An Embedded Toolset for Human Activity Monitoring in Critical Environments

Marco Di Benedetto[1,*], Fabio Carrara[1], Luca Ciampi[1], Fabrizio Falchi, Claudio Gennaro, Giuseppe Amato

Institute of Information Science and Technologies of the National Research Council of Italy (ISTI-CNR)
name.surname@isti.cnr.it

Abstract

In many fields of human working and recreational activities, there exist scenarios where both individual and collective safety have to be constantly checked and properly signaled, as occurring in dangerous working places or during pandemic events like the recent COVID-19 disease. From wearing personal protective equipment to filling physical spaces with an adequate number of people, it is clear that a possibly automatic solution would help to check compliance with the established rules. Based on an off-the-shelf compact and low-cost hardware, we present a deployed real use-case embedded system capable of perceiving people's behavior and aggregations, and able to supervise the appliance of a set of rules relying on a configurable plug-in framework. Working on indoor and outdoor environments, we show that our implementation of counting people aggregations, measuring their reciprocal physical distances, and checking the proper usage of protective equipment is an effective yet open framework for monitoring human activities in critical conditions.

Keywords: Deep Learning, Computer Vision, Machine Learning, Personal Protective Equipment, Counting, Homography, Embedded System

1. Introduction

As occurs during a severe health emergency event, there exist scenarios in which ensuring compliance to a set of guidelines becomes crucial to secure a safe living environment in which human activities can be conducted. In fact, as evidenced during the recent COVID-19 pandemic, wearing medical masks, avoiding the creation of large gatherings in confined places, and keeping a certain physical distance among people were the most common rules every government applied in their jurisdiction territories. However, this task could not always be guaranteed by human supervision, especially in crowded scenes where checking usage of personal protection equipment or enforcing a strict social behavior has to be continuously assessed to preserve global health.

In the past decades, Computer Vision applications have shown astonishing results in several daily life tasks. Automatic image analysis aimed at classifying, locating and counting objects, as well as estimating the distance between different instances of objects, are typical examples of applications of Computer Vision technology, which can be a valuable tool to automatically monitor human activities in critical environments through images captured by networked cameras.

In this work, we present an embedded modular Computer Vision-based and AI-assisted system that can carry out several tasks to help monitor individual and collective human safety rules, processing the captured images directly on an off-the-shelf commercial and low-cost device. Our solution consists of multiple modules, each responsible for a specific functionality that the user can easily enable and configure. In particular, exploiting one of these modules, or combining some of them, our framework makes available many capabilities. These range from the ability to estimate the so-called social distance (i.e., the physical distance among pedestrians) to the skill of estimating the number of people present in the monitored scene,

*Corresponding author
[1]Authors contribute equally.

as well as the possibility to localize and classify personal protective equipment (PPE) worn by people (such as helmets, high-visibility clothing, and face masks) that the World Health Organizations has recommended as one of the primary tools able to contrast the spread of the disease, like, for example, the recent COVID-19 pandemic.

To validate our solution, we test all the functionalities that our framework, deployed on an embedded device, makes available, exploiting two novel datasets that we collected and annotated on purpose and that represent another contribution of our work. Specifically, we gathered a first dataset comprising images captured by a smart camera located in a public square in the city of Pisa, Italy, that represents a typical scenario for which it is crucial to check compliance with the safety rules, such as the maintaining of the social distance or the monitoring of the area occupancy. Moreover, we collected and annotated a second dataset comprising images containing pedestrians with and without PPE, such as helmets, high-visibility vests, and face masks. The peculiarity of this dataset is that part of the images are gathered from the GTA V video game and automatically annotated by the graphical engine. Experiments show that our system can effectively carry out all the functionalities that the user can set up, providing to be a valuable asset to automatically monitor compliance with safety rules.

To summarize, the main contributions of this work are the following:

- We introduce an expandable and flexible Computer Vision-based and AI-assisted embedded system, deployed in a real use-case scenario, capable of automatically monitoring human activities in critical environments, where individual and collective safety must be constantly checked. We base our solution on modules responsible for specific tasks that the user can easily configure and add to the whole system, making many available functionalities such as the capability of estimating the number of pedestrians present in the scene, measuring the social distances among people, and detecting PPE worn by the individuals.

- We collect and annotate two novel datasets that we exploit to validate our framework. The first one, named *CrowdVisorPisa*, is gathered from a camera in a public square of the city of Pisa, Italy, and represents a typical scenario for which it can be essential to monitor compliance with the safety rules, such as the respect of social distance. The second is instead a collection of images, partially synthetic, representing pedestrians with and without wearable PPE.

- We conduct experiments evaluating all the modules and the functionalities, which our framework makes available *in an embedded and deployed off-the-shelf device*, showing that our solution may be a valid aid to monitor and handle critical environments drastically reducing human supervision.

We organize the rest of this paper as follows. We review similar works in Section 2, and we introduce our modular framework and its plug-ins in Section 3. In Section 4, we describe the exploited datasets along with the adopted training procedures. In Section 5, we show our experiments, also discussing and analyzing the obtained results. Finally, we conclude the paper with Section 6, suggesting some insights on future directions.

2. Related Work

Due to the COVID-19 pandemic, many Computer Vision-based works have been recently published to help monitor human activities analyzing images, especially on the specific task of evaluating the social distance between people. For example, the Inter-Homines system, presented in [13], evaluates in real-time the contagion risk in a monitored area by analyzing video streams. The system includes occlusion correction, homography transformation, and people anonymization. People are located in the space exploiting the CenterNet [46] object detector, and interpersonal distances are then calculated. Results are evaluated on the JTA dataset [12] (i.e., in a virtual world). In [31], the YOLO9000 detector [25] has been exploited to detect people, and centroids of the found bounding boxes are computed to evaluate the distance between them. Similarly, in [2], a platform for social distance tracking in top perspective video frames based on YOLOv3 [26] was presented. Here too, centroids of the bounding boxes are used to estimate distances. A subset of the authors of [2] also presented in [1] a social distance framework based on the Faster-RCNN detector [27]. Instead, YOLOv3 [26] to detect humans and Deepsort [38] to track people are exploited in [24]. Experiments are conducted on the Oxford town center surveillance footages [4]. The usage of Faster R-CNN [27] and YOLOv4 [6] to detect pedestrians are discussed in [40] to monitor social distancing and density. Monitoring of workers to detect social distancing violation that uses Mobilenet-V2 [30] to detect people is introduced in [18].

Another task recently tackled in literature, again related to the COVID-19 pandemic, is face mask detection. For example, authors [19] present an edge computing-based mask identification framework (ECMask). It consists of three main stages: video restoration, face detection (inspired by FaceBoxes [43]), and mask identification (based on Mobilenet-V2 [30]). Deep learning models were trained and evaluated on the Bus Drive Monitoring Dataset, which unfortunately is not publicly available. Authors in [11] developed a deep learning-based computer vision system able to perform face mask detection but also face-hand interaction detection. A more comprehensive literature review of applications of artificial intelligence in battling against COVID-19 is given in [34] including social distancing and face mask detection.

Differently from most other works, in this paper, we present a *modular* and *expandable* Computer Vision-based embedded system that can fulfill *multiple* tasks to help monitor compliance of individual and collective human safety rules in critical scenarios, like the one caused by the COVID-19 pandemic. The main peculiarities are that it runs directly on a low-cost computing device and that the user can easily enable and configure the available functionalities, ranging from computing social distances, estimating the number of people present in the scene, or detecting PPEs, combining more modules and building more complex tasks.

3. Modular Framework

The general purpose of our monitoring system is to be embeddable on low-cost devices and, above all, to be expandable to different features on demanding situations. To this end, we designed a framework able to orchestrate a set of internal and user-defined *plug-ins*, each dedicated to a single task. By specifying inputs and outputs, it is possible to create a dependency graph where each sub-module represents a node, and each pair of matching input-output represents an edge. In this way, given the desired output, a *topological sort* is executed to minimize and linearize the sequential execution of computations. Although an easier solution may exist in the context of the plugins we developed and hereafter describe, this methodology is rather simple to implement and, moreover, allows for sequencing complex compute graphs with additional sub-modules.

