/2022	ENG	Document accepted; only minor changes suggested
Internal review	27/04/2022	ACCELI	Document accepted; only minor changes suggested

Disclaimer

Executive Summary

The purpose of D4.5 is to present the final solutions chosen for face detection and recognition, object detection, activity recognition and object detection at the edge. Initially, an overview of the finalized user requirements is presented which, along with the key result objectives, are used as a guideline for the implemented solutions. Any recent related work not mentioned in previous version (D4.1) is also included, along with a brief introduction of all the relevant algorithms whenever considered necessary. A detailed presentation of the models, algorithms and techniques that were developed then follows along with the reasoning behind each choice and the benefits and compromises that were made. The evaluation experiments that were carried out throughout the development process and the produced results, both quantitative as well as qualitative, are also reported here in order to explore the applicability of the tools. The performance of the algorithms has been evaluated in terms of their accuracy (mean Average Precision) and detection latency (frames per second) as described in DoA (KPI 2.2.1).

Table of Contents

List of figures

List of Tables

Definitions and acronyms

AP	Average Precision
AR	Activity Recognition
CCTV	Closed Circuit TV
CNN	Convolutional Neural Networks
COCO	Common Objects in Context
DCEP	Data Collection and Edge Processing
DSFD	Dual Shot Face Detector
FCNT	Fully Convolutional Networks Tracker
FD	Face Detection component
FDDB	Face Detection Data Set and Benchmark
FDR	Face Detection and Recognition component
fps	frames per second
FR	Face Recognition component
GCEP	Geospatial Complex Event Processing Engine
IoU	Intersection over Union
KCF	Kernelized Correlation Filters Tracker
KR	Key Result
LFW	Labelled Faces in the Wild
MOT	Multiple Object Tracking
NOA	National Observatory of Athens
ODE	Object Detection at the Edge component
PFE	Probabilistic Face Embeddings
PUC	Pilot Use Case
SORT	Simple Online and Realtime Tracking
SRN	Selective Refinement Network
SSD	Single Shot Detector
UAF	United Alert Format
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
UR	User Requirement(s)
VOD	Video-based Object Detection component
WP	Work Package
YOLO	You Only Look Once

1. Introduction

1.1. Scope

The 7SHIELD project aims to be an integrated framework for providing cyber-physical protection services of ground segments. The proposed critical infrastructures protection solutions can be integrated within the existing legacy systems or function autonomously. Work Package 4 (WP4) aims at providing crisis management tools for the detection on cyber and physical threats. In Figure 1-1 the general architecture of 7SHIELD is presented. The technologies included in this WP are Face Detection and Recognition, Object Detection at server or at the Edge, Activity Recognition, cyber-attack detection methods, infrared and thermal image processing for man-made disasters, ground and aerial thread detection via laser-based technologies and finally a combination of cyber and physical early warning mechanism.

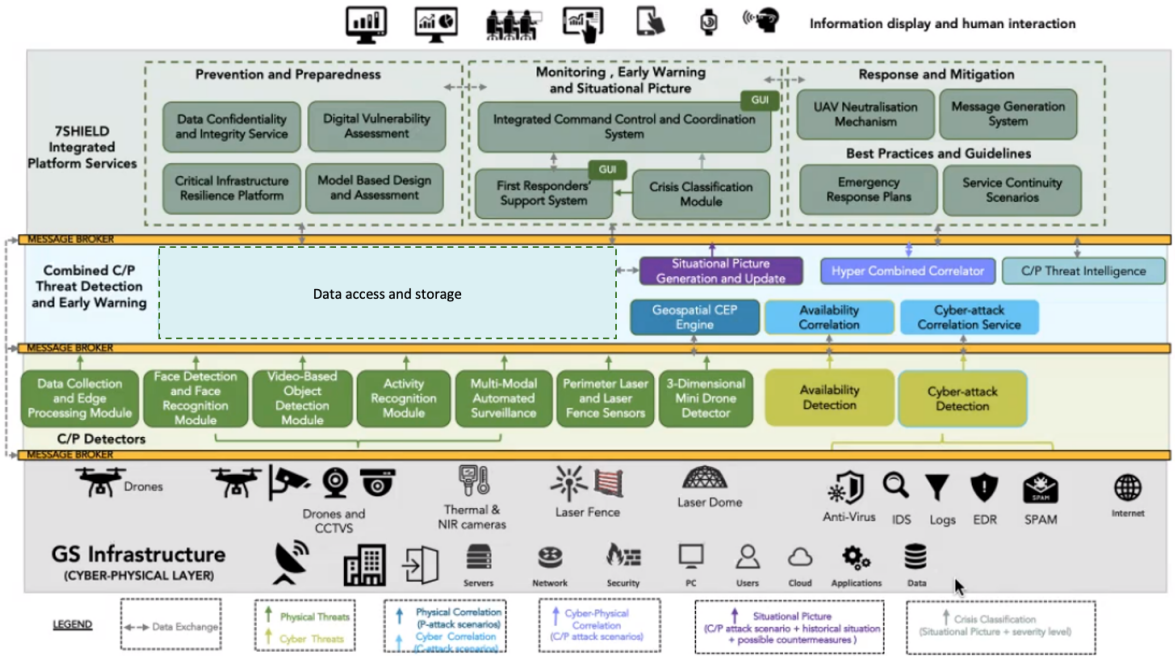

Figure 1-1 - General architecture of 7SHIELD

This deliverable covers two tasks, namely ***Task 4.2: Face Detection and Recognition*** and ***Task 4.3: Object Detection and Activity Recognition.*** Task 4.2 aims at providing solution for early detection and recognition of unauthorised persons in specific areas. This task is mapped to the Key Result ***KR06: Face detection and face recognition module***. Task 4.3 on the other hand has as objective to detect all object of interest in the provided input and also evaluate the relevant action any detected person performs. This task is mapped to the Key Result ***KR07: Video-based object and activity recognition***.

1.2. Deliverable structure

This deliverable consists of the following sections:

- In Section 1, the scope of the deliverable along with a mention to the contained tasks and references to other activities and documents are presented.
- In Section 2 is dedicated to Face Detection and Recognition. After presenting the final user requirements and objectives connected with them, the architecture, data sources and an evaluation of the public datasets are described. Next, the final version of the component's architecture is presented and the developed face detection and recognition techniques are described in detail along with evaluation and results.
- Section 3 is dedicated to Object Detection and Activity Recognition. A similar approach is followed which includes a brief presentation of user requirements and objectives, followed by a related work presentation (which adds any new method not mentioned in previous version) and finally, it concludes with the presentation and analysis of all implemented algorithms in Object detection at workstation and at the edge and activity recognition.
- Finally, Section 4 wraps up our findings and conclusions are drawn about the developed 7SHIELD video-based surveillance techniques.

1.3. Reference to other activities and documents

This deliverable is the main result of the activities carried out under Task 4.2 and Task 4.3 and stands as a continuation of deliverable ***D4.1 - Video Surveillance Techniques: Initial Release***, which has been submitted on M8 of project's lifespan. This deliverable is also connected with the EURES deliverables ***D4.2 - 7SHIELD Combined Physical and Cyber Threat detection*** and ***D6.2 – System integration and interoperability v1 (1st prototype)*** released at M9 and M10 respectively, which contain details concerning the messages to be exchanged between modules in the 7SHIELD ecosystem.

Furthermore, a relation with the ***Task 4.1: Data collection from UAVs and processing at the edge*** has been established based on the embedded artificial intelligence algorithms that are going to be fully embedded into the 7SHIELD Unmanned Aerial Vehicle (UAV). Details on that had been mentioned at deliverable ***D4.3 - Data collection from UAVs and processing at the edge techniques*** which released at M14 and also contained in this deliverable in 3.3.3.1.

2. Face Detection and Recognition

2.1. Introduction

2.1.1. User & Functional Requirements

Following the second round of End User Requirements (UR) during the end user workshop, several requirements were examined regarding their relevance to KR06 which is the main KR produced by Task 4.2. The final list is presented in Table 2-1 each one with its assigned priority, description and Pilot Use Cases (PUC).

ID	Type	Priority	Description	PUC	Crisis Management Phase
FR_SCE_04	Functional	MUST	7SHIELD must use facial recognition to identify target as authorized or unauthorized person.	PUC1, PUC2, PUC3	Detection
FR_SCE_22	Functional	MUST	IT personnel must receive an alert on the dashboard in case of unauthorized access to a Data Centre or antenna shelter.	PUC2	Detection
FR_SCE_23	Functional	MUST	IT personnel must receive a second alert from the 7SHIELD platform confirming unauthorized access to Data Centre in case face recognition system cannot match person with authorised personnel.	PUC2	Detection

Table 2-1 - Final UR for KR06

2.1.2. Objectives

All the final requirements are functional and of highest priority. The related URs are discussed below in the context of the KR06 objective:

- Functional requirement FR_SCE_04 states "7SHIELD must use facial recognition to identify target as authorized or unauthorized person" which is well aligned with the

description of KR06, thus, it was recognized as the main UR which will be fully supported by KR06. Its description is straight forward setting the ultimate goal of the module which is to detect and characterize targets that appear in CCTV streams as authorized or unauthorized.

- Functional requirement **FR_SCE_22** is related to the alerts that KR06 will produce which will be fed to other 7SHIELD modules, specifically the higher-level correlators and will ultimately reach the operators after correlation with other cyber or physical threats.
- Functional requirement **FR_SCE_23** is similar with **FR_SCE_22** and refers to the alert system in 7SHIELD which should be supported by KR06 when matching of the detected faces cannot be performed successfully with authorized personnel.

2.2. Architecture, Data Sources and Outputs

2.2.1. Architecture

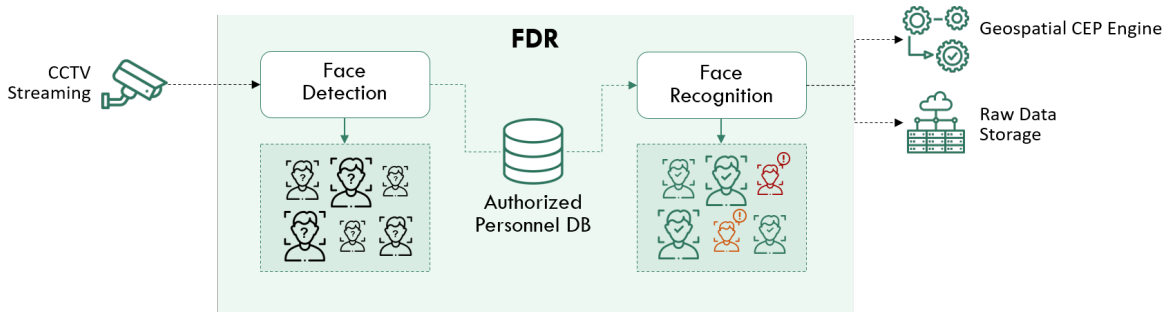

Figure 2-1 - 7SHIELD FDR architecture and connectivity diagram

The Face Detection and Recognition (FDR) component architecture is shown in Figure 2-1. The system is initialized with a video stream, but it also can support on-demand analysis of video files in an offline processing scenario. The main components of FDR have not been altered from the previous version and still remain the Face Detection (FD) and Face Recognition (FR) components.

Processing begins on the FD component. FD is responsible to detect patches inside the input frame where faces are tightly enclosed. The acquired face patches are instantly characterized as *unknown* and are immediately provided to the FR component for further processing. FR cooperates with a pre-existing gallery of *known* faces and is responsible to decide which of the *unknown* faces can be matched and with what certainty with any *known* face from the gallery. *Known* individuals can be either (a) authorized personnel expected to appear in the area covered by the CCTV camera, or (b) unauthorized individuals which are not allowed to be in this location. This information, along with photos depicting the *known* faces must be previously defined during the initialization of the FDR module and the gallery creation.

