



## D6.5 Models for Emergency Prediction and Handling v2

<b>Deliverable No.</b>	D6.5	<b>Due Date</b>	31/07/2024
<b>Description</b>	The models of emergency prediction and handling.		
<b>Type</b>	Report	<b>Dissemination Level</b>	PU
<b>Work Package No.</b>	WP6	<b>Work Package Title</b>	High Level Ecosystem for AI Operations
<b>Version</b>	1.0	<b>Status</b>	Final Version



## Authors

Name and surname	Partner name	e-mail
Alex Barnadas	INETUM	<a href="mailto:abarnadas1@gmail.com">abarnadas1@gmail.com</a>
Sofia Granda	INETUM	<a href="mailto:Sofia.granda@inetum.com">Sofia.granda@inetum.com</a>
Luis Carrascal	INETUM	<a href="mailto:luis.carrascal@inetum.com">luis.carrascal@inetum.com</a>
Antonio Gamito	INETUM	<a href="mailto:antonio-jesus.gamito@inetum.com">antonio-jesus.gamito@inetum.com</a>
Miguel Angel Romalde	INETUM	<a href="mailto:miguel-angel.romalde.ext@inetum.com">miguel-angel.romalde.ext@inetum.com</a>

## History

Date	Version	Change
01/02/2024	0.1	Initial ToC
28/02/2024	0.2	Inetum's Content completed
29/02/2024	0.3	Formatting
15/04/2024	0.4	First Draft
08/05/2024	0.5	Version ready for peer review
10/06/2024	0.6	Document after peer review
03/06/2024	0.7	Corrections peer review
04/06/2024	0.8	Ready for Quality check
01/10/2024	0.9	Corrections after Quality check
09/10/2024	1.0	Final Version

## Key data

Keywords	Disaster preparedness, computer vision, COVID management, IoT
Lead Editor	Alex Barnadas i Morera
Internal Reviewer(s)	FORTH, THL



## Abstract

This deliverable reports the efforts made to develop T6.2 (data interpretation and emergency management handling) of WP6 (High level ecosystem for AI operations) regarding the reporting period from month 24 to 36. Risk management is a subject that must be approached by all organisations, especially hospitals. This deliverable D6.5, has the responsibility to create a base for including innovative and state-of-the-art technologies into hospital systems emergency management and handling.

This deliverable contains a report of the development of the final AI solutions for emergency prediction and handling. It consists of a basic contextualization and explanation of two compatible and complementary approaches of how to solve the stated problem by the pilots.

This deliverable complements the work stated in D6.4, which provides a broader contextualization of the problem and further refines the issues and proposed solutions. Additionally, D6.5 utilizes the analysis and visualization of the emergency prediction systems' outcomes conducted in D6.1 and D6.2.

## Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

# Table of contents

- TABLE OF CONTENTS ..... 5**
- LIST OF TABLES ..... 6**
- LIST OF FIGURES..... 7**
- 1 INTRODUCTION ..... 8**
  - 1.1 OBJECTIVES ..... 8
  - 1.2 EXECUTIVE SUMMARY ..... 8
  - 1.3 ODIN CONTEXT ..... 9
- 2 AI INSIDE ODIN'S ARCHITECTURE ..... 10**
  - 2.1 DATA..... 10
- 3 EMERGENCY PREDICTION SYSTEMS..... 12**
  - 3.1 AVAILABLE SYSTEMS ..... 12
    - 3.1.1 *The camera network* ..... 12
    - 3.1.2 *The server* ..... 12
    - 3.1.3 *The ODIN platform*..... 12
  - 3.2 PEOPLE COUNTER ..... 13
    - 3.2.1 *Models used*..... 13
    - 3.2.2 *Conditions needed*..... 14
    - 3.2.3 *Parallel multicamera approach* ..... 14
    - 3.2.4 *Frame synchronization problem*..... 14
  - 3.3 SCENE UNDERSTANDING..... 14
    - 3.3.1 *AI Models needed*..... 15
    - 3.3.2 *Perspective transform* ..... 16
    - 3.3.3 *Improvable fields*..... 19
    - 3.3.4 *What is the resultant data and how is it treated* ..... 19
- 4 CONCLUSION ..... 20**

## List of tables

TABLE 1: DELIVERABLE CONTEXT ..... 9

## List of figures

FIGURE 1: UC7 INSIDE ODIN'S KER ARCHITECTURE DIAGRAM. ....	10
FIGURE 2: UC7 EMERGENCY PREDICTION SYSTEM ARCHITECTURE. ....	12
FIGURE 3: YOLOV8 BENCHMARKS.....	14
FIGURE 4: EXAMPLE OF OBJECT DETECTION, INCLUDING POSE DETECTION. ....	15
FIGURE 5: PERSPECTIVE TRANSFORMATION CONCEPT. ....	16
FIGURE 6: CALIBRATION LINES OVERWRITTEN IN THE IMAGE.....	17
FIGURE 7: HYPOTHETIC PERSPECTIVE TRANSFORMATION OF AN IMAGE .....	17
FIGURE 8: EXAMPLE 1 OF HOW AN OBJECT RESHAPES AFTER THE TRANSFORMATION. ....	18
FIGURE 9: EXAMPLE 2 OF HOW AN OBJECT RESHAPES AFTER THE TRANSFORMATION. ....	18
FIGURE 10: EXAMPLE OF HOW THE TRACKS COULD BE PLACED IN A BIRD VIEW MAP. ....	18