3.1. Detecting Objects

The object detector is the system's main component on which almost all other plug-ins rely on. Its primary purpose is to localize and classify pedestrian instances from input images. These detections constitute the main data that will be exploited, in different ways, by the other nodes of the system.

We base our object detector on *Faster R-CNN* [27], a popular state-of-the-art CNN-based object detection system. It operates as a two-stage algorithm, exploiting two different modules during the different phases of its detection pipeline. In the first stage, a CNN acts as a backbone by extracting input image features. Starting from this features' space, the Region Proposal Network (RPN) is responsible for generating the region proposals that might contain objects, slicing pre-defined region boxes (called anchors), and ranking them, suggesting the ones most likely containing objects. The second module is the Fast R-CNN detector [15] that uses the proposed regions, and it is in charge of classifying and localizing the objects inside them, outputting class scores and bounding boxes coordinates.

We modify this detection system to meet our needs, making it lighter and more suitable for running on computational- and resource-limited devices. Moreover, we specialize it in detecting people instances, performing a supervised domain adaption that exploits several pedestrian datasets. We detail all these strategies in Section 4.

3.2. Tracking

Object tracking can be an essential tool to increase the robustness to spurious detections and achieve temporal consistency in video analysis. To this end, we implement and apply an object tracker over pedestrian detection to reidentify people among consecutive video frames. This step is particularly useful for assessing temporal rules, such as raising an alarm after the same pedestrian occupies a forbidden area for more than a predefined amount of time.

The implementation of the tracker follows the formulation of DeepSort [38]; it is based on SORT [5], a simple causal tracking algorithm for 2D objects in which targets are represented by the position and area of the bounding box and their speed of variation. The state of each target (also known as *tracklet*) is updated with available detections using a Kalman filter framework. DeepSort builds upon SORT by adding a matching scheme between predicted and actual targets based on feature vectors that describe the appearance of tracked objects; tracklets can be confirmed if the cosine score between feature vectors of the predicted and actual target is above a programmable threshold. Feature vectors in DeepSort must be provided by extracting representations from detected regions with an additional pretrained network. In our implementation, we avoid this step by reusing the feature vectors of detected regions that the object detection network has already extracted; in particular, we perform a Region of Interest (RoI) average pooling of the features extracted by the CNN backbone using only the regions provided by the pedestrian detection module.

3.3. Crowd Counting

In some scenarios where individual and collective safety has to be constantly monitored, like people aggregations during the recent COVID-19 pandemic, estimating the number of people present in a region of interest is crucial to monitor the area occupancy. By measuring, and limiting, the number of people who can visit a location at any one time, it is possible to drastically reduce the likelihood of setting up people gatherings and, consequently, minimizing human virus transmission. Our solution relies on a dedicated plug-in that can work in two different modalities that the user can conveniently pick out, depending on the considered scenario. The first one, named *Counting by Instances*, is better suited for not particularly crowded environments and relies on the object detector described in Section 3.1. The second, named *Counting by Density Estimation*, is instead a more holistic approach most appropriate for highly crowded scenarios; it aims to compute a mapping between the features of the captured image and its pedestrian density maps, skipping the detection of the single instances. The estimated number of people present in the controlled area can then be obtained by integrating this density map. In the following paragraphs we describe in detail both modalities.

Counting by Instances. This counting modality depends entirely on the pedestrian detector. The pedestrian detection module provides the module's input, consisting of the localized pedestrian instances. The counting plug-in, when this modality is enabled, is only responsible for counting them. As already mentioned, this approach has some limitations in highly crowded scenarios since, in this case, people instances are heavily occluded and not easily identifiable.

Counting by Density Estimation. This modality tackles the counting task as a supervised regression problem from the image features to an associated density map, following the seminal work [20], avoiding the detection of individual object instances. As mentioned above, this approach is particularly attractive in highly congested scenarios, where the instances of the objects are not completely visible due to the occlusions. The module's input consists directly of the captured image, so this node, when this modality is enabled, did not depend on the object detector module.

In this scenario, the most widely used labels needed for the supervised training are the dotted annotations, obtained by putting a single dot on each object instance in each image. Formally, we assume to have a set of N training images $I_1, I_2, ..., I_N$. We also assume that each image I_i is labeled with a set of 2D points $\mathbf{P}_i = P_1, ..., P_{K(i)}$, where $K(i)$ is the total number of annotated objects (in our case pedestrians). For a training image I_i, we define the ground truth density map as

$$\forall p \in I_i, \quad H_i(p) = \sum_{P \in \mathbf{P}_i} \delta(p - P). \qquad (1)$$

Here, p denotes a pixel, while a point identifying a pedestrian is represented as a delta function. Converting it into a continuous density function with Gaussian kernel G_σ we obtain

$$\forall p \in I_i, \quad F_i(p) = \sum_{P \in \mathbf{P}_i} \delta(p - P) * G_\sigma. \qquad (2)$$

The sum of the density map is equivalent to the total number of pedestrians. It is worth noting that the Gaussian spread parameter σ depends on the size of each pedestrian in the image, considering the perspective transformation. However, it is almost impossible to obtain the occluded object's size manually in a high-density environment. So this parameter is a dataset-specific quantity empirically estimated. Then, given a set of training images together with their ground truth densities, we aim to learn a transformation of the feature representation of the image that approximates the density function at each pixel so that it minimizes the sum of the mismatches between the ground truth and the estimated density functions (the *loss* function).

We build our density map estimator upon the Congested Scene Recognition Network (CSRNet) [21], a CNN-based algorithm that can understand highly congested scenes and perform accurate density estimation. It comprises two major components. For the image features extraction, it exploits a modified version of the well-known VGG-16 network [33], where the final classification part, i.e., the final fully-connected layers, is removed. The output size of this front-end network is 1/8 of the original input size. Following other works [41, 7, 8], a back-end composed of dilated convolutional layers are stacked upon this front-end to extract deeper information of saliency and, at the same time, maintaining the output resolution. Using dilated convolutions, we can deliver larger reception fields while replacing pooling operations (e.g., the max pool operation) that are often responsible for losing quality in the density generation procedure.

3.4. Measuring Social Distances

A critical condition that must be kept under control in dynamic environments where an infection is ongoing is represented by the physical distance among individuals. In the case of air-borne diseases, it is thus very common to issue rules to avoid people gatherings in confined places and keep a specific reciprocal separation to contrast the spread of pathogen agents. Although crowd counting is effective in monitoring aggregations, measuring distances among people becomes critical during pandemic events. Assuming that individuals mostly hang out on the same planar floor, we decided to measure their actual distance by applying a simple pre-calibrating step to the fixed monitoring camera, using a proper geometrical transformation that places detected items on a common system of reference. Our solution is to pre-compute a mapping between real points in the scene whose relative position is known, and their projection on the acquired frame image. This process is well known in Computer Vision and consists of finding a *homography*, i.e., a perspective transformation that projects on two different points of view a set of 3D points lying on the same plane.

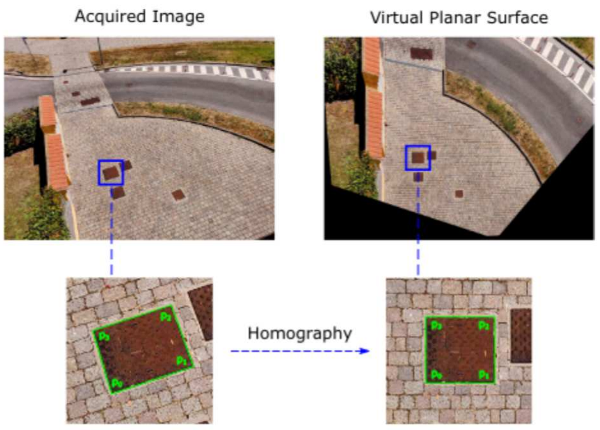


Figure 1: An example of homography. A set of four points in the acquired image is mapped through homography to its projection on an euclidean metric space laying on a virtual planar surface.

