Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Technical University of Cartagena

School of Industrial Engineering

Human motion estimation and analysing feasible collisions between individuals in real settings by using conventional cameras an Deep Learning-based algorithms

Bachelor's Final Degree Project

Rafael Sojo García

Project author

Assoc Prof. Dr. Nieves Pavón Pulido
Assoc Prof. Dr. Virginijus Baranauskas

Supervisor

Cartagena, Kaunas 2022

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Technical University of Cartagena

School of Industrial Engineering

Human motion estimation and analysing feasible collisions between individuals in real settings by using conventional cameras an Deep Learning-based algorithms

Bachelor's Final Degree Project

Intelligent Robotics Systems (6121EX013)

Industrial Electronics and Automation Engineering (5071)

Rafael Sojo García

Project author

Assoc. Prof. Nieves Pavón Pulido
Assoc Prof. Dr. Virginijus Baranauskas

Supervisors

Assoc. Prof. Dr. Arūnas Lipnickas

Reviewer

Cartagena, Kaunas 2022

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Technical University of Cartagena

School of Industrial Engineering

Rafael Sojo García

Human motion estimation and analysing feasible collisions between individuals in real settings by using conventional cameras and Deep Learning-based algorithms

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Rafael Sojo García.

APPROVED BY:

KTU Faculty of Electrical and Electronics Engineering

Head of the Department of Automation

Assoc. prof. dr. Gintaras Dervinis

2022 05 04

TASK OF FINAL PROJECT OF UNDERGRADUATE (BACHELOR) STUDIES

Issued to the Student:	*Rafael Sojo García*	Group ERB8/2

1. Project Subject:

Lithuanian Language:	Žmogaus judėjimo įvertinimas ir galimų susidūrimų analizė tarp asmenų realiomis sąlygomis, naudojant įprastą kamerą ir giliojo mokymosi algoritmus
English Language:	Human motion estimation and analysing feasible collisions between individuals in real settings by using conventional cameras and Deep Learning-based algorithms

Approved 2022 April . 29d. Decree of Dean Nr. *V25-03-10*

2. Goal of the Project: To develop a collision prevention system

3. Specification of Final Project: The work must meet the methodological requirements for the preparation of final projects of KTU Faculty of Electrical and Electronics Engineering.

4. Project's Structure. *The content is concretized together with supervisor, considering the format of the final project, which is listed in 14 and 15 points of Combined Description of Preparation , Defence and Keeping of Final Projects Methodical Requirements*

4.1 Analyse computer visio, object detection, object tracking algorithms
4.2 Create human detection and tracking algorithm
4.3 Make experimental testings of created algorithm

5. Economical Part. *If economical substantiation is needed; content and scope is concretized together with supervisor during preparation of final projects*

none

6. Graphic Part. *If necessary, the following schemes, algorithms and assembly drawings; content and scope is concretized together with supervisor during preparation of final projects*

Show experimental results and explane them

5. This Task is Integral Part of Final Project of Undergraduate (Bachelor) Studies

6. The Term of Final Project Submission to Defense Work at a Public Session of Qualification Commission. *until 2022-05-13* *(date)*

I received this task:	***Rafael Sojo García***	*2022-02-08*
	(student's name, surname, signature)	*(date)*
Supervisors:	**Assoc. Prof. Nieves Pavón Pulido** **Assoc. Prof. Dr. Virginijus Baranauskas**	*2022-02-08*
	(position, name, surname, signature)	*(date)*

Sojo García, Rafael. Human motion estimation and analysing feasible collisions between individuals in real settings by using conventional cameras and Deep Learning-based algorithms. Bachelor's Final Degree Project / supervisors: Prof. Dr. Nieves Pavón Pulido, Assoc. Prof. Dr. Virginijus Baranauskas; School of Industrial Engineering; Technical University of Cartagena; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): electronics and electrical engineering, technological science, engineering.

Keywords: Artificial Intelligence, object detection, object tracking, collision prevention, YOLOv5, DeepSORT.

Kaunas, Cartagena, 2022, 93 pages.

Summary

The aim of the final project is to develop a collision prevention system. This project includes: an introduction, which the reader is made aware of the problematic to be addressed; theoretical foundation, with which to provide the necessary knowledge required by the models used, with special emphasis on their mathematical foundations; the proposed method section. Experimental section develops the four main steps 1) camera calibration, 2) correction of possible distortion during image acquisition, 3) detection and tracking of the people involved in the images obtained and 4) obtention of alerts in potentially dangerous situations throughout the test. The experimental section show the results obtained in the process of detection, tracking and obtention of alerts. Conclusions and a list of the bibliographical references are given.

Sojo García, Rafael. Estimación del movimiento humano y analizador de posibles colisiones entre individuos en entornos reales usando cámaras convencionales y algoritmos de Deep Learning. Trabajo de Fin de Grado / supervisores: Prof. Dr. Nieves Pavón Pulido, Prof. Dr. Virginijus Baranauskas; Universidad Politécnica de Cartagena, Facultad de Ingeniería Industrial; Universidad Tecnológica de Kaunas, Facultad de Ingeniería Eléctrica y Electrónica.

Campo de estudio y área: ingeniería electrónica e ingeniería eléctrica, ciencia tecnológica, ingeniería.

Keywords: Inteligencia artificial, detection de objetos, seguimiento de objetos, prevención de colisiones, YOLOv5, DeepSORT

Kaunas, Cartagena 2022, 93 páginas.

Sumario

El objetivo del proyecto final es desarrollar un sistema de prevención de colisiones. Este proyecto incluye: una introducción, con la que se hace partícipe al lector de la problemática que se pretende abordar; una fundamentación teórica, con la que se aportan los conocimientos necesarios que requieren los modelos utilizados, haciendo especial hincapié en sus fundamentos matemáticos; la sección del método propuesto. La sección experimental desarrolla los cuatro pasos principales 1) calibración de la cámara, 2) corrección de la posible distorsión durante la adquisición de la imagen, 3) detección y seguimiento de las personas implicadas en las imágenes obtenidas y 4) obtención de alertas en situaciones potencialmente peligrosas a lo largo de la prueba. El apartado experimental muestra los resultados obtenidos en el proceso de detección, seguimiento y obtención de alertas. Se presentan las conclusiones y una lista de referencias bibliográficas.

Sojo García, Rafael. Žmogaus judėjimo įvertinimas ir galimų susidūrimų analizė tarp asmenų realiomis sąlygomis, naudojant įprastą kamerą ir giliojo mokymosi algoritmus. Bakalauro baigiamasis projektas / vadovai / doc. dr. VIrginijus Baranauskas, doc. dr. Nieves Pavón Pulido; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, Universidad Politécnica de Cartagena, Pramonės inžinerijos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektronikos ir elektros inžinerija, technologijos mokslai, inžinerija.

Reikšminiai žodžiai: dirbtinis intelektas, objektų aptikimas, objektų sekimas, susidūrimų vengimas, YOLOv5, DeepSORT.

Kaunas, Cartagena 202. p. 93

Santrauka

Baigiamojo projekto tikslas – sukurti susidūrimų prevencijos sistemą. Darbą sudaro: įvadas, kuriame skaitytojas supažindinamas su problema, kurią reikia spręsti; teorinis pagrindimas, kuris suteikia reikiamų žinių, reikalingų naudojamiems modeliams, ypač pabrėžiant jų matematinius pagrindus; autoriaus siūlomas metodas. Praktinėje dalyje išplėtoti keturi pagrindiniai žingsniai: 1) kameros kalibravimas, 2) galimų iškraipymų koregavimas vaizdo gavimo metu, 3) žmonių, dalyvaujančių gautuose vaizduose, aptikimas ir sekimas ir 4) įspėjimų gavimas, testuojant potencialiai pavojingas situacijas. Eksperimentinėje dalyje rodomi perspėjimų aptikimo, sekimo ir gavimo rezultatai. Pateikiamos išvados ir naudotų literatūros šaltinių sąrašas.

Table of contents

List of figures

List of tables

List of abbreviations and terms

Terms:

AGV – Autonomous Guided Vehicles

AMR – Autonomous Mobile Robot

ANN – Artificial Neural Network

CmBN – Cross mini-Batch Normalization

CNN – Convolutional Neural Network

CSPNet – Cross Stage Partial Network

DPM – Deformable Parts Model

FPN – Feature Pyramid Network

FPS – Frames Per Second

HSV – Hue Saturation Value

IOU – Intersection Over Union

mAP – Mean Average Precision

PANet – Path Aggregation Network

R-CNN – Region-based Convolutional Neural Network

ReLU – Rectified Lineal Unit

RGB – Read Green Blue

ResNet – Residual Network

SKU – Stock-Keeping Unit

SORT – Simple Online Real-time Tracking

SSE – Sum-Squared Error

SSP – Spatial Pyramid Pooling

VOC – Visual Object Classes

YOLO – You Only Look Once

1. Introduction

Nowadays, Computer Vision (CV), has become a very powerful tool since technological improvements done in recent years are bringing a strong rise in the use and popularity of Artificial Intelligence (AI) systems. Moreover, CV systems have a lot of different possible applications in many fields typically led to the human responsibility, like Health Care or Security Industries. Thus, allowing us to focus our attention in the task of increasing productivity, whereas these systems take care of those aspects in which we are either more likely to commit errors, or just more expensive than the robotization of the job in the long term.

On the other hand, in industrial environments where vehicles are constantly sharing space with pedestrians, any distraction or visibility reduction can end into a forklift-related accident. This is either an economic and a work safety problematic due to the risk for the worker and the subsequent hospitalization costs, which in many cases might end into serious injuries.

Therefore, the aim of this project is to address this problematic through a CV-based human trajectory estimation for collision preventions. This solution takes advantage of some CV and models based on Deep Learning (DL), focused on performing real-time object detections, as well as a mathematical and statistical object tracking implementation with a deep association metric.

This project is entirely focused on human-to-human interactions, but the results are completely extrapolable for human-to-vehicle and vehicle-to vehicle-situations.

1.1. Objectives

This project aims to develop a collision prevention system, whose function is to contribute to the work safety in the manufacturing industry. For that reason, this system should be able to return warnings depending on the location of several individuals within a predefined area.

To accomplish this goal, the following partial objectives are set:

- Camera calibration: for the camera parameters extraction.
- Image undistortion: for the correct data acquisition.
- Object detection and tracking: based on YOLOv5 and DeepSORT.
- Security area definition: setting the angle, radius and orientation of a circular sector in front of each human at each frame.
- Warning system definition: the alerts are given depending on the trajectories and locations of the individuals.

2. State of Art

2.1. Accidents involving Forklifts.

Industrial environments are characterized by the fact that work safety has an important role in every task, since the machinery is capable of causing enormous amounts of energy. However, the workers and vehicles moving around are something that may go unnoticed, but according to the Occupational Safety and Health Administration (OSHA) in 1995, up to 34.900 serious injuries, 61.800 non-serious injuries and around 85 fatalities, were caused by forklifts trucks in the United States since 1981 on annual average. At least a 10% of the fatal accidents as a result of people being struck by a forklift, see Table 1.

Type accident	Percent
Crushed by tipping vehicle	42
Crushed between vehicle and a surface	25
Crushed between two vehicles	11
Struck or run over by vehicle	10
Struck by falling material	8
Fall from platform on forks	4
Accidental activation of controls	2

Table 1. Fatal accident causes involving forklifts. [1]

More recently, the Bureau of Labor Statistics, U.S. Department of Labor, shared in 2019 that more than 8.500 nonfatal injuries involving forklifts still occurring on average every year, resulting in several days out of work. Furthermore, with barely no change in the number of fatal injuries and nearly the 20% of them as a direct cause of forklift struck, see Fig 1 and Fig 2.

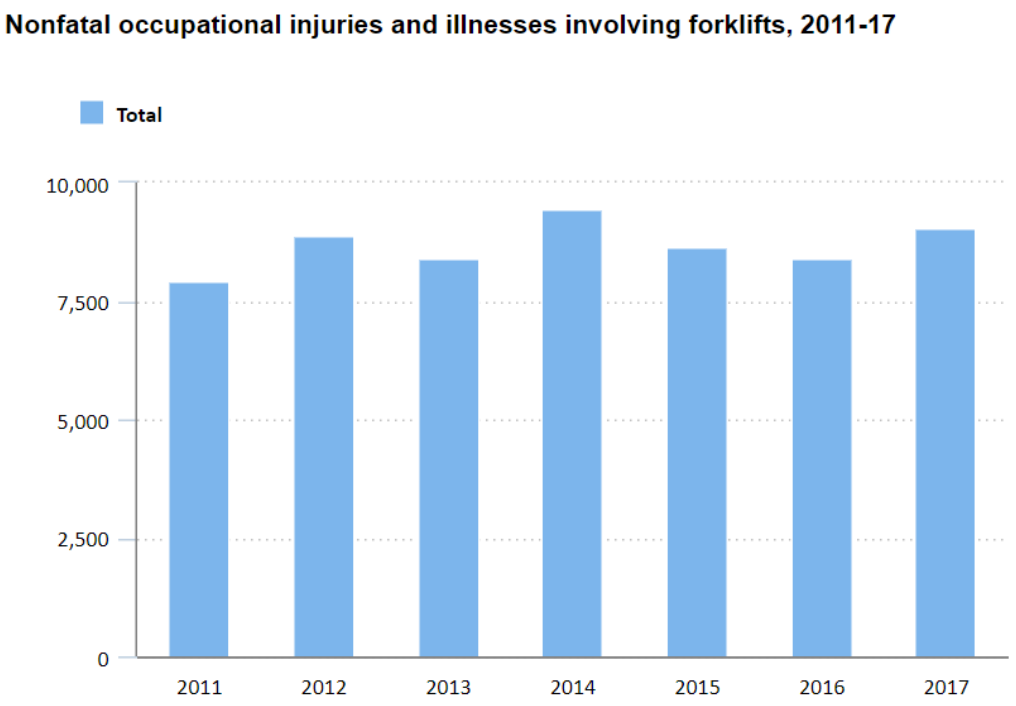


Fig. 1. Nonfatal injuries involving forklifts, 2011-2017. [2]

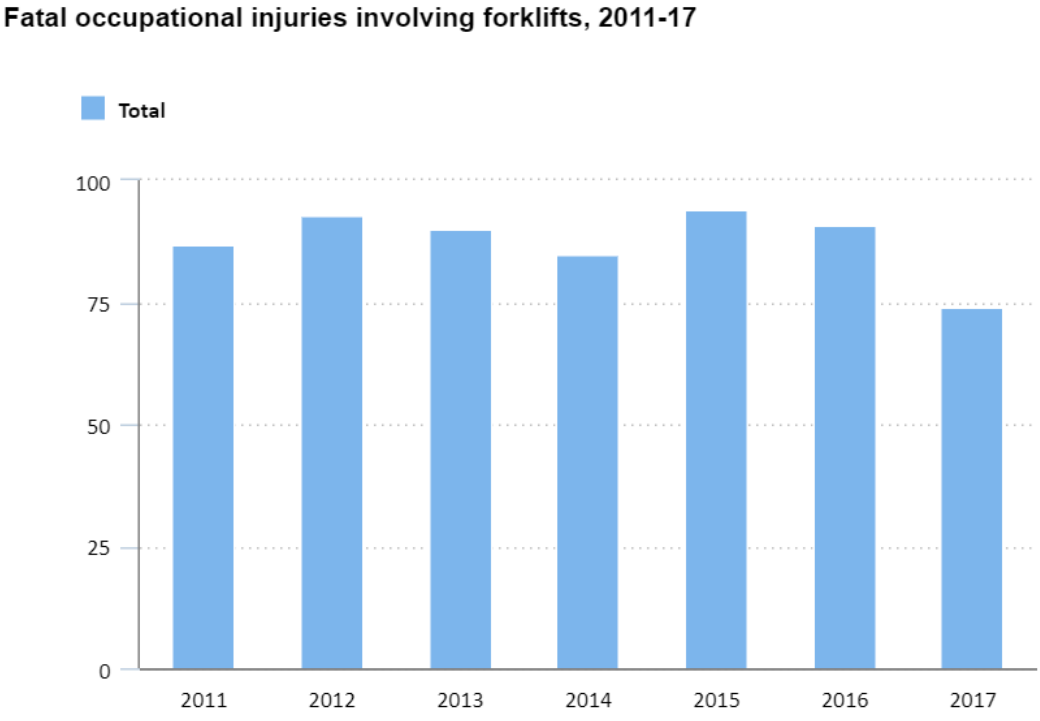


Fig. 2. Fatal injuries involving forklifts, 2011-2017. [2]

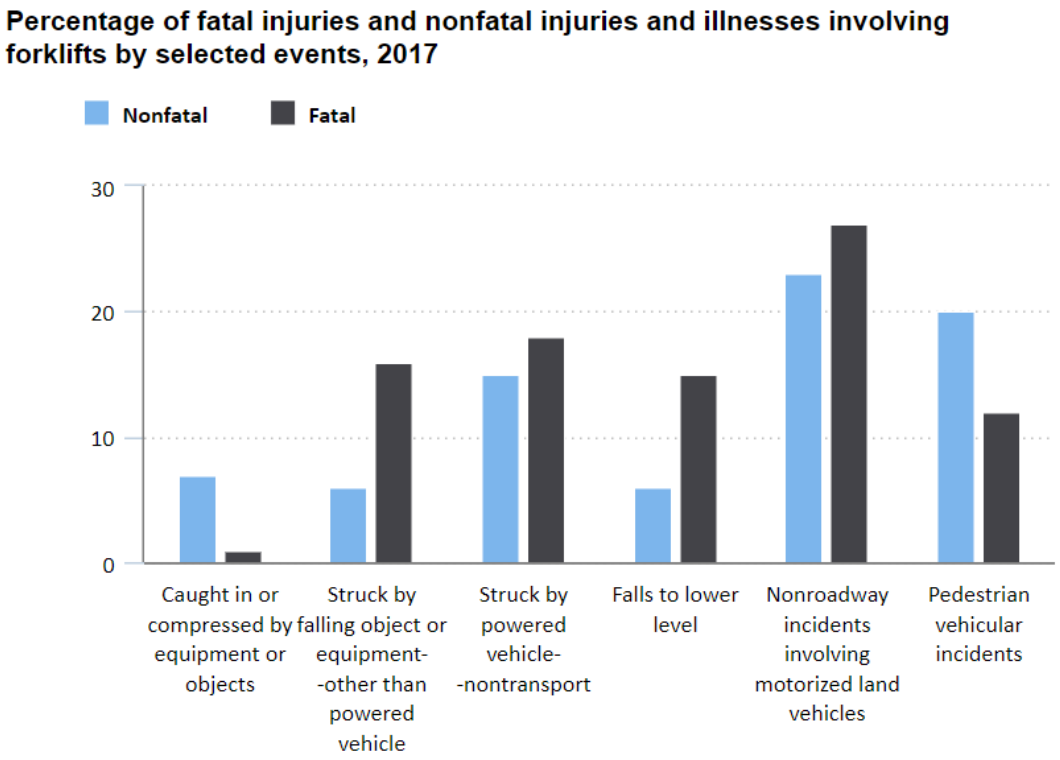


Fig. 3. Accident causes involving forklifts, 2017. [2]

2.2. Strategies for addressing the problem

Forklifts struck are one of the principal cause of accidents in the manufacturing industry as it was shown above. However, there are different ways of addressing the problem for reducing the number of injuries. Some of the most promising strategies are the use of autonomous robots and the application of CV approaches.

2.2.1. Autonomous Mobile Robots

According to Intel‘s description, “an autonomous mobile robot is a type of robot that can understand and move through its environment independently” [3]. These robots, use artificial intelligence systems to calculate the best route based on the environmental inputs received from their sensors and cameras. This behavior increases the fluency of travelling, since they are able to avoid obstacles and find faster routes depending on the situation without the need of an operator, as with the classical AGVs, which independency was subject to the conditions of the tracks and the correct oversight.

Furthermore, an AMR can be used for different tasks than simply moving inventory, like assisting in the picking process or performing sortation, thus, increasing productivity and order accuracy. Due to the fact that an AMR model can be equipped with different handling technologies, the sortation usability is managed by a barcode reader on the robot, and a rapid displacement to the storage location, where the item is tilted.

For the assistance in the picking process, there are two ways of using the AMRs:

- Conventional picking.

- Goods-to-Person Picking.

In the picking process, an operator must move from location to location, lossing a big amount of time. When using an AMR, it is possible to equip the robot with an order basket, so that the robot can perform these movements while the operator of every location takes the inventory. The Goods-to-Person Picking, on the contrary, consist on the movement of the entire shelve to the pick station. Then, the robot indicates the correct picks and quantities.

In any case, a fleet of AMRs can significantly reduce the number of forklift-related accidents within a warehouse if the number of SKUs is big enough [4].

2.2.2. Computer Vision approaches

2.2.2.1. VIA Mobile360

The company VIA Technologies, Inc., announced in October of 2021 the VIA Mobile360 Forklift Safety System [5]. The system uses voice and sounds alerts to warn the forklift driver of possible collisions through three IP67 HD cameras, two of them wide-angle cameras situated in front and rear of the forklift, and one inside the cabin.

According to the company, the area coverture has a radius of 4-5 meters far, which can be configured with either two or one warning zones. The alert is given if either a pedestrian is to close of the forklift safety zone (see Fig. 4), or the cabin camera detects signs of fatigue, smartphone usage or smoking.