# 1 Introduction

## 1.1 Objectives

The main objective of the Reference Use Case (RUC) C is to prepare hospitals for the considered disasters, but also to increase patient and staff safety and security. For this reason, the following applications have been considered.

### Disaster preparedness: COVID measures and sanitary safety

COVID measures, such as minimum interpersonal distance or wearing a mask, are now globally accepted common practices and rules in every hospital sometimes can be hard to make sure everyone is following them. The use of AI could lead to an easier way to remind everyone that these rules are still to be followed by everyone.

### Safety: Evacuation risks

Evacuation is a scenario that can end up being catastrophic; this might be due to the lack of information on the capacity in every single room. To mitigate this risk, we purpose to track the amount of people in each floor/room and trigger an alarm at the moment that its limit is exceeded to keep the capacity of the hospital in its safe values. This way we can have a more precise control on how many people are in the hospital and how many more can enter in a defined area.

### Security: Overcrowding avoidance

An overcrowded scene is a potential risk by itself and not only in case of an emergency, for example, when people try to escape from a fire or there is not spacious enough for people in the waiting room, so when the ambulance arrives, and ambulance stretcher cannot pass between the people. Specially in a hospital where emergencies are a daily matter.

## 1.2 Executive Summary

In this Deliverable, as we said before, the main objective is to prepare the hospital for the considered disasters increasing the patient and staff safety and security.

We will focus on the emergency handling and prediction systems; this is autonomous and does not need another ODIN component. Now we use threads, it is a big improvement in performance, which makes the application able to process more cameras at the same time.

For this use case, we are using the YOLOv8 library, YOLOv8 has several models available, in which it is possible to choose between the trade-off of performance and execution time.

The output of this computer vision algorithm is the input of the evacuation prediction model built by the Sant'Anna School of Advanced Studies. Its motivation is, in fact, to satisfy the needs of this algorithm, which just needs the initial position of all the people in a room, as well as its direction of movement and velocity. There is a perspective transform that converts what the camera can see in a rectangle to have a better knowledge about the coordinates.

## 1.3 ODIN context

Table 1: Deliverable context

PROJECT ITEM IN THE DOA	RELATIONSHIP
Project Objectives	D6.5 is strongly related to ODIN’s Objective 1 (O1) that aims “to deliver an open and secure decentralized platform supporting a suite of federated services and Key Enabling Resources (KERs) empowered by robotic solutions, augmented by IoT environments, and enhanced by extensible specialized AI”.
Exploitable results	The results of this deliverable will be directly exploited by WP3 (Platform integration, Privacy, Security and Trust + knowledge + cognition) and WP7 (ODIN pilots design, deployment, evaluation and validation).
Workplan	<p>This deliverable is part of the progress of T6.2 (Data interpretation and emergency prediction handling) regarding the reporting period of month 24-36 and is the second version of Emergency prediction and handling.</p> <p>The progress of the disaster preparedness is strongly related to the use case that the pilots consider relevant in their dependencies. Thus, the deliverable will be in a kick-off for the pilots to perform AI enhanced disaster preparedness plans.</p>
Milestones	D6.5 is linked to the Milestone MS2 (ODIN technologies catalogue defined and full version of the platform and IoT robotics and AI components).
Deliverables	<p>The current deliverable is related to D6.1 (The data model ecosystem for AI operations and modules implementation) and D6.2 (Data results interpretation and data integration services) that provide the data results analysis and interpretation for the AI-based models. D6.5, is also related to D7.10 (Pilot Studies Use Case Definition and Key Performance) where the pilots’ requirements and needs where described in detail. Additionally, D6.6 is related to D3.12 (ODIN Platform v3) that defines the AI as a KER in the ODIN platform and its connection to HIS. And D3.5 where it is defined the security, privacy and trust.</p> <p>Deliverable D6.5 is a second release of D6.4.</p>
Risks	The potential risks in this deliverable are mainly Risk 2 (Technical problems during component/module development), Risk 4 (Risk of time consuming due to multiple technology) and Risk 9 (Legal restrictions imposed in the execution of the trials).

## 2 AI inside ODIN's architecture

The AI algorithms in which is based the emergency handling and prediction systems will have to run on premises due to the unavailability to take the image stream out of the hospital. It is then convenient to set up a way to run the models used by the system. The system is relatively autonomous with respect to the rest of ODIN components, meaning that it does not need any other component than the ODIN platform, since it does not store images, so it does not need a database. In figure 1 it can be appreciated where the use case takes place inside the ODIN KERS architecture.