More in detail, a d-dimensional homography represents the linear operation

$$x' = Hx \quad x, x' \in \mathbb{R}^d$$

and is expanded in homogeneous 3D coordinates as an approximation of the projection operation of a pinhole camera, represented by a 4×4 matrix with the translation and projective components added. In the case of coplanar points, we can consider the input z component as a constant and thus eliminate it from the above formulation:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Thus, the eight h_{ij} unknown can be found if we can relate two sets of four 2D points through homography. To calculate its elements, we minimize the matching error of the two sets after perspective division (i.e., dividing by the homogeneous component) to bring coordinates from homogeneous to euclidean space:

$$\begin{aligned} w' &= h_{31}x + h_{32}y + h_{33} \\ x' &= \frac{h_{11}x + h_{12}y + h_{13}}{w'} \\ y' &= \frac{h_{21}x + h_{22}y + h_{23}}{w'} . \end{aligned}$$

We can set up the projected image of four coplanar (but not collinear) points to a virtual flat surface and then use the found transformation matrix to approximate the relative location of any point laying on the same plane, as shown in Figure 1. This calibration step is kept as straightforward as possible for an untrained operator, as it is easy to select four points of a known-sized quadrangle (e.g., a standardized manhole). As perturbations may occur with point selection, we also allow to select more than four coplanar points whose relative position is known, and then use a *random sample consensus* (RANSAC [14]) algorithm to iterate through a random selection of four points for a first approximation of the plane equation, which is eventually refitted to minimize the error.

3.5. Detecting Personal Protective Equipment

A simple intervention for protecting health and well-being is wearing Personal Protective Equipment (PPE). This is particularly true in dangerous working environments, such as wearing harnesses and helmets on construction sites, but it also became evident in light of the recent COVID-19 pandemic, where wearing face masks can prevent infections. Therefore, we implement a module dedicated to detecting worn PPE, essential for ensuring compliance with regulations that imply personal protection.

Our solution for PPE detection follows the same methodology already adopted for pedestrian detection: specifically, we adopt the same detector architecture based on Faster-RCNN (see Section 3.1 for details). Differently from the pedestrian detector, the PPE detector network takes as input a rather small image depicting a pedestrian, and it is trained to distinguish and detect several classes of worn PPE, i.e., surgical/face masks, helmets, and high-visibility vests. The module's input is provided by the pedestrian detection module: once detected, the patch of the video frame depicting a pedestrian is given as input to the PPE detector that provides bounding boxes of PPE if the pedestrian wears them.

Conceptually, the detection of PPE could be tackled by the object detector module by adding the PPE classes to the base detection model. However, we empirically noticed that merging the PPE detection with the pedestrian detection module leads to performance degradation in both tasks. Using separate modules provides a more flexible solution in which the input image resolution of both detectors can be adjusted separately and better adapted to the monitored scenario. For example, when wide areas are monitored, PPE detection can be performed several times at each processed video frame depending on how many pedestrians have been detected, while the PPE detection network can be configured to operate on smaller input sizes, maintaining an affordable computational cost.

4. Datasets and Architecture Adaptions

A key point in producing a verification system that can generalize on a broad spectrum of working conditions is to generate a training set based on an adequately large amount of environmental conditions. In our case, this means being able to access a massive amount of images involving people under different environmental scenarios. Since manually annotating new collections of images is expensive and requires a notable human effort, a recently promising approach is to gather data from virtual world environments that resemble the characteristics of the real-world scenarios and where the labels can be acquired with an automated process. Thus, in this work, we build vast training datasets, considering both real-world and synthetic images from public datasets when available, and collecting others when needed, covering a multitude of different scenarios and contexts. Hereafter, we describe these data, dividing them according to the module for which they are employed. Furthermore, we describe the exploited training procedures, highlighting the changes we made to the architectures to adapt them to our specific scenarios.

4.1. Datasets for Object Detection

To train the object detector module, we use many popular publicly available pedestrian detection datasets. Furthermore, we introduce a novel dataset, named *CrowdVisorPisa*, that we also employ for evaluating our solution. In the following, we detail all the exploited datasets.

Virtual Pedestrian Dataset (ViPeD) [3, 9]. The Virtual Pedestrian Dataset is a *synthetic* collection of images generated exploiting the highly photo-realistic graphical engine of the video game Grand Theft Auto V (GTA V) by Rockstar North. It comprises about 500K images belonging to 512 different urban environments (256 for training and 256 for testing) characterized by various weather conditions, illumination, perspectives, viewpoints, and density of people. Labels are *automatically* generated and consist of bounding boxes precisely localizing the pedestrians present in the scenes.

MOT17Det [23] and MOT20Det [23]. The MOT17Det and MOT20Det datasets are two collections of images (5,316 and 8,931, respectively), annotated with bounding boxes, taken from multiple sequences describing crowded scenarios having different characteristics, like viewpoints, weather conditions, and camera motions. The authors provided training and test subsets, but they released only the ground-truth labels for the former. The main difference between MOT20Det compared to MOT17Det is that the first contains more crowded scenarios.

CityPersons [42]. The CityPersons dataset consists of a set of stereo video sequences recorded from a moving car in streets from different cities in Germany and neighboring countries. In particular, the authors provide 5,000 frames from 27 cities labeled with bounding boxes and split across train/validation/test subsets.

CrowdHuman [32]. CrowdHuman is a benchmark dataset for pedestrian detection. It comprises 15,000, 4,370, and 5,000 images for training, validation, and testing, respectively, describing diverse, crowded scenarios, with an average number of persons in an image of 22.6. The authors annotated each human instance with a head bounding box, a human visible-region bounding box, and a full-body bounding box.

PRW [45]. The PRW dataset contains 11,816 frames where 932 different pedestrian identities are annotated with their bounding boxes. The authors provide the training and the test splits.

CUHK-SYSU [39]. The CUHK-SYSU is a large-scale benchmark dataset containing 18,184 images, 8,432 different identities, and 96,143 pedestrian bounding boxes. It is divided into training and test subsets.

CrowdVisorPisa (ours). The CrowdVisorPisa dataset is a novel collection of images that we collected and annotated on purpose for this work. In particular, we stored 15 different sequences gathered from a webcam located in a public square of the city of Pisa, Italy, each of which comprises ten images captured with a time interval of 1 second. We manually labeled all frames, localizing the pedestrian instances with bounding boxes. Furthermore, we also annotated each sequence taking track of the different pedestrian entities entering or exiting the scene. We divided the dataset into train and test splits, considering 10 and 5 different sequences, respectively. The former split is exploited to train the object detector module, while the latter to evaluate the performance of some modules of our framework. It is worth noting that, due to camera positioning not modifiable for local restrictions, this dataset represents a particularly challenging scenario as people instances are small and sometimes difficult to localize. A sample of the dataset is shown in Figure 2.