The output of FDR in the final version besides being stored in the same local storage as the deployed module, is forwarded through the 7SHIELD message bus to the Geospatial Complex Event Processing Engine (GCEP) which is responsible to combine physical and cyber threats. After the recognition process, a detailed report can be produced with the detection and recognition details and is represented with the 7SHIELD United Alert Format (UAF) format. The produced UAF alert from FDR have been initially established in deliverable D4.2. The UAF alerts of FDR include information for the camera location and type in the "Analyzer", "Sensor" and "Target" fields as well as the actual analysis results in the "Vector" field. Each item in the "Vector" array represents an unauthorized entity (person) from a portion of a video or a streaming sequence that has been detected by the FDR module. The entities may be linked with several attachments, depending on how many face instances were detected and found to belong to each particular entity by the FDR module. The "Observable" array provides for each instance additional details related to the detection confidence of FDR. The UAF alert is enclosed to the dedicated *DL.FDR.UNAUTHORIZED* Kafka topic and is send through the message bus. More details about the message bus can be found in deliverable D6.2.

2.2.2. Data Requirements

Usually, ground stations have to monitor specific key areas like server rooms or other closed spaces with important equipment. The most effective way to monitor who has access is by placing CCTV cameras at all the possible entrances of the monitored space. Each space may be associated with its own list of individuals who are authorized to enter.

In order for the system to recognize the authorized individuals that are allowed to enter a restricted area, a photo gallery of faces of the authorized persons must be assembled. The gallery must contain pictures of the authorized personnel from various perspectives. The collected photos are used in order to match the detected faces with authorized personnel. In order to create the gallery specific requirements, have to be met when capturing the photos. Several photos of the known individuals' faces must be collected. To achieve maximum performance and robustness to pose variations each person must capture at least 10 pictures as shown in Figure 2-2.

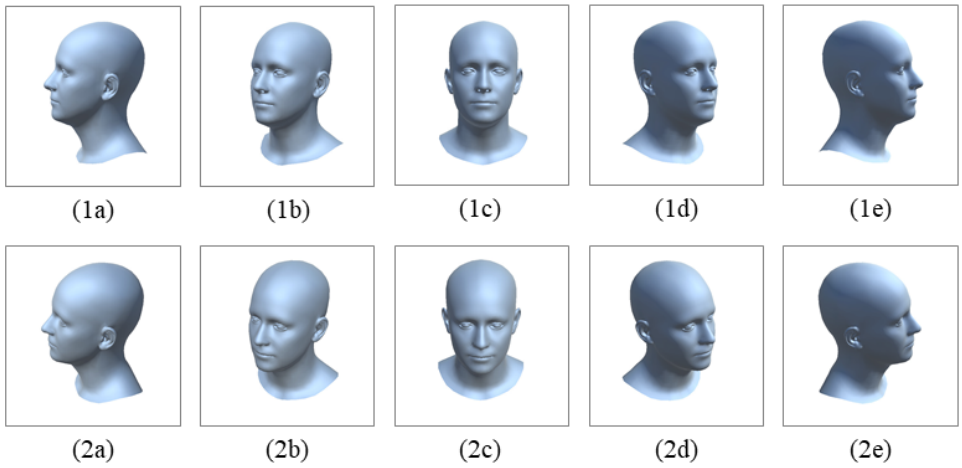

Figure 2-2 - FDR gallery photos with different head pose and camera angles.

Each person is asked to capture two (2) sets of photos with a different placement of the camera in each set: (a) the camera pointing straight at the person, height close to shoulder level, and (b) the camera pointing straight at the person, height above shoulder level - simulating the installed camera placement in the stakeholder's premises (person not looking at the camera).

A minimum of five (5) photos for each set (a total of 10 photos) that clearly show the person's face in a neutral facial expression must be provided:

a) One (1) photo with the face looking at a -90-degree angle (full left profile).

b) One (1) photo with the face looking at a -45-degree angle (half-left profile).

c) One (1) photo with the face looking straight in front.

d) One (1) photo with the face looking at a +45-degree angle (half-right profile).

e) One (1) photo with the face looking at a +90-degree angle (full right profile).

The faces must be illuminated adequately with natural or ambient light, so that all facial features are captured properly. The distance of the camera from the face must be moderate, e.g., close to 1-2 meters depending on the lens. The person's shoulders must be inside the frame. Extreme close ups and adding smartphone filters as well as other distortions must be avoided.

Note that the provision of other personal information is not required by FDR in order to perform any type of analysis in the scope of 7SHIELD and the photos are not linked with any other personal or identifiable information. The default policy is to store the data inside the deployed FDR module and to pseudonymize the photos, meaning that the pictures are assigned numerical identifiers (e.g., "person_0" or similar). Similarly, the products of FDR do not reveal real persons' identities, but only the event that is taking place, that one or

more unauthorized persons have been detected, and the detections are also assigned numerical identifiers.

2.2.3. Public Evaluation Datasets

As far as existing benchmarks, we have presented in the previous release of this deliverable several datasets available for face detection and face recognition which are publicly available. Some of them are dedicated to the detection or the recognition task, while others offer images with metadata suitable for both tasks.

Dataset	Task	# faces	# people
LFW [8]	Face Recognition	13233	5749
WIDER FACE [20]	Face Detection	393703	-
FDDB [9]	Face Detection	5171	-
Chokepoint [19]	Face Recognition	64204	29

Table 2-2 - Summary of face detection and recognition datasets used. The number of annotated face instances as well as the number of individuals depicted are reported.

Table 2-2 provides a summary of the face detection and face recognition datasets that were acquired and used in order to evaluate the performance of the models. For a dataset made for face recognition (or face verification), the number of face instances which appear in the images as well as the number of people that make up that dataset is given. In the case of face detection, the number of faces means the number of annotated bounding boxes in the entire collection of images (more than one face may exist in a single image), while the number of people is not given since it is irrelevant.

The Chokepoint dataset [19] is a new addition in this release. It was used to produce a demo video for FDR during preparations for the mid-term review. The advantage of this dataset is that it provides videos from a fixed camera which shows people walking towards an entrance of a room which is presumably monitored. This scenario suites the use cases of FDR in the 7SHIELD project very well and thus the dataset was acquired for testing.

The video dataset was designed for experiments in person identification/verification under real-world surveillance conditions. It includes footage from an array of three cameras placed above several portals (natural choke points) that capture subjects walking through each portal. The captured faces have variations in terms of illumination conditions, pose and sharpness. The dataset consists of 48 sequencies with a total of 29 subjects (23 male and 6 female) walking through the choke point. The videos were captured using a frame rate of 30 frames per second (fps) and a resolution of 800X600 pixels.

2.3. Final Algorithms and Results

Figure 2-3 shows the final version of FDR internal components' architecture to be described.

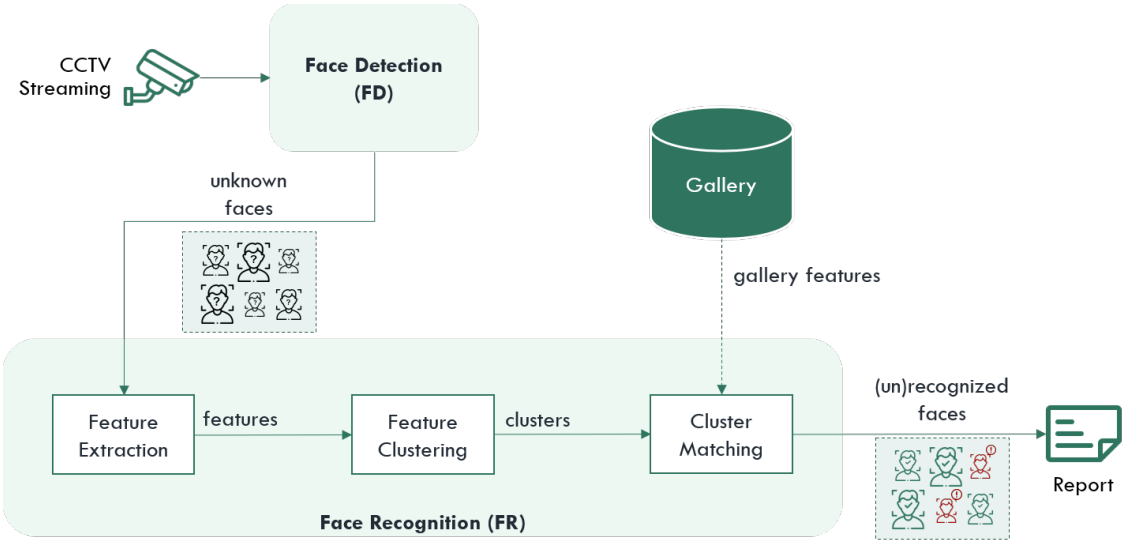

Figure 2-3 - 7SHIELD FDR final version

2.3.1. Face Detection

The FD component is responsible to detect all the faces in the current video frame. Usually, this is the heaviest task of the FDR pipeline, since state-of-the-art object detection is achieved using very deep pre-trained Convolutional Neural Network (CNN) architectures that carry out multiple tasks at once: (a) filtering the input and feature extraction, (b) candidate box proposal generation, (c) binary classification of candidates (face or no face) and (d) box pixel coordinate regression. An assortment of state-of-the-art object detection techniques has been presented in detail in the literature review of D4.1.

The evaluation metric used for face detection in the literature is Average Precision (AP). It is taken by calculating the area under the Precision-Recall curve. Precision is defined as the proportion of true positives out of all the detected faces and Recall as the proposition of true positives out of all the annotated faces. In other words, precision measures the accuracy of the detector and recall measures its ability to retrieve the existing faces. Whether a bounding box detection counts as a true positive is decided based on its overlap with a ground truth box. The overlap is measured by the Intersection over Union (IoU) metric. Thus, detected faces must have a good alignment with true faces in order to be considered correct. The precision and recall metrics are calculated for every IoU threshold (from most relaxed to most strict) to draw the Precision-Recall curve.

Face Detection Method	Accuracy @WIDERFACE (AP %)	Speed of Detection @1080p (fps)
TinyFaces (2017) [7]	90.7	1.8
PyramidBox (2018) [16]	94.3	4.3
DSFD (2019) [11]	95.5	4.2
SRN (2018) [2]	94.8	2.6

Table 2-3 - Performance evaluation of face detection models.

In this iteration we enriched our evaluation with another state-of-the-art face detection model, namely Selective Refinement Network (SRN) [2]. This model introduced two-step classification and regression operations selectively into an anchor-based face detector to reduce false positives and improve localization accuracy simultaneously. Its performance is equally impressive (over 94%) as the other three models, however DSFD remains the highest performing detector in WIDER Face dataset by a small margin (Table 2-3).

In addition, we measured the speed of detection for all models using the fps@1080p metric which indicates how many video frames of size 1080p (1920 pixels horizontally and 1080 pixels vertically) the model can analyse in a single second. In this regard, however all models perform poorly, bellow 5 fps. When considering frame-by-frame processing such speeds are nowhere near real time detection. This just proves how heavy the task is however and is not affecting the performance of FDR which still runs in real time speed.