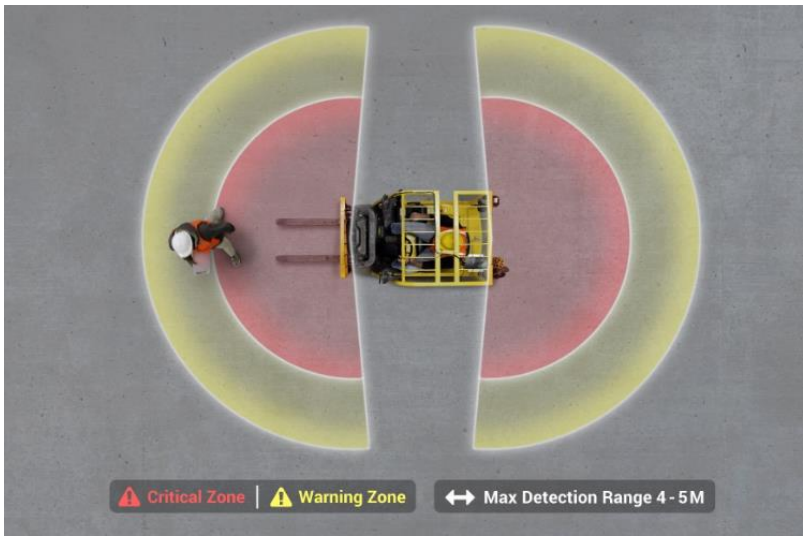


Fig. 4. VIA Mobile360 Warning and Critical Zones.

2.2.2.2. Warny

Warny [6] is a product developed by the Australian computer vision company Bigmate, focused on reducing the number of heavy vehicles-related accidents, as well as unexpected situations. They provide a full ecosystem to enhance work safety through three applications:

- Vehicle collision avoidance
- Safety zone alerting
- Thermal analysis of people and industrial systems

Warny (see Fig. 5), is capable of detecting combustions, fires, equipment overheating, and especially, a real time movement analysis, for example, alerting operators out of the sight of an upcoming driver. The system takes images from multiple standard CCTV cameras, which are used to calculate the ground-truth position of the object detected. Once the real positions are obtained, their velocities and relative distances are calculated in real time, thus, performing an analysis of the situation and giving the corresponding alert to the worker‘s device.

According to the article published in VentureBeat website, “Early into installation, one factory in Singapore saw a 22% drop in incidents (...) Across the board, on average, Bigmate is seeing a 80% drop in incidents in their clients‘ work environment” [7]. They ensure a network with nearly 100% accuracy and milliseconds of latency for the alerts. All the data is also collected to allow workplace analysis.

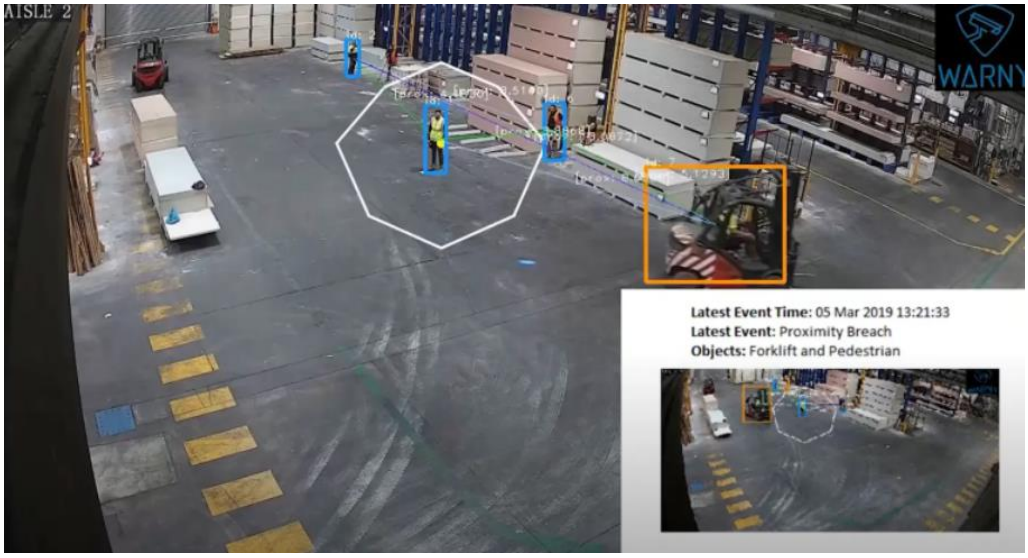


Fig. 5. Warny camera detections.

2.3. Theoretical concepts

In previous sections, two CV-based approaches to ensure work safety were presented. IBM gives the following description to this term. “Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and

other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand" [8]. Therefore, CV is the branch of artificial intelligence that teaches computer systems how to extract and understand visual information from the environment.

2.3.1. How does CV work?

In order to extract this information, CV-based systems take advantage of a DL concept called CNN, which is used to get features from the image in a set of convolutional layers, whose level of complexity increases with the depth of the architecture. These layers are used to create a feature map with the most meaningful information of the image. Then, the model takes this feature map to understand the context of the image through a classification model.

2.3.2. Digital images

First, it is important to know what a digital image is. A digital image is a matrix of elements from 0 to 255, in which the 0 represents the minimum level of luminosity and 255 de maximum. For example, in gray scale images, the 0 is a black pixel, whereas 255 is a white pixel, see Fig. 6.

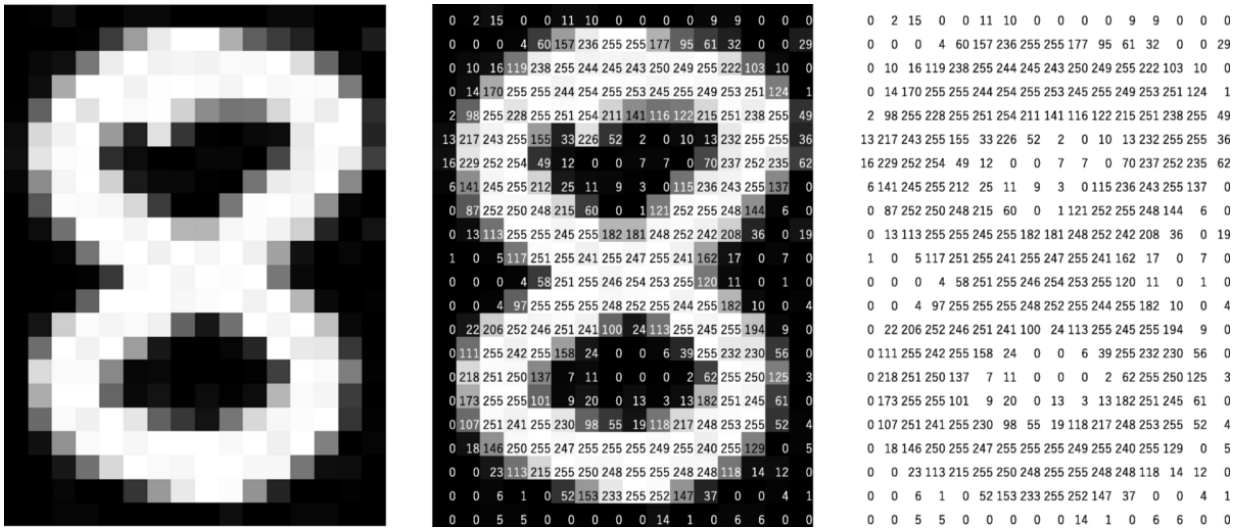


Fig. 6. Gray scale digital image.

However, digital images are usually composed by different color channels, where the scale does not necessarily mean the brightness, but some source of information. The most common color space for visualization is RGB (see Fig. 7), where each channel corresponds to the brightness of each color (red, green, and blue). Other color spaces like HSV, LAB or LUV are not suitable for visualization, but they contain useful information to increase the robustness against light changes in CV-based applications, like hue or saturation channels in HSV space.

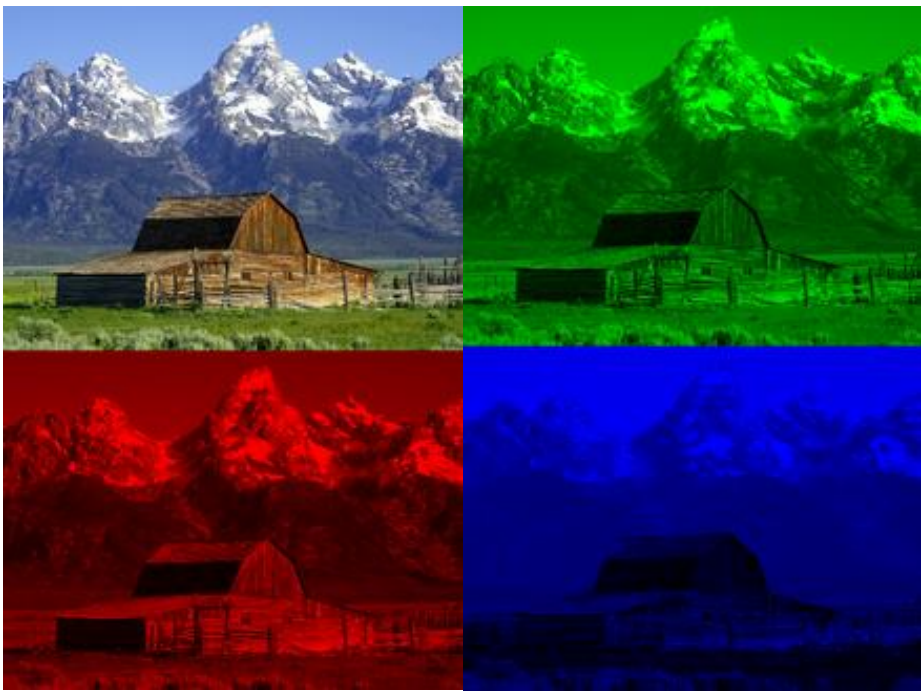


Fig. 7. RGB image.

2.3.3. The Convolution

A convolution is a sum of element-wise products between two matrixes. In computer vision, these two matrixes are the raw image and the kernel (see Fig. 8), where the kernel is an odd smaller matrix, whose elements are used to highlight some features of the raw image. The kernel is also named as filter.

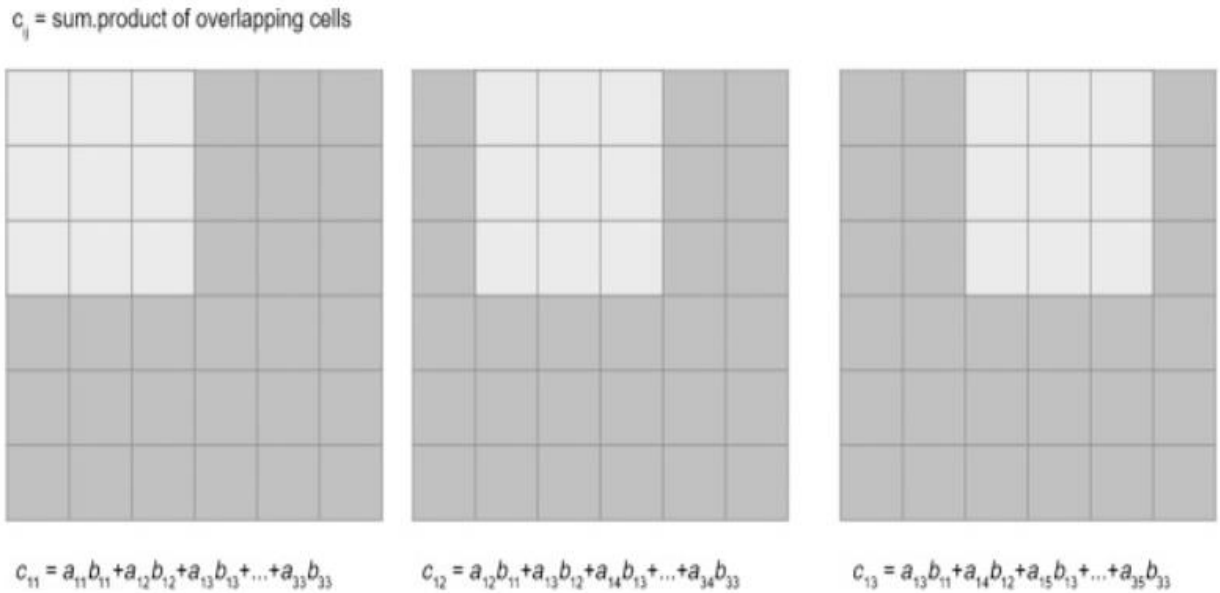


Fig. 8. Convolution [9]

Then, the kernel convolves through the different cells replacing the number that falls in the center of the overlapped elements. However, this will result in a reduction of the input size. To avoid so, a padding process is applied to the image during the convolution. This padding process consist of adding pixels to the border of an image, for example, a 3x3 kernel operation might want to use 1-unit padding, which will add one extra pixel to each one of the edges. There are different types of paddings, but the most common is the zero padding, which fills these extra cells with zeros, see Fig. 9.

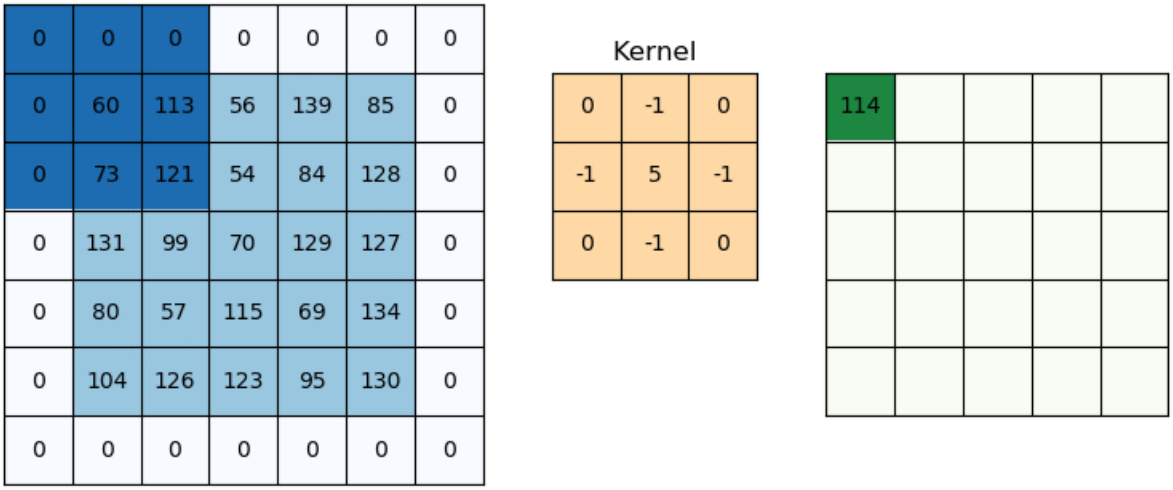


Fig. 9. Convolution with zero padding.

Another important parameter in relation to the output size is the stride, which is defined as the number of cells that the kernel moves after each iteration.

2.3.4. Pooling

Sometimes, in CV-based systems it may be desirable to get rid of non-significant information, resulting in a smaller summarized picture. This is done with the pooling operation, which fuses meaningful information across the image using a filter. Depending on how this operation is performed, one result or another will be obtained, two main types are Max Pooling and Average Pooling.

The max pooling takes the maximum value of the area covered by the filter, thus, extracting the most significant features of each region, while the average pooling takes the mean [10]. In Fig. 10, there is an example of both.

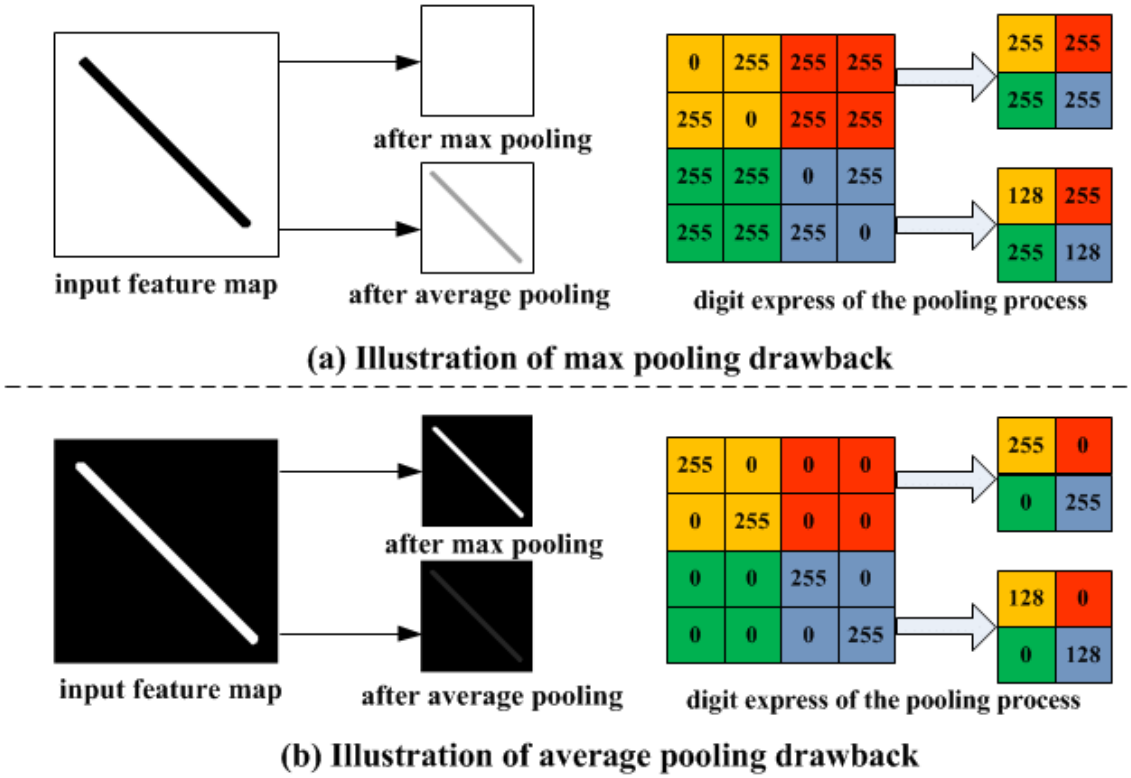


Fig. 10. Max and Average Pooling examples.

2.3.5. Linear Regression

The concepts shown above are important to understand how the features of an image can be extracted. Nevertheless, the following sections are more related to learning process in AI systems.

Linear regressions are one of the most important and basic concepts in supervised Machine Learning (ML). Basically, a linear regression is a statistic model that tries to find the line that best fits a set of supposedly related values, where the y-axis corresponds to the dependent variable, and the x-axis the independent variable. In supervised machine learning problems, the aim is to predict this y value, given a single x input, or set of inputs. Therefore, there are simple, equation 1, and multiple linear regressions, equation 2.

$$y = b_0 + w_1 x \quad (1)$$

$$y = b_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \quad (2)$$

The line that best fits the model (see Fig. 11) is obtained through the slope and the intercept, which are usually referred as weight (w) and bias (b). These two parameters are calculated for a given training set as the optimal values for which the error is minimum, where the error is the distance between the predicted and the actual value. Then, the accuracy of the model is validated in a test set.

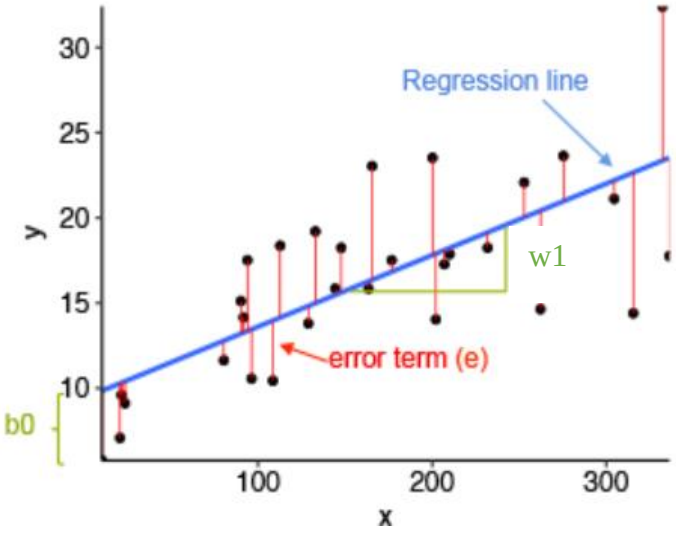


Fig. 11. Simple linear regression [11].

2.3.6. Logistic Regressions

However, in some problems it would be necessary to predict the probability of belonging to a certain class, since the dependent variable can be categorical, for example, in binary classifications. Here, a linear regression approach might cause some problems by the fact, that a linear model allows the result to be out of the boundaries of the problem, i.e. probabilities higher than one and lower than zero, see Fig. 12. It can also bring problems with the outliers, since the line that best fits the model can be shifted to decrease the distance error between the farthest outliers and the rest of the data, causing miss-classifications.

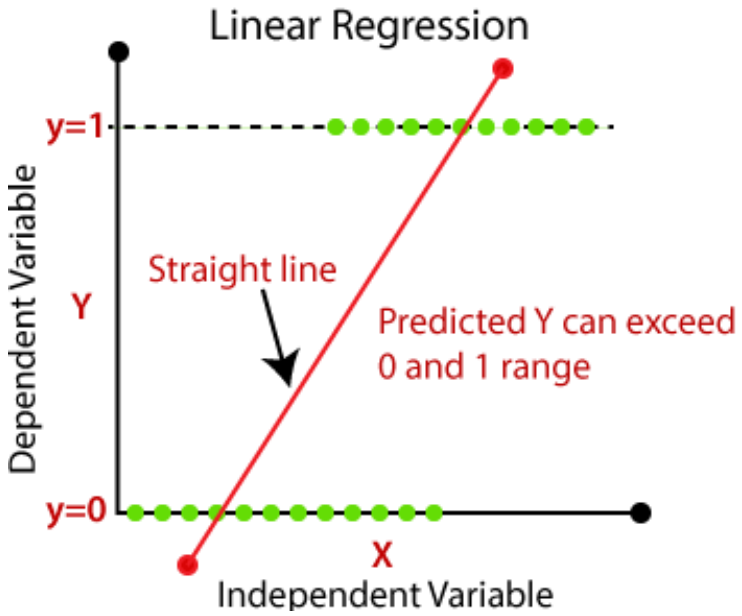


Fig. 12. Linear regression for a binary classification problem.