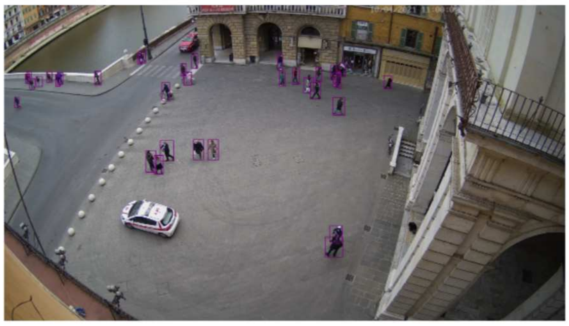


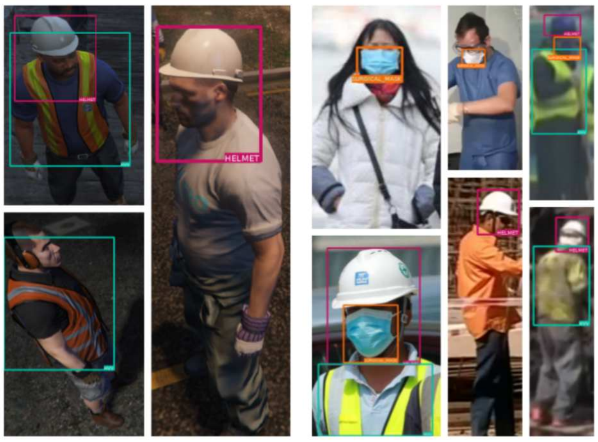
Figure 2: A sample from our novel *CrowdVisorPisa* dataset, together with bounding box annotations localizing pedestrians.

4.2. Datasets for PPE Detection

CrowdVisorPPE (ours). We collect and annotate a novel dataset (named *CrowdVisorPPE*) to train and evaluate our PPE detection module. It comprises 54,017 images representing pedestrians with and without wearable PPE. Roughly half of the dataset comprises synthetic images procedurally generated using the GTA V video game engine as in [10], whereas the other half comprises photographic real-world images taken from the Web and manually annotated. The PPE classes of interest, i.e., helmets, high-visibility vests (HVVs), and face masks, are annotated with bounding boxes. The real-world subset is the only source of face mask instances since they are not available for rendering in GTA V. We hold out a subset of real images as the test split, whereas synthetic images and the remaining real ones form the training split. We show the dataset details in Table 1 and some samples in Figure 3.

		# PPE instances		
Train Split	# img	Helmet	HVV	Mask
GTA V (**V**)	28,078	9,575	21,374	0
Web (**R**)	21,820	10,673	10,686	1,630
Test Split				
Web (**R**)	4,119	2,163	2,017	271

Table 1: Details of the *CrowdVisorPPE* dataset. **V** = virtual/synthetic data; **R** = real/photographic data.



(a) GTA V (Virtual) (b) Web (Real)

Figure 3: Samples from our novel *CrowdVisorPPE* dataset. PPE classes are color coded: helmet, high-visibility vest, face mask.

4.3. Datasets for Crowd Counting by Density Estimation

To train the module responsible for crowd counting by density estimation, we exploit many popular publicly available datasets, detailed in the following.

GTA5 Crowd Counting (GCC) [36]. The GCC dataset is a large-scale and diverse synthetic crowd counting dataset, gathered from the video-game Grand Theft Auto V (GTA5) and automatically annotated. It consists of 15,212 images, with a resolution of 1080×1920, containing 7,625,843 persons in 400 different scenarios with various locations, weather conditions, and crowd densities. Compared with the existing datasets, GCC is a more large-scale crowd counting dataset in both the number of images and persons.

ShanghaiTech [44]. The ShanghaiTech dataset is a large-scale crowd dataset of nearly 1,200 manually dot-annotated images with a total of 330,165 people with centers of their heads. This dataset consists of two parts: part A, containing 482 images crawled randomly from the Internet, and part B, composed of 716 images taken from the busy streets of metropolitan areas in Shanghai. The crowd density varies significantly between the two subsets, making this dataset more challenging. The two parts are divided into training and testing subsets: 300 images of part A are used for training, and the remaining 182 images for testing, while 400 images of part B are for training and 316 for testing.

UCF-QNRF [17]. The UCF-QNRF dataset is a collection of images gathered from three sources: Flickr, Web Search, and the Hajj footage. The authors performed the entire annotation process in two stages, the first one for the labeling and the second one for the verification, for a total of 2,000 human-hours spent through to its completion. This dataset comprises 1,535 images with more than 1 million dot-annotations on the centers of the pedestrian's heads, divided into training and test subsets.

NWPU-Crowd [35]. The NWPU dataset is a large-scale congested crowd counting and localization dataset consisting of 5,109 images, in a total of 2,133,375 annotated heads with points and boxes. Compared with other real-world datasets, it has the most extensive density range. Another peculiarity of this dataset is that it also comprises some negative samples like high-density crowd images to assess the robustness of models.

4.4. Adaptation for Object Detection

To make the object detector able to run efficiently directly on computational- and resource-limited devices, we employ, as the backbone of Faster R-CNN, the *ResNet50* architecture, a lighter version of the popular *ResNet101* [16]. We start considering the detector pre-trained on the *COCO* dataset [22], a large collection of images depicting complex everyday scenes of ordinary objects in their natural context, categorized into 80 different classes. In our case, we have to localize and identify objects belonging to just one class (i.e., pedestrian). To this end, we further simplify the model by reducing the number of the final fully convolutional layers responsible for classifying the detected objects, making our detector lighter. We call *Light* this modified version of the object detector module to distinguish it from the *Full* original one, having instead the *ResNet101* backbone and a larger number of fully connected layers.

Intending to specialize the detector in finding the specific pedestrian object category, we adopt a supervised *domain adaptation* strategy, exploiting the datasets described in Section 4.1 and fine-tuning the network to this specific task. Following Ciampi et al. approach [9], we employ the *Balanced Gradient Contribution* (BGC) [28, 29] strategy, where, during the training phase, we mix the synthetic data, taken from *ViPeD*, and the real-world images gathered from the remaining datasets. In this way, we take advantage of the great variability and size of *ViPeD*, and at the same time, we mitigate the existing domain shift between these synthetic data and the real-world ones. In particular, during the training phase, we exploit batches composed of 2/3 of synthetic images and 1/3 of real-world data, thus considering statistics from both domains throughout the entire procedure and where the real-world data acts as a regularization term over the synthetic data training loss.

4.5. Adaptation for PPE Detection

As in the object detector, we adopt the Faster R-CNN model with the *ResNet50* backbone as the PPE detector architecture. The methodology used to obtain the trained PPE detector follows Di Benedetto et al. approach [10] and reaches a comparable detection performance: we start from a detector pre-trained on COCO with a new detection head that matches the number of the PPE classes, and then we use a mixture of synthetic and real images of pedestrians with PPE when training the model, to finally testing it on real data only. The only difference concerning the object detector and [10] is that we perform PPE detection only on pre-segmented patches containing a single pedestrian instead of searching for PPE in the full-frame. This simplifies the task for the model and enables us to save computational budget by processing smaller images.

4.6. Adaptation for Crowd Counting by Density Estimation

To train the crowd counter by density estimation module, we adopt a supervised domain adaptation strategy consisting of training the network with the synthetic data and then fine-tuning it exploiting the real-world images, as explained in [9]. In particular, we set the initial weights of the network layers with values coming from a Gaussian distribution with 0.01 standard deviation. Then, we train the network exploiting the GCC dataset, and, finally, we fine-tune it using the real-world data.

5. Experiments and Results

We evaluate all the modules making up our framework, considering different scenarios and exploiting appropriate metrics depending on the relating task. For all the experiments, we consider the *Light* version of our object detector module since it has shown similar performance compared with the *Full* version, and it is more appropriate to be used in combination with low-cost and computational-limited hardware.

Being our target a deployable monitoring system, we selected the NVIDIA Jetson TX2 embedded device as the hardware host. It is composed of two 64 bit CPUs with two and four cores each, an NVIDIA Pascal GPU with 256 CUDA cores, 8 GB of RAM shared between the system and the graphics accelerator, and a 32 GB solid-state storage volume. The operating system is the NVIDIA's Linux4Tegra (L4T) distribution based on Ubuntu. We installed Python 3.8 with OpenCV 4.5 and the deep learning framework PyTorch 1.8. As detailed in Table 2, memory usage is kept within 5 GB of both system and GPU RAM. An external USB camera completes the whole installation.