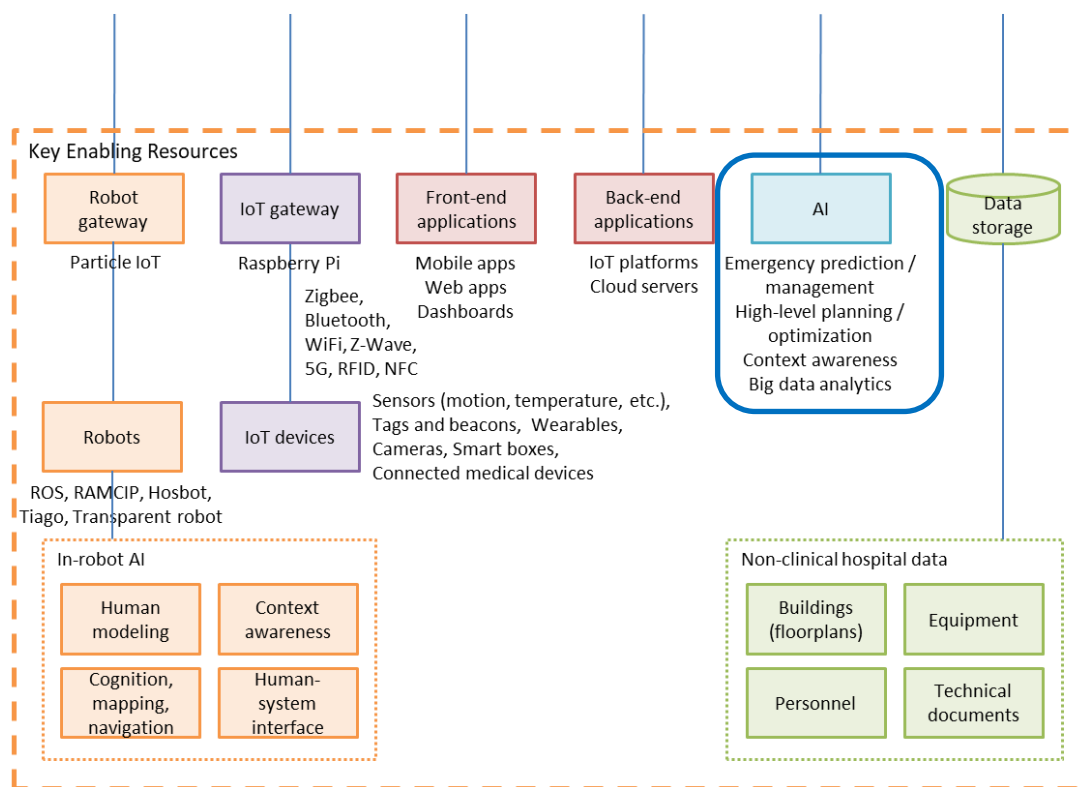


Figure 1: UC7 inside ODIN's KER architecture diagram.

### 2.1 Data

Data treatment is always a challenge when speaking of real-time applications, in our case, several challenges have had to be solved to make things work. Mainly due to the use of threads, which is the most notable improvement with respect to the previous deliverable (D6.2), that makes the application able to process more cameras at the same time, but also introduces the problem of frame synchronization. For this reason, a semaphore approach has been built which seems to work fine.

Race conditions occur when multiple threads attempt to read and write shared data simultaneously. By using a semaphore, it ensures that only one thread can access and modify the shared resource at a time, thus preserving data integrity. Semaphores also help to avoid deadlocks, where threads are stuck waiting for each other to release resources. The semaphore

is acquired by a running thread if it's not blocked, and once the thread completes its task, it releases the semaphore. After its release, other processes can acquire and reserve it.

This description highlights the importance of semaphores in preventing data corruption and ensuring the smooth execution of concurrent processes. terms of data privacy all possible efforts to respect the GDPR have been made discarding all the approaches that including the identification, either biometrical or in any other way, of the people detected.

See deliverable D6.2 for more contextualization in terms of data treatment.

## 3 Emergency prediction systems

### 3.1 Available systems

The main goal is to alert when an exit has to be used and is overcrowded.

To accomplish the task purposed, the pilot is assumed to count on a number of different systems that will allow the method purposed to run on its premises.

#### 3.1.1 The camera network

Since part of the solution is based on a computer vision algorithm it is essential for it to be deployed to have a camera network installed in the hospital and accessible through the GPU computing server, to process the images in real time. The architecture of the network looks like in figure 2.