Therefore, a logistic regression approach is more suitable for this type of problems, where rather than a straight line it is used an S-curve, so the predictions can fall within the boundaries of the problem, and the farthest outliers do not affect much to the model [12]. To do so, a logistic function must be applied to the linear calculations.

$$\sigma(z) = \frac{1}{1+e^{-y}} \quad (3)$$

For example, equation 3, corresponds to the sigmoid function, it makes the output of the linear prediction to fall between 0 and 1, thus, resulting in a much more accurate model. A threshold value it is also set to classify the predictions, see Fig. 13.

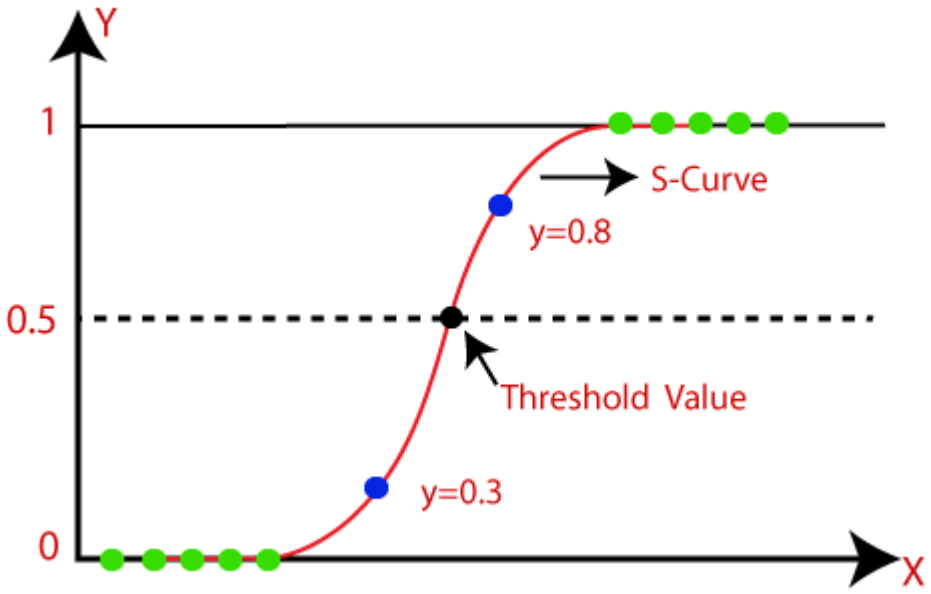


Fig. 13. Sigmoid function applied to the linear regression with a threshold of 0.5.

2.3.7. Cost functions.

To train a model in supervised learning algorithms, the output has to be evaluated iteratively until it reaches a good accuracy. This is done with a cost function, either called loss. The cost function compares the output of the model with the real output of the dataset in order to get the error of the prediction. To do so, there are many approaches depending on the type of problem, but for regression problems, the loss is a distance-based error calculation.

Based on this distance, there are two famous approaches, the MSE (equation 4), or L2 loss, and, MAE (equation 5), or L1 loss [13].

$$MSE = \frac{1}{N} \cdot \sum_{i=0}^{n} (y - y')^2 \quad (4)$$

$$MAE = \frac{1}{N} \cdot \sum_{i=0}^{n} |y - y'| \quad (5)$$

The fact that the L2 loss is squared, means that it penalizes higher the outliers of the dataset. However, the L1 loss is more robust in this aspect and also performs well with noisy data.

2.3.8. Artificial Neural Networks

An Artificial Neural Network is a deep learning model arquitecture directly inspired from the human central nervous system. The information in the human body flows in form of electrical impulses through billions of interconnected cells called neurons, see Fig. 14.

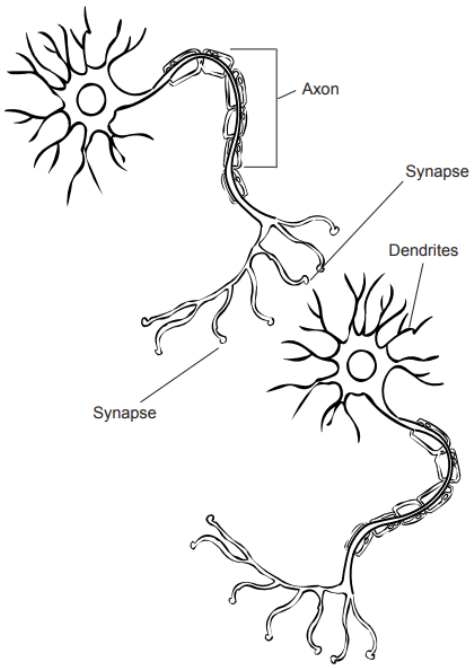


Fig. 14. Structure of a Neuron

Every single neuron has a cell body from which a set of small branches called dendrites grow, as well as a long fiber known as the axon. The information flows from the whiskers of the axon, i.e. the synapses, to the dendrites of another neuron. Then, the information from many neurons is processed in its body cell and joined together to be sent to a new neuron. Thus, constantly activating

certain neuron connections over the others, the brain learns which neurons will be more likely to use to get the expected output.

An artificial neural network works pretty much the same way, where there is a neuron that receives inputs from many different others, and send a response according to them. Its architecture is composed by an input layer, a set of hidden layers and the outputs, see Fig. 15.

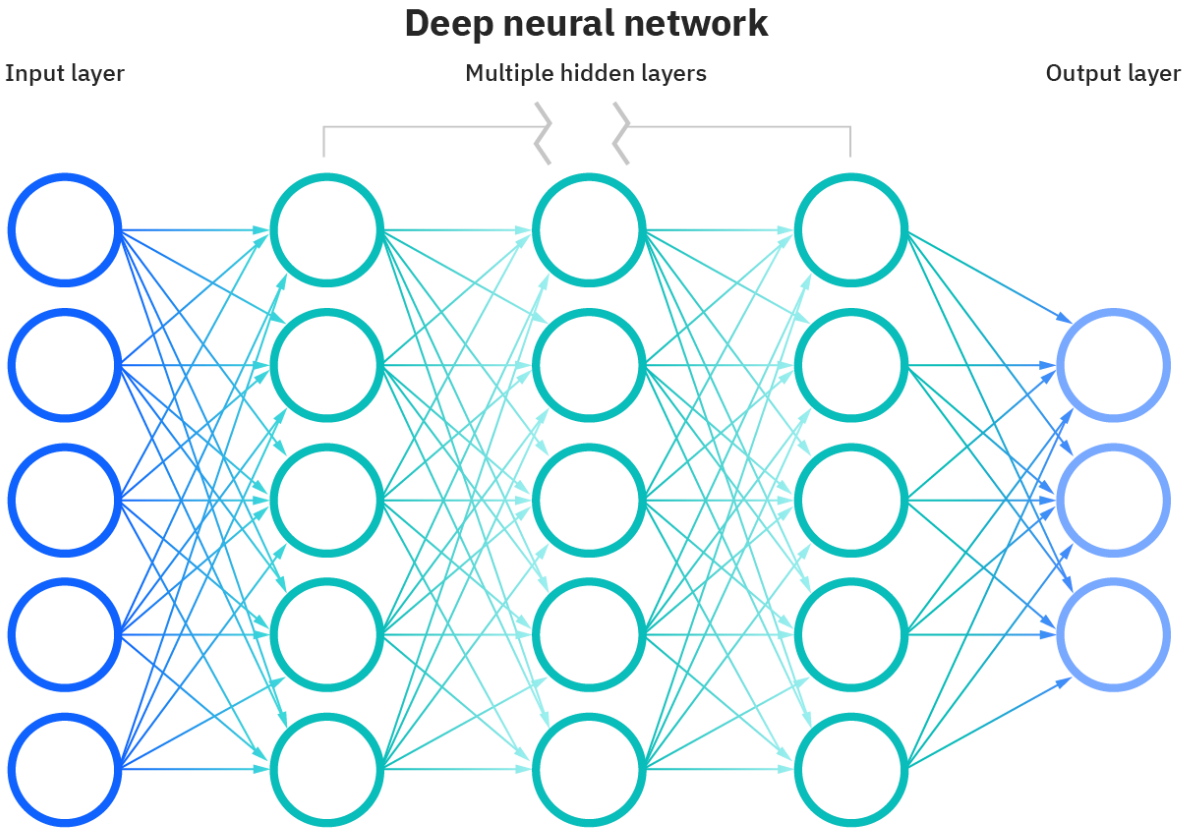


Fig. 15. Structure of a Deep Neural Network

The hidden layers can be understood as the central nervous system, where the inputs of the neuron are the dendrites and the outputs the synapses. Every neuron of an artificial neural network performs a linear regression followed by an activation function, which can be either linear, hyperbolic or logistic as seen above. On the contrary, rather than electrical impulses, the information flows in form of weighted values, which are adjusted depending on the neuron. The output of a neuron is defined in equation 6.

$$o = f(z) = f\left(\sum_i w_i x_i + b\right) \quad (6)$$

Each neuron might find different associations for the input data that can end into the expected value or not. For this reason, after reaching the output layer and calculate the loss, the error of the neurons

is back-propagated to search the path that has had the biggest implication in the result to reinforce it. This process is known as backpropagation.

The weights and bias of the neurons are randomly initialized and automatically adjusted through this backpropagation algorithm. Once the network has arrived with a result, is more efficient to search the error implications of each neuron backwards than calculating forward every change in the net. Therefore, if every neuron in the final layer has an implication in the error, it will be more likely to have a neuron that has contributed more in the result than de others. This process can be repeated for each layer discarding the neurons with less contribution in the result, thus, back propagating the error until the first layer. This way can be obtained the parameters for the calculations of the gradient descent in the training process [15], see equation 7.

$$\frac{\partial C}{\partial b^L}, \frac{\partial C}{\partial w^L} \rightarrow l = 0,1,2, \dots, L; \tag{7}$$

Here, $\frac{\partial C}{\partial b^L}$ and $\frac{\partial C}{\partial w^L}$ are the partial derivatives of the cost with respect the bias and weight respectively.

2.3.9. Optimization functions.

After the cost calculations, the parameters of the function must be optimized to find the minimum possible error. In convex functions, it can be obtained matching the cost derivative to zero, but for non-convex functions, this approach faces the problem of finding several points where the slope can be zero, like peaks or saddle points. Hence, the Stochastic Gradient Descent or ADAM approaches are betters methods for the optimization task, especially in ANNs.

The Stochastic Gradient Descent consists on a partial derivative calculation of the parameters of the function. This gradient can be subtracted on a certain percent, the learning rate, to find a minimum in the function [14].

$$\theta_{n+1} = \theta - \eta \nabla f \quad \vee \quad 0 < \eta < 1 \tag{8}$$

Where

$$\nabla f = \begin{bmatrix} \frac{\partial C}{\partial \theta_1} \\ \frac{\partial C}{\partial \theta_2} \\ \dots \\ \frac{\partial C}{\partial \theta_n} \end{bmatrix} \tag{9}$$

Here, η is the learning rate.

For small learning rates, the algorithm might never converge, while for larger ones the jump after each iteration can be such, that it turns impossible to find the minimum. The following picture, Fig 16, shows a 3-Dimentional representation of this process.

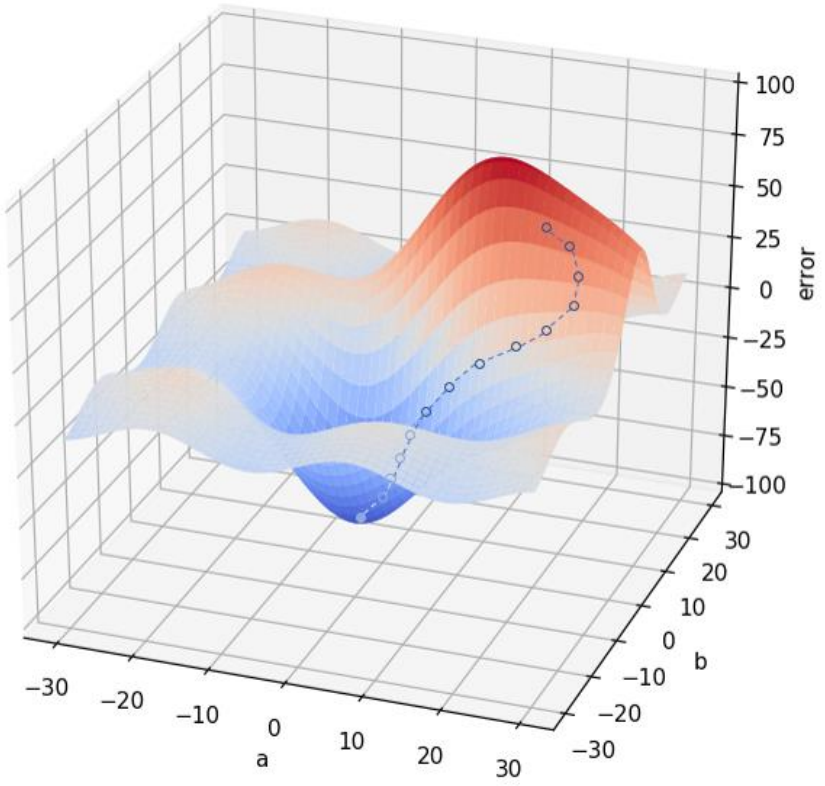


Fig. 16. Gradient Descent 3D representation.

2.3.10. Activation Functions.

As mentioned above, every linear regression in a neuron comes with an activation function. However, a common issue in deep neural networks is the vanishing gradient problem during backpropagation. Here, the chain rule is used for the gradient descent calculations, this means that after multiple concatenations, the gradients from the first layers will be inevitably smaller than the lasts, tending to vanish at a certain point if the activation output is also constrained within small boundaries. Hence, the activation function has a direct implication in the performance of the network. For this reason, there are some functions which are more likely to perform better under certain circunstances than others. Non-linear functions tend to give better results, for example, a very popular activation function in CNNs is ReLU, which maximizes positive inputs, but returns zero otherwhise, i.e. $R(z) = \max(0, z)$.

$$R(z) = \begin{cases} 0\,, & x \leq 0 \\ x, & x > 0 \end{cases} \qquad (10)$$

It performs well in most of the cases and avoids the vanishing gradient problem. Nevertheless, negative neurons are completely forgotten during the training. Leaky ReLU (see Fig. 17) solves this problem multiplying all negative inputs for a small factor, α, while keeps maximizing positive inputs [16].

$$LR(z) = \begin{cases} \alpha x\,, & x \leq 0 \\ x, & x > 0 \end{cases} \quad (11)$$

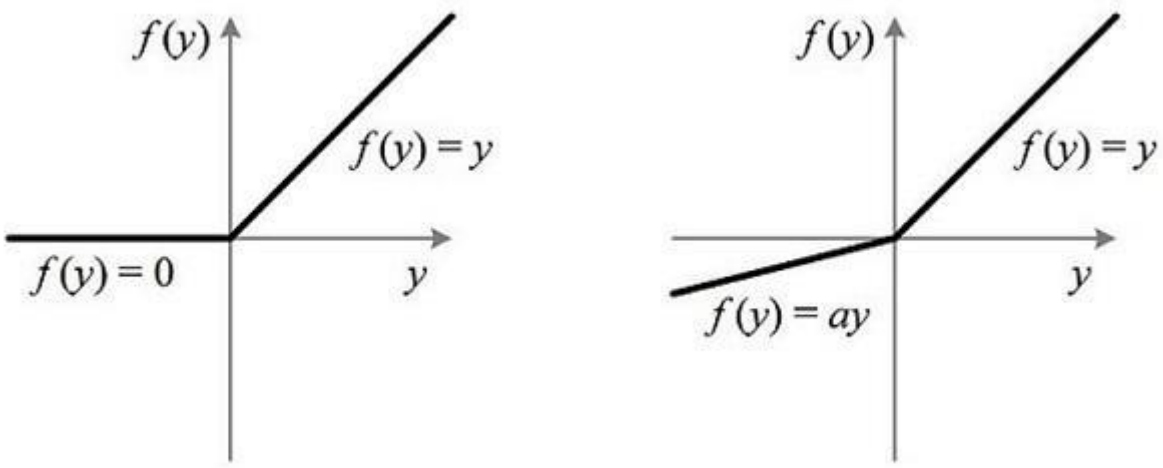


Fig. 17. ReLU and Leaky ReLU activation functions.

Another famous activation function that has been already explained is the Sigmoid function. It works well for binary classification problems, but despite of its popularity, it is computationally expensive and converges slower than, for example, the Tanh activation function (see Fig. 18), which is zero-centered between -1 and 1.

$$\tanh(z) = \frac{2}{1 + e^{-2z}} - 1 \quad (12)$$

On the contrary, for multi-classification problems, Softmax gives better results, so it is used at the end of the network. It returns a probability between 0 and 1 like the sigmoid [17].

$$Softmax(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (13)$$

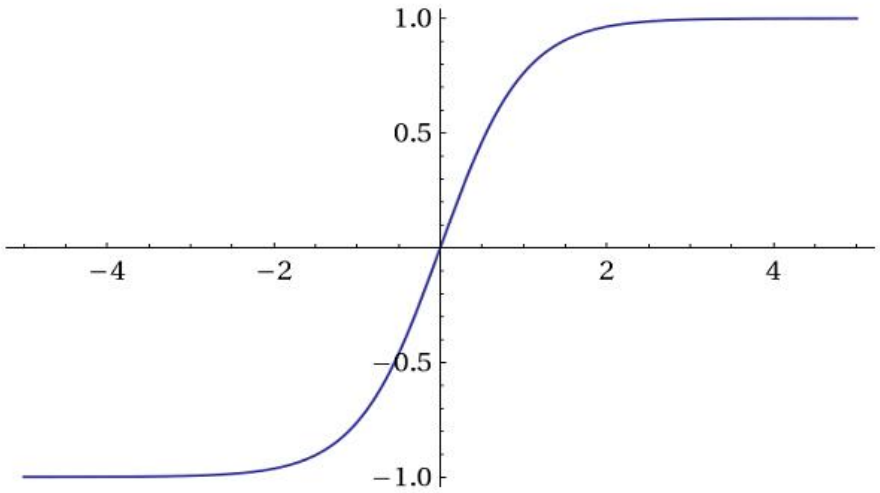


Fig. 18. Tanh activation function.

2.3.11. CNNs

All things consider, a DL model can be configured in multiples ways so that the network learns to perform different tasks based on the functions used and the connections of its layers. There are dozens of different configurations and ways of building an ANN, one of the most famous is named CNN and is fundamental for CV-based applications, as it was mentioned in previous sections.

"The first work on modern convolutional neural networks occurred in 1990, Gradient Based Learning Applied to Document Recognition" [18] . In this paper, it was found that the concatenation of filters throughout different layers ended in a progressive transformation from simple to complex feature maps. These feature maps being incremented along with the depth of the network. In this approach, the increment was from 6 to 16 after the first subsampling, but in modern CNNs the final layers are made by hundreds of them. Then, the feature map is usually flattened and each cell is used as an input for the classification model. A ReLU activation function is commonly used here as well as during the convolutional layers, but the process usually ends with a logistic function for the object classification, for example, the Softmax function if there are multiple classes. Therefore, two parts can be distinguished in a CNN, a feature learning, and a classification part. Here is a visual representation of the structure just explained, see Fig. 19.

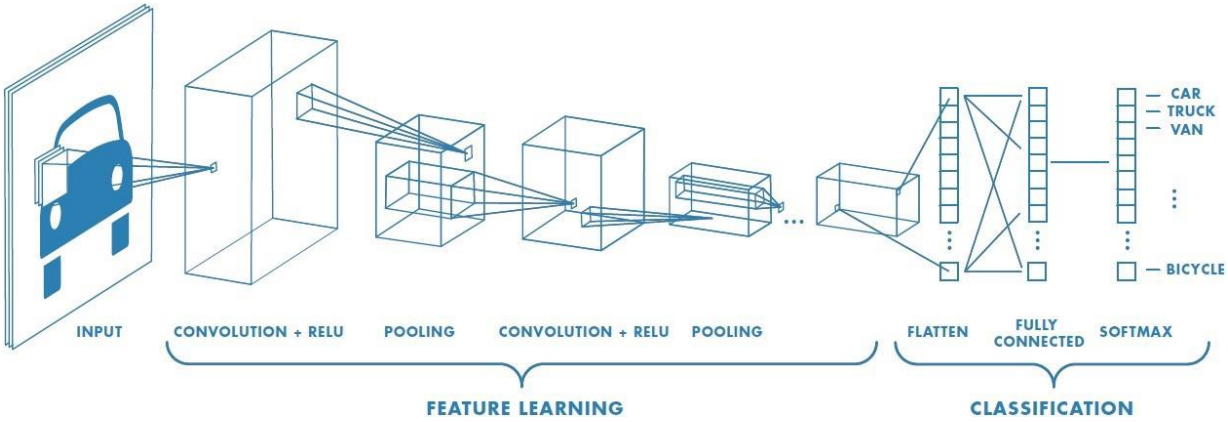


Fig. 19. Typical structure of a CNN

The learning process for the image classification requires a dataset of labeled images. These images must be different and should have different angles and different levels of luminosities, so that the network could learn to perform in different scenarios.

2.4. Object Detection

For the time being, the process of image classification has been explained, but this networks are still unable to know where the object is. To do find the object, there is something called object detection, which by making use of a proposal area or bounding box, teaches the network to also locate the object. In this section, an overview of some approaches to this problem are going to be presented, as well as the object detector used.

2.4.1. Sliding Windows

The first method ever used to find the location of an object was done by the sliding-windows technique. This technique, consists on taking a crop of the image to predict whether a particular object is on that window or not. Then, the window is moved along the image with different strides and sizes performing this predictions until the network ends finding a window that contains the object that was looked for, see Fig. 20.