5.1. Counting by Instances

In this setting, we test and evaluate the counting by instance functionality. We consider our *CrowdVisorPisa* dataset and, in particular, the five sequences belonging to the test subset, performing two different sets of experiments over it: the first one that involves only the object detector module and the second that instead takes also into account the tracker module. More in detail, in the first case, we evaluate the effectiveness of our framework to estimate the number of people present in the single frames, while in the second scenario we also consider the temporal relation that exists between consecutive images, tracking the found pedestrian instances over time.

We report in Figure 4 the obtained results concerning the first scenario. Each row of the figure represents a different sequence. In the first column

Table 2: System and GPU Memory Usage in GB. **OD** = object detector model type; **DC** = whether the density counter module is active; **PPE** = whether the PPE detector module is active. The modular framework is assumed to always use the object detector in its *Light* or *Full* models, along with the enabled distance measure plug-in that consumes a fixed and negligible (less than 1 MB) amount of memory. Video stream size is 1173 × 880 RGB pixels. System memory is calculated with `/usr/bin/time -f "%M"`, GPU memory with `torch.cuda.max_memory_allocated()`.

OD	DC	PPE	SysRAM	GpuRAM
Light	✗	✗	2.36	0.55
	✓	✗	2.44	0.86
	✗	✓	2.35	2.10
	✓	✓	2.44	2.20
Full	✗	✗	2.51	0.62
	✓	✗	2.52	0.94
	✗	✓	2.51	2.20
	✓	✓	2.51	2.30

we show the number of people that our detector module can localize for each frame making up a sequence, varying the detection threshold. In the second column we instead illustrate the errors in terms of counting for each frame, changing again the detection threshold. We also report, for each sequence, the best Mean Absolute Error (MAE), i.e., the mean of the sum of the absolute errors, obtained with a specific threshold. As can be seen, we obtain a MAE close to 1 or 2, depending on the considered scenario, demonstrating that the module provides a reliable estimation of the number of pedestrians present in the monitored scene.

On the other hand, in Figure 5 we show the results concerning the second scenario. Each row of the figure corresponds to a different sequence. We report the results about the single frames making up a sequence for three different detection thresholds, one for each column. In particular, we indicate the pedestrians that enter and exit from the scene at each frame, exploiting the tracklets provided by the tracker module that represents the recognized identities of the people instances over time. We notice that with a low (resp. high) threshold value our system tends to overestimate (resp. underestimate) the total number of people present in a sequence, finding thus its optimal threshold values in the 0.5 - 0.6 range. We also note that false positives detections tend to create spurious peaks in the people count that are however often recovered in the immediate/next following prediction/frame.

Table 3: Evaluation metrics of the Object Detection (**OD**), Density Counter (**DC**), and **PPE** Detector modules, measured on the corresponding test sets. The mean Average Precision (mAP) measures the average precision of the detection when varying the score threshold in detection-based modules (**OD**, **PPE**). For **DC**, MAE and RMSE measure the counting error, while SSIM measures the quality of the predicted density map.

OD	**PPE**	**DC**		
mAP ↑	mAP ↑	MAE ↓	RMSE ↓	SSIM ↑
0.836	0.606	92.28	365.4	0.79

5.2. *Counting by Density Estimation*

Given that the annotation procedure for labeling datasets having these characteristics is highly costly in terms of manual human effort, we exploited the test subsets of the already publicly available datasets described in Section 4.

We report in Table 3 the obtained results in terms of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). It is worth noting that, as a result of the squaring operation, RMSE effectively penalizes large errors more heavily than small ones, thus more suitable when outliers are particularly undesirable. Furthermore, we also compute the Structural Similarity Index Measure (SSIM) [37] to measure the density map quality, which measures images' similarities under three aspects: brightness, contrast, and structure. The value of SSIM is in the $[0, 1]$ range: the larger it is, the less distortion of the image is measured. Finally, in Figure 6 we show some examples of the considered images, together with the ground truth and the predicted density maps.

5.3. *Detecting Pedestrians and Personal Protection Equipment*

We validate the detection of pedestrian and worn PPE, performed respectively by the Object Detector and the Personal Protection Equipment Detector modules. For the former, we focus on the five test sequences of our *CrowdVisorPisa* dataset, whereas for the latter, we consider the *CrowdVisorPPE* test subset.

We evaluate the two modules using the mean Average Precision (mAP), a popular metric in measuring the accuracy of object detectors that computes the average precision value for recall values spanning 0 to 1. We set an Intersection over Union

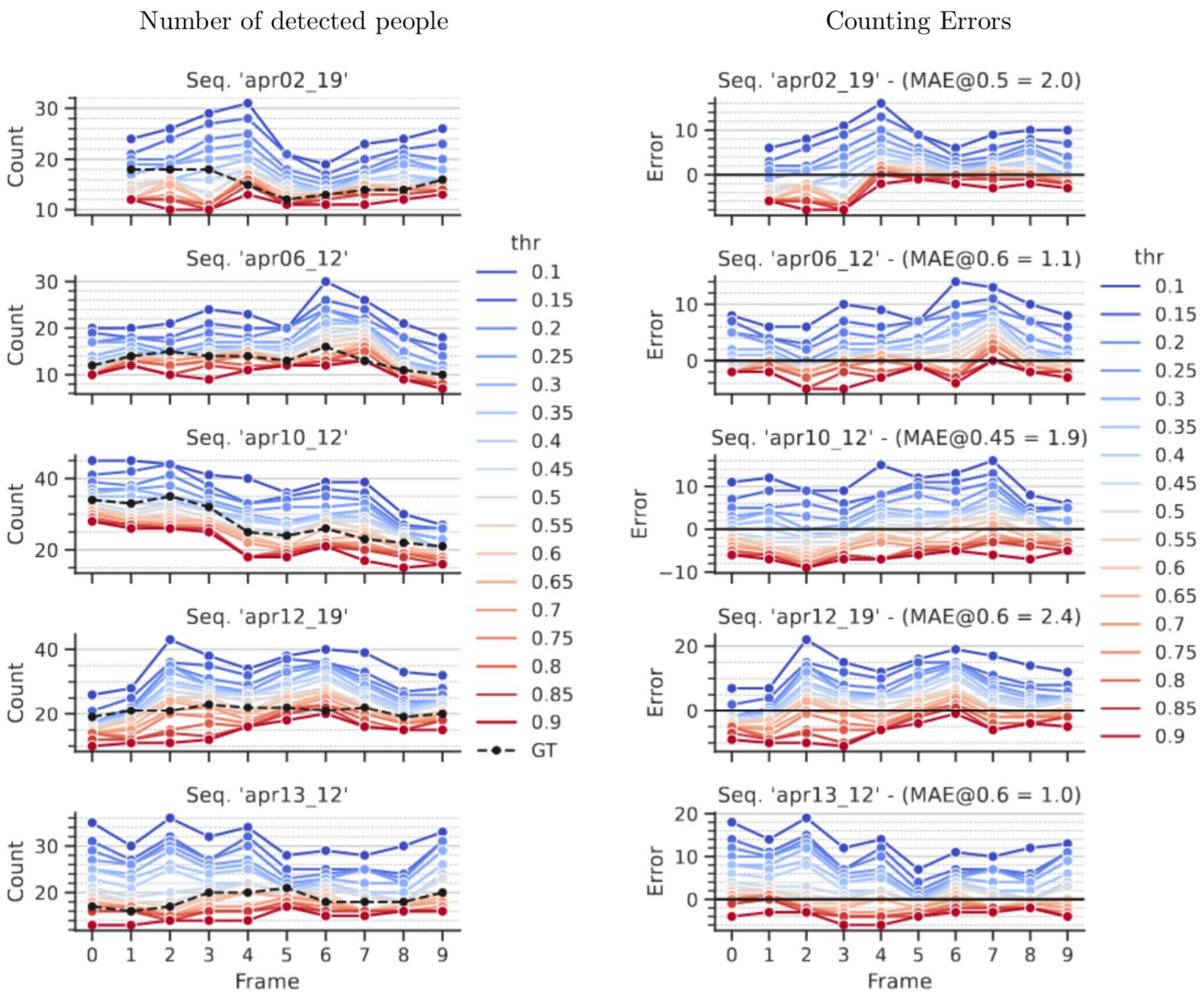


Figure 4: Evaluation of *counting by instances* functionality of our framework, considering the single still frames of the five test sequences of our *CrowdVisorPisa* dataset. In the first column we report the number of people located by our detector, varying the detection thresholds. In the second column we show the counting errors and the best MAE obtained with a specific detection threshold.