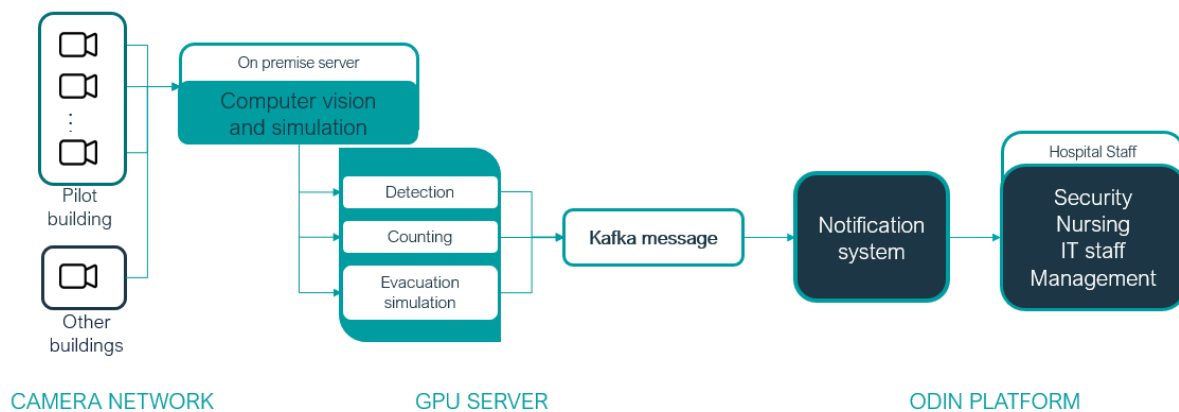


Figure 2: UC7 Emergency Prediction System architecture.

#### 3.1.2 The server

Given that the platform itself is designed to be running on premises it is essential to dispose a dedicated server that is capable to run it, when needed. This server is the middle point between the camera network and the ODIN platform, since there it will be calculated the designed risks with the image information given by the cameras and forward the results of the evaluation to the ODIN platform.

In the case of the SERMAS pilot, a server with a Nvidia GPU 3070 with 8GB of GPU memory is provided. Although this would not be the ideal case, since this is not a server GPU, but a PC GPU not designed to support long periods of time running, it is more than enough to run the algorithm and it gives us an ideal environment in which to test in a real scenario for the solution purposed. It is then suitable for a small use case, of 10 cameras, but not to cover the whole hospital area for 24/7 monitoring.

#### 3.1.3 The ODIN platform

In order for the solution to be usable, it must be linked to the platform since this is the way the hospital people will use the information given by the solution. The ESB is based on the usage of Kafka, to exchange messages between the components of the ODIN platform. WP4 provided different alternatives to connect KERs to the ESB. Since the computer vision algorithm was an in-

house development, it was easy to include in the code a Kafka Producer that could send the necessary information to the ESB. For this purpose, the kafka-python library was used<sup>1</sup>.

```
def KafkaNotify(topic='my-topic2', message=b'No message has been defined'):  
    producer = KafkaProducer(bootstrap_servers=['10.0.21.11:32100'])  
  
    # Asynchronous by default  
    future = producer.send(topic, message)  
  
    # Block for 'synchronous' sends  
    try:  
        record_metadata = future.get(timeout=10)  
    except KafkaError:  
        # Decide what to do if produce request failed...  
        log.exception()  
        pass
```

## 3.2 People counter

For this use case we are using the YOLOv8 library<sup>2</sup>, an open-source library maintained by the private enterprise Ultralytics. This library is free to use, although the license does not allow to commercialize products that use the software and it is mandatory to open source the code that uses this library, that is why the code implemented will have to remain open source and free to use, if commercialization is needed, YOLOv7<sup>3</sup> is a better approach with similar (in some cases better) performance.

### 3.2.1 Models used

For the detection task we have had to consider several factors such as the size of the video that the cameras provide, as well as the frame rate and the number of cameras that are running at the same time. Fortunately, YOLOv8 has several models available, in which it is possible to choose between the trade-off of performance and execution time, we can appreciate a comparison between those models (*n*, *s*, *m*, *l* and *x*), as well as a comparison with other popular models in terms of number of parameters/latency and precision in figure 3. In this case an efficient model has been chosen, in order not to saturate the GPU and still get a reliable detection performance. Due to a further use case, it is used the pose version, so the name of the model is “yolov8m-pose”.