Usually, video footage is streamed at rates close to 25-30 fps. None of the four models tested are capable of real time frame-by-frame object detection, since the incoming frames are arriving at a faster rate than the processing rate of the model. This simply means that frames that arrive between processing times are dropped (skipped) until the model is available again to process another frame. However, this is not a problem since monitoring cameras, and especially stationary ones, can capture a single person multiple times in subsequent frames. Specifically in the case of DSFD, the model will process 4 frames of the total of 30 that will arrive within a second, which means that frame dropping will not happen for more than 6 or 7 consecutive frames. It is impossible for a person to move in and out of the camera footage in less than 0.23 seconds, as such each person's face will be analysed by the detector for multiple frames despite the frame dropping effect.

2.3.2. Face Recognition

FR is responsible to take input from FD in the form of cropped portions of images or video frames, each one containing a single face, and characterise them as being known (authorized) or unknown (unauthorized). In the initial version of the component each

detected face was processed independently from the others using a retrieval pipeline where each query face was first analysed to extract facial features and was then matched against all the faces (images) from the gallery. Then, based on whether the Euclidean distance between the query's feature vector and the one from the closest gallery match exceeded a predefined threshold or not, the detection was marked as unknown or known. In general, this is a widely accepted process in the facial recognition literature regarding face verification or identification tasks. Face identification is closely related to classification, meaning that there is a model which has been trained to associate facial features with specific labels, i.e., the persons' identities. Face verification on the other hand refers to the task of deciding if two faces belong to the same person without knowledge of who the person is. This can be done with the process described before: (a) feature extraction, which provides the representation of the face images, (b) feature matching, which calculates the distances between feature vectors and (c) distance thresholding, which decides the outcome. The face verification task is more similar to what the FDR module in 7SHIELD is expected to do. In this process, the analysis of the faces is caried out by the facial feature extractor. This unit is using deep CNNs which have been trained to extract high level features such that those can uniquely represent the face of each person even in extreme variations of pose, camera angle, occlusions, or bad image quality. The performance of feature extraction has a very high impact on the performance of the system since all decisions are taken on the basis of higher similarity between facial feature vectors that belong to the same person.

2.3.2.1. Facial Feature Extraction

For facial feature extraction we have adopted the CNN FaceNet architecture [14], which produces high quality face representations and is designed to exhibit robustness to variations imposed by shot angle, face part occlusion and luminance variations. Based on our initial evaluation of feature extractors, which is shown in Table 2-4, FaceNet exhibits exceptional performance in LFW dataset as well as the other two evaluated methods with very little margin between them. The evaluation metric for face verification is Accuracy. The dataset is split to 10 equal parts, where the first 9 are used for cross validation to select the optimal distance threshold to achieve top accuracy. The 10th part is the test set from which pairs of queries are given and the model decides if they belong to the same person or not.

Face Recognition Method	LFW Accuracy (%)
Facenet (2015) [14]	99.4
PFE (2019) [15]	99.6
Arcface (2019) [3]	99.7

Table 2-4 - Performance evaluation in face verification task.

The FaceNet model has been trained on a set of large-scale images that include faces that are labelled with the person to whom they belong. The goal of learning is to extract features for each individual's face such that they uniquely represent it, making it distinguishable from faces of other individuals. In each cycle of training, after features are extracted by a forward pass through the neural network, they are fed into triplets inside an objective function. The objective function's purpose is to minimize the Euclidean distance between two feature vectors of faces the belong to the same individual and at the same time maximize the Euclidean distance between two feature vectors of faces of different individuals. The Facenet model was obtained pre-trained on the face recognition dataset VGGFace2 [1], which contains images of thousands of personalities. Note here that the training dataset contains a very large amount of face images in order to generate a sufficiently large number of triplets capable of training the model.

2.3.2.2. Gallery Size Impact

In order to examine how the authorized personnel gallery may affect recognition speed, the recognition process was repeatedly executed by incrementally increasing the gallery size in each step. In this experiment, the full-size gallery was comprised by images from a selection of 96 individuals, from the LFW dataset, with at least 10 images (the first 10 to populate the gallery and the other 5 for testing). The minimum gallery size was set to two randomly selected persons, and the initial set was 20 photos. The gallery was then enriched with one additional identity in each cycle using random order. In order to measure the performance, the face verification task was performed in each cycle in randomly ordered images of the remaining identities from the LFW dataset and the average single sample recognition time was recorded. Figure 2-4 (left) shows that the execution time for a single face increases linearly with the gallery size and that the maximum time reached was 1ms which can be considered negligible compared to face detection speeds. Finally, in case of batch processing (i.e., an image that contains multiple faces), the execution time is affected by the number of faces as shown in Figure 2-4 (right).

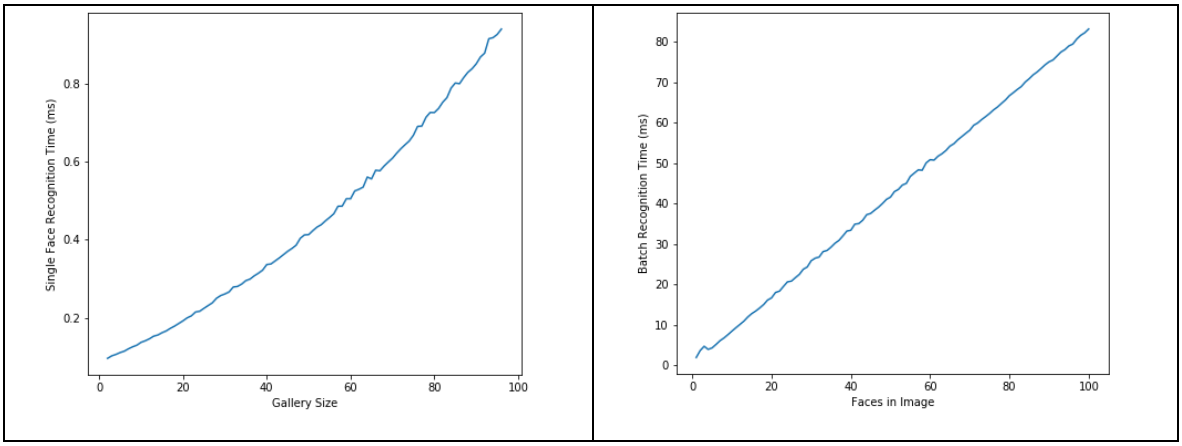

Figure 2-4 - Gallery size impact in face recognition speed.

2.3.2.3. Challenges of the Previous Release

So far, we have described the face verification task and how it can be used to process the detections independently, one after another, as well as the core processing unit that enables this process by producing face representations which are then manipulated appropriately. This type of processing is also mostly apparent in the literature of face recognition and is a simplified approach best suited for model evaluation. In a real application however, where we process online video streams, a large number of instances that belong to the same person may appear multiple times in an input video segment through sequential video frames. It is expected that if a person stays in front of the camera even for a single second, FD should detect at least four face boxes of the same person based on the measured detection speed. This can result to an excessive number of samples to pass through the FR component, which will be tasked to process and produce a decision for each one, even though it is certain that many will belong to the same person. However, it is enough if an unknown person is detected once to produce an alert. In addition to the added overhead, multiple alarms for the same unknown person may be produced.

Moreover, deciding on each detected instance of a person's face provides no means of information aggregation and is non-intuitive. As a result, it may also be a process that is sensitive to outliers. Depending on the characteristics of the monitored space the detected face instances may be captured with high variation in terms of illumination or occlusions, due to unevenly distributed light sources in the space or the existence of obstacles between the captured faces and the camera respectively and especially closer to frame borders (e.g., in the case that a face has not completely entered the frame yet). Some of these instances are good candidates for outliers and may cause false positives or false negatives and making decisions about whether a person is known or unknown from a poorly illuminated face or a face which is half occluded for example will certainly lead to errors and false alerts.

2.3.2.4. Facial Feature Clustering

To solve the problem of the added overhead of FR and to lessen the impact of outliers the final version of the component is equipped with a facial clustering mechanism, in order to cluster similar faces together and base decisions on clusters instead of single samples. The processing pipeline of FR has been adapted accordingly to support clustering and now consists of four main functions (a) facial feature extraction, (b) feature clustering, (c) cluster matching and (d) distance thresholding. After feature extraction, the extracted feature vectors are stored over time and then clustered at regular intervals in order to obtain groups where each one encloses features of the same person. It is not guaranteed that all face instances of a single person will be assigned to a single group, however the purpose of clustering is not to discriminate between persons (i.e., there may be more than one cluster with instances from the same person), but to provide clusters of samples into the matching process instead of single samples.

After extracting features for all faces in a set of frames these are given as input to the clustering unit. Hierarchical clustering is a general family of clustering algorithms that develops groups hierarchically by sequentially merging or splitting them. Agglomerative clustering is a hierarchical clustering technique using a bottom-up approach where each observation starts in its own cluster and the clusters are merged sequentially with each other in an iterative process. The linkage criterion defines the metric that determines the merging path. In particular, maximum or complete linkage minimizes the maximum vector distance between any two observations of a cluster and is used in our clustering algorithm to guarantee a safe distance between features is not surpassed inside a single cluster. In theory, since the feature extractor has been trained to maximize the Euclidean distance between samples of different individuals, it is sufficient to set an upper threshold of Euclidean distance that should verify the complete linkage criterion for all clusters.

The clustering result depends on the choice of an appropriate distance threshold for which the linkage criterion is satisfied for all clusters. By varying the distance threshold, the user can obtain clusters of different coherence. A low threshold creates clusters where the features between samples show high similarity as they should be very close to the Euclidean hyperspace of features. Conversely, a higher threshold creates clusters with looser requirements on the similarity of features internal to the cluster. It is undesirable to set the threshold either a too low or too high as, the former sets too strict similarity criteria and will lead to an overestimation of the number of individuals, while the latter sets too loose similarity criteria and will lead to instances of different persons to be merged into a single cluster. Therefore, the optimal threshold value is not known a priori but can be based on experimental results. Empirically, by observing the cluster outcomes on a number of runs in the ChokePoint dataset it has been set to 1.15, which is enough to provide coherent face clusters.

The final two steps of FR have been replaced with cluster matching and distance thresholding. Instead of matching single instances, the average Euclidean distance between samples inside a cluster and gallery entries is calculated in order to match detected face clusters with authorized identities from the gallery. In this way, even if an outlier is present inside the cluster, the matching depends on the average distance of all samples and its influence is lessened. The final step has been adapted from the previous release to characterise each face cluster as unauthorized or authorized depending on whether the average distance exceeds the predefined threshold or not. If an unauthorized decision is declared, the cluster's instances are reported appropriately inside the alert format in the UAF message.

3. Object Detection and Activity Recognition

3.1. Introduction

3.1.1. User Requirements

Since the previous version of User and Functional Requirements there has been a new release. Nevertheless, in the aspects that involve the Object Detection and Activity Recognition there were no additional requirements and, thus, the ones reported in the previous version of the document are the ones being valid. In this aspect and in order not to repeat unnecessary information we omit the inclusion of the same Requirements and address the interested reader to the previous version of the deliverable (D4.1 Video Surveillance Techniques Initial Release).

3.1.2. Objectives

This module comprises two separate subtasks: a) object detection and b) activity recognition. They are closely connected to each other and in fact the latter has been defined as an expansion of the former. The main outcomes of these subtasks are the modules that named as Video-based Object Detection (VOD) and Activity Recognition (AR). The choice for defining AR as a VOD extension is the UR which involved many requirements related to objects movements and, thus, directed the AR to this direction. The general idea is that AR will be combined with VOD and increase the whole awareness of the system.

The general objective for the first subtask still remains the development of a framework for identifying and localizing object of interest in the provided video streams. For specific objects the second subtask is also applicable and the results of the first subtask are propagated to the AR submodule.