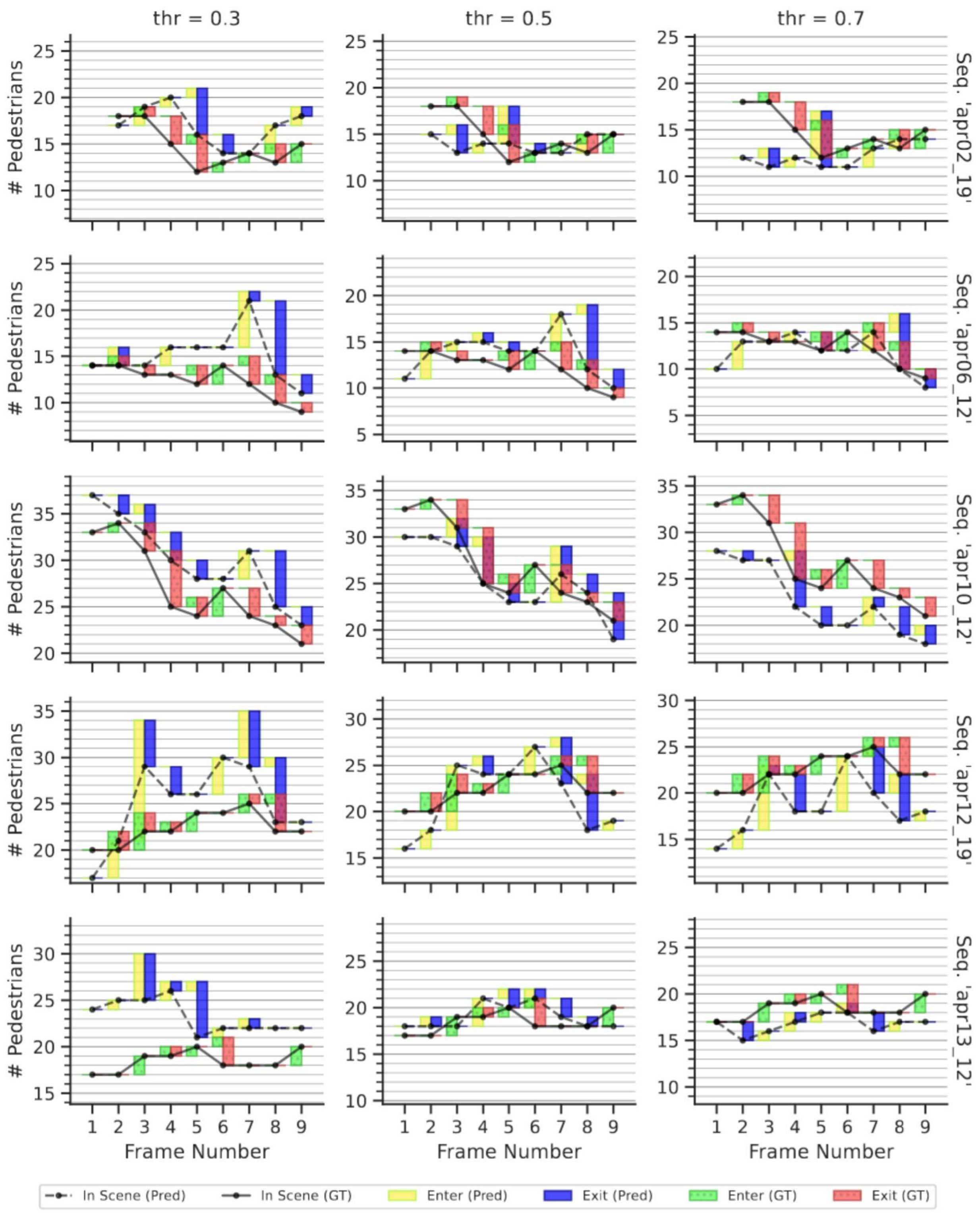


Figure 5: Evaluation of the counting by instances functionality of our framework, taking into account also the tracker module. We consider the five test sequences of our *CrowdVisorPisa* dataset, reported one for each row. Columns represent the results obtained for three different detection thresholds. For each plot, we show the pedestrians that enter and exit from the scene, relying on the tracklets describing the recognized identities of the people instances over time.

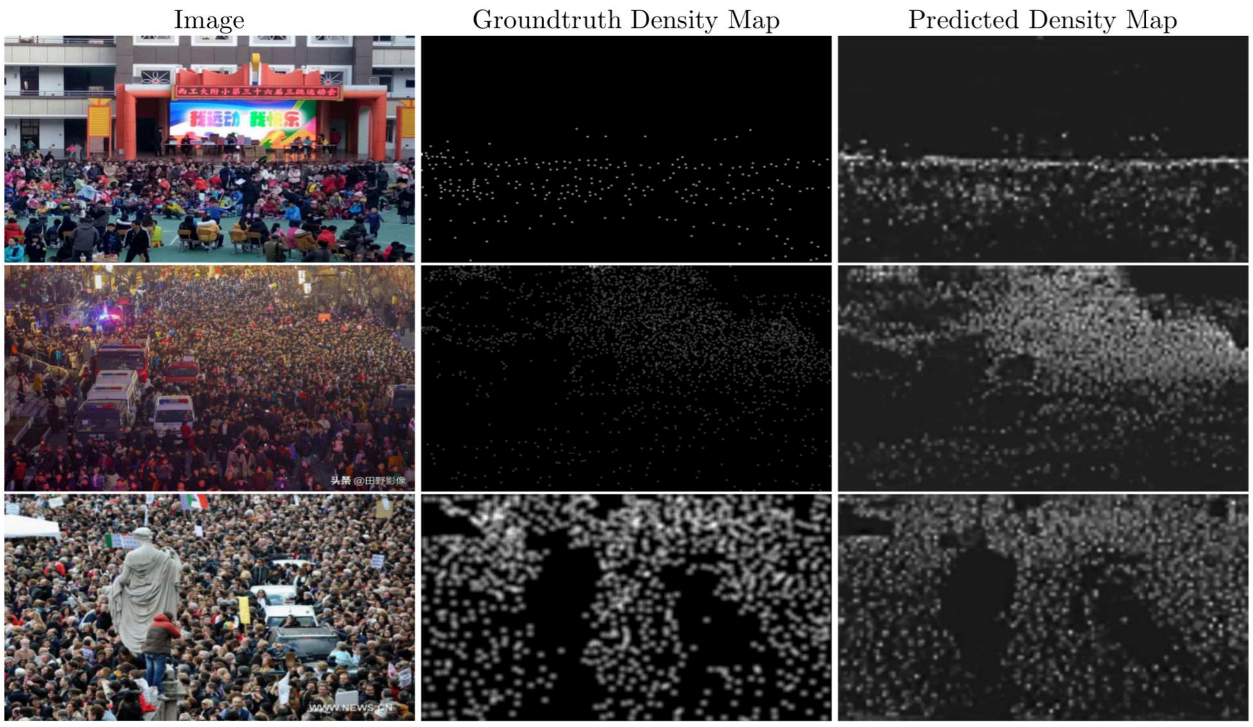


Figure 6: Some samples of the test subsets exploited for the evaluation of our *counting by density* estimation functionality, together with the ground truth and the predicted density maps. Integrating these density maps, i.e., summing up the pixel values, we can obtain an estimation of the people present in the image.