---

<sup>1</sup> <https://kafka-python.readthedocs.io/en/master/>

<sup>2</sup> <https://github.com/ultralytics/ultralytics>

<sup>3</sup> <https://github.com/WongKinYiu/yolov7>

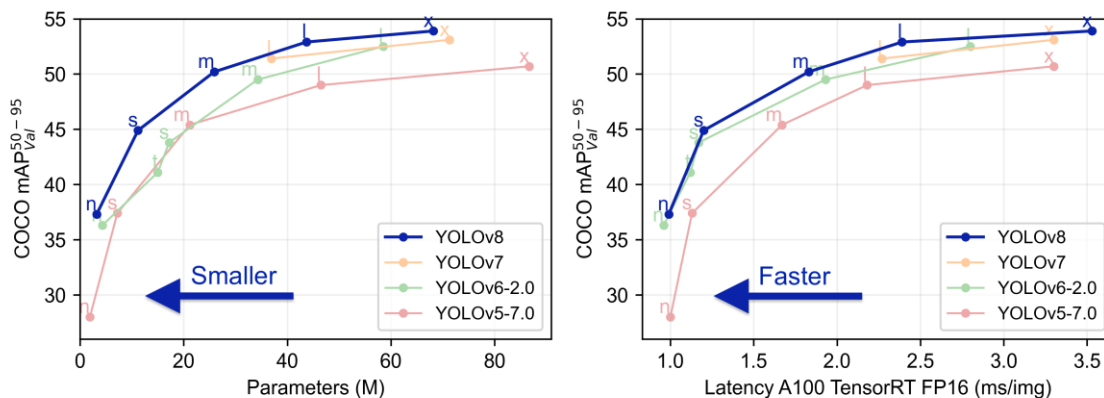


Figure 3: YOLOv8 benchmarks.

### 3.2.2 Conditions needed

Since the system needs to identify univocally all the people it would be problematic if the cameras overlapped the scene framed, otherwise, more people than the present could be counted. Thus, it is necessary to carefully position the angles of the cameras as such that they do not overlap the scene framed.

This problem could be faced with a reidentification algorithm, but since it was not needed, it was decided to leave for a future improvement in case the pilot was successful.

### 3.2.3 Parallel multicamera approach

Different approaches could have been taken in regard to the problem of processing number of cameras. In the previous iteration of the use case the straightforward approach was taken. The model was simply fed with all the frames that were given by the camera and one only iteration was needed for all the cameras, this leads to a desynchronization problem and it was a poor usage of the resources.

### 3.2.4 Frame synchronization problem

With the current approach a thread is created for every camera address passed, iteratively. Although this is a more elaborated solution and is a better practice it also has its own problems, since the processing time leads to delays that might be carried with time and the consequent discoordination between the different threads. This makes it hard to add up how many people are currently in camera, since different cameras are computing different instants. This has been solved using two adjustments, the first is to try to match the processing time and the frame rate, then we are minimizing the delay of the processing. And the other is to synchronize the people counter of all the different threads, independently of which time it was processed, only taking into account what time the frame was grabbed.

## 3.3 Scene understanding

This section might work as a prelude for the following section. Since the output of this computer vision algorithm is the input of the evacuation prediction model built by the University of Santa Anna (SSSA). Its motivation is in fact to satisfy the needs of this algorithm that just needs the initial position of all the people in a room, as well as its direction of movement and velocity. Considering that the hardware available is not capable to accurately calculate the real position of the people framed, and, in the opinion of the author, a very clever approach to solving this problem involves a manual calibration of the scene in the camera.

### 3.3.1 AI Models needed

Two models are needed to satisfy the needs of the use case. One additional detection model to correctly localize the people detected, and a tracking method to calculate the velocity and direction.

#### 3.3.1.1 Position

Having a calibrated scene, one might think that the position of the detection could be deducted simply by transforming the coordinates of the centre bottom rectangle detected. But this led to problems, since when the full body is not framed, i.e. when the legs of the person detected are occluded, the position would change drastically, that would make the position, velocity, and direction, drastically inconsistent, noisy, and unbelievable. Thus, a pose estimation is included.

The pose detection algorithm is nothing more than an object detection for the different parts of the human body, its output is a stickman in the position of the human detection. This is very suitable for our use case since we want to know where the feet of each person detected meets the floor. The midpoint between both feet is taken as the position of the person.

This solution sacrifices the detection whenever the feet are not framed, as for example in figure 4, but it is anyways better than the previous solution. A joint solution was tested, in which whenever the feet were occluded the bottom position was estimated considering the width of the detection, but the results were not precise enough.

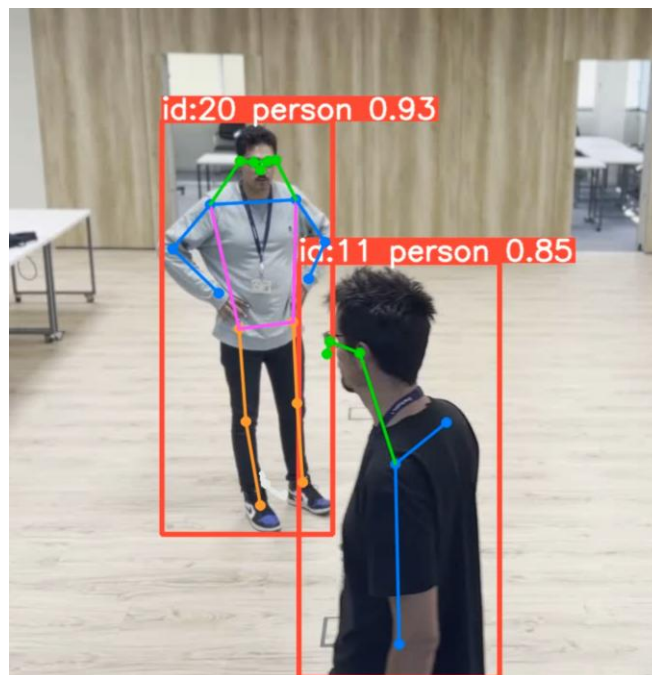


Figure 4: Example of object detection, including pose detection.