Since the same UR are applicable to this module, the focus is mainly on the detection of objects along with their relevant movements. These are included in UR FR_SCE_02, FR_SCE_08, FR_SCE_14. As mentioned before not all detected objects are of the same importance and, thus, for AR we only focus on human activity recognition. Other objects are not included in the latter submodule. So, only movement of persons is analysed in this subtask.

As mentioned before, deriving from the UR a list of objects of interest have been compiled which include a generic person class, and the most common ground transport vehicle: car, truck, bus and motorcycle.

Besides the movement detection we have also incorporated in our modules the ability to control zone crossing in the Video analytics rules set at FR_SCE_76 UR. In order to set this requirement, we used fixed cameras which do not change the scenery they capture and, thus, the zone can be safely defined.

3.1.3. Architecture

The module plays a crucial role in the 7SHIELD architecture as it is a critical part of the modules for detecting physical threats. Object detection is responsible for automatically translating objects, via detections, in visual input into meaningful messages to the project pipeline. It is, thus, connected to various other modules of the project which involve:

- The GCEP, as the responsible module for retrieving the correlation between geospatial events.
- The Crisis Classification Module, to determine whether there is a crisis situation present.
- The 7SHIELD Knowledge Base with purpose of storing information in order to be available to other modules.
- And, finally, the Raw Data Storage which is aimed at storing data to be used for visualization purposes from VOD and AR.

3.2. Related Work

3.2.1. Object Detection

As it has been mentioned in the previous version of the deliverable object detection algorithms are typically separated into two distinct categories:

- single-phase ones,
- and two-phase ones

The distinctive factor between those two being the existence of a subnetwork named Region Proposal Network. In summary, the former is faster, lighter and less effective while the latter are considered more effective and robust, yet slower and heavier. The interest of the community has shifted to the single-phase ones in the later years because of their benefits and in an attempt to cover the distance in performance with the two-phase detectors.

Typical, examples of the two-phase detectors are Fast RCNN, Faster RCNN and R-FCN, while for the single-phase ones are Single Shot Detector (SSD), You Only Look Once (YOLO) detector with its various successors and EfficientDet. YOLO has produced a series of detectors since its initial release. The more recent published work being YOLO v4.

In general, all object detectors perform the same task: localize and identify all objects of interest inside the given image. The results are presented via bounding boxes around each object, accompanied by a corresponding confidence score and a label which depicts the class the enclosed object belongs to. In the following images the process of producing the bounding box inside an image is depicted (Figure 3-1). As it is obvious several candidates

bounding boxes are produced initially, yet with low confidence score and, thus, can be filtered out in order to only keep the actual detected objects. For a more extended presentation of object detection related works the reader can refer to the previous version of the deliverable if interested.

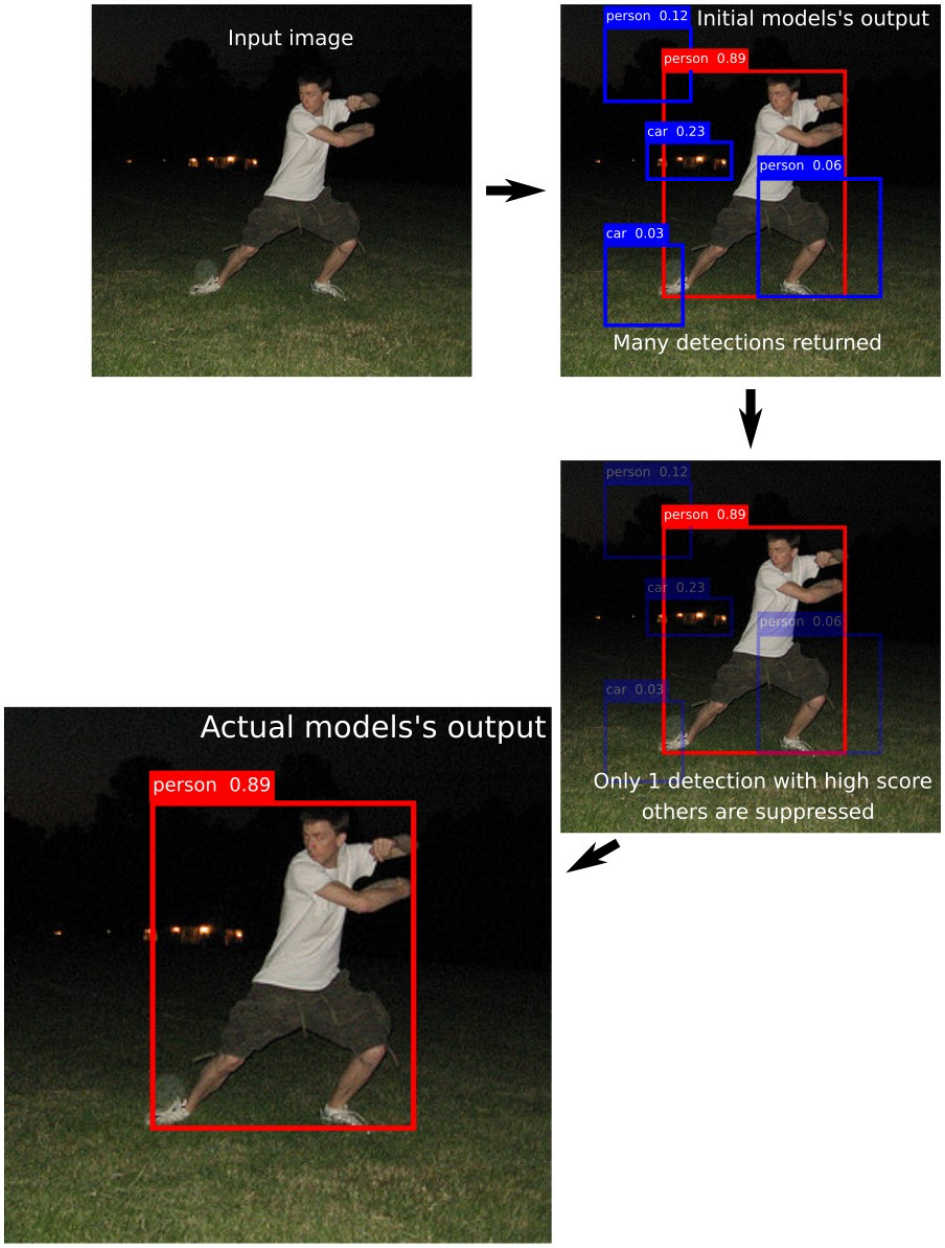

Figure 3-1 - Process of filtering the bounding box inside an image

3.2.2. Activity Recognition

Firstly, we will define our problem. Both the terms action and activity are often found in the literature, but the concept of action has predominated. In this document, the two terms will be used synonymously. According to the Merriam-Webster's Dictionary action is "the accomplishment of a thing usually over a period of time, in stages, or with the possibility of repetition". Additionally, when it comes to early detecting security threats or detecting

something that should not be done, the term "abnormal behavior" is frequently used. Finally, in the context of computer vision, activity recognition is the task to determine the action being executed in the specified number of frames.

Although deep learning methods have prevailed for solving this problem as well, instead we will make use of simpler heuristics methods to solve the required task. After analyzing the UR, we conclude that the only actions that should be recognized are those that have to do with the movement of a person, such as running, walking and standing. Hence, our model for activity recognition will be implemented as a natural continuation and extension of the already successful VOD module, i.e., we will take as input the results of VOD and try to use them to draw a conclusion about the movement or not of a person. The main advantage of this technique is, the rapid development but also at the same time the fast execution, without the need to consume large computing resources.

In conclusion, it is worth mentioning that there is no restriction on the rules to use, nor is there a general formula to follow. In these cases, we usually combine several rules together and after several trials we come up with the combination that works best for our problem. More specifically for our case, since we need to make use of the information from VOD, some of the rules could contain information such as distance (spatial and temporal), the bounding boxes, angles and unique identifiers. Finally at a more advanced level, if we had access to the frame itself, we could also use information from the image itself using for example an optical flow algorithm.

3.2.3. Tracking

As mentioned in the previous version of the deliverable (D4.1) we have included a tracker in the pipeline which provided the necessary connection between detections in consecutive frames. Essentially, tracker functions as a pipeline modifier which takes distinct frame level object detections and outputs video object detection providing the missing connection between the bounding boxes. Object detection functions on a timeless state ignoring any temporal information which is provided by the tracker which combines multiple such frame-level detection results to a complete video object detection output.

In the previous version we have been using a traditional computer vision tracker, namely Kernelized Correlation Filters Tracker (KCF) [5] which is a good candidate for its speed and its low requirements on resources since there is no demand for any GPU card. On the other hand, it is targeted on single object tracking which has different properties than Multiple Object Tracking (MOT). The latter takes into consideration the existence of various tracked objects in the scene and acts accordingly.

In the following 3 images a sequence of 3 frames is depicted, where object detection has been applied. The objects of interest are persons which are present in all 3 frames. Nevertheless, there is occlusion in the second frame which make simple tracking more

challenging. The second image presents a case of occlusion where a previously visible object (person) is now partially occluded. This may lead to object being missed by the detector or produce inaccurate bounding boxes. In this challenging cases it is crucial to use a robust tracker which can afford such deviations of the trajectories they follow. In those cases, the tracker can retrieve the expected bounding box positions and, thus, reconnect otherwise broken trajectories and provide a more stable and consistent object id assignment.

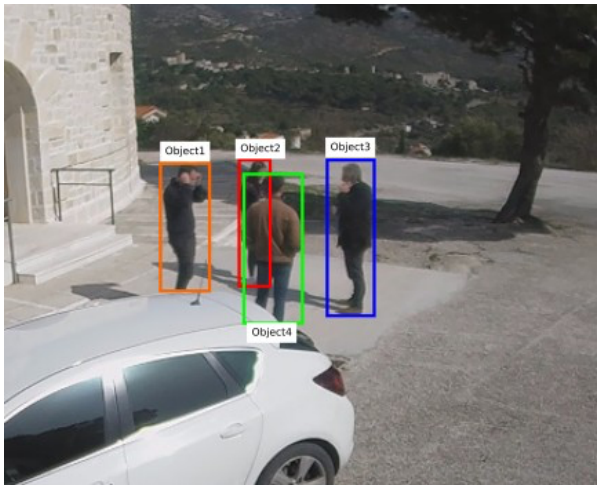

Figure 3-2 - Example with frame #n and 4 objects (persons) being tracked.

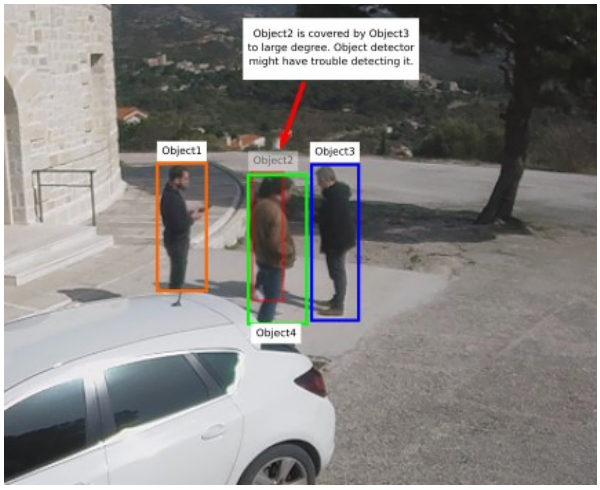

Figure 3-3 - Example with frame #n+1 with the same 4 objects being tracked. 1 object is occluded.