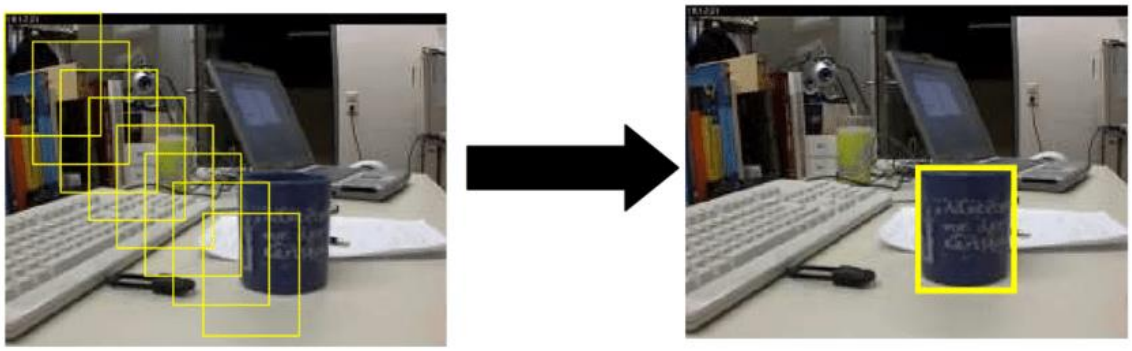


Fig. 20. Sliding Window visual example.

2.4.2. DPM

One of the classical approaches that use this technique is Deformable Parts Model (DPM). This approach takes into account that either a person or an object can have different shapes, poses and colors. In real situations, an object can be partially occluded, what might lead into an accuracy reduction. DPM, uses the sliding windows technique to find the bounding box of an object and then, doubles the size of the image to take a crop of the different parts within that bounding box from where to extract more features. Within these crops, it is later computed how far are the parts from the body of the object. The following picture, Fig. 21, shows a diagram of this process.

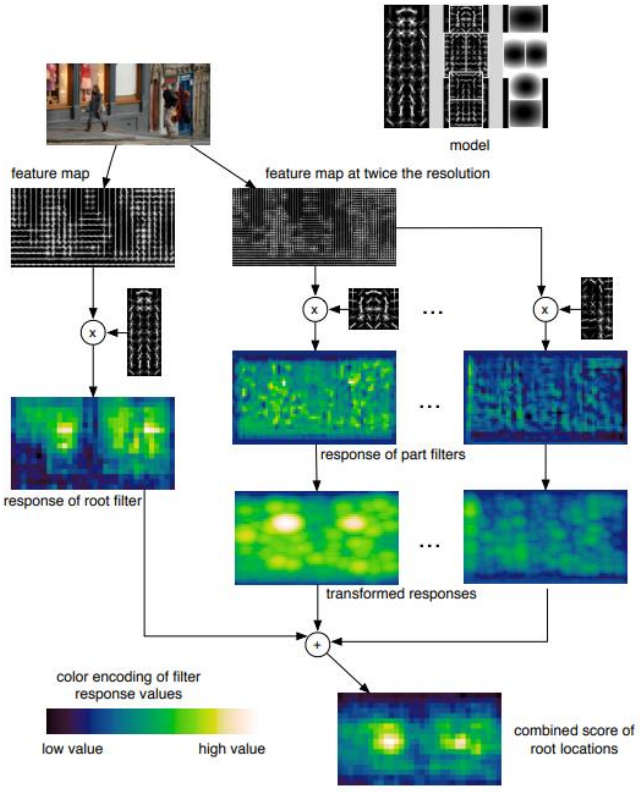


Fig. 21. DPM score process.

2.4.3. Region Proposals Network

Another famous technique for choosing possible bounding boxes within an image, is a region proposal method, which is done by a selective search of areas at different scales ratios. These areas, are found through a segmentation at different scales, resulting on a variety of several shapes for which the bounding box that fits that region is proposed. The regions are proposed by an hierarchical grouping of the most similar regions at different color spaces. Then, the color similarities are computed through their histogram intersections, as well as the textures similarities using SIFT measurements and Gaussian derivatives. The size of the region is also computed, allowing the smaller regions to merge earlier. Finally, the way in which each region fits into the

other is taken into account for the filling process. Fig. 22 shows an example of this process at different scales.

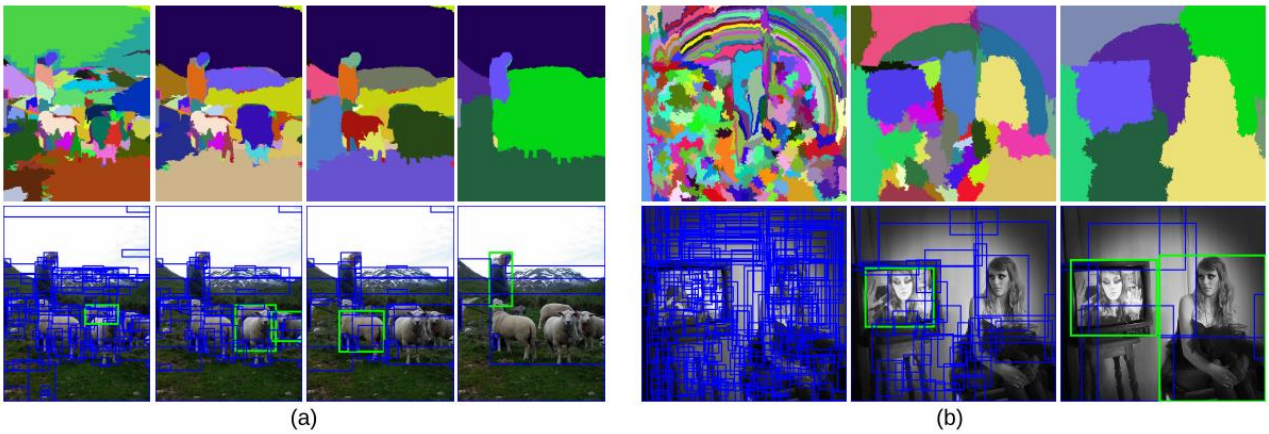


Fig. 22. Two examples of selective search at different scales. Note that (b) has detected the girl on the TV. [20]

A famous approach that implements this region proposal network is the R-CNN family, a two stages object detector where the bounding boxes for the feature extraction are got using this selective search, and a SVM algorithm is used for the classification process.

2.4.4. YOLO

The main problem with these object detection models, is the high computational cost of their methods, which makes them incredibly slow. Even the improvements done by Fast R-CNN and Faster-RCNN, were so far away from being real-time approaches. This is the purpose for which YOLO was intended. The developers of this approach, managed to solve this frame rate problem by reframing it as "a single regression problem" [21], with a lot less bounding boxes instead of the conventional region-based and sliding windows techniques.

YOLO is the choice for the object detection task in this project, but in order to comprehend how it works nowadays, it is important to first walk through its different versions.

2.4.5. YOLOv1

The first version of this algorithm, began "simultaneously predicting bounding boxes and class probabilities for those boxes" [21]. It worked over the entire image, thus, resulting on the encoding of the contextual information linked to the prediction. This was a great feature, because it allowed the model to be more generalizable in a wide range of situations while maintaining larger frame rates.

To accomplish this achievement, YOLO introduced an $S \; x \; S$ grid whose cells were responsible for providing these predictions. Every cell returned a B number of bounding boxes and a C number of class probabilities. Each bounding box contains the information of the x and y coordinates relatives

to that cell, the width and height relatives to the entire image and the IOU confidence that there was an object, ideally 1. Therefore, the prediction was an $S\ x\ S\ x\ (B * 5 + C)$ tensor limited to one object per cell. This limitation in the number of objects that can be detected was important, because it directly affected to the performance with smaller objects.
Nevertheless, what made YOLO extremely fast was not only the reduction in the number of bounding boxes proposals compared to other techniques, but its network design, which was closely based on the GoogLeNet architecture. The main difference here, remained in the fact that YOLOv1 uses just $1\ x\ 1$ reduction layers before the $3\ x\ 3$ convolutional layers instead of the inception modules. This was named Darknet. Darknet's architecture was built under "24 convolutional layers followed by 2 fully connected layers" [21] and took $448\ x\ 448$ RGB images as inputs, see Fig. 23.

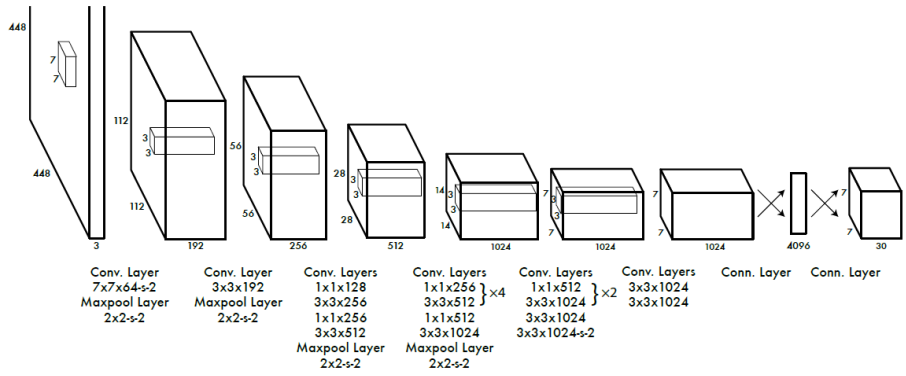


Fig. 23. Darknet [21].

Proceeding with the activation functions, a linear activation function was used for the final layer and a leaky rectified linear activation was used for the rest.

This first approach was trained with a grid size of $S = 2$, using the Pascal VOC Dataset, which contains a list of 20 different classes, each grid cell returning $B = 2$ bounding boxes, making a total of 98 proposals, much less than the thousands of proposals in previous methods. Finally, the model was optimized with the sum-squared error loss.

The results were quite impressive in terms of speed, achieving around 45 FPS, but it lacked in terms of recall and mAP, see Table 2.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN	2007+2012	70.0	0.5
Fast R-CNN VGG-16	2007+2012	73.2	7
Fast R-CNN ResNet	2007+2012	76.4	5
YOLO	2007+2012	63.4	45
SSD300	2007+2012	74.3	46
SSD500	2007+2012	76.8	19

Table 2. Detection frameworks on PASCAL VOC 2007 [22]

2.4.6. YOLOv2

Later, YOLOv2 was released and with it, several changes within its architecture. One of the most significant changes made in YOLOv2 with respect to YOLOv1 to improve the location problems, was the implementation of anchor boxes (see Fig. 24) instead of predicting the coordinates directly.

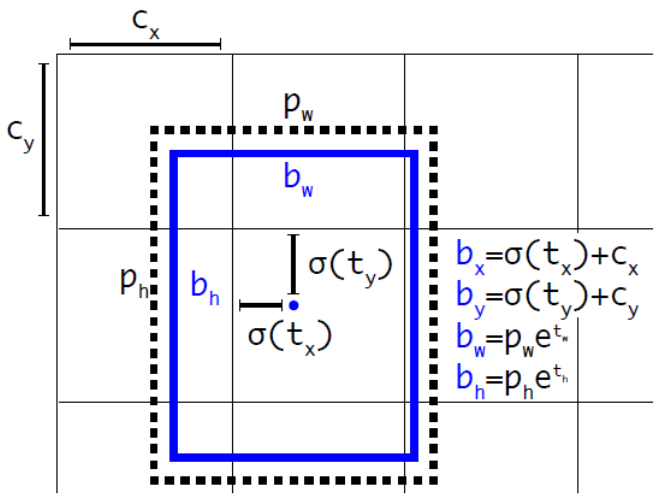


Fig. 24. Bounding boxes with dimension priors and location prediction [22]

This anchor boxes allow the network to slightly adjust their size, which is much easier than predicting bounding boxes from scratch. These anchors, are chosen running a "k-means clustering on the training set bounding boxes to automatically find good priors" [22], thus, improving the results in the average IOU compared to hand-picked priors, see Table 3.

Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes	9	60.9
Cluster IOU	9	67.2

Table 3. Average IOU of boxes to closest priors on VOC 2007 [22]

Here, it is showed how, indeed, the clusters can achieve the same results than the hand-picked priors but using almost half of them.

However, since the anchor boxes led to model instability, the approach made in YOLOv1 was kept in terms of maintaining location coordinates relative to each grid cell. Therefore, t_x, t_y, t_w, t_h and t_o were defined as the new five predictions for each bounding box, whose x and y coordinates, width, height and IOU correspond to the following terms.

$$b_x = \sigma(t_x) + c_x \quad (14)$$

$$b_y = \sigma(t_y) + c_y \quad (15)$$

$$b_w = p_w e^{t_w} \quad (16)$$

$$b_h = p_h e^{t_h} \quad (17)$$

$$\Pr(object) * IOU(b, object) = \sigma(t_o) \quad (18)$$

Where p_w and p_h are the width and height of the anchor box and $c_x$and c_y the offsets of that cell.

Another important change made in YOLOv2, was the new network design, Darknet 19 (see Fig. 25), in which the class predictions were "decouple (…) from the spatial location" [22]. This meant, that the prediction then became an $S\ x\ S\ x\ B(5 + C)$ tensor. This fact indicated that the loss would not only depend on the cell, but the index of the anchor too. On the other hand, the fully connected layers were also removed due to the use of anchor boxes for the bounding box predictions, and the input resolution was reduced to 416, downsampling the output image by a factor of 32. This way, allowing the implementation of a multi-scale training for multiples of 32 from 320 to 608. The multi-scale training, allowed YOLOv2 to learn how to perform in different scenarios, becoming faster at lower resolutions, and more accurate at higher, see Fig. 26.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Fig. 25. Darknet 19 [22]

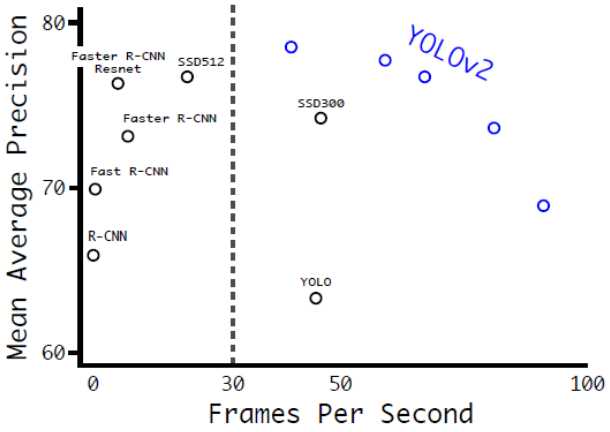


Fig. 26. Accuracy and speed on VOC 2007 [22]

2.4.7. YOLOv3

In YOLOv3 there is just one important change with respect to YOLOv2, since the implementation of the anchor boxes, the new Darknet 19 and the multi-scale training were incredibly successful on improving recall and mAP without losing speed in the process. It was a new version of Darknet, Darknet-53 (see Fig. 27), which merged the previous structure used in Darknet-19 with the residual blocks introduced in ResNet [23] within an FPN. These residual blocks, allow the network to become deeper while solving the vanishing gradient problem. This is the main reason of the new three different levels of predictions, where the feature map is upsampled after each one of them. Thus, allowing the net to perform better with smaller boxes. For the class predictions, each bounding box may now have a multilabel classification, using independent logistic classifiers, and the binary cross-entropy loss during training.
Table 4 shows how YOLOv3 reaches the same mAP than other methods, but clearly overcoming them in terms of speed.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 27. Darknet 53 [24]

Method	mAP-50	time(ms)
SSD321	45.4	61
DSSD321	46.1	85
R-FCN	51.9	85
SSD513	50.4	125
DSSD513	53.3	156
FPN FRCN	59.1	172
RetinaNet-50-100	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.2	198
YOLOv3-320	51.5	22
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Table 4. Speed/mAP-50 between YOLOv3 and other methods [24]

2.4.8. YOLOv4

Previously, it was mentioned that object classifiers are composed by two main parts, in object detection, these two parts are named as backbone, which extract the features, and head, which perform predictions, but ever since most recent develops in object detection are including a way of mix together feature maps from different stages, a "neck" has started to take part in the structure. Thus, the last version of YOLO, YOLOv4, implemented several new features to Darknet-53 to improve its performance on single conventional GPUs. The following picture, Fig. 28, shows this structure.

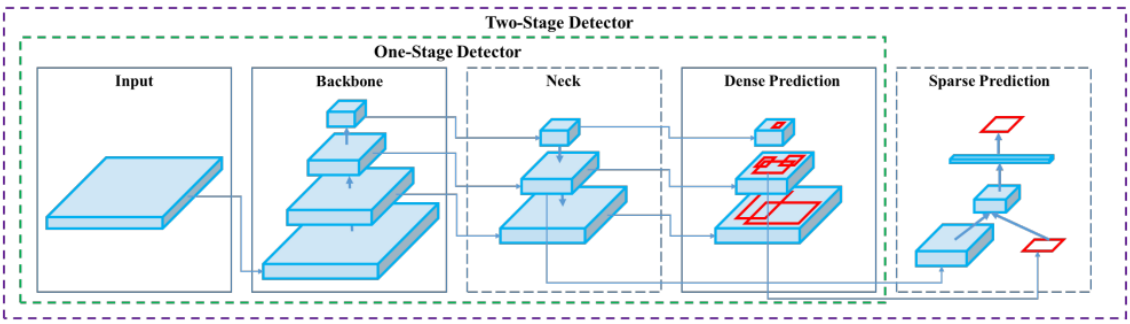


Fig. 28. Object detector [25]

The new backbone in YOLOv4 maintained Darknet-53, but merging it with a CSPNet on top of its basis. The CSPNet, takes just one portion of each output before the next convolutional layer, thus, resulting in lighters convolutional blocks. The rest, is concatenated in the transition layers, see Fig. 29.

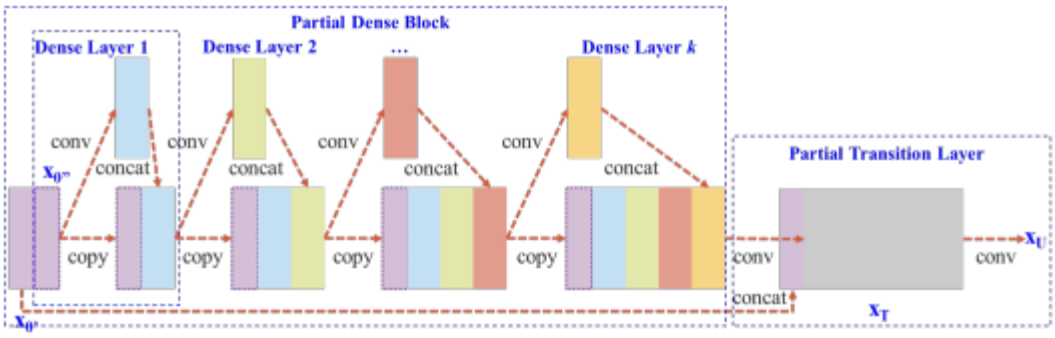


Fig. 29. CSPDenseNet example.

Usually, network lightening leads to worst performances, but on the contrary, this technique results in a reduction of the computational cost from 10% to 20% and still outperforming in terms of accuracy, also reducing memory costs and "removing computational bottlenecks" [26]
Furthermore, YOLOv4 adds a modified version of the SPP model to its model neck, setting a multi-scale max pooling with different kernel sizes but stride 1, and a subsequential concatenation of those outputs. This block not only improves the receptive field but screens the most significant context features with nearly 0 speed reduction. The neck also includes a modified PANet [27] for the aggregation of parameters, using element concatenations instead of element-wise additions. The head remains the same as in YOLOv3

Ultimately, a Mosaic data augmentation was included, next to a Self-Adversarial Training and a modified version of the CBN, it was called CmBN. Mosaic data augmentation helps to add more context information within one image, reducing the necessity of larger mini-batch. This new mosaic data augmentation works well with Cross Batch Normalization, as the tendency of higher resolution images has grown in recent years. In this case, the CmBN "collects statistics only between mini-batches with a single batch" [25].

Fig. 30 shows YOLOv4 performance.

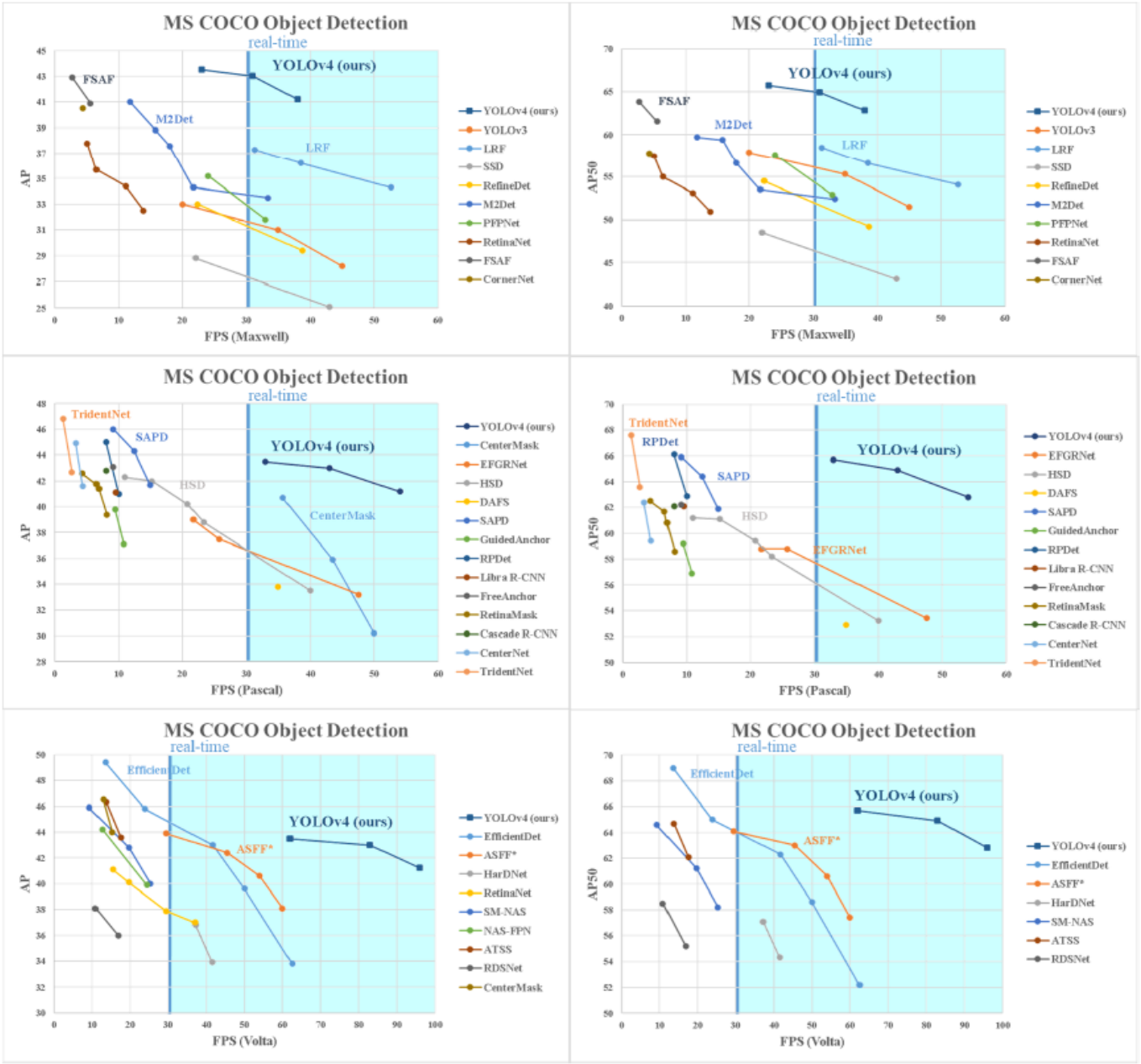


Fig. 30. Comparison of the speed and accuracy of different object detectors [25]

This figure shows how YOLOv4 outperforms different object detectors in terms of speed and works in par in terms of AP due to its previously explained model design, being a very balance and suitable option for real-time object detections.