#### 3.3.1.2 Velocity and direction

Once the real-world position is calculated, then it is a matter of calculating the difference between frames in terms of distance divided by time. To do this a tracking algorithm is needed, fortunately YOLOv8 comes with an implemented solution of tracking in which the BoT-SORT model<sup>4</sup> is used.

<sup>4</sup> <https://github.com/NirAharon/BoT-SORT>

This allows to consistently relate each detection between frames; thus, velocity and direction can be computed.

Since the velocity and the direction is computed in real time if we calculated it straight forward between each pair of frames a very noisy result will be made, that is why it is calculated between several frames, by now 20 frames has given very good results covering all the noise and correctly calculating these values.

### 3.3.2 Perspective transform

Once the detection system is implemented, the problem to be solved is how to approximate the real-world position to the detected coordinates on the screen. To do so, an intrinsic transformation of the coordinates is purposed<sup>5</sup>. This method is nothing more than a perspective correction of the ground in which we transform the image to see the floor as if we were right above it.

To reproduce this type of transformation it is needed the information of where the floor plane is located, which in our case, it is obtained through a manual calibration. This calibration just needs the position of 4 points that match the 4 vertices of a known rectangular equilateral area.

With these 4 points, we can use the OpenCV function `getPerspectiveTransform` to calculate the transformation matrix needed to map the coordinates. This function applies geometrical transformation of images. With the input of the coordinates of the original image (the coordinates selected manually) and the perspective we look for (the Cartesian coordinates), the function calculates the matrix needed to transform it. Then, we can use the function `warpPerspective` from the same library, with the matrix calculated, to obtain the new image. A conceptual example of this transformation can be appreciated in Figure 5.

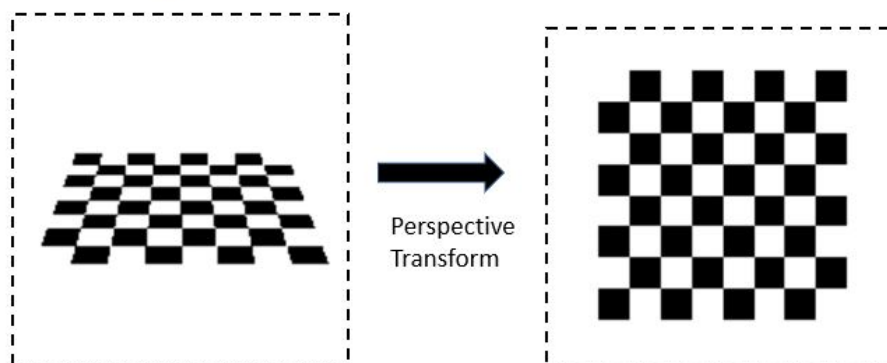


Figure 5: Perspective transformation concept.

If we want to apply the transformation to an image, we will need to manually specify the position of the real-world corners of the square in the image. In figure 6 it can be appreciated how those are specified.