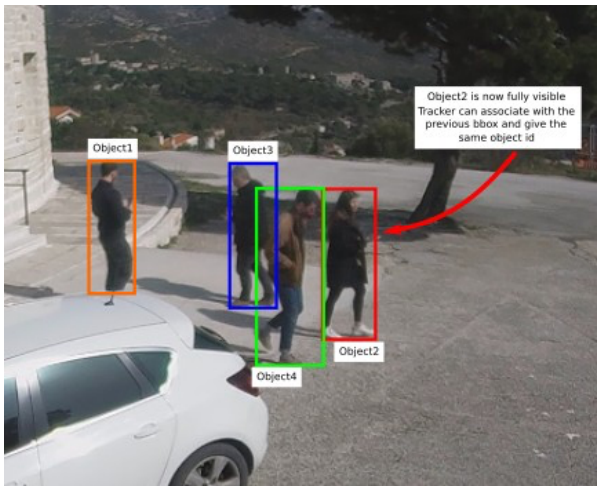

Figure 3-4 - Example of frame #n+2 where all 4 objects are again visible

As mentioned in the previous deliverable, the state-of-the-art in single object tracking is acquired by deep neural networks like GOTURN [4] which can achieve 100 fps running on

GPU or Fully Convolutional Networks Tracker (FCNT) [17] with over 50 fps on GPU also. While developing our work we shifted our interest in MOT in order to facilitate the Activity Recognition task. A simple yet powerful example of this work is Simple Online and Realtime Tracking (SORT) which does not use any deep learning techniques, but it achieves good performance on 60 fps. An extension of this algorithm which uses a deep feature association to boost the performance on expense of some efficiency, is a well-known approach called as DeepSORT [18]. Thus, it achieves process speed equal to 40 fps. DeepSORT uses the same steps which include Kalman filters, Hungarian algorithm and data association performed using a deep learning feature extractor to find similarities in human figures.

It should be mentioned that regarding Figure 3-2, Figure 3-3, and Figure 3-4, they were all captured from an actual operational test on the National Observatory of Athens (NOA) at 30 March 2022. All persons depicted in them are partners of the consortium and have provided their consent for using their figures in the deliverables of the project.

3.2.4. Datasets

As mentioned in the previous version, we have collected a set of datasets which seem to adhere to the expected input images for the project's necessities. The preferred position for the capturing camera is considered to be either UAV perspective (with more focus on inclined view angles than top-down view) and fixed cameras perspective. The latter, at least for surveillance purposes is typically an elevated camera to some degree which attempts to capture a wide area. The scene captured is, thus, at some degree comparable with the footage of a UAV at a low orbit with a medium angle of view. We present some of the public datasets acquired as follows (Table 3-1).

- The **VisDrone** dataset [13] comprised of images collected from UAV perspective. It contains in total of **10,209** static images. The initial annotation included some classes that was not relevant with the tasks of 7SHIELD and thus an adjustment of the images and annotations was performed to make it available for the project's purposes.
- As mentioned before **UAV123** is a dataset focusing on tracking. Nevertheless, it contains objects captured from a low-altitude point of view and it is suitable for our object detection task.
- The **UCF-Lockheed-Martin UAV Data Set** [6] is one of the first UAV datasets created. It contains various actions captured from a 400-450 feet height. We sampled the actual videos and manually adjusted the annotated objects depicted in them.
- The **VIRATv1** is a large dataset focusing on real life videos. We have used sampled annotated images from the videos provided.

Dataset	Task	# images
VisDrone train set	Object detection from drone point of view	6471
VisDrone test set	Object detection from drone point of view	548
UAV123 (car-truck etc.)	Tracking (modified for object detection)	1137
UCF-Lockheed-Martin UAV Data Set	Activity recognition from drone point of view	407
Person datasets (UAV123, VIRATv1 etc.)	Person instances from drone point of view	9478

Table 3-1 - Object detection datasets

3.3. Methodology

3.3.1. Object Detection

In D4.1 two object detection module was implemented and used. The first one was a Faster RCNN while the second one was an EfficientDet variation (Figure 3-5). Since, we focus on solution which are fast we abandoned the Faster RCNN model and replaced it with YOLO v4 (Figure 3-6). Thus, we still have two models for object detection but both are fast and lighter than Faster RCNN. Both models are single-phase detectors which make them fast and lightweight.

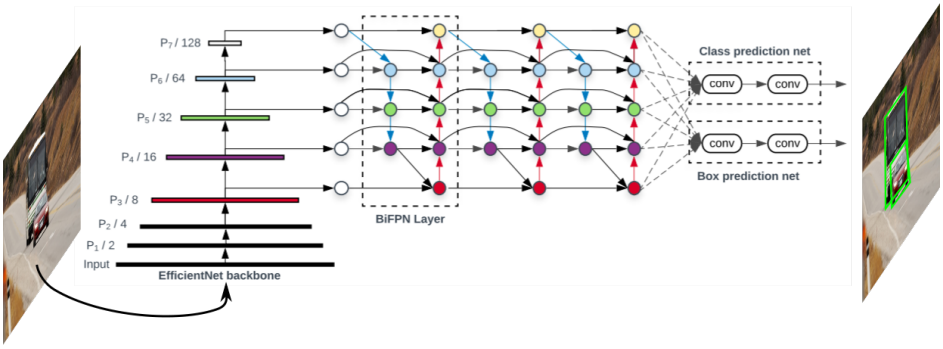

Figure 3-5 - EfficientDet architecture

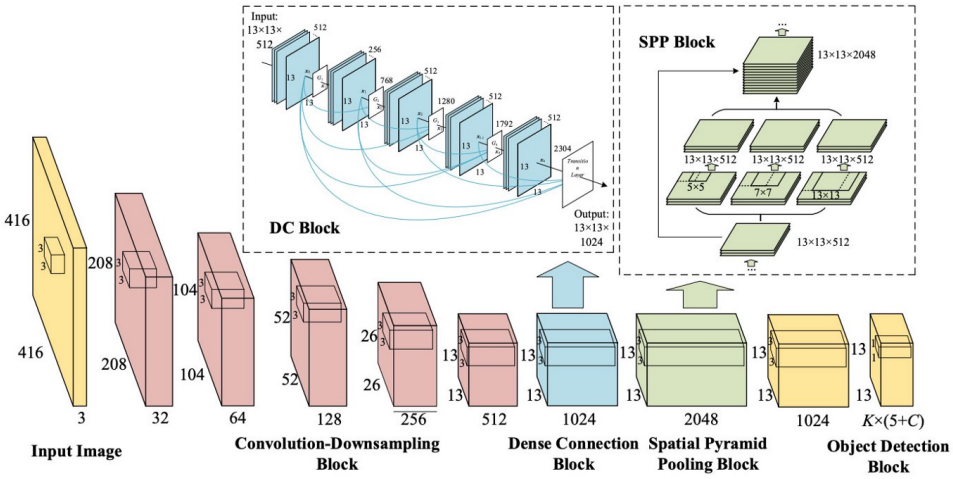

Figure 3-6 - YOLO v4 architecture

As mentioned before, Object Detection is performed in every frame which means that is performed on still images. Every such procedure is unaware of relevant detection in previous or next frames. So, in order to have a connection between the bounding boxes produced in every frame we provide the Object Detector output to a tracker which has the responsibility to find their connection.

After detecting an object, VOD provide its output to the GCEP correlator which combines the cyber and physical threats. The messages are being aggregated over a certain number of frames (typically 60) and then being send periodically for as long as there are detected objects. The format being used for reporting the detection is 7SHIELD UAF. Also, the visualization output is also being stored in the same device the module is being executed from in order to provide a different way to analyze the monitored scenery.

3.3.2. Activity Recognition

For the needs of the project, based on the UR for action recognition, a model has been developed that is an extension of VOD module which uses heuristic rules for motion recognition. More specifically, we tried to identify whether a person is moving or not and in addition we attempted to distinguish if moving whether he/she is walking or running. This solution, although simple, adequately satisfies the project requirements and has the advantages of fast implementation and execution.

Hence, initially VOD takes as input a stream of frames and results in the output for each frame, the bounding boxes in which the people are located, a unique identifier for each person and a confidence score, which shows the percentage of certainty that the bounding box contains a person. Then, the above results are distributed through Kafka platform. Therefore, the AR module has as its input the results of VOD module, i.e., the bounding boxes for each person, a unique identifier, and the frame number in which they appeared. An example of VOD output is shown in Figure 3-7.

Figure 3-7 - Object Detection Output as used by Activity Recognition

Then, we will analyze in more detail the logic with which the AR module works. Since we want to predict the movement made by each person individually in the provided surveillance video, we filter the results of the VOD, based on the unique identifier of each person (shown in Figure 3-7). More specifically, AR reads the messages from VOD with its predictions periodically, and for a certain number of frames for example every 15 frames and then for each unique person identifier, creates a queue where it stores the following information: the frame number in which the particular person appeared as well as the bounding boxes, i.e., exactly where the particular person appeared. Consequently, we have two kinds of information, the temporal information, in which frame number person appeared and the spatial information, in which point of the image exactly appeared.

First, we look at each time a prediction is made if there is a sufficient number of frames in each queue for each person to make a prediction. This number is inextricably linked to how well VOD works and from now on, we call it prediction window length. Hence in VOD, there is a chance that it will fail to assign the same identifier to the same person after some time. This can happen for example because an obstacle or occlusion appeared in the video or even because VOD simply failed to make a correct identification. For this reason, this number can be neither too large nor too small, since if it is small, we will have very frequent predictions, while if it is large, we may have wrong predictions since the identifiers will be confused. After testing with the VOD module, we concluded that the optimal number for our case is 15 frames.

Then, another similar problem that may arise with the identifiers provided by VOD is the following: it may be that, for example, while a person is indeed in consecutive frames, VOD may eventually fail to recognize him in all but a few of them. Hence, we should introduce some tolerance in our own AR module with respect to successful recognition in consecutive frames. Hence, after we read 15 frames and create the queues for each person, then we

check what percentage of those frames each person appears in. If, for example, it only appears in the first and last one, then it is quite likely that VOD has made a mistake and therefore a prediction cannot be made because that particular identifier is not present in a sufficient number of consecutive frames. Beyond that, an identifier may be found in all 15 frames or it may have failed to find it in for example the 2nd and 13th frame. Therefore, since it does not make sense to check one by one the frames for the presence of the unique identifier, as such a thorough analysis does not offer any additional value, we decided that we will simply check for the presence of this identifier in a large percentage of these 15 frames of the window. After testing, we came up with the value of 80% percentage, i.e., we allow a tolerance of VOD to miss up to 3 frames (20% x 15). Again, this number can be easily changed in case of a change in the implementation of VOD.

So far, we are mainly concerned with the temporal information as well as each time we are going to predict the motion. Now, we will see how the prediction of the motion is done, if the reasons we mentioned earlier for the number of frames apply. Thus, now we will make use of the spatial information, i.e., the bounding boxes provided for each person identifier. Essentially, the bounding box defines a rectangle within which the person is located in the given frame. Consequently, if in subsequent time frames we receive bounding box for the same person identifier, then there are 2 cases: if the particular person is standing still, then the frames will be approximately at the same point, and if he has moved then they will have moved as well. Since the frames provided by VOD are not constant in size even in consecutive frames for a standing man, we will use the areas defined by the frames for a person. In this way we manage to have easier calculations, since we do not need to calculate exactly how much a frame has moved. Additionally, we also mitigate the problem of unstable in size frames, since we introduce some tolerance in our system, which will be easier to understand later in the analysis of our calculations.