2.5. Object Tracking

Object detection is often coupled with object tracking for many reasons, since we may not only need to know where a certain object is, but its trajectory. This has become a very important task that must be taken into consideration if we talk, for example, about autonomous vehicles or security systems for collision avoidances.

2.5.1. DeepSORT

One of these approaches often coupled with YOLO is DeepSORT, a deep learning improvement of a famous object tracker released in 2007 called SORT.
SORT

The methodology followed by SORT, takes advantage of the nice performance of another two famous methods for motion estimation and combinational optimization, the Kalman Filter [28] and Hungarian algorithm.

SORT methodology has 4 main steps:
1. Detection;
2. Estimation;
3. Association;
4. Creation and Deletion of Track identities.

The detections can be made by any object detector, the better detections, the greater performance in SORT.

2.5.2. Estimation model. Kalman Filter.

The Kalman Filter is a probabilistic state estimator that fuses information from different sources to give a final estimate for unknown states. The estimate is calculated through both the predicted motion model and measurement model distributions, taking into consideration a certain noise for each one of them.
The estimation model is done by the Kalman filter using the following state vector.

$$X = [u, v, s, r, vu, vv, vs]^T \quad (19)$$

Where u and v are the horizontal and vertical centre coordinates of the target, s is the scale area of the bounding box and r the aspect ratio of that bounding box. The rest of the parameters are their respective velocities.
For a 1-Dimentional state, the predictions are given by.

$$x_k^{pred} = F_k x_{k-1} + B_k u_k \quad (20)$$

$$P_k^{pred} = F_k P_{k-1} F_k^T + Q_k \quad (21)$$

Where

$$F_k = \begin{pmatrix} 1 & \delta t \\ 0 & 1 \end{pmatrix} \quad (22)$$

$$G_k = \begin{pmatrix} \frac{1}{2}\delta t^2 \\ \delta t \end{pmatrix} \quad (23)$$

Where x_k is the state vector, P_k the covariance, u_k a control input noise, and F_k, B_k and Q_k are the transition, control and covariance noise matrix, respectively.

Then, the measurent is used to correct the prediction based on the innovation and some gain. This kalman gain is also used to propagate the state covariance.

$$x_k^{corrected} = x_k^{pred} + K_k * Innovation \quad (24)$$

$$P_k^{corrected} = (I - K_k H_k) P_k^{pred} \quad (25)$$

Where

$$K_k = P_k^{pred} H_k^T \left(H_k P_k^{pred} H_k^T + R_k\right)^{-1} \quad (26)$$

$$Innovation = (y_k - H_k x_k^{pred}) \quad (27)$$

Here, y_k corresponds to the measurement, K_k is the Kalman Gain, H_k is the relation matrix between the predicted state and the measurements in case of having different input units, and R_k is the measurement covariance noise.

2.5.3. Data Association. Hungarian Algorithm.

For the data association, a cost matrix is calculated based on the IOU distance between all the predicted and detected objects. Later, the Hungarian algorithm matches the bounding boxes.
The Hungarian algorithm is an optimization algorithm which finds the minimum values for each not repeated row and column within a given $N\ x\ N$ matrix in assigment problems. The fundamental of this problem is that the optimal assigment for a reduced cost matrix, is also the optimal assigment for the original cost matrix.

2.5.4. Track Identities.

Finally, the new identities are created only if their IOU is greater than a fixed minimum distance and remain for a certain amount of time. Then, if the identity is lost for a T_{Lost} number of frames, it is eliminated.

Fig. 31 shows SORT performance.

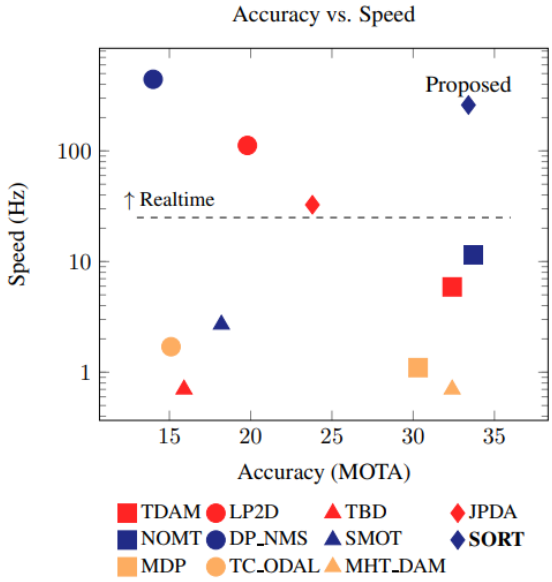


Fig. 31. SORT performance in relation to several baseline trackers. [29]

As shown, SORT had a nice performance in comparison to other object trackers. The problem in SORT was the high number of switches, since it does not perform well under high state uncertainties, for example, during occlusions.

2.5.5. Deep Association metric for SORT

To improve this behavior, in 2017 was introduced a deep association metric for SORT. They included an appearance feature descriptor to consider along with the state estimation in order outperform SORT previous results. There are also some little changes with respect the original SORT, for example, the state vector.

$$y_i = (u, v, \gamma, h, vu, vv, v\gamma, vh) \quad (28)$$

Where γ is the aspect ratio and h the height of the bounding box.
Besides, the squared Mahalanobis distance is used instead of the Hungarian algorithm due to that change.

$$d_{i,j}^{(1)} = (d_j - y_i)^T S_i^{-1} (d_j - y_i) \quad (29)$$

Here, S_i is de covariance matrix and d_j is the bounding box detection.
The problem with the Euclidean distance to perform the data association, is that it does not work well when covariances are involve. Therefore, Manhalanobis distance is a better choice for rescaling the problem, so that these covariances can vanish. In this case, it is used a threshold of 95% the Manhalanobis distance instead of the IOU minimum.

2.5.6. Feature Descriptor.

However, as used to happen in the original SORT, the Manhalanobis distance is not suitable enough under high uncertainties, since the Kalman filter estimates are not so accurate. This is where the appearance descriptor vector takes part in.
Fig. 32 shows the architecture of the obtention of the feature descriptor.

Name	Patch Size/Stride	Output Size
Conv 1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv 2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
Batch and ℓ_2 normalization		128

Fig. 32. CNN architecture for the feature descriptor [30]

The feature descriptor is obtained throughout the above architecture, for which the nearest neighbor queries are used to process the relation between the measurement and the existing track, using the smallest cosine distance as the metric and a binary variable for the unlikely associations.

$$d_{i,j}^{(2)} = \min\left\{1 - r_j^T r_k^{(i)} \,\middle|\, r_k^{(i)} \in R_i\right\} \quad (30)$$

Where r_j is the appearance descriptor for the bounding box and $r_k^{(i)}$ the associate appearance descriptor for the i-th track.
Then, both state and appearance metrics are combined using a weighted sum.

$$c_{i,j} = \lambda d_{i,j}^{(1)} + (1 - \lambda) d_{i,j}^{(2)} \quad (31)$$

Nevertheless, the creation and deletion of identities follows the same logic as the original SORT.
Fig. 33 shows some DeepSORT results.

		MOTA ↑	MOTP ↑	MT ↑	ML ↓	ID ↓	FM ↓	FP ↓	FN ↓	Runtime ↑
KDNT [16]*	BATCH	68.2	79.4	41.0%	19.0%	933	1093	11479	45605	0.7 Hz
LMP_p [17]*	BATCH	**71.0**	**80.2**	**46.9%**	21.9%	434	**587**	7880	**44564**	0.5 Hz
MCMOT_HDM [18]	BATCH	62.4	78.3	31.5%	24.2%	1394	1318	9855	57257	35 Hz
NOMTwSDP16 [19]	BATCH	62.2	79.6	32.5%	31.1%	**406**	642	**5119**	63352	3 Hz
EAMTT [20]	**ONLINE**	52.5	78.8	19.0%	34.9%	910	**1321**	**4407**	81223	12 Hz
POI [16]*	**ONLINE**	**66.1**	79.5	**34.0%**	20.8%	805	3093	5061	**55914**	10 Hz
SORT [12]*	**ONLINE**	59.8	**79.6**	25.4%	22.7%	1423	1835	8698	63245	**60 Hz**
Deep SORT (Ours)*	**ONLINE**	61.4	79.1	32.8%	**18.2%**	**781**	2008	12852	56668	40 Hz

Fig. 33. DeepSORT tracking results [30]

It can be seen a reduction in the lost percentage and number of identity switches with respect the original SORT, and an increase in the tracked percentage while maintaining a real-time runtime. Definitely, a more robust and accurate version than SORT, and still faster than other options. Therefore, the choice for the object tracking.

3. Project Development and Results

The images used on this project are recorded with an Asus Xtion Pro Live and the further post processing is done in Python using Google Colaboratory and Visual Studio Code. Here, there is a brief explanation of these working tools.

3.1. Asus Xtion Pro Live

The Asus Xtion Pro Live [31] is an RGB camera with a depth measurement sensor for distances in between 0.8 m and 3.5 m. It is thought to operate indoor and has a field of view of 58° H, 45° V, 70° D (Horizontal, Vertical, Diagonal). It has a power consumption below 2.5 W and uses OpenNI2 as software.
For this project, the image acquisition was done at a resolution of 640x480 and 30 FPS.

3.2. Programming Language. Python

"Python is an interpreted, object-oriented, high-level programming language" [32] with dynamic typing and programming structure. It is popular because of its portability and simple syntax, which facilitates the readability of the code and the learning process. It has one of the largest programming communities, hence, there are a lot of different libraries available for almost every scenario. It has multiple applications in artificial intelligence and computer vision problems.

3.3. Programming environments

Although the aim of this project is to develop a collision avoidance algorithm, first, an object detection and tracking must be done. For that reason, Google Colaboratory is used. However, the rest of the project has been developed using Visual Studio Code instead.
Google Colaboratory is a free Jupyter Notebook based environment for cloud programming released by Google, and enabled for everybody with an internet connection without the need of installing anything. It is given the possibility of using a random GPU or TPU from their servers for deep learning applications. Therefore, it is perfect for the object detection and tracking tasks of the individuals appeared in this project.

3.3.1. Visual Studio Code

Visual Studio Code, on the other hand, is a free code editor from Microsoft. It supports almost any programming language and allows the customization of its interface throughout an extension store. It also includes git connection, a debugging window, and a command promnt. Hence, one of the best choices for app and web developing.

Due to the fact that Google Colaboratory is run on the cloud, it has some limitations with respect other code editors, like for example some OpenCV functions. Thus, it is justified the use of Visual Studio Code during the rest of the project, i.e. camera calibration, image acquisition, error measurement and collision avoidance scripts.

3.4. Proposed Solution

In this section, the proposed solution for the collision avoidance between humans and vehicles will be explained. That said, during the introduction, it was mentioned that this solution was focused on human-human interactions. However, later it will be explained how it could be applied for human-vehicle situations.

First of all, a calibration must be done to get the intrinsic parameters of the camera. It will be done running a calibration script which returns a YAML file with the OpenCV calculations. Then, these parameters will be used to correct the distortion of the image, and calculate the error on the ground projection between the original and the undistort images.

After the camera calibration and error calculations for the ground projections, a set of frames will be recorded as inputs for the object detection and tracking scripts. The model is taken from a public github repository from Mikel Brostrom, it will be later explained in more detail.

Finally, the standing points of the individuals detected corresponding to the tracked bounding boxes will be mapped from the pixel coordinate system to the ground plane. This ground projections, will be used to set the orientation of a safety area according to the trajectory of the detection. Therefore, the collision alerts will be given depending on these safety areas and their trajectories.

3.4.1. Camera Calibration

In computer vision applications, it is important to perform a camera calibration from the operating point of the camera, since the image might have a positive or negative radial distortion, which is related to the field of view of the camera lens, see Fig. 34.

Fig. 34. Type of distortions [33]

This calibration is used to get the intrinsic parameters of the camera, as well as the distortion parameters, and rotation and translation vectors, so that any possible distortion on the image could be corrected beforehand in order to perform accurate measurements or any needed calculation.

This is possible due to the fact that the camera performs a 2D representation of the 3D world, which in fact, is directly related to their intrinsic parameters like the focal length and the location of its optical center. The pinhole camera model (see Fig. 35) mathematically describes this relationship as:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (32)$$

Where (f_x, f_y) are the focal lengths in pixels, (c_x, c_y) are the coordinates of principal point, and the r_{mn} parameters are part of the rotation-translation matrix, which is related to the scene.

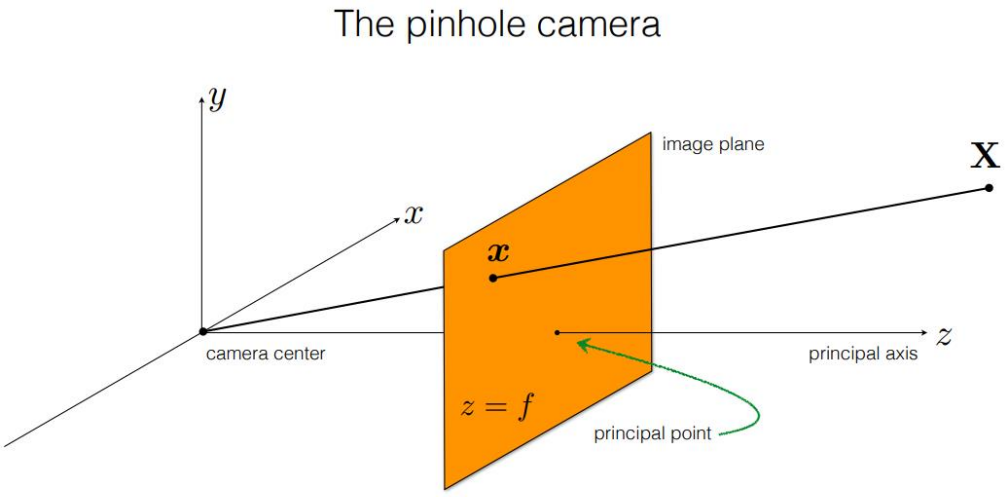


Fig. 35. Pinhole camera model.

3.4.2. Chessboard Method

OpenCV has some functions to calculate these intrinsic and extrinsic parameters throughout the corners of a chessboard and their relationship within a set of images at different positions, see Fig. 36.

Fig. 36. Two examples of the calibration images with the chessboard at different positions

Table 5 shows to get the images for the calibration process.

Table 5. Loop to get the calibration images.

```
# initialize the camera
openni2.initialize("C:\Program Files\OpenNI2\Redist")
dev = openni2.Device.open_any()

color_stream = dev.create_color_stream()
color_stream.set_video_mode(c_api.OniVideoMode(pixelFormat =
c_api.OniPixelFormat.ONI_PIXEL_FORMAT_RGB888, resolutionX = int(640),
                                                resolutionY = int(480), fps = 30))
color_stream.start()

i=0
start = time.time()
while True:

 cframe = color_stream.read_frame()
 cframe_data = cframe.get_buffer_as_uint8()
 cimg = np.ndarray((cframe.height, cframe.width, 3), dtype=np.uint8,
                       buffer=cframe_data)
 cimg = cv.cvtColor(cimg, cv.COLOR_BGR2RGB)

 end = time.time()

 #stores the frame every 5 seconds
 if int(round(end-start,2)) == 5:
  cv.imwrite("calibration_images/"+str("{}".format(i)+"chessboard") +
              ".png",cimg)
  i+=1
  start = end
 cv.imshow("color", cimg)

 #breaks the loop after 10 images
```

```
31. if len(glob.glob('calibration_images/*.png')) == 10:
32.  break
33.
34. if cv.waitKey(10) & 0xFF == ord('q'):
35.  break
36.color_stream.stop()
37.openni2.unload()
38.cv.destroyAllWindows()
```

For this process, ten chessboard images were got in loop, storing a frame every five seconds so that the chessboard could be situated at different positions. To measure the time, the function time is used. In line 23, is set the statement for which, if the difference between the end variable, line 20, and start variable, line 11, is equal to five, it would mean that five seconds were spent.

After that, some preparation for the calibration process is needed. Table 6 shows the code for this preparation.

Table 6. Chessboard preparation.

```
1. chessboardSize = (9,7)
2. frameSize = (640,480)
3.
4. # prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
5. objp = np.zeros((chessboardSize[0] * chessboardSize[1], 3), np.float32)
6. objp[:,:2] = np.mgrid[0:chessboardSize[0],
7.                       0:chessboardSize[1]].T.reshape(-1,2)
8.
9. size_of_chessboard_squares_mm = 10
10.objp = objp * size_of_chessboard_squares_mm
11.# Arrays to store object points and image points from all the images.
12.objpoints = [] # 3d point in real world space
13.imgpoints = [] # 2d points in image plane.
```

The window size and the number of x and y-axis corners of the chessboard should be defined. These corners are also used to prepare an array with the location in millimeters of each one of them. Moreover, two empty lists are defined to store the image and real world corners in case they are found, see lines 12 and 13. Now, in Table 7 is presented the code for the calibration process.

Table 7. Camera calibration.

```
1. # termination criteria
2. criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
```

```
3. images = glob.glob('calibration_images/*.png')
4.
5. for i, image in enumerate(images):
6.  img = cv2.imread(image)
7.  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
8.
9.   # Find the chess board corners
10.  ret, corners = cv2.findChessboardCorners(gray, chessboardSize, None)
11.
12.  # If found, add object points, image points (after refining them)
13.  if ret == True:
14.
15.   objpoints.append(objp)
16.   corners2 = cv2.cornerSubPix(gray, corners, (11,11), (-1,-1), criteria)
17.   imgpoints.append(corners)
18.   # Draw and display the corners
19.   cv2.drawChessboardCorners(img, chessboardSize, corners2, ret)
20.
21.  cv2.imwrite("calibration_images/calibrated/"+str("{}".format(i)+"calib")
22.                 + ".png",img)
23.cv2.destroyAllWindows()
24.# CALIBRATION
25. ret, cameraMatrix, Distortion_Params, Rotation_vec, Translation_vec =
   cv2.calibrateCamera(objpoints, imgpoints, frameSize, None, None)
```

The calibration process works as follows. After the chessboard is prepared, a rough estimation of the corners in the calibration image is done, line 10. If this estimation is done correctly, the corners locations are refined with another function in line 16. Then, both 3D and 2D lists of chessboard corners are stored for each calibration image. These two lists as well as the frame size, are used as inputs for the camera calibration function in line 25.
Fig. 37 shows the calibration points.

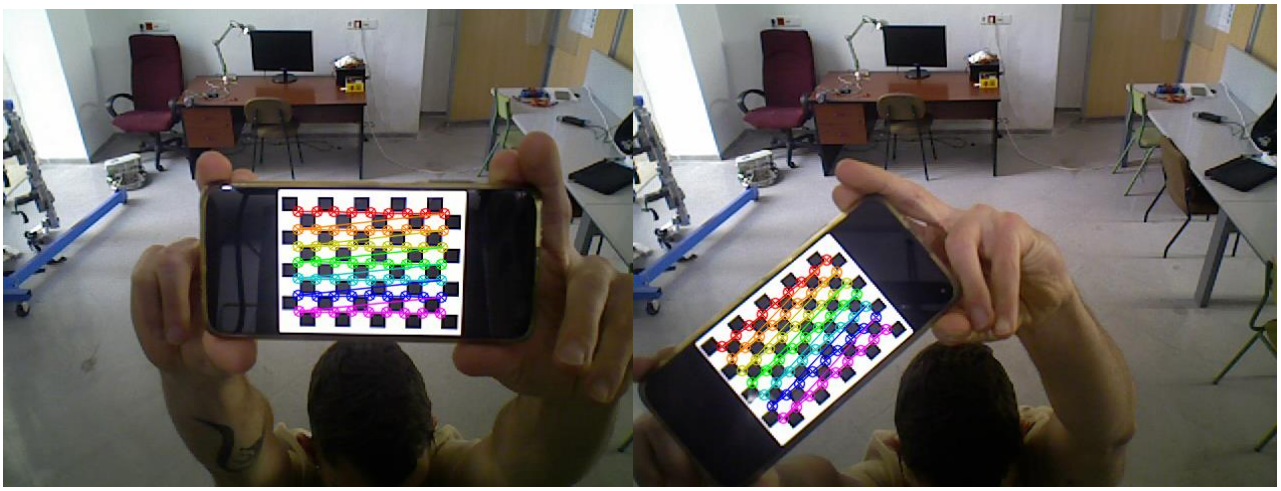


Fig. 37. Corners found within the calibration images.