<sup>5</sup> <https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec>

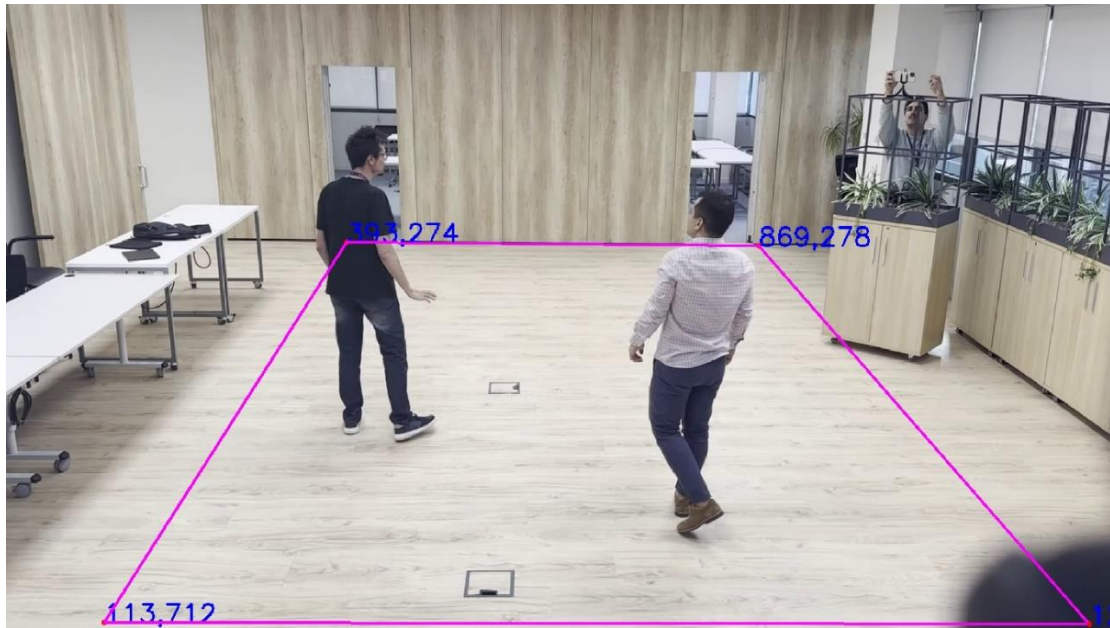


Figure 6: Calibration lines overwritten in the image.

This would not be the ideal case, since the area selected is not known to be a regular square, and it is only approximated. To test how good the approximation is, a whole example image is transformed, see Figure 7. If we close up, in Figures 8 and 9 we can see how a shape, which we do know to be a perfect square has reshaped to be more rectangular. The closest to an equilateral square we draw the rectangle in the calibration process, the more equilateral will look equilateral shapes placed on the floor such as the ones in Figures 8 right and 9 right. Also, as we can see, the further the real-world shapes are from the camera, the more appreciable will appear the distortion. As we can see this while differentiating Figure 8 from Figure 9 that represent the same shape, at different distances with respect to the camera position.

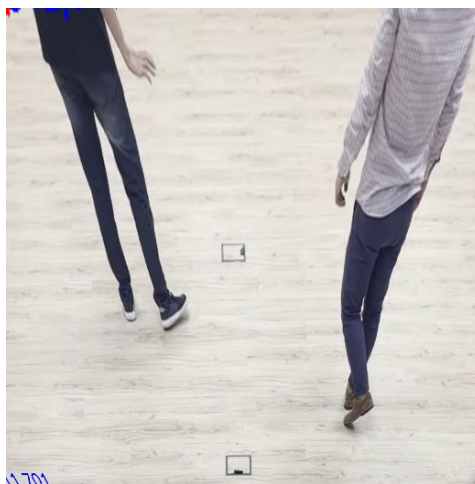


Figure 7: Hypothetic perspective transformation of an image

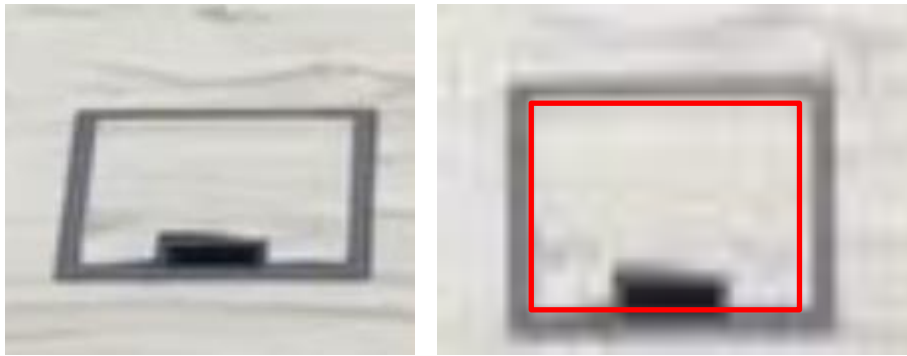


Figure 8: Example 1 of how an object reshapes after the transformation.

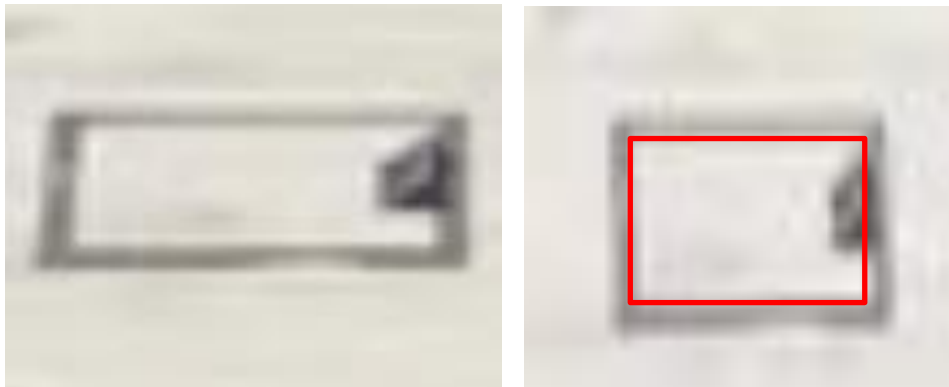


Figure 9: Example 2 of how an object reshapes after the transformation.