More specifically, as we mentioned, we will use the areas defined by the frames to track how much a frame has moved to the next time. Essentially, for each person ID, we will take its bounding box in the first and last frame window that appears and calculate the area of IoU of the two bounding boxes. As a consequence, if this person did not move at all during this time, then the above metric will return a value close to 1, since the frames will almost match. If on the other hand the person has moved a lot, he/she will have a value close to 0. We have used the above technique to decide whether there was a movement (standing or moving) and if so, how large it was (movement: walking or running). So, three classes were introduced this way: standing, walking and running.

Activity Recognition module also reports its results via the UAF format. The module analyzes the output from VOD module and provide an estimation (if it possible to be deduced under the restriction being mentioned previously) of the action being executed by the detected persons. The analysis for the actions and the reporting is done in packets of 15 frames.

3.3.3. Object Detection at the Edge

In the aspect of project's objectives and requirements a dedicated module for performing object detection on-board the project's UAV assets is demanded. The main difference is the constraints on the resources which are available. We use a Jetson AGX Xavier UAV for our purposes in 7SHIELD. Apart from the restricted resources other things which ought to be changed were the way the Object Detection at the Edge (ODE) was triggered. Since, the detection is performed entirely on board a method for direct communication between the Data Collection and Edge Processing (DCEP) and ODE should have been searched. The actual communication has been presented in the following subsection 3.3.3.1.

ODE also provide its output via the Kafka messaging system (as the other T4.3 modules, namely VOD and AR). Similarly, to them, it provides alerts every time it encounters detection on the monitored area and after the detection it send them periodically every 60 frames. In order to complete the message that should be send it retrieves telemetry information from the UAV (GCEP) with which it communicates internally using UDP packet for robust and fast communication. The messages are being formatted in order to follow the usual 7SHIELD UAF so as to be compliant with the norms the projects have set.

3.3.3.1. Communication process between DCEP and ODE

The purpose of this implementation is the communication between the DCEP and the ODE component. More specifically for any detection made by the ODE during the flight it is necessary to know some of the flight data of the UAV such as the attitude data (latitude, longitude, altitude). For this reason, a joint implementation of cooperation was developed between these two components DCEP and ODE through the exchange of datagram packets. The messages as well as their description are as follows:

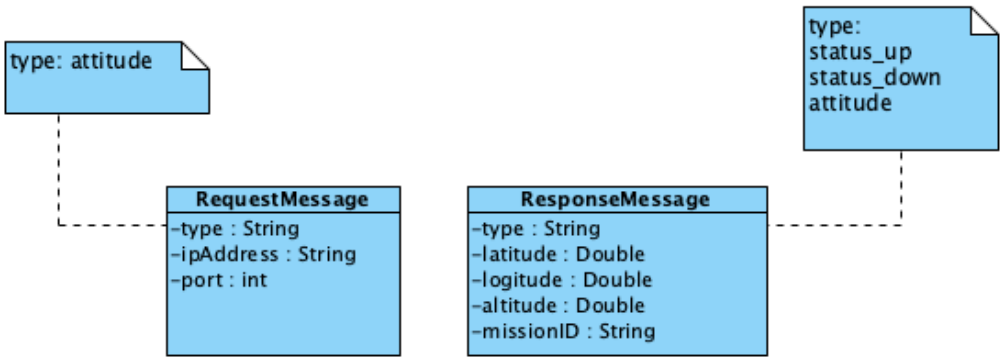

Figure 3-8 - Type of messages for on Edge inter-communication

The exchange as mentioned above is done through datagram packets in JSON format. Below we can see some examples of these messages:

- Request Message (From ODE to DCEP)

{"type":"attitude","ipAddress":"127.0.0.1", "port": 20000}

- Response Message Example (From DCEP to ODE)

  {"type":"status_up","latitude":0.0,"longitude":0.0,"altitude":0.0, "missionID": "mission_01"}

  {"type":"attitude","latitude":37.907446,"longitude":23.752632,"altitude":28, "missionID": "mission_01"}

  {"type":"status_down","latitude":0.0,"longitude":0.0,"altitude":0.0, "missionID": "mission_01"}

In the following diagram we can see the sequence of these messages:

Figure 3-9 - Sequence of messages between DCEP and ODE

As we can see in the diagram above the sequence is as follows:

1. When the UAV takes off, the DCEP component informs the ODE component about the status of the UAV. Status is up in this case.
2. The ODE then runs the object detecting algorithm for each frame of video that reads real time from the UAV camera while also feeding the results to the tracker.
3. In any case an object is detected, ODE requests the flight information (longitude, latitude and altitude) of that time.
4. The DCEP then sends the flight information of that time.
5. ODE compiles the information from DCEP and the actual detection and periodically, after accumulating a certain amount of frame outputs send them via the project's messaging system (Kafka) to other modules.

6. The latter three steps are repeated as long as the flight has not terminated, and objects are detected.
7. Finally, during landing the DCEP component informs the ODE component about the UAV landing and ODE stops any process on data. Status is down now.

In the diagram below we can see the connection between the components DCEP and ODE in relation to the UAV materials with which they interact:

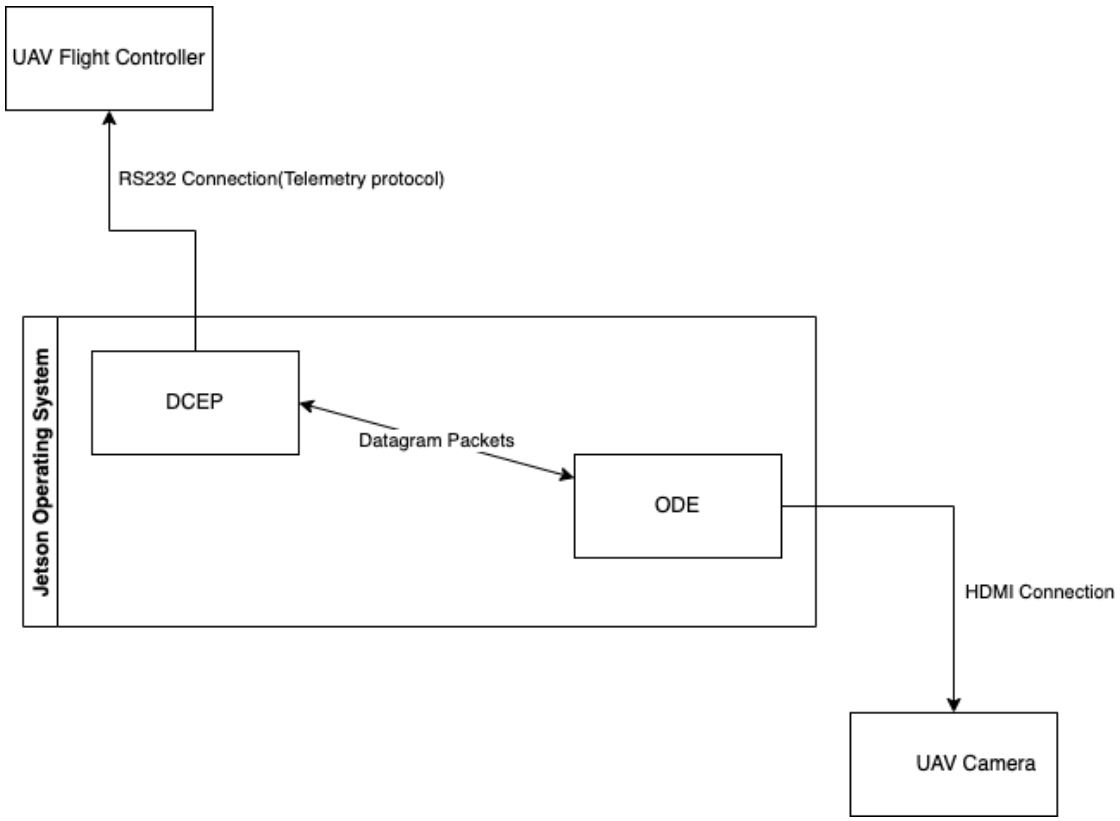

Figure 3-10 - Connection between the components DCEP and ODE in relation to the UAV materials

3.4. Experiments and Results

3.4.1. Object detection effectiveness results

We have separated the task of the Object Detector into two distinct cases:

1. The case where the detection should be performed in an outdoor environment and
2. The case where only indoor object detection is of interest.

The reason for separating these two cases is that this way we can deal more easily with the objects to be detected (the chance of appearing a truck indoor is limited for example) and achieve higher performance. Also, the sceneries monitored in those cases are exclusive to each other. The outdoor scenes often must capture smaller, blurrier and in general more challenging instance of objects, while indoor person detection is more confined and controlled and, thus, easier as task. Finally, it should be noted that indoor does not

necessary means inside a building but it's used in a rather extended form and implies any scenery which is not on an open area, which include also environments like patios, outdoor surfaces in stations, balconies etc. We have adopted the chosen models also for this case, choosing a heavier model for outdoor detection and a lighter one for indoor.

3.4.1.1. Outdoor detector

We have conducted a series of experiments in order to evaluate the performance we achieve using the two aforementioned models: EfficientDet and YOLO v4. There are 5 classes of interest which have been defined a-priori and on which the models were trained on. These include:

1. 1 generic Person detection class
2. 4 common land vehicles:
    a. Car
    b. Bus
    c. Truck
    d. Motorcycle

The effectiveness evaluation of each model was performed in a custom created evaluation set. The training set was also a custom compiled dataset which included images captured with the preferred angle of view and elevation. In total 1803 images were used to train and evaluate the model. The split between training and validation set is shown in Table 3-2.

Custom Dataset statistics	
Training set samples:	Evaluation set samples:
16820	983
Total number of images	
17803	

Table 3-2 - Custom Object Detection dataset statistics

Regarding the effectiveness performance the results for the two models are shown in the Table 3-3, Table 3-4, Table 3-5 and Table 3-6 for EfficientDet and YOLO v4 respectively. The first and third tables are baseline performance values which are the values we are competing against. The second and the fourth tables contain the highest performances value we were able to achieve. EfficientDet model achieves a mAP of just about 65% while YOLO v4 achieves a just under 80% value. The mAP is evaluated on a relatively difficult dataset since it contains a lot of small (or even tiny) object instances and, thus, the overall performance is hindered by this fact. In generally, YOLO v4 seems to outperform

EfficientDet in every class including the mAP but the difference varies depending on the class. The larger difference is observed for Person class which is almost 7% while the smaller difference occurs in Truck class which is just above 1%. For the YOLO v4 case we see that there is a huge difference between the baseline and the improved model values. This is because the baseline was trained on a more generic dataset derived from COCO [12] while the improved on our custom dataset. Nevertheless, we see that YOLO v4 achieves a satisfactory precision even in the challenging custom dataset.

baseline outdoor EfficinetDet phi2 Average Precision (AP)		
51.11% (person)	70.35% (bus)	74.68% (car)
47.97% (motorcycle)	67.82% (truck)	**mAP 62.39%**

Table 3-3 - Baseline EfficientDet mAP performance

improved EfficinetDet phi2 evaluation experiments Average Precision (AP)		
52.42% (person)	75.36% (bus)	76.68% (car)
52.94% (motorcycle)	69.22% (truck)	**mAP 65.32%**

Table 3-4 - EfficientDet mAP performance

baseline medium YOLO v4 evaluation experiments Average Precision (AP)		
70.47% (person)	81.16% (bus)	59.88% (car)
64.2% (motorcycle)	56.22% (truck)	**60.93% (mAP)**

Table 3-5 - Baseline YOLO v4 mAP performance

medium YOLO v4 evaluation experiments Average Precision (AP)		
59.21% (person)	80.72% (bus)	81.0% (car)
56.39% (motorcycle)	70.55% (truck)	**mAP 69.57%**

Table 3-6 - YOLO v4 mAP performance

The mAP performance is not affected by the machine it is tested on, so it is the same if examined on a workstation or at the edge (on board a UAV for example).