The functions used to get the chessboard corners were:

cv2.findChessboardCorners(image, patternSize[, corners[, flags]]) [34]**:** It finds the positions of internal corners of the chessboard.

- Image: source chessboard view. It must be an 8-bit grayscale or color image.
- patternSize: number of inner corners per a chessboard row and column.
- corners: output array of detected corners.
- flags: various operation flags.

cv2.cornerSubPix(image, corners, winSize, zeroZone, criteria) [35]**:** It refines the corner locations.

- image: input single-channel, 8-bit or float image.
- corners: initial coordinates of the input corners.
- winSize: half of the side length of the search window.
- zeroZone: half of the size of the dead region in the middle of the search zone over which the summation in the formula below is not done.
- criteria: criteria for termination of the iterative process of corner refinement.

Finally, the optimal camera matrix is obtained and and everything is stored in a YAML file. Table 8 shows the code for this process.

Table 8. Optimal parameters.

```
# GETTING OPTIMAL PARAMETERS
h, w = img.shape[:2]
newCameraMatrix, roi = cv2.getOptimalNewCameraMatrix(cameraMatrix,
                                          Distortion_Params, (w,h), 1, (w,h))

print('Saving parameters !')
cv_file = cv2.FileStorage('cameraIntrnsic.yaml', cv2.FILE_STORAGE_WRITE)
cv_file.write('Camera_Matrix', cameraMatrix)
cv_file.write('newCamera_Matrix', newCameraMatrix)
cv_file.write('Roi',roi)
cv_file.write('Distortion_Params', Distortion_Params)
cv_file.write('Rotation_Vec', np.array(Rotation_vec))
cv_file.write('Translation_Vec', np.array(Translation_vec))
```

The functions used in the calibration process were:

cv2.calibrateCamera(objectPoints, imagePoints, imageSize, cameraMatrix, distCoeffs[, rvecs[, tvecs[, flags[, criteria]]]]) [36]**:** It finds the camera intrinsic and extrinsic parameters from several views of a calibration pattern.

- objectPoints: it is a vector of vectors of calibration pattern points in the calibration pattern coordiante space.
- imagePoints: it is a vector of vectors of the projections of calibrating pattern points.
- imageSize: size of the image used to initialize the camera intrinsic matrix.
- cameraMatrix: input/output 3x3 floating-point camera intrinsic matrix
- distCoeffs: input/output vector of distortion coefficients.
- rvecs: output vector of ration vectors.
- tvecs: output vector of translation vectors.
- flags: different flags.
- criteria: termination criteria for the iterative optimization algorithm.

cv2.getOptimalNewCameraMatrix(cameraMatrix, distCoeffs, imageSize, alpha[, newImgSize[, centerPrincipalPoint]]) [37]**:** It returns the new camera intrinsic matrix based on the free scaling parameter.

- cameraMatrix: input camera intrinsic matrix.
- distCoeffs: input vector of distortion coefficients.
- imageSize: original image size.
- alpha: free scaling parameter between 0 and 1.
- newImgSize: image size after rectification.
- centerPrincipalPoint: optional flag that indicates whether in the new camera intrinsic matrix the principal point should be at the image center or not.

In this case, I got a mean error during the calibration of 0.09005, see Table 9

Calibration Error
0.09005

Table 9. Camera calibration error.

3.4.3. Correcting distortion in images.

Now, since the camera calibration has been performed, the following task is to use the camera parameters to undistort the image, see Table 10.

Table 10. Image undistortion.

```
1. frame = cv2.imread('error_measurement/test0/1color.png')
```

```
2. frame_copy = frame.copy()
3.
4. # Undistort image
5. dst = cv2.undistort(frame_copy, cameraMatrix, Distortion_Params, None,
6.                     newCameraMatrix)
7.
8. # crop the image
9. x, y, w, h = roi
10.dst = dst[y[0]:y[0]+h[0], x[0]:x[0]+w[0]]
11.dst = cv2.resize(dst,(640,480))
```

In line 5, the image is undistorted, and right after, it is cropped to eliminate the resulting black borders. Both images are shown in Fig. 38 and Fig. 39.

The function which performs the undistortion is:

cv2.undistort(src, cameraMatrix, distCoeffs[, dst[, newCameraMatrix]]) [38]**:** It transforms an image to compensate radial and tangential lens distortion.

- src: input image..
- dst: output corrected image.
- cameraMatrix: input camera Matrix.
- distCoeffs: input vector of distortion coefficients.
- newCameraMatrix: camera matrix of the distorted image.

Fig. 38. Original image.

Fig. 39. Undistort image

3.4.4. Ground projection.

Now, to get the actual location of the individuals, it would be necessary to map the image coodinates to another reference system, there are multiple ways to so.

One of those ways, is via homography calculations, which is commonly used to relate 2D images of the same view at different angles. In this case, the idea is to use the homography matrix to project any falling point within the boundaries of a ground layout in the input image, into an upper view of the same area, so that the actual location could be accurately obtained .

Therefore, the new coordinates could be calculated as follows.

$$\begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} = H_{3x3} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad (33)$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} \cdot \frac{1}{z_a} \quad (34)$$

Where $(x_1, y_1, 1)$ are the coordinates within the boundaries of the original image and H_{3x3} is the homography matrix.

OpenCV has two functions to do so:

cv2.findHomography(srcPoints, dstPoints[, method[, ransacReprojThreshold[, mask[, maxIters[, confidence]]]]]) [39]**:** It finds a perspective transformation between two planes.

- srcPoints: coordinates of the points in the original plane
- dstPoints: coordinates of the points in the target plane
- method: method used to compute a homography matrix
- ransacReprojThreshold: maximum allowed re-projection error to treat a point pair as an inlier
- mask: optional output mask set by a robust method
- maxIters: the maximum number of RANSAC iterations
- confidence: confidence level

cv2.warpPerspective(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) [40]**:** It transforms the source image using the specified matrix

- src: input image
- M: 3x3 transformation matrix
- dsize: size of the output image
- dst: output image
- flags: combination of interpolation methods (INTER_LINEAR or INTER_NEAREST) and the optional flag WARP_INVERSE_MAP, that sets M as the inverse transformation (dst→src)
- borderMode: pixel extrapolation method.
- borderValue value used in case of a constant border.

3.4.5. Homography matrix functions.

Here, two homography functions are defined. The first, is set for the original image, whereas the second is for the undistort one. See Table 11.

Table 11. Homography calculation functions.

```
1. width,height = (640,480)
2. n,m = (246,320)
3.
4. def homography()
5.  #corners of the ground layout
6.  corners = np.array([[178,105], [465,105], [0,height], [width, height]])
7.  #corners of the new perspective
8.  cornersN = np.array([[0,0],[n,0],[0,m],[n,m]])
9.  corners=np.int64(corners)
10. cornersN=np.int64(cornersN)
```

```
11.
12. h,_=cv2.findHomography(corners,cornersN)
13.
14. return h, corners
15.
16.def homography_dist():
17.
18. corners = np.array([[161,95], [461,95], [0,height], [width, height]])
19. cornersN = np.array([[0,0],[n,0],[0,m],[n,m]])
20. corners=np.int64(corners)
21. cornersN=np.int64(cornersN)
22.
23. h,_=cv2.findHomography(corners,cornersN)
24.
25. return h, corners
```

In relation to the homography calculations, the real plane has a width and height measures of 246x320 cm, these measurements are used to define the window size of the new perspective, which means that the scale is 1:1 for the upper view

3.4.6. Homography error calculations.

The next step, is to measure the error in the location estimation after finding the homography between the upper view of the real plane and the original image taken from the camera point of view. To do so, both original and undistort images are loaded and reframed to the new perspective. Then, a set of 20 points with respect the measurement of the slabs in the real plane is drawn, and the deviation within the x and y-axis is measured.

Table 12 corresponds to this process.

Table 12. Homography error measurement.

```
1. #load test images
2. frame = cv2.imread('error_measurement/test0/1color.png') #distort
3. dst = cv2.imread('error_measurement/test0/0color.png')   #undistort
4. wrap_dist = frame.copy()
5. wrap_undist = dst.copy()
6.
7. #show ground corners
8. h, layout = homography()
9. cv2.circle(frame, layout[0], 5, (0,0,255), 2)
10.cv2.circle(frame, layout[1], 5, (0,255,0), 2)
11.cv2.circle(frame, layout[2], 5, (255,0,255), 2)
12.cv2.circle(frame, layout[3], 5, (255,0,0), 2)
13.
```

```
14. h1, layout2 = homography_dist()
15. cv2.circle(dst, layout2[0], 5, (0,0,255), 2)
16. cv2.circle(dst, layout2[1], 5, (0,255,0), 2)
17. cv2.circle(dst, layout2[2], 5, (255,0,255), 2)
18. cv2.circle(dst, layout2[3], 5, (255,0,0), 2)
19.
20.#ground perspective
21. ground=cv2.warpPerspective(wrap_dist,h,(n,m),flags=cv2.INTER_LINEAR)
22. ground_undist=cv2.warpPerspective(wrap_undist,h1,(n,m),
23.                                   flags=cv2.INTER_LINEAR)
24.#set 20 ground points in the intersection of the slabs
25. ground_grid(ground)
26. ground_grid(ground_undist)
```

In lines 8 and 14, the homography matrixes are obtained and used to reframe the view in lines 21 and 22. Then, the ground points for the error measurement are set in lines 25 and 26, This is done throughout the following function, see Table 13.

Table 13. Ground grid function

```
1. def ground_grid(image):
2.  pt1 = (45, 13) #offset to the first corner
3.  slab  = 56
4.
5.  for row in range(0,4):
6.   for col in range(0,5):
7.    ptx = (pt1[0]+slab *row, pt1[1]+slab *col)
8.    cv2.circle(image,(int(ptx[0]),int(ptx[1])),2,(0,255,0),2)
```

Every slab is a 56 cm square, and the first corner is situated at a distance of 45 cm in the x-axis and 13 cm in the y-axis with respect the origin point, which is the top left corner of the original image. The following Fig. 40 and Fig. 41 correspond to the original image and its ground projection, while Fig. 42 and Fig. 43 correspond to the undistort ones.

Fig. 40. Original image ground corners.

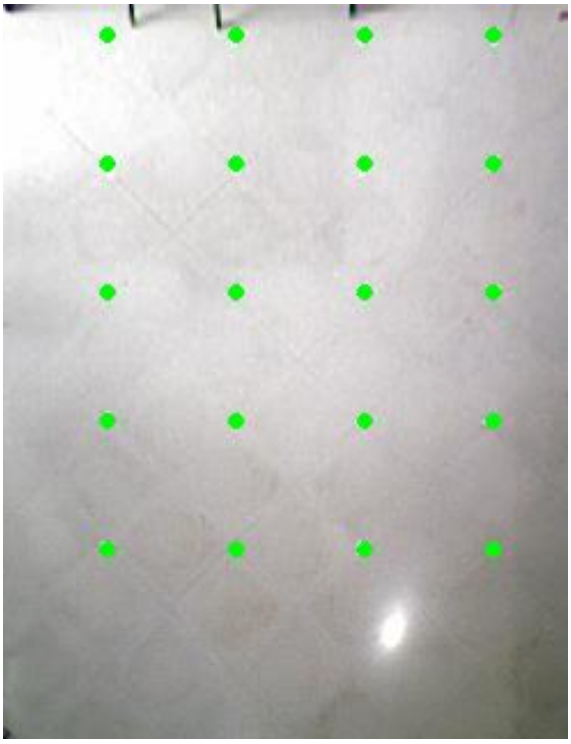


Fig. 41. Upper view of the ground after using the warpPerspective function in the original image

Fig. 42. Undistort image ground corners

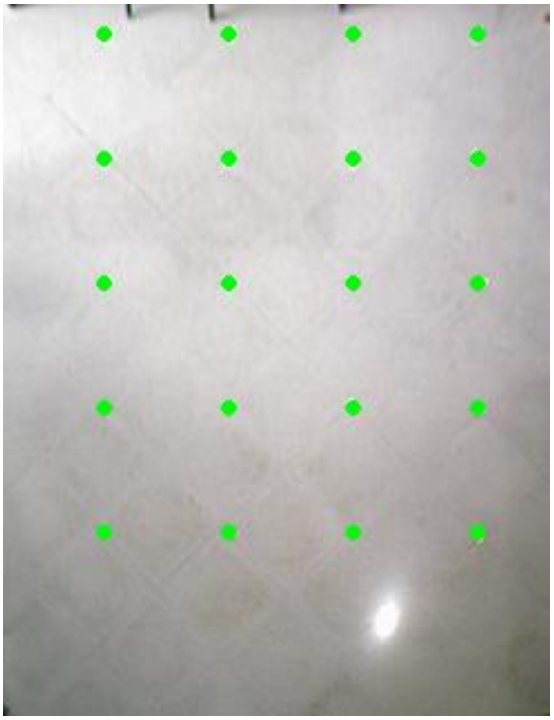


Fig. 43. Upper view of the ground after using the warpPerspective function in the undistort image

For the original image, the following x and y deviations in centimeters were obtained, see Table 14. Likewise, for the undistort image those are shown in Table 15.

Table 14. Original image error

$$X_{dev}^{original} = \begin{bmatrix} -3 & 0 & 0 & 0 \\ -3.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.5 \\ 0 & 0 & 2 & 3 \\ 2 & 0 & 4.5 & 5 \end{bmatrix} ; Y_{dev}^{original} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & 4 & 3.5 \\ 0 & 2.5 & 1.5 & 1.5 \\ 0 & 0 & 2.5 & 3 \\ 0 & 0 & 1 & 1.5 \end{bmatrix}$$

Table 15. Undistort image error

$$X_{dev}^{undistort} = \begin{bmatrix} 0 & -2 & -2 & -2 \\ 0 & 0 & 0 & 0 \\ 2.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 1.5 & 0 \end{bmatrix} ; Y_{dev}^{undistort} = \begin{bmatrix} -1 & -2 & -2 & -2.5 \\ -0.5 & 0 & 0 & 0 \\ -1.5 & -2.5 & -1 & -2 \\ 0 & 0 & 0 & -1.5 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Table 16 shows the error of the ground projections.

Image	**Mean Absolute Error X axis**	**Mean Absolute Error Y axis**
Undistort	0.6	0.875
Original	1.275	1.25

Table 16. Mean absolute error of the ground projection

Since the original image had nearly zero distortion, the mean absolute error in both cases is really small. There are various points with no deviation, while the rest are deviated less than 5 centimeters in one or both axis. However, the maximum deviation reached in the original image is slightly higher. In any case, it is a good accuracy for the location estimation.

3.4.7. Object Detection and Tracking

The object detection is done with YOLOv5, whose model structure is the same as YOLOv4, but entirely built in Pytorch and using a focus CSPDarknet53 as backbone, see Table 17.

Table 17. YOLOv5m model summary

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Model                                    --                        --
├─Sequential: 1                          --                        --
│    └─Detect: 2                         --                        --
│    │    └─ModuleList: 3-1              --                        343,485
│    └─Conv: 2-1                         [2, 48, 160, 160]         --
│    │    └─Conv2d: 3-2                  [2, 48, 160, 160]         5,184
│    │    └─BatchNorm2d: 3-3             [2, 48, 160, 160]         96
│    │    └─SiLU: 3-4                    [2, 48, 160, 160]         --
│    └─Conv: 2-2                         [2, 96, 80, 80]           --
│    │    └─Conv2d: 3-5                  [2, 96, 80, 80]           41,472
│    │    └─BatchNorm2d: 3-6             [2, 96, 80, 80]           192
│    │    └─SiLU: 3-7                    [2, 96, 80, 80]           --

                                              (...)                                .

│    └─Conv: 2-22                        [2, 384, 10, 10]          --
│    │    └─Conv2d: 3-59                 [2, 384, 10, 10]          1,327,104
│    │    └─BatchNorm2d: 3-60            [2, 384, 10, 10]          768
│    │    └─SiLU: 3-61                   [2, 384, 10, 10]          --
│    └─Concat: 2-23                      [2, 768, 10, 10]          --
│    └─C3: 2-24                          [2, 768, 10, 10]          --
│    │    └─Conv: 3-62                   [2, 384, 10, 10]          295,680
│    │    └─Sequential: 3-63             [2, 384, 10, 10]          2,952,192
│    │    └─Conv: 3-64                   [2, 384, 10, 10]          295,680
│    │    └─Conv: 3-65                   [2, 768, 10, 10]          591,360
│    └─Detect: 2-25                      [2, 6300, 85]             --
==========================================================================
Total params: 21,190,557
Trainable params: 21,190,557
Non-trainable params: 0
Total mult-adds (G): 12.22
==========================================================================
Input size (MB): 2.46
Forward/backward pass size (MB): 371.06
Params size (MB): 84.76
Estimated Total Size (MB): 458.28
==========================================================================
```

Fig. 44 shows a comparison of the different versions in YOLOv5.

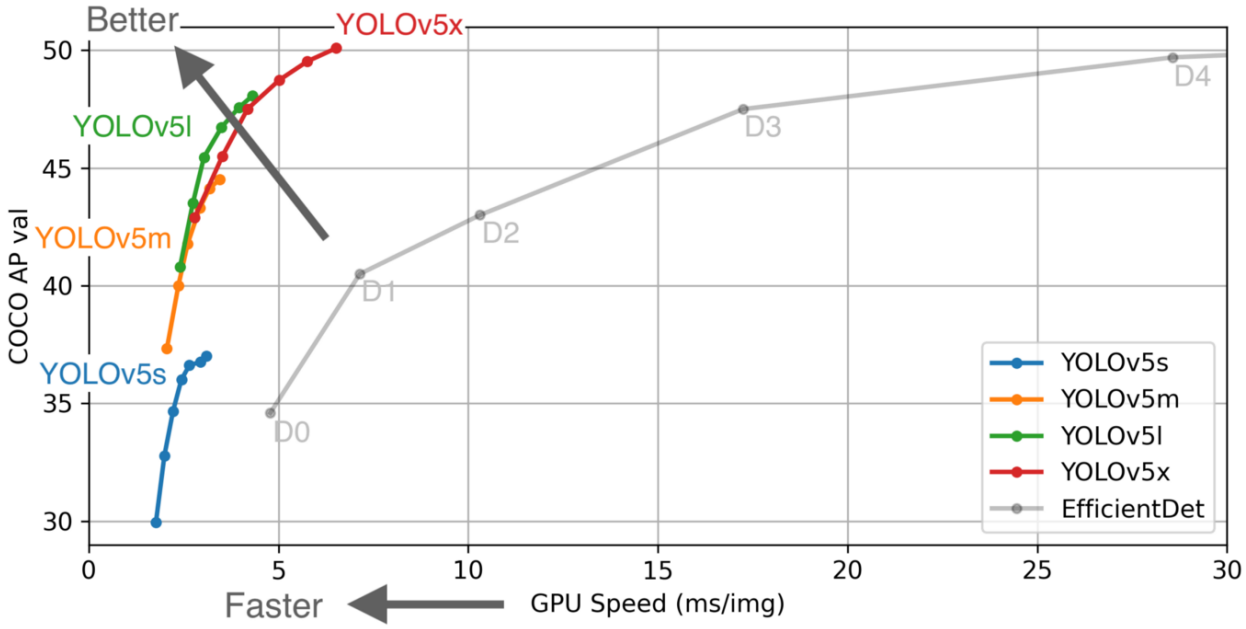


Fig. 44. YOLOv5 variants

More specifically, this project uses the weights from the variant YOLOv5m, which sets a good balance between speed and AP, as can be seen above. Its size is estimated in about 458 MB and has a total 21,190,557 trainable parameters

On the other hand, the object tracking and re-identification is done with DeepSORT in combination with OSNet, see Table 18.

Table 18. OSNet_x0_25 model summary

Layer (type:depth-idx)	Output Shape	Param #
OSNet	--	--
├─ConvLayer: 1-1	[2, 16, 160, 160]	--
│ └─Conv2d: 2-1	[2, 16, 160, 160]	2,352
│ └─BatchNorm2d: 2-2	[2, 16, 160, 160]	32
│ └─ReLU: 2-3	[2, 16, 160, 160]	--
├─MaxPool2d: 1-2	[2, 16, 80, 80]	--
├─Sequential: 1-3	[2, 64, 40, 40]	--
│ └─OSBlock: 2-4	[2, 64, 80, 80]	--
│ │ └─Conv1x1: 3-1	[2, 16, 80, 80]	288
│ │ └─LightConv3x3: 3-2	[2, 16, 80, 80]	432
│ │ └─Sequential: 3-3	[2, 16, 80, 80]	864
│ │ └─Sequential: 3-4	[2, 16, 80, 80]	1,296

```
│    │    └─Sequential: 3-5                  [2, 16, 80, 80]       1,728
│    │    └─ChannelGate: 3-6                 [2, 16, 80, 80]       49
│    │    └─ChannelGate: 3-7                 [2, 16, 80, 80]       (recursive)
│    │    └─ChannelGate: 3-8                 [2, 16, 80, 80]       (recursive)
│    │    └─ChannelGate: 3-9                 [2, 16, 80, 80]       (recursive)
│    │    └─Conv1x1Linear: 3-10              [2, 64, 80, 80]       1,152
│    │    └─Conv1x1Linear: 3-11              [2, 64, 80, 80]       1,152
│    └─OSBlock: 2-5                          [2, 64, 80, 80]       --
│    │    └─Conv1x1: 3-12                    [2, 16, 80, 80]       1,056
│    │    └─LightConv3x3: 3-13               [2, 16, 80, 80]       432
│    │    └─Sequential: 3-14                 [2, 16, 80, 80]       864
│    │    └─Sequential: 3-15                 [2, 16, 80, 80]       1,296
│    │    └─Sequential: 3-16                 [2, 16, 80, 80]       1,728
│    │    └─ChannelGate: 3-17                [2, 16, 80, 80]       49
│    │    └─ChannelGate: 3-18                [2, 16, 80, 80]       (recursive)
│    │    └─ChannelGate: 3-19                [2, 16, 80, 80]       (recursive)
│    │    └─ChannelGate: 3-20                [2, 16, 80, 80]       (recursive)
│    │    └─Conv1x1Linear: 3-21              [2, 64, 80, 80]       1,152

                                             (...)