In an ideal case we could place marks on the floor to label a perfect squared area, for example a tiled floor would make an ideal environment. Then measuring the distance of each side of the square would allow the positions estimated to be converted in meters, and thus the velocities from pixels/second to metres/second.

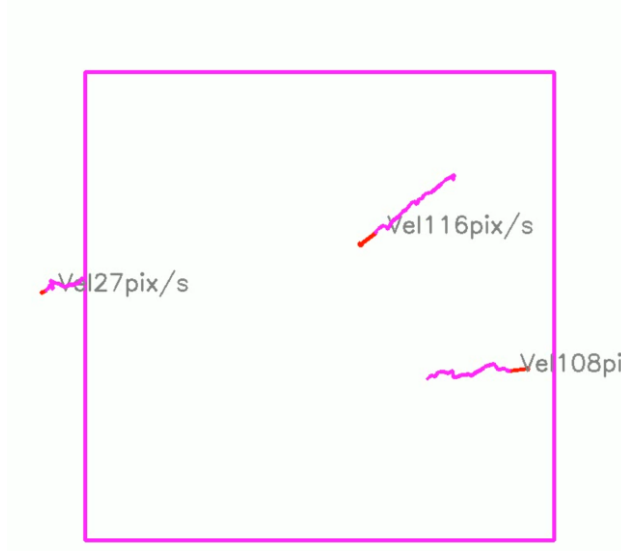


Figure 10: Example of how the tracks could be placed in a bird view map.

Lastly, if we measured the position of each of the four points and placed it in a map, we could map all the locations of the detected people in a real map of the building, as it has been exemplified in figure 10, to perform the evacuation simulation.

### 3.3.3 Improvable fields

Although the solution is still to be tested it is supposed that there are a few fields in which the precision of the final result might be noisy with respect to the real-world position. Mainly there are three ways that the solution is susceptible to a considerable lack of precision:

- o Distance to the camera:

Since the angular perspective of the camera implies that the further, we are from the camera, the more distance we need to move to perceive a difference, it is legitimate to think that the error will be magnified with distance.

- o Size of the calibration area

In a similar way, the area that we choose to perform the calibration will have to cover the most room possible, since with a small square, the error will also be propagated to the next step since the beginning.

- o Lens aberration

The type of the lens could also be a cause of error, even though measures have been taken to reverse these aberrations, it is still something to be bothered about. Wide angle lenses might provoke the fisheye effect, that would distort the whole image and thus the coordinates calculated.

### 3.3.4 What is the resultant data and how is it treated

The outcome of the scene understanding algorithm is a constant flow of information consisting of row per detection on each frame, this row consists of the real-world position, the velocity and orientation of the movement of the person detected.

	A	B	C	D	E	F
1	FrameNumber	ScreenCoords	SceneCoords	Velocity	Orientation	
2						
3	63	[1082, 236, 160, 264]	[286, -53]	1	90	
4	63	[279, 210, 150, 336]	[-195, 440]	30	-117	
5	63	[874, 147, 112, 254]	[4, -188]			
6	66	[1083, 237, 161, 266]	[284, -52]	5	-78	
7	66	[279, 209, 150, 336]	[-195, 439]	24	-139	
8	66	[874, 148, 112, 253]	[3, -190]			
9	69	[1083, 237, 159, 262]	[284, -51]	9	-83	
10	69	[278, 209, 148, 333]	[-191, 439]	23	-140	
11	69	[870, 147, 113, 253]	[4, -191]			
12	72	[1082, 237, 158, 261]	[285, -47]	7	-81	
13	72	[278, 208, 147, 331]	[-191, 438]	3	-90	
14	72	[865, 147, 114, 253]	[4, -192]			
15	75	[1083, 237, 159, 261]	[285, -49]	6	-71	
16	75	[278, 208, 147, 330]	[-192, 435]	2	0	

Figure 11: Example of the resultant data

To ease out things on the merging process with the following states, by now this data is being constantly written and overwritten every few frames to a file stored in disk. This approach could be improved for the sake of good practices, but in practical terms it works perfectly fine, and a different approach is not needed in any practical aspect since the evacuation simulation algorithm does not have to deal with the real-time challenge.

## 4 Conclusion

The goal of the task T6.2 is to define and purpose solutions for the emergencies in the sanitary context that could leverage from AI. Several options have been reviewed and some of them have been tested, and after analysing all the purposed solutions the mentioned solution has been the most viable application.

To summarize, the solution on emergency prediction and handling counts on a detection tool, a simulation tool, and a notification tool that will all run in a server within the ODIN platform which will be able to set up alarms, that will be triggered by the system itself in case one of the defined conditions is met.

The solution is being tested in the SERMAS dependencies and is thought to be easily upgradable and transferrable to other pilots in case it is requested.

All three solutions combined are a proof of concept of a potential game changer in the current approaches taken by hospitals at this time to prevent the targeted disasters.