Below a few examples of the outdoor object detector are provided. As it can be seen to those images the actual object instance in many cases is relatively small as mentioned before which makes their detection a challenging task. Figure 3-11 to Figure 3-15 are cropped images from VisDrone (in order to comply with our UR) and Figure 3-16 is from Viratv1 dataset.

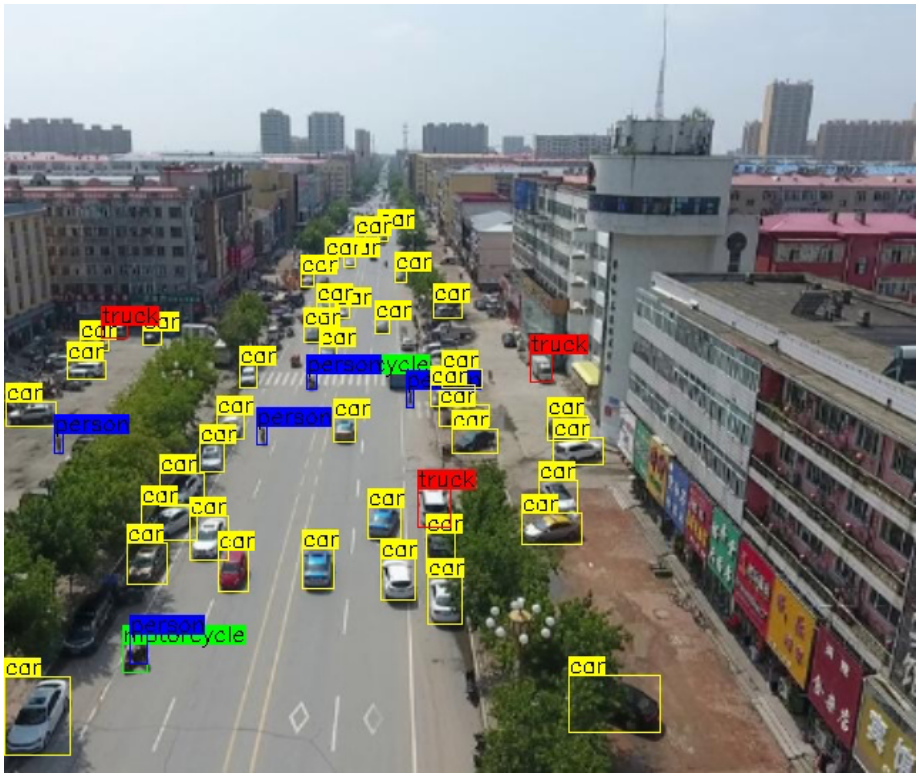

Figure 3-11 - Example of outdoor Object detection using YOLO v4

Figure 3-12 - Example of outdoor Object detection using YOLO v4

Figure 3-13 - Example of outdoor Object detection using YOLO v4

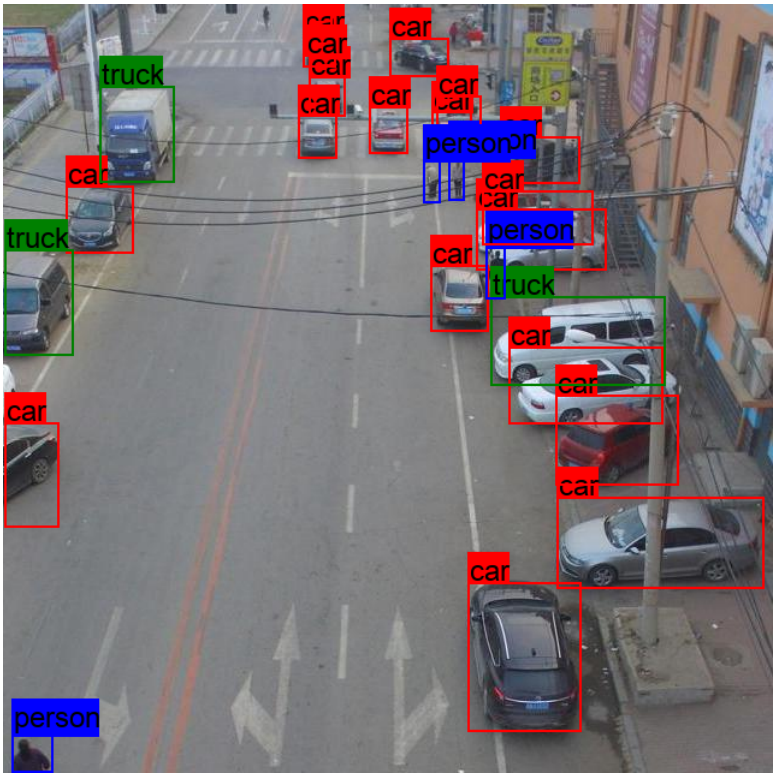

Figure 3-14 - Example of outdoor Object detection using EfficientDet

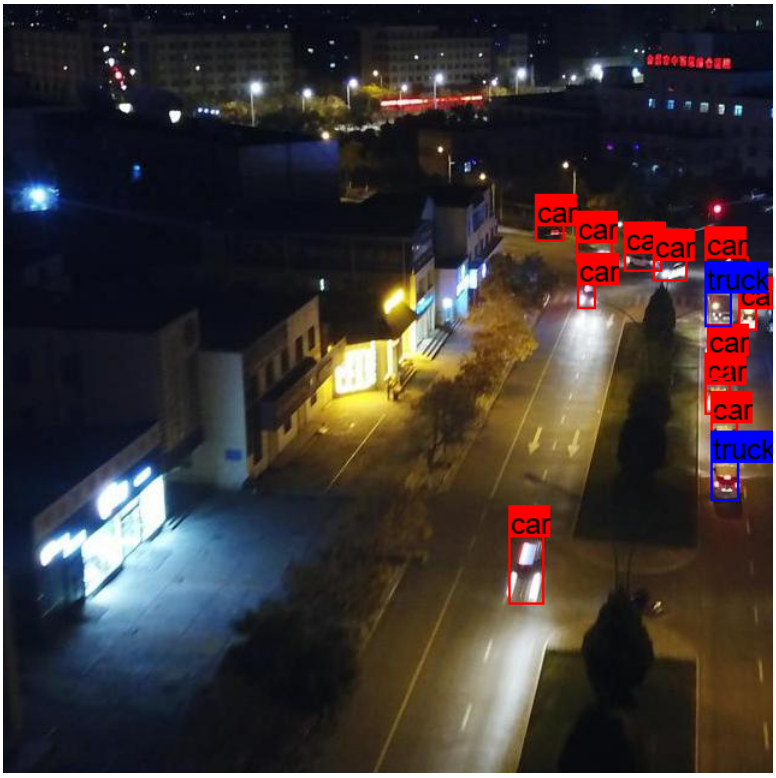

Figure 3-15 - Example of outdoor Object detection using EfficientDet

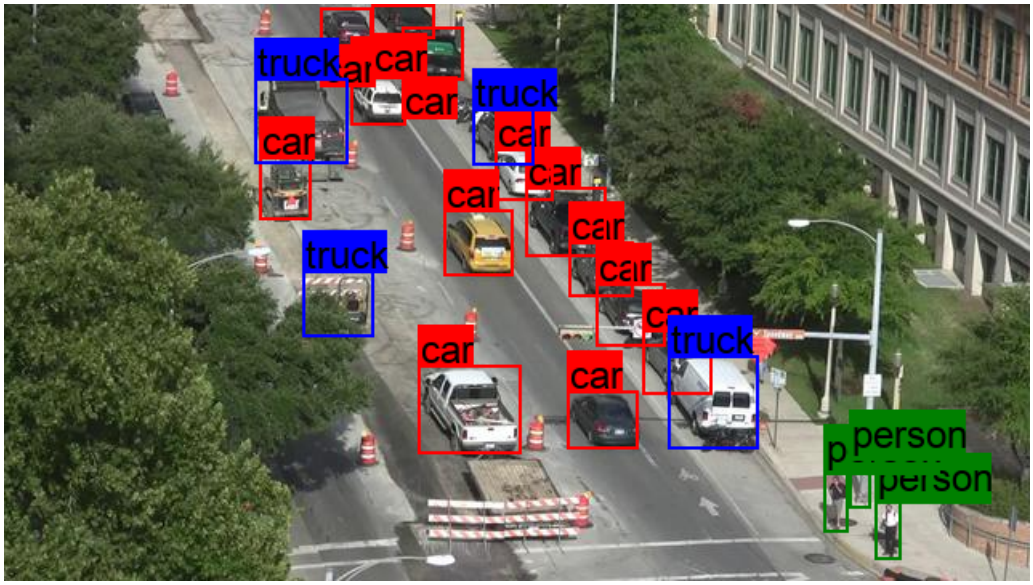

Figure 3-16 - Example of outdoor Object detection using EfficientDet

3.4.1.2. Indoor detector

Regarding the indoor object detector, we have also conducted a series of experiments to evaluate the performance of the model. In this case we did not train the YOLO detector since we considered that the results from EfficientDet were satisfactory both in efficiency as well as in effectiveness measurements.

In this case, the objects of interest were just two:

1. A generic person class and
2. A backpack class

For the training and evaluation of the indoor object detection also a custom dataset has been compiled. It includes images from Open Images Dataset [10], COCO and Virat v1 datasets.

Custom Dataset statistics	
Training set samples:	Evaluation set samples:
6046	109
Total number of images	
6155	

Table 3-7 - Custom Object Detection dataset statistics

baseline indoor EfficinetDet phi1 Average Precision (AP)		
86.26% (person)	83.28% (packpack)	**mAP 84.77%**

Table 3-8 - baseline indoor EfficientDet mAP performance

improved model indoor EfficinetDet phi1 Average Precision (AP)		
86.40% (person)	85.07% (packpack)	**mAP 85.73%**

Table 3-9 - improved indoor EfficientDet mAP performance

Below some output samples of the indoor object detector have been included. All images depicted below Figure 3-17, Figure 3-18, Figure 3-19, are from Open Images and COCO datasets.

Figure 3-17 - Example of indoor EfficientDet detector

Figure 3-18 - Example of indoor EfficientDet detector

Figure 3-19 - Example of indoor EfficientDet detector

3.4.2. Object Detection efficiency experiments

3.4.2.1. Outdoor detector

Regarding the efficiency evaluation we have also tried to estimate how fast each model is. For this reason, we have tested the models on both a workstation and jetson asset. The results in this case of course are heavily affected by the hardware available and, thus, we have estimated the processing time of each model on the different architectures we expect to use for the project's needs. The workstation has a GPU Nvidia GeForce RTX 3090 while the Jetson was a Jetson AGX Xavier. Since we have chosen to use YOLO v4 in Jetson also we haven't tested EfficientDet efficiency on Jetson and the relevant value is missing in Table 3-10.

EfficientDet efficiency results	
Workstation Nvidia Geforce RTX 3090	Jetson Jetson AGX Xavier
YOLO v4: 25 fps	YOLO v4: 8.4 fps
EfficientDet: 30 fps	N/A

Table 3-10 - Efficiency results

3.4.2.2. Indoor detector

Regarding the efficiency of the indoor model, we have tested its processing speed and the results are provided in the following Table. It should be noted that since this indoor model is not expected to be used for ODE, we haven't checked its speed on the Jetson. As it can be seen the fps are above real time capabilities which means the processing of input images can be performed in a satisfactory manner.