├─Conv1x1: 1-6                               [2, 128, 20, 20]      --
│    └─Conv2d: 2-12                          [2, 128, 20, 20]      16,384
│    └─BatchNorm2d: 2-13                     [2, 128, 20, 20]      256
│    └─ReLU: 2-14                            [2, 128, 20, 20]      --
├─AdaptiveAvgPool2d: 1-7                     [2, 128, 1, 1]        --
├─Sequential: 1-8                            [2, 512]              --
│    └─Linear: 2-15                          [2, 512]              66,048
│    └─BatchNorm1d: 2-16                     [2, 512]              1,024
│    └─ReLU: 2-17                            [2, 512]              --
==========================================================================
Total params: 201,864
Trainable params: 201,864
Non-trainable params: 0
Total mult-adds (M): 514.20
==========================================================================
Input size (MB): 2.46
Forward/backward pass size (MB): 249.06
Params size (MB): 0.81
Estimated Total Size (MB): 252.32
==========================================================================
```

The OSNet model chosen for the re-identification task is its tiniest version, OSNet_x0_25. In the above table, it can be seen that it has a size of 252 MB and a total of 201.864 trainable parameters. Both detection and tracking models, are already trained in humans with the COCO VOC Dataset.

3.4.8. Google Colaboratory implementation

As mentioned previously, this object detector and tracker is taken from a public repository from Mikel Brostrom, so the first step is to clone the repository and install the dependencies. The following code from Table 19, corresponds to the first cell from Google Colaboratory.

Table 19. Cloning repository.

```
!git clone –recurse-
submodules https://github.com/mikelbrostrom/Yolov5_DeepSort_OSNet.git
# clone repo
%cd Yolov5_DeepSort_OSNet
%pip install -qr requirements.txt  # install dependencies
```

Now, to get the actual position of the detections, it is made the assumption that the middle point of the width and the highest value of the height, corresponds to the standing point of the human in the image, which is the closer location to the feets. Table 20 shows the calculation of this standing point.

Table 20. Standing point calculation.

```
# draw boxes for visualization
if len(outputs[i]) > 0:
 for j, (output) in enumerate(outputs[i]):

  bboxes = output[0:4]
  id = output[4]
  cls = output[5]
  conf = output[6]

  if save_txt:
  # to MOT format
   bbox_left = output[0]
   bbox_top = output[1]
   bbox_w = output[2] - output[0]
   bbox_h = output[3] - output[1]
   bbox_groundx = bbox_w/2 + output[0]
   bbox_groundy = output[3]
   # Write MOT compliant results to file
   with open(txt_path + '.txt', 'a') as f:
    f.write(('%g ' * 10 + '\n') % (frame_idx + 1, id,
              bbox_groundx,bbox_groundy, bbox_w, bbox_h))
```

In the following table, it can be seen a modified fragment from the track.py file from Yolov5_DeepSort_OSNet folder obtained after cloning the repository. Here, the variables from lines 12 and 13 were stored in line 21, line 206 of the original code. However, since the needed values are the ones corresponding to the standing point of the object detected, the variables calcualted in lines 16 and 17 are saved instead.

To run the model, it can be done throughout the track.py file, see Table 21.

Table 21. Running command.

```
1. !python track.py  --yolo_model yolov5m.pt
2.                   --source '/content/drive/MyDrive/TFG/prueba0/*'
3.                   --save-vid
4.                   --save-txt
5.                   --classes 0
```

By default, the weights from YOLOv5s and OSNet_x0_25 are set. Therefore, the YOLO model is changed to YOLOv5m, whereas the DeepSORT model is let by default. Likewise, there are many other default parameters that can be change by typing some commands in the running line. In this case, the detections are limited to the class 0 corresponding to humans, and the commands --save-vid and --save-txt are specified so that the video and text record could be obtained. In relation to the text record, the information is stored in rows per detection, as seen in table 18. The first two positions of each row, corresponding to the frame number and tracking id of the object detected, and the next four, to the standing x and y points, and the width and height lengths of the bounding box respectively.
To end the section, it is important to mention another set of parameters corresponding to the DeepSORT configuration file, see Table 22.

Table 22. DeepSORT configuration file.

```
1. DEEPSORT:
2. MODEL_TYPE: "osnet_x_25"
3. MAX_DIST: 0.2
4. MAX_IOU_DISTANCE: 0.7
5. MAX_AGE: 30
6. N_INIT: 3
7. NN_BUDGET: 100
```

Here, in lines 3, 4 and 5, it is set a maximum distance for the ID matching of 0.2, an IOU threshold of 0.7 and 30 frames of ID storing after losing the detection. Furthermore, the minimum number of matching frames for considering a valid ID is set to 3 and the maximum size of the appearance descriptor is fixed to 100.

3.4.9. Tracking preparation

Finally, the last step is to use these results to set the orientation of the safety zones for the collision alerting.

To do so, the tracking file is loaded, split and saved as a numpy array. Then, this array is split into another two, so that the frame numbers and detection information could be separated, see the following code from Table 23.

Table 23. Detection array preparation 1.

```
1. #import object detection info
2. f = open('detections/test3/tracks/prueba3.txt').read()
3. detections = f.split('\n')[:-1]
4.
5. record = np.zeros((len(detections),6))
6. for i, rec in enumerate(detections):
7.  rec = np.array(rec.split())
8.  record[i:] = rec
9.
10.#prepare the number tracks per frames
11.frames = np.array([np.array(list(j)).astype(int) for i, j in
12.groupby(record[:,0])], dtype=object)
13.max_detects  = max([len(frame) for frame in frames])
14.
15.ground_points = record[:,1:].astype(int)
16.tracks = np.zeros((3,frames.shape[0], max_detects), dtype = object)
```

In line 3, the tracking record is split per row, but it is also needed to split the columns, which are 6. This is done in the for loop from line 6. Then, once this process has finished, the frames, whose number is the same, are clustered together. The objective here, is to separate the detections that were done at each frame. After that, the maximum number of detections made at certain point and the number of frames, are used to define a 3-Dimentional array. This array, containing the ground location of all the human detections at each frame and their corresponding ID.
Table 24 shows the last part of the tracking array preparation.

Table 24. Detection array preparation 2.

```
1. row = 0
2. for n1, frame in enumerate(frames):
3.  ids = list(ground_points[row:row+len(frame),0])
4.  tx = list(ground_points[row:row+len(frame),1])
5.  ty = list(ground_points[row:row+len(frame),2])
6.
7.  tracks[0,n1,:len(frame)] = tx
8.  tracks[1,n1,:len(frame)] = ty
9.  tracks[2,n1,:len(frame)] = ids
```

```
10.
11. if len(frame) < max_detects:
12.  tracks[0,n1,len(frame):] = np.nan
13.  tracks[1,n1,len(frame):] = np.nan
14.  tracks[2,n1,len(frame):] = np.nan
15.  row += len(frame)
```

Depending on the length of each frame, there will be more or less detections. This information is used to separate the IDs, as well as the ground locations, as it can be seen in lines 3,4 and 5 from the above code. However, due to the fact that number of columns is fixed to a maximum value, in case of less number of detections, the last position is always set to a None value, see line 11.

3.4.10. Local plane coordinates calculation.

Now, in Table 25 the real coordinates are obtained from the ground projections.

Table 25. Real coordinates calculation in the main code.

```
1. images = glob.glob('tests/prueba3/*.png')
2. images = natsort.natsorted(images)[6:]
3. for i, image in enumerate(images):
4.
5.  if i <= tracks.shape[1]-1:
6.   img = cv2.imread(image)
7.   ground = np.zeros_like(img)
8.   ground=cv2.warpPerspective(ground, h,(n,m), flags=cv2.INTER_LINEAR)
9.
10.  for person in range(0,len(frames[i])):
11.   if np.isnan(tracks[0,i,person]) != True:
12.
13.    x = tracks[0,i,person]
14.    y = tracks[1,i,person]
15.    id = tracks[2,i,person]
16.
17.    xy_pixels = np.array([x,y],dtype=int)
18.
19.    xyreal = real_coords(xy_pixels,h)
20.    xyreal = xyreal.reshape(1,2).astype(int)[0]
21.
22.    cv2.circle(img, xy_pixels, 2, (0,0,255),2)
```

After the tracking array preparation, the ground location of each detection is got, see lines 10 to 15. This location is converted to the ground projection throughout the homography matrix, as mentioned in section 3.4.4. For that reason, the function in line 19 is defined as follows in Table 26.

Table 26. Real coordinates function definition

```
1. def real_coords(pxl_point,h):
2.  coords = np.array([[pxl_point[0]], [pxl_point[1]], [1]])
3.
4.  real_coords = np.dot(h, coords)
5.  real_coords = real_coords/real_coords[2]
6.  return real_coords[:2]
```

Notice that, as it was shown in formula 34, to get the new positions the coordinates are divided by the z value. Now, these real coordinates can be used to get direction of the human detected.

3.4.11. Safety areas calculation

Table 27 correspond to the obtention of the security areas.

Table 27. Safety area calculation in the main code

```
1.     if id not in tracks_dict.keys():
2.      tracks_dict[id] = xyreal
3.
4.     else:
5.      t1 = tracks_dict[id]
6.      t2 = xyreal
7.      tracks_dict[id] = xyreal
8.
9.      xy_edges,xy_half_edges = limits_safety_zone(t2, t1, ground)
10.     tracks_edges[id] = xy_edges, xy_half_edges
11.     cv2.circle(ground, xyreal, 2, (0,0,255),2)
```

First of all, two locations are needed, for that reason, an if-else conditional is set to check whether or not the algorithm has seen the current identity of the loop. In case of a new identity, its first location is stored, otherwise, both previous and current positions are named as t1 and t2, see lines 5 and 6. Then, a circular sector of 50 degrees is got in the same direction of the movement. This sector is obtained throughout the function defined in line 9, see Table 28.

Table 28. Safety areas function definition.

```
1. def limits_safety_zone(xyreal_t2,xyreal_t1, ground):
2.  xydiag = xyreal_t2 - xyreal_t1
3.
4.  diag = np.sqrt(xydiag[0]**2 + xydiag[1]**2)
5.  x = xydiag[0]
6.  y = xydiag[1]
7.
8.  if diag==0:
```

```
9.    return [],[]
10.
11. theta = math.acos(x/diag)
12. theta = theta*180/math.pi
13.
14. if (x >= 0 and y >= 0) or (x <= 0 and y >= 0): #1 quadrant(4), 2 quadrant(3)
15.  start2 = 335-theta
16.  end2 = start2 + 50
17.
18. else: #3 quadrant(2), #4 quadrant(1)
19.  start2 = theta - 25
20.  end2 = start2 + 50
21.
22. xy_edges = np.zeros((int(int(end2)-int(start2-1)), 2))
23. xy_half_edges = np.zeros_like(xy_edges)
24.
25. for i,angle in enumerate(range(int(start2),int(end2))):
26.  rads = angle*(math.pi/180)
27.
28.  # external edge
29.  xedge = xyreal_t2[0] + 100 *np.cos(rads)
30.  yedge = xyreal_t2[1] - 100 *np.sin(rads)
31.
32.  xy=np.array([int(xedge),int(yedge)])
33.  xy_edges[i,:] = xy
34.
35.  # middle edge
36.  xedge = xyreal_t2[0] + 50*np.cos(rads)
37.  yedge = xyreal_t2[1] - 50*np.sin(rads)
38.
39.  xy_half=np.array([int(xedge),int(yedge)])
40.  xy_half_edges[i,:] = xy_half
41.
42.  cv2.circle(ground, xy,2,(255,255,0),2)
43.  cv2.circle(ground, xy_half,2,(0,0,255),2)
44.
45. return xy_edges, xy_half_edges
```

Here, the distance from t1 to t2 is calculated, but if t1 and t2 are equals, then two empty lists are returned, see line 9. Nevertheless, when these two variables are different, the angle of displacement is obtained as the arccosine of the triangle formed between these two points. This angle is eventually used to get the starting and ending points with respect to that orientation, see lines 11 to 20.

Finally, a set points with a separation of one degree each, are situated at two different radius of distance. These two radius corresponding to the limits of the security area, one being the half of the other. In this case, since the ground area is not so big, this radius is set to one meter long. Notice

that in lines 30 and 37, the radius is substracted from the current y position instead of being added. This is due to the fact, that the y-axis of the window grows downward.

3.4.12. System of Alerts.

Finally, the alerting system is defined in Table 29.

Table 29. Alerting system in the main code.

```
1.      ids = tracks[2,i,:]
2.      for id_list in ids:
3.       if id_list in tracks_edges.keys() and id != id_list:
4.        alert = alerts(tracks_edges, alert, id, id_list)
5.        #actual id of the loop, other ids
6.
7.        if alert[2][id] >= 6:
8.         print('ALERT 2: inminent collision for object {}'.format(id) )
9.         cv2.putText(img, 'ALERT', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1,
10.                                          (0, 0, 255), 2, cv2.LINE_AA)
11.       elif alert[1][id] >= 6:
12.        print('ALERT 1: object {}  is on the way to collide
13.                                       alert'.format(id) )
14.        cv2.putText(img, 'ALERT', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1,
15.                                           (0, 255, 255), 2, cv2.LINE_AA)
16.       elif alert[0][id] >= 6:
17.        print('ALERT 0: object {} has someone close, take care'.format(id))
18.        cv2.putText(img, 'ALERT', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1,
19.                                             (0, 255, 0), 2, cv2.LINE_AA)
```

For the alerting system, there are three different alerts depending on the distance between the limits of the security areas. In lines 7, 11, and 15, it is showed that they are named as ALERT 2, ALERT 1 and ALERT 0 respectively, for which the danger increases with the number of the alert.

The alerts are given throughout the alert function in line 4, see Table 30. It takes as inputs a dictionary with the arrays containing the limits of the safety areas of each individual, the previous alert dictionary, the current id, and an id from the list of identities seen in that instance. The idea is to compare the position of the current safety area with the rest of areas from the map, so that the system could know which identity is in danger with which other. Besides, it is set a minimum number of 6 consecutive alerts, to avoid fake warnings. he alert function definition.

Table 30. Alerting system function definition.

```
1. def alerts(tracks_edges, alert, id, id_list):
2.  alrt_ID0 = alert[0]
3.  alrt_ID1 = alert[1]
4.  alrt_ID2 = alert[2]
```

```
5.
6.  id_xyedges, id_xy_halfedges = tracks_edges[id]
7.  id_list_xyedges, id_list_xy_halfedges = tracks_edges[id_list]
8.
9.  if len(id_xyedges) == len(id_list_xyedges) and len(id_xyedges) != 0:
10.
11.   distance_far_edges = id_xyedges - id_list_xyedges
12.   distance_farclose_edges_1 = id_xyedges - id_list_xy_halfedges
13.   distance_farclose_edges_2 = id_xy_halfedges - id_list_xyedges
14.
15.   euclid_far_distance = np.sqrt(distance_far_edges[:,0]**2 +
16.                                 distance_far_edges[:,1]**2)
17.   euclid_farclose_distance_1 = np.sqrt(distance_farclose_edges_1[:,0]**2 +
18.                                        distance_farclose_edges_1[:,1]**2)
19.   euclid_farclose_distance_2 = np.sqrt(distance_farclose_edges_2[:,0]**2 +
20.                                        distance_farclose_edges_2[:,1]**2)
21.
22.   dist_mean_far = np.mean(euclid_far_distance)
23.   dist_mean_farclose_1 = np.mean(euclid_farclose_distance_1)
24.   dist_mean_farclose_2 = np.mean(euclid_farclose_distance_2)
25.
26.   if dist_mean_farclose_2 < 50 or dist_mean_farclose_1 <50 :
27.
28.    #ALERTS 2
29.    if id not in alrt_ID2.keys():
30.     alrt_ID2[id] = 1
31.     alert[2] = alrt_ID2
32.    else:
33.     n = alrt_ID2[id]
34.     n += 1
35.     alrt_ID2[id] += n
36.     alert[2] = alrt_ID2
37.
38.   elif dist_mean_far < 90:
39.
40.   #ALERTS 1
41.    if id not in alrt_ID1.keys():
42.     alrt_ID1[id] = 1
43.     alert[1] = alrt_ID1
44.
45.    else:
46.     n = alrt_ID1[id]
47.     n += 1
48.     alrt_ID1[id] += n
49.     alert[1] = alrt_ID1
50.
51.   elif dist_mean_far < 120:
52.   #ALERTS 0
53.    if id not in alrt_ID0.keys():
54.     alrt_ID0[id] = 1
55.     alert[0] = alrt_ID0
```