(IoU) threshold of 0.5 to assess whether a prediction is a true positive or a false positive. We report the obtained results in Table 3, showing that our modules can reach a mAP of 0.836 and 0.606 for the pedestrian detection and the PPE detection tasks, respectively. Figure 7 shows some predictions of PPE detections on the CrowdVisorPPE test set.

5.4. Measuring Social Distances

To establish a correspondence between the acquired image and a planar *metric* surface onto which objects (i.e., pedestrians) positions can be evaluated, we used a known-sized manhole in the monitored scene to calculate the homography exploited to unwarp pixel position into real-world relative locations. The homography reprojection is a closed-form mathematical process, so no previous training is needed.

Our measuring results can be seen in Figure 8: in the examples shown, the precision of measurements is relative to both the initial calibration (i.e., manhole real size mapped to its projection on the input image) and the accuracy of the pedestrian bounding boxes (i.e, rectangles) predicted by the object detector. We measured the manhole with an upper bound precision of 1 cm and a pixel area error of about 3 cm, thus confining the overall measurements below the 10 cm error. For pedestrian positions, we used the midpoint of the lower edge of its predicted bounding box. As can be seen from the gridded unprojection of the examples, results are consistent within the above error gap.

6. Conclusion

In this work, we presented a modular framework based on Computer Vision and AI technologies, deployed in a real use-case scenario on a low-cost off-the-shelf embedded platform and aimed at monitoring human activities in critical conditions. As an effective setup, we implemented a set of visual-based modules for pedestrian detection, tracking, aggregation counting based on instances and density maps, social distancing calculations, and personal protection environment detection. Specifically, we trained artificial neural models with publicly available and, for the purpose of the physical device installation, custom datasets; at the same

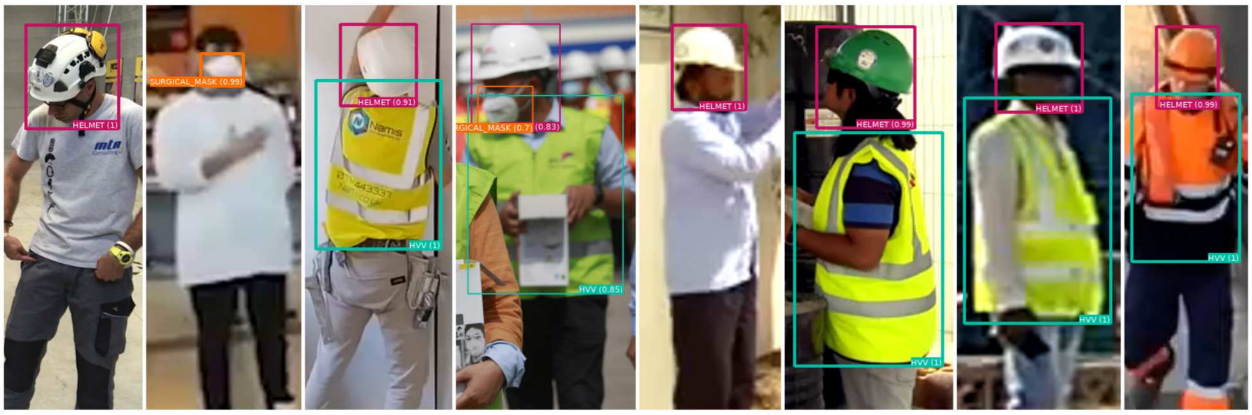


Figure 7: Examples of predictions of the PPE Detection module on our CrowdVisorPPE test set. PPE classes are color coded (helmet, high-visibility vest, face mask), and the detection score is reported in parenthesis.

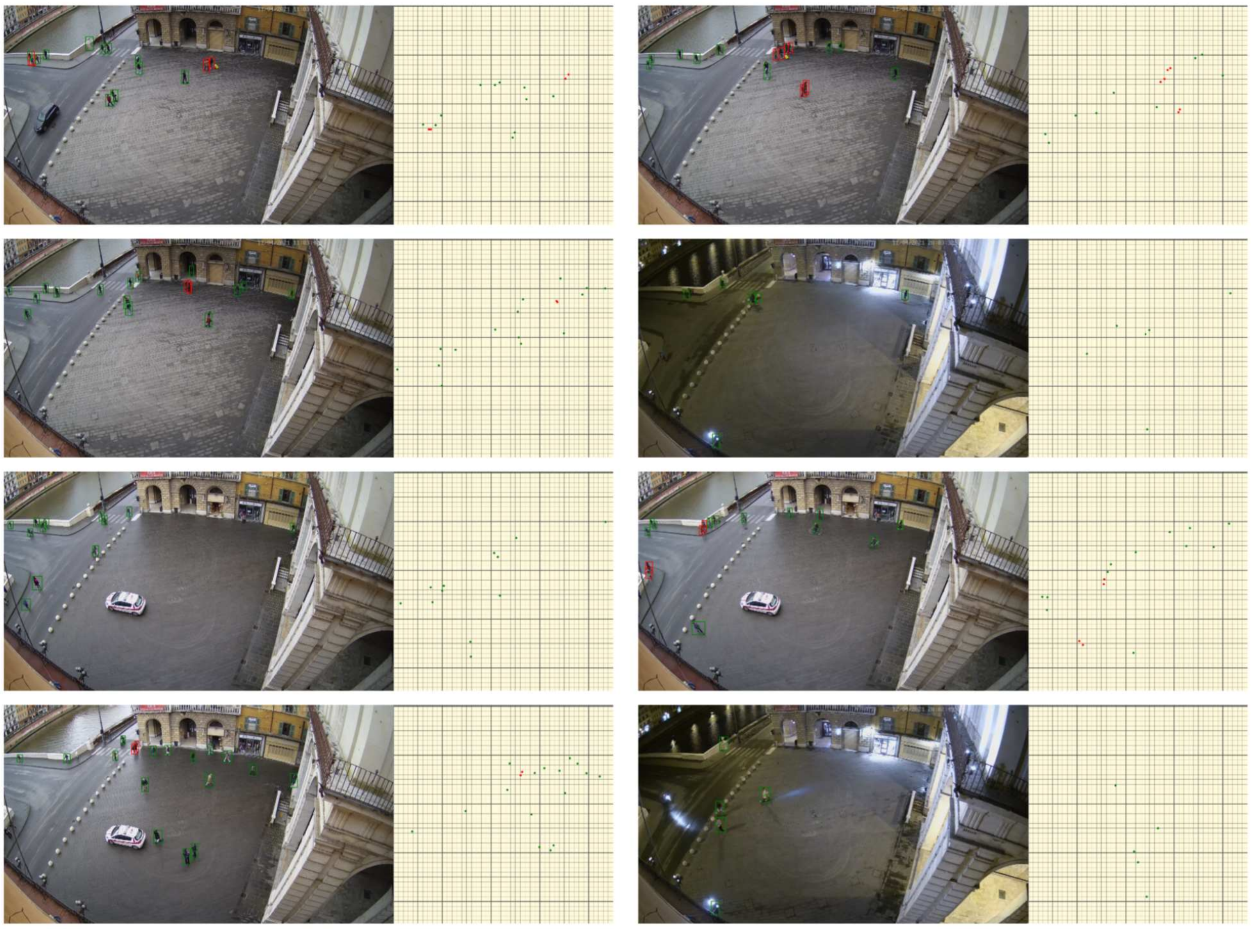


Figure 8: Examples of detections and distance warnings under different lighting conditions. Each of the eight images represents, in its left side, pedestrians detected and tracked in the example scenario, while showing on its right side their 2D projection on a virtual planar surface obtained through homography, with a reference 1-meter-spacing overlay grid. Green color means a safe placement, red color indicates violations of the 1-meter physical distance rule.

time, we applied a transfer learning approach to expand detection capabilities by using computer-generated training imagery. To test the effectiveness of our solution, we monitored a known place in Italy during the restrictions imposed from the COVID-19 pandemic, proving satisfactory accuracy in terms of detection, counting, and physical distance measurements. In addition, the modularity of our framework allows us to embed enhanced or more target-specific plug-ins in novel system installments.