EfficientDet efficiency results	
Workstation Nvidia Geforce RTX 3090	EfficientDet: 35 fps

3.4.3. Activity Recognition effectiveness results

For the Activity Recognition (AR) module we have also conducted a series of experiments to evaluate its performance. Since AR is defined as an extension of Object Detection, there are no training weights in this module and, thus, the processing speed is very high. Also, taking into account the latter fact we only evaluated the effectiveness (accuracy for this case) of the module. Finally, as stated before, the results are dependent of the Object Detector being used before and, thus, are highly sensitive to changes on the performance of VOD. The results are demonstrated in the following table.

Activity Recognition Accuracy results	
EfficientDet detector	YOLO v4 detector
77%	81%

Figure 3-20 - Example of correct AR labeling for EfficientDet output

Figure 3-21 - Example of correct AR labeling for YOLO output

Figure 3-22 - Example of correct and false AR labeling for EfficientDet output

Figure 3-23 - Example of correct AR labeling for YOLO output

Figure 3-24 - Example of correct AR labeling for EfficientDet output

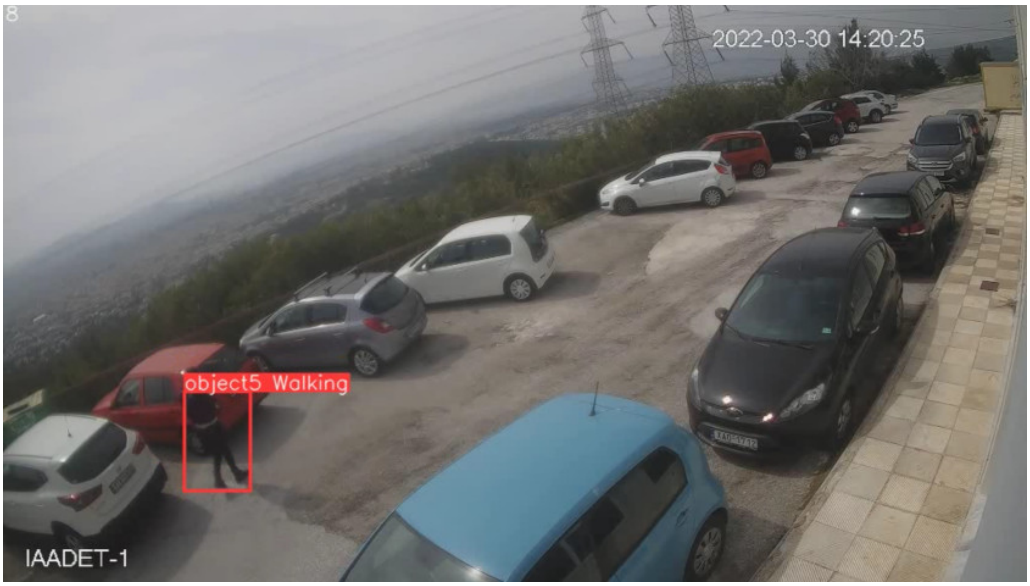

Figure 3-25 - Example of correct AR labeling for YOLO output

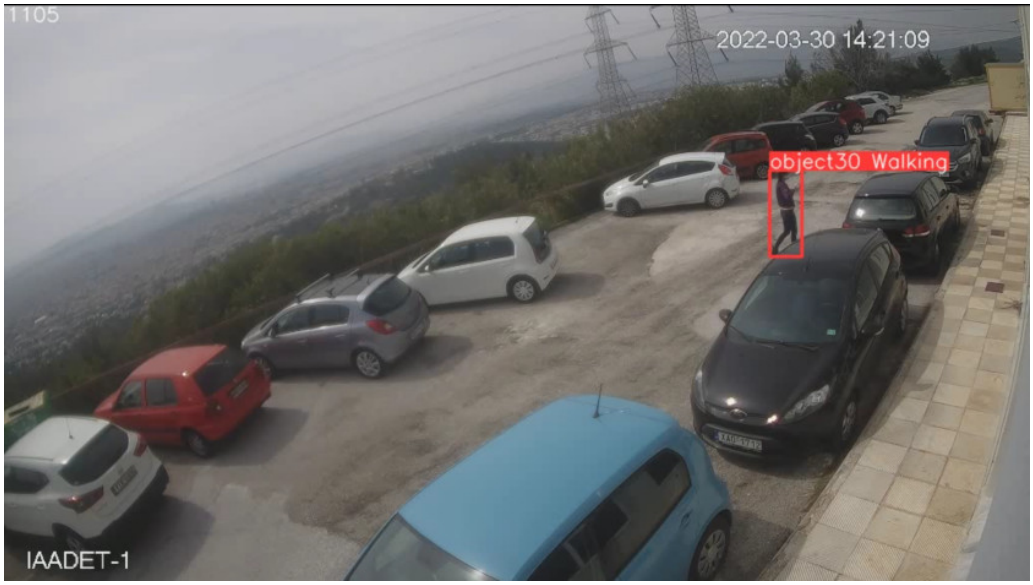

Figure 3-26 - Example of correct AR labeling for YOLO output

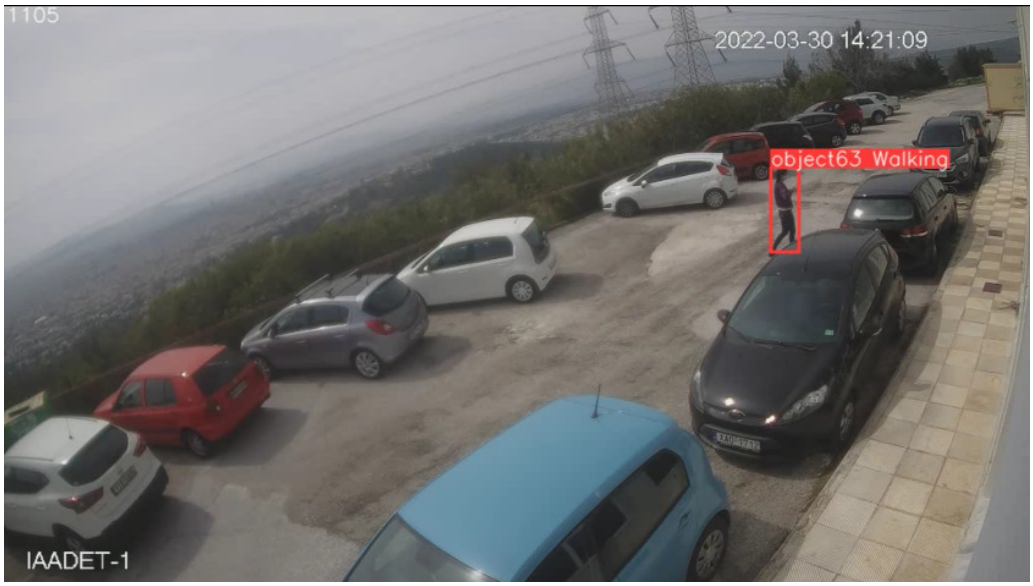

Figure 3-27 - Example of correct AR labeling for EfficientDet output

4. Conclusions

This section draws conclusions for Task 4.2 and Task 4.3 (including a part of Task 4.1 related to ODE) and their corresponding key results KR06 and KR07.

After the revised user requirements have been reviewed, we have adapted the acquired state-of-the-art solutions for face detection and face recognition specifically for the objectives of KR06 in the context of the 7SHIELD platform and with the defined use cases of the project in mind. In addition to enriching performance evaluation of some key aspects of our solution (e.g., detection speed, gallery size impact), the final release was also redesigned with a heavy focus in the recognition component and in order to provide an initial level of information aggregation using the facial clustering algorithm which is also now more robust to outliers and false alarms. Regarding the main benefits of our approach, the face gallery can be expanded dynamically with additional persons, without the need to retrain the deployed detection and recognition models. Moreover, in a large-scale environment with multiple cameras, a separate gallery can be created per camera in order to support customized monitoring for each specific area.

Regarding the object detection module, state-of-the-art algorithms have been utilized to address the objectives of KR07. The general task was split into two different detectors to better cover the requirements (indoor and outdoor detector) while being fast and lightweight. In order to deliver a video object detector (as the requirement demanded) a tracker was introduced, DeepSORT, which although not trained for the project provided the necessary link between consecutive frames. It also, provided a robustness for misdetection and occlusion and helped the activity recognition task which rely on the video object detector.

For activity recognition, after validating the user requirements, the solution chosen was to be implemented as an extension to the video object detector. The actions indicated by the users include motions which could be handled with relatively ease from this extension. The solution is fast, since no deep neural network is involved in the activity recognition process, while maintaining high performance.

Finally, as regards the object detection at the edge, special care was taken for delivering models which could function in a satisfactory way on board the project's UAV. Also, a new submodule for delivering a fast and reliable inter-communication between ODE and GCEP was implemented in order to address the requirements.

5. References

[1] Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018, May). Vggface2: A dataset for recognising faces across pose and age. In 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018) (pp. 67-74). IEEE.

[2] Chi, C., Zhang, S., Xing, J., Lei, Z., Li, S. Z., & Zou, X. (2019, July). Selective refinement network for high performance face detection. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, No. 01, pp. 8231-8238).

[3] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4690–4699.

[4] Held, D., Thrun, S., & Savarese, S. (2016, October). Learning to track at 100 fps with deep regression networks. In European conference on computer vision (pp. 749-765). Springer, Cham.

[5] Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2014). High-speed tracking with kernelized correlation filters. IEEE transactions on pattern analysis and machine intelligence, 37(3), 583-596.

[6] https://www.crcv.ucf.edu/data/UCF_Aerial_Action.php

[7] Hu, P., & Ramanan, D. (2017). Finding tiny faces. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 951-959).

[8] Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008, October). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In Workshop on faces in'Real-Life'Images: detection, alignment, and recognition.

[9] Jain, V., & Learned-Miller, E. (2010). Fddb: A benchmark for face detection in unconstrained settings (Vol. 2, No. 4, p. 5). UMass Amherst technical report.

[10] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., ... & Ferrari, V. (2020). The open images dataset v4. International Journal of Computer Vision, 128(7), 1956-1981.

[11] Li, J., Wang, Y., Wang, C., Tai, Y., Qian, J., Yang, J., ... & Huang, F. (2019). DSFD: dual shot face detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5060-5069).

[12] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

[13] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Qinghua Hu, Haibin Ling. Vision Meets Drones: Past, Present and LFFuture. arXiv preprint arXiv:2001.06303 (2020).

[14] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

[15] Shi, Y., & Jain, A. K. (2019). Probabilistic face embeddings. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 6902-6911).

[16] Tang, X., Du, D. K., He, Z., & Liu, J. (2018). Pyramidbox: A context-assisted single shot face detector. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 797-813).

[17] Wang, L., Ouyang, W., Wang, X., & Lu, H. (2015). Visual tracking with fully convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 3119-3127).

[18] Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In 2017 IEEE international conference on image processing (ICIP) (pp. 3645-3649). IEEE.

[19] Wong, Y., Chen, S., Mau, S., Sanderson, C., & Lovell, B. C. (2011, June). Patch-based probabilistic image quality assessment for face selection and improved video-based face recognition. In CVPR 2011 WORKSHOPS (pp. 74-81). IEEE.

[20] Yang, S., Luo, P., Loy, C. C., & Tang, X. (2016). Wider face: A face detection benchmark. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5525-5533).

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883284