```
56.
57.    else:
58.     n = alrt_ID0[id]
59.     n += 1
60.     alrt_ID0[id] += n
61.     alert[0] = alrt_ID0
62.   else:
63.    alrt_ID0[id] = 1
64.    alrt_ID1[id] = 1
65.    alrt_ID2[id] = 1
66.
67.    alert[0] = alrt_ID0
68.    alert[1] = alrt_ID1
69.    alert[2] = alrt_ID2
70.
71. else:
72.   alrt_ID0[id] = 1
73.   alrt_ID1[id] = 1
74.   alrt_ID2[id] = 1
75.   alert[0] = alrt_ID0
76.   alert[1] = alrt_ID1
77.   alert[2] = alrt_ID2
78.
79. return alert
```

Here, the Euclidean distances between the external, and between the external and internal limits of the security areas are calculated. After that, the mean of each set of distances is obtained and the results are used to get a rough estimation of the overall distance, see lines 22, 23 and 24. Considering the distances between the external and internal boundaries as the most important, if any of those values is less than 50 cm, it would mean that the hazard of collision is higher than any other. Likewise, if the distance is not that close, but the external limits are less than 90 cm, then, a medium alert in the scale of danger is given, whereas if the distance is in between 90 and 120, the alert is just a suggestion of velocity moderation, because another human is close.

On the other hand, the count in the number of consecutive alerts is done individually at each level of danger. If the dictionary of a certain alert for the given current id has saved that id previously, its value is incremented, otherwise is set to one. Moreover, the count of the three types of alerts is reset for the given id every time that there is not a hazard, what forces a minimum continuity to avoid fake warnings, see lines 72 to 77. These three dictionaries are saved within another dictionary named as alert, which is the output of the function. Now that this function is clearer, it is better understood the alerting system from Table 30.

3.5. Experimental Investigation.

To evaluate the collision avoidance system, four tests for which the distortion has been corrected have been recorded. In these tests, the trajectories are within the boundaries of the area defined previously, i.e. a rectangle of 246 cm width and 320 cm height. The first two tests, correspond to a

single person walk and there are a total number of 465 and 617 frames, whereas for the remaining two, there are two people at the same time and a total 707 and 419 frames respectively.

The following figures, Fig. 45 and Fig. 46, are examples of the image acquisition process.

Fig. 45. Frames from test 1, on the left, and test 2, on the right.

Fig. 46. Frames from test 3, on the left, and test 4, on the right.

As mentioned previously, the detections are done in Google Colaboratory with the weights from the YOLOv5m model and OSNet_x0_25. Likewise, the DeepSORT configuration file is set as it was showed in table 20.

Some examples of human detections are presented in Fig. 47 and Fig. 48, followed by an extract of the 20 firsts detections made at each test in Tables 31, 32, 33 and 34.

Fig. 47. Detection examples from test 1, on the left, and test 2, on the right.

Fig. 48. Detection examples from test 3, on the left, and test 4, on the right.

Table 31. Test 1. Detections from the 20 firsts frames

1. 3 1 388 475 260 391
2. 4 1 384 474 256 400
3. 5 1 380 476 250 408
4. 6 1 376.5 477 233 415
5. 7 1 374 478 220 424
6. 8 1 371.5 478 205 437
7. 9 1 367.5 479 201 448
8. 10 1 363.5 477 199 451
9. 11 1 360.5 476 197 455
10. 12 1 357 464 200 455
11. 13 1 359 451 194 449
12. 14 1 361 448 194 448
13. 15 1 363 446 192 446
14. 16 1 368.5 445 191 445
15. 17 1 375 445 190 445
16. 18 1 385.5 444 189 444
17. 19 1 394.5 440 187 440
18. 20 1 401.5 429 185 429

Table 32. Test 2. Detections from the 20 first frames

1. 1 281.5 478 349 207
2. 4 1 290 478 336 221
3. 5 1 300 478 336 234
4. 6 1 307 478 334 243
5. 7 1 313 479 332 251
6. 8 1 320.5 478 325 260
7. 9 1 330 478 320 268
8. 10 1 340.5 478 317 276
9. 11 1 355 479 316 282
10. 12 1 365.5 479 313 284
11. 13 1 382 479 314 284
12. 14 1 397 479 310 286
13. 15 1 404 479 310 288
14. 16 1 406.5 479 309 289
15. 17 1 409.5 479 299 294
16. 18 1 409 479 298 300
17. 19 1 406 479 290 310
18. 20 1 398.5 479 245 343

Table 33. Test 3. Detections from the 20 first frames.

1. 3 1 167.5 478 311 302
2. 3 2 587.5 356 103 355
3. 4 1 166.5 478 311 306
4. 4 2 583 351 112 350
5. 5 1 166 479 312 308
6. 5 2 578.5 344 121 343
7. 6 1 166 479 308 307
8. 6 2 575.5 340 127 340
9. 7 1 164.5 479 301 307
10. 7 2 572.5 338 133 338
11. 8 1 164.5 479 293 306
12. 8 2 570 336 138 335
13. 9 1 164 479 280 308
14. 9 2 568 335 142 333
15. 10 1 162 479 268 312
16. 10 2 567.5 333 143 331
17. 11 1 161.5 479 253 319
18. 11 2 567.5 332 141 329
19. 12 1 160.5 479 247 327
20. 12 2 568 332 140 329
21. 13 1 158.5 478 231 336
22. 13 2 568.5 331 139 328
23. 14 1 154.5 479 207 354
24. 14 2 568.5 332 141 328
25. 15 1 152 478 204 363
26. 15 2 568 332 142 326
27. 16 1 150 478 186 375
28. 16 2 568.5 333 141 324
29. 17 1 148 478 184 388
30. 17 2 568.5 334 139 328
31. 18 1 146 479 190 396
32. 18 2 568.5 335 141 333
33. 19 1 145.5 479 267 421
34. 19 2 568 335 142 329
35. 20 1 139 479 270 432
36. 20 2 568.5 335 139 332

Table 34. Test 4. Detections from the 20 first frames.

1. 1 136 479 264 336
2. 2 540 308 144 307
3. 4 1 127.5 479 247 350
4. 4 2 544.5 304 151 303
5. 5 1 116.5 479 227 364
6. 5 2 547 303 154 301
7. 6 1 108 479 216 372
8. 6 2 548.5 302 157 300
9. 7 1 95.5 479 187 381
10. 7 2 548.5 300 157 298
11. 8 1 96 479 192 391
12. 8 2 548 298 154 297
13. 9 1 93.5 479 187 400
14. 9 2 545.5 297 147 296
15. 10 1 95 479 190 406
16. 10 2 543.5 298 145 297
17. 11 1 97 479 194 411
18. 11 2 542.5 298 145 298
19. 12 1 100.5 478 201 419
20. 12 2 542 298 144 298
21. 13 1 102.5 478 205 428
22. 13 2 542 297 142 297
23. 14 1 103.5 479 207 434
24. 14 2 541 298 140 298
25. 15 1 110.5 479 221 436
26. 15 2 541.5 299 143 299
27. 16 1 117 479 234 438
28. 16 2 542.5 301 147 301
29. 17 1 120.5 479 241 439
30. 17 2 543 303 146 303
31. 18 1 123 479 246 441
32. 18 2 543.5 304 147 304
33. 19 1 125 478 250 450
34. 19 2 545 304 152 304
35. 20 1 126 478 252 454
36. 20 2 546.5 305 153 305

Some information can be extracted from the detections made. In all the cases, the detection confidence is above 80%, even if the most distant individuals cannot be fully seen.

On the other hand, the tracking identities of test 3 and 4, remained the same during the first 20 frames. However, DeepSORT eventually ended switching these identities due to some long occlusions occurred during both tests. In test 3, these identities ended as human 3 and human 8, while in test 4, the identities ended as human 3 and human 5 respectively. Nevertheless, this was not a problem, since after each switch, that identity remained enough time.

With respect to the visualization of the standing points and safety areas, as it was mentioned previously, there are two radious of danger, one being the half of the other, where the longest radius is one-meter long. Here, there are some examples from each test, see Fig. 49, Fig. 50, and Fig. 51.

Fig. 49. Test 1 localization.

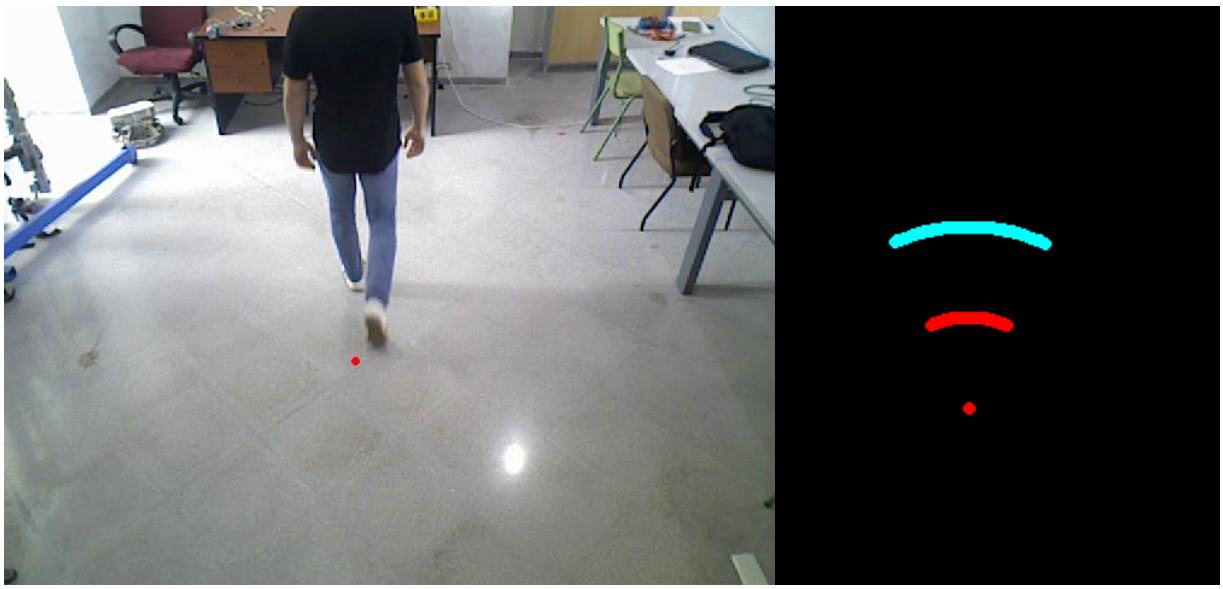


Fig. 50. Test 2 localization.

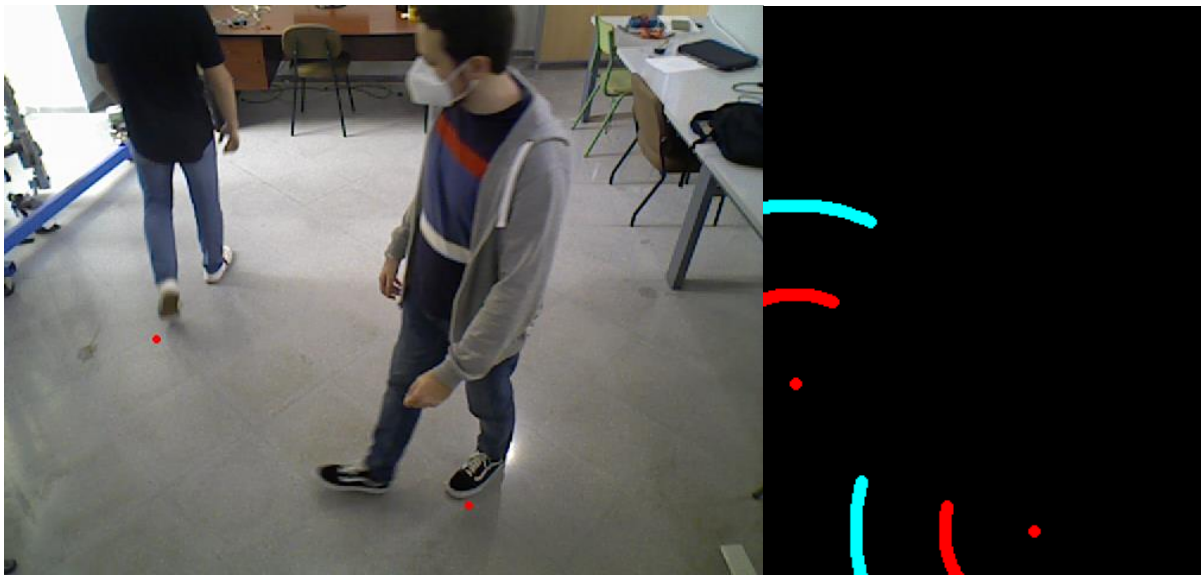
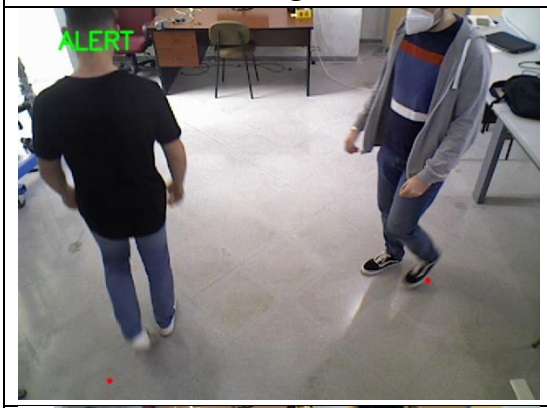


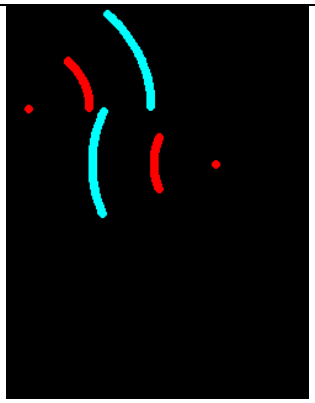
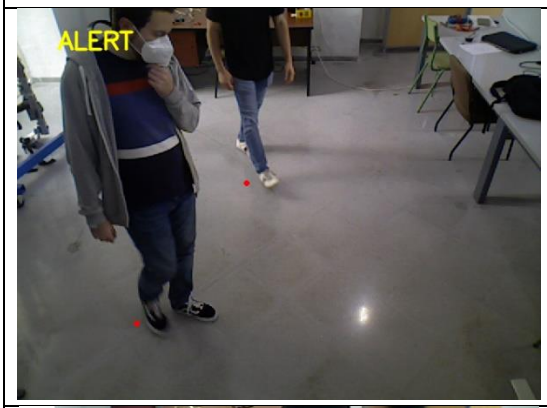
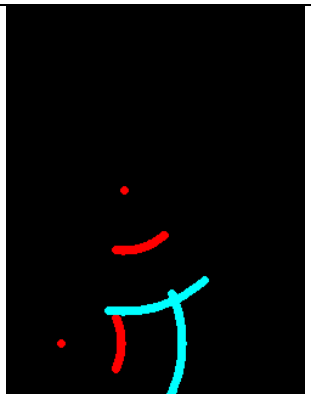
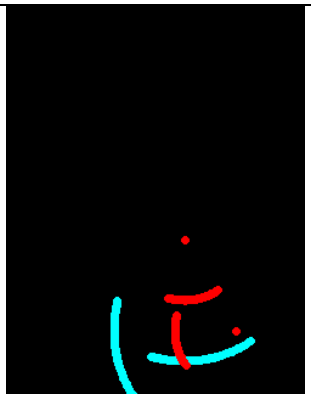
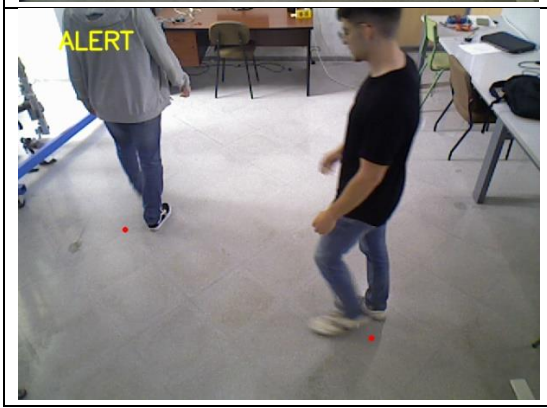
Fig. 51. Test 3 localization.

Fig. 52. Test 4 localization.

Finally, three types of warning are given depending on the distance and trajectory of the individuals, as it was explained in section 3.4.12. In Tables 35 and 36, the results obtained for the dangerous situations in both test 3 and 4 are shown, where the type-0 is drawn in green letters, type-1 in yellow and type-2 in red.

Table 35. Warnings from test 3

Image	Ground	Type and Frame
		Type-0. Frame 33.
		Type-0. Frame 118.

		Type-1. Frame 231.
		Type-1. Frame 237.
		Type-2. Frame 478.
		Type-1. Frame 533.

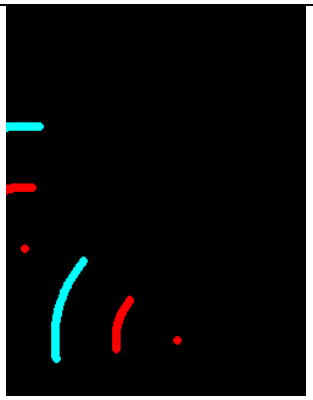
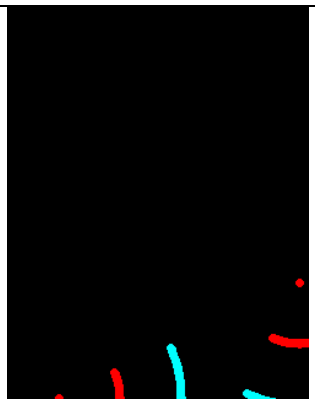
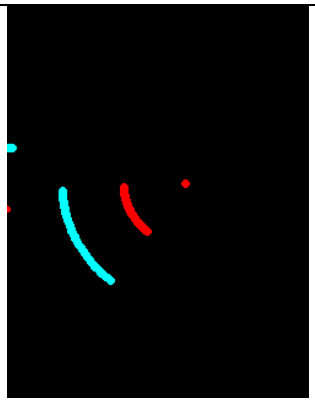
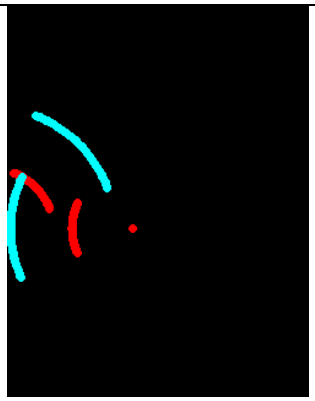
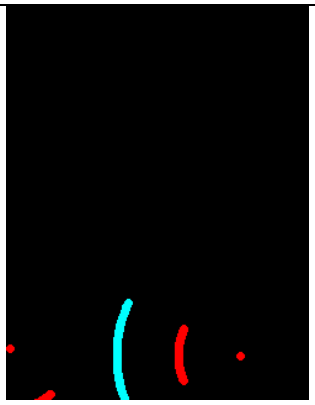
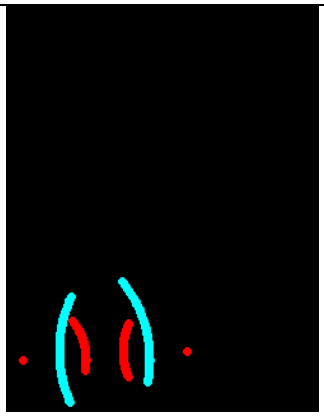
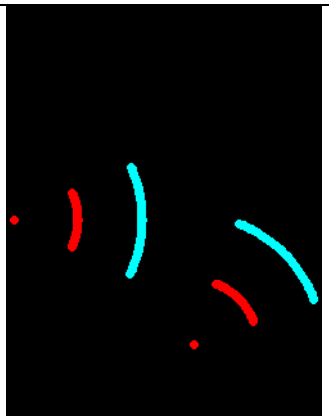
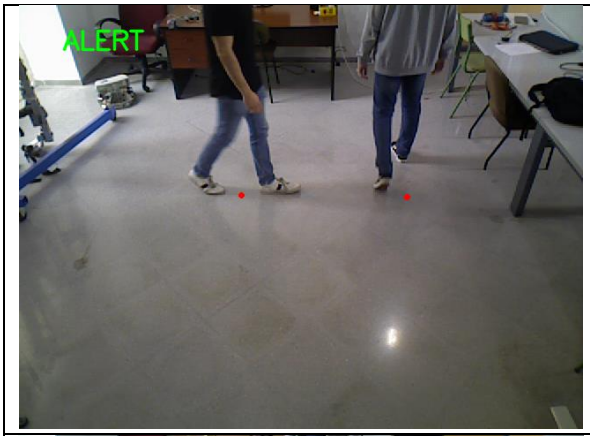
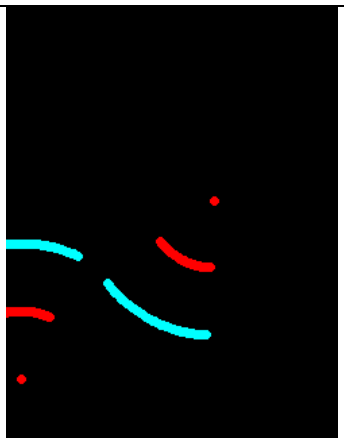
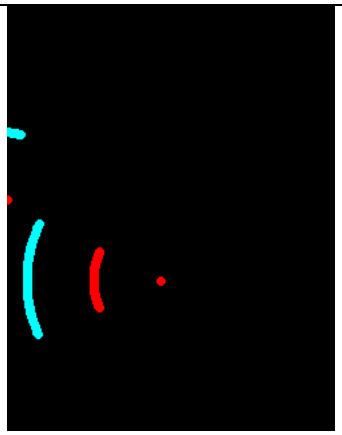
		Type-2. Frame 538.

Table 36. Warnings from test 4

		Type-0. Frame 13.
		Type-0. Frame 117.
		Type-2. Frame 124.
		Type-0. Frame 197.

		Type-2. Frame 206.
		Type-1. Frame 241.
		Type-1. Frame 249.
		Type-1 Frame 261.

		Type-0 Frame 268.
		Type-0 Frame 292.
		Type-2 Frame 310.
		Type-1 Frame 367.

		Type-0 Frame 384.

There are a total of 7 situations in which the algorithm returned an alert on test 3, and a total of 13 on test 4. In test 3, there is just one error, which is in frame 118. Here, there is a false positive from a type-0 alert. Likewise, the same error occurred in frames 241 and 292 from test 4, for which a type-1 and type-0 alerts were given respectively.

On the other hand, the situation in frame 268 from test 4 was set as a type-0 alert, when it was a type-2 situation. Something similar can be said from the frame 367, were a type-1 alert was given for a type-0 situation.

Therefore, there are a total of 20 alerts, for which 3 were not needed, and 2 returned the wrong type of alert. From these errors, the most important is the error on frame 267, which is classifying the situation as less dangerous than it is.

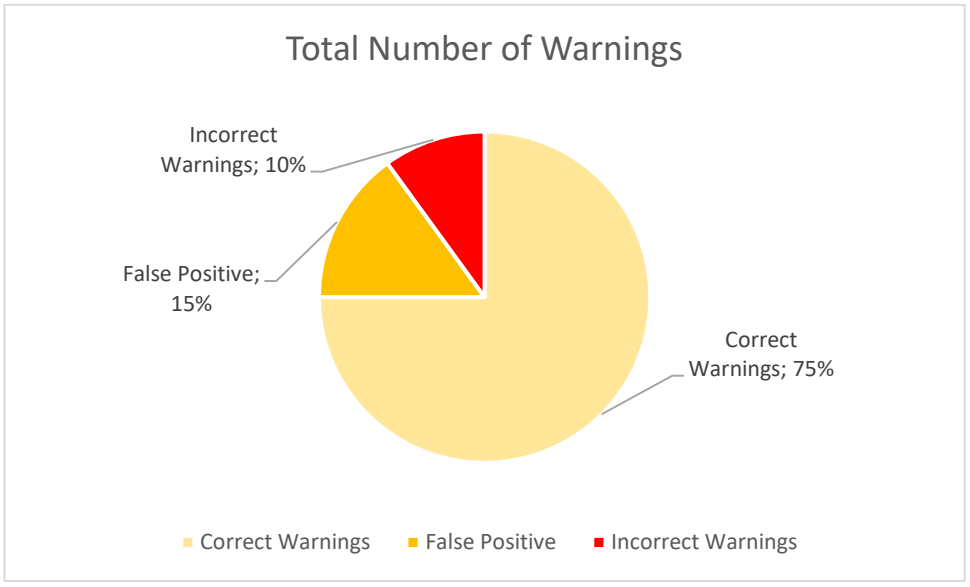


Fig. 53. Pie chart of warnings.

On the other hand, the safety areas sometimes are slightly shifted to a different sight, but in general terms, it has a good performance.

These results can be completely extrapolable for human-vehicles and vehicles-vehicles situations, if the detection model is trained in forklifts. Likewise, the security areas can be easily extended considering bigger angles and longer radius. Therefore, as a future work, I would like to improve the performance on the orientation of the security areas, as well as testing the algorithm in real-time, and train the detection model to be able of recognizing these type of vehicles, which in conclusion, is translated into a more robust collision avoidance algorithm.

Conclusions

1. The aim of the project is achieved throughout two test, on which two people have walked around a predefined area simulating possible collision situations. Here, the algorithm, has been able to effectively classify most of them with the correct type of alert, depending on the distance between the individuals appeared.
2. After reading and comparing the performances in terms of speed and accuracy of several object detection models, the usage of YOLO instead of any other was a key factor. Even though it was not test in real time, YOLOv5 is extremely fast and allow a real-time implementation, which is crucial for this problematic.
3. The researches done in chapters 3, 4 and 5, were essential to understand the basis of a CNN, as well as the fundamentals of an object detector and object tracker respectively.
4. Due to the hardware limitations, Google Colaboratory helped to obtain the detection and tracking files needed for the development of the project. Therefore, it was another key factor in this project.
5. The collision avoidance algorithm has demonstrated to perform accurately, correctly returning a warning for all the 17 situations with a real hazard of collision, although one of those was classified as less dangerous.
6. According to the goal of the project, these results can be use for future work safety developments in the manufacturing industry.

List of references

[1] «Powered Industrial Truck Operator Training,» 14 03 1995. [On Line]. Available: https://www.osha.gov/laws-regs/federalregister/1995-03-14.

[2] «Occupational Injuries, Illnesses, and Fatalities Involving Forklifts,» 10 6 2019. [On Line]. Available: https://www.bls.gov/iif/oshwc/cfoi/forklifts-2017.htm.

[3] «Intel,» [On Line]. Available: https://www.intel.com/content/www/us/en/robotics/autonomous-mobile-robots/overview.html#:~:text=An%20autonomous%20mobile%20robot%20(AMR)%20is%20a%20type%20of%20robot,to%20a%20fixed%2C%20predetermined%20path..

[4] E. Romaine, «Conveyco,» 27 August 2020. [On Line]. Available: https://www.conveyco.com/types-and-applications-of-amrs/.

[5] «Viatech,» [On Line]. Available: https://www.viatech.com/en/products/ai-systems/mobile360-