6.1. Future Work

We plan in the next iteration of this project to develop an algorithm that can automatically select the best counting modality between instancing and density map, which is currently chosen manually by the user. At the same time, we will try to integrate and expand modules with further visual analyses, like gesture/posture recognition, and the assessment of appropriate PPE wearing. Finally, we will attempt to apply a transfer learning approach to predict physical distances among people by using an automatically labeled computer-generated training set based on a rendering engine simulation.

Acknowledgements

This work was partially supported by the H2020 projects AI4EU (GA 825619) and AI4Media (GA 951911), and by the Tuscany POR FSE 2014-2020 project AI-MAP (CNR4C program, CUP B15J19001040004). We would like to thank Visual Engines S.r.l. - https://www.visualengines.com - for supporting this research and giving us access to their data sources collected for the CrowdVisor project.

References

[1] Ahmed, I., Ahmad, M., and Jeon, G. (2021a). Social distance monitoring framework using deep learning architecture to control infection transmission of covid-19 pandemic. *Sustainable Cities and Society*, 69:102777.

[2] Ahmed, I., Ahmad, M., Rodrigues, J. J., Jeon, G., and Din, S. (2021b). A deep learning-based social distance monitoring framework for covid-19. *Sustainable Cities and Society*, 65:102571.

[3] Amato, G., Ciampi, L., Falchi, F., Gennaro, C., and Messina, N. (2019). Learning pedestrian detection from virtual worlds. In Ricci, E., Bulò, S. R., Snoek, C., Lanz, O., Messelodi, S., and Sebe, N., editors, *Image Analysis and Processing - ICIAP 2019 - 20th International Conference, Trento, Italy, September 9-13, 2019, Proceedings, Part I*, volume 11751 of *Lecture Notes in Computer Science*, pages 302–312. Springer.

[4] Benfold, B. and Reid, I. D. (2011). Stable multi-target tracking in real-time surveillance video. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 3457–3464. IEEE Computer Society.

[5] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE.

[6] Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934.

[7] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848.

[8] Chen, L., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587.

[9] Ciampi, L., Messina, N., Falchi, F., Gennaro, C., and Amato, G. (2020). Virtual to real adaptation of pedestrian detectors. *Sensors*, 20(18):5250.

[10] Di Benedetto, M., Carrara, F., Meloni, E., Amato, G., Falchi, F., and Gennaro, C. (2020). Learning accurate personal protective equipment detection from virtual worlds. *Multimedia Tools and Applications*, pages 1–13.

[11] Eyiokur, F. I., Ekenel, H. K., and Waibel, A. H. (2021). A computer vision system to help prevent the transmission of COVID-19. *CoRR*, abs/2103.08773.

[12] Fabbri, M., Lanzi, F., Calderara, S., Palazzi, A., Vezzani, R., and Cucchiara, R. (2018). Learning to detect and track visible and occluded body joints in a virtual world. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, volume 11208 of *Lecture Notes in Computer Science*, pages 450–466. Springer.

[13] Fabbri, M., Lanzi, F., Gasparini, R., Calderara, S., Baraldi, L., and Cucchiara, R. (2020). Inter-homines: Distance-based risk estimation for human safety. *CoRR*, abs/2007.10243.

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

[15] Girshick, R. B. (2015). Fast R-CNN. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1440–1448. IEEE Computer Society.

[16] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

[17] Idrees, H., Tayyab, M., Athrey, K., Zhang, D., Al-Máadeed, S., Rajpoot, N. M., and Shah, M. (2018). Composition loss for counting, density map estimation and localization in dense crowds. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*, volume 11206 of *Lecture Notes in Computer Science*,

pages 544–559. Springer.

[18] Khandelwal, P., Khandelwal, A., Agarwal, S., Thomas, D., Xavier, N., and Raghuraman, A. (2020). Using computer vision to enhance safety of workforce in manufacturing in a post COVID world. *CoRR*, abs/2005.05287.

[19] Kong, X., Wang, K., Wang, S., Wang, X., Jiang, X., Guo, Y., Shen, G., Chen, X., and Ni, Q. (2021). Real-time mask identification for covid-19: An edge computing-based deep learning framework. *IEEE Internet of Things Journal*, pages 1–1.

[20] Lempitsky, V. S. and Zisserman, A. (2010). Learning to count objects in images. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 1324–1332. Curran Associates, Inc.

[21] Li, Y., Zhang, X., and Chen, D. (2018). Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1091–1100. IEEE Computer Society.

[22] Lin, T., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. In Fleet, D. J., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer.

[23] Milan, A., Leal-Taixé, L., Reid, I. D., Roth, S., and Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831.

[24] Punn, N. S., Sonbhadra, S. K., Agarwal, S., and Rai, G. (2020). Monitoring covid-19 social distancing with person detection and tracking via fine-tuned yolo v3 and deepsort techniques. *arXiv preprint arXiv:2005.01385*.

[25] Redmon, J. and Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525. IEEE Computer Society.

[26] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

[27] Ren, S., He, K., Girshick, R. B., and Sun, J. (2017). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149.

[28] Ros, G., Sellart, L., Materzynska, J., Vázquez, D., and López, A. M. (2016a). The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3234–3243. IEEE Computer Society.

[29] Ros, G., Stent, S., Alcantarilla, P. F., and Watanabe, T. (2016b). Training constrained deconvolutional networks for road scene semantic segmentation. *CoRR*, abs/1604.01545.

[30] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4510–4520. IEEE Computer Society.

[31] Saponara, S., Elhanashi, A., and Gagliardi, A. (2021). Implementing a real-time, ai-based, people detection and social distancing measuring system for covid-19. *Journal of Real-Time Image Processing*, pages 1–11.

[32] Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., and Sun, J. (2018). Crowdhuman: A benchmark for detecting human in a crowd. *CoRR*, abs/1805.00123.

[33] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[34] Tayarani-N, M.-H. (2020). Applications of artificial intelligence in battling against covid-19: a literature review. *Chaos, Solitons & Fractals*, page 110338.

[35] Wang, Q., Gao, J., Lin, W., and Li, X. (2021). Nwpu-crowd: A large-scale benchmark for crowd counting and localization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(6):2141–2149.

[36] Wang, Q., Gao, J., Lin, W., and Yuan, Y. (2019). Learning from synthetic data for crowd counting in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8198–8207. Computer Vision Foundation / IEEE.

[37] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612.

[38] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE.

[39] Xiao, T., Li, S., Wang, B., Lin, L., and Wang, X. (2017). Joint detection and identification feature learning for person search. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3376–3385. IEEE Computer Society.

[40] Yang, D., Yurtsever, E., Renganathan, V., Redmill, K. A., and Özgüner, Ü. (2020). A vision-based social distancing and critical density detection system for COVID-19. *CoRR*, abs/2007.03578.

[41] Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[42] Zhang, S., Benenson, R., and Schiele, B. (2017a). Citypersons: A diverse dataset for pedestrian detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4457–4465. IEEE Computer Society.

[43] Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., and Li, S. Z. (2017b). Faceboxes: A CPU real-time face detector with high accuracy. In *2017 IEEE International Joint Conference on Biometrics, IJCB 2017, Denver, CO, USA, October 1-4, 2017*, pages 1–9. IEEE.

[44] Zhang, Y., Zhou, D., Chen, S., Gao, S., and Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*

2016, Las Vegas, NV, USA, June 27-30, 2016, pages 589–597. IEEE Computer Society.

[45] Zheng, L., Zhang, H., Sun, S., Chandraker, M., Yang, Y., and Tian, Q. (2017). Person re-identification in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3346–3355. IEEE Computer Society.

[46] Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. *CoRR*, abs/1904.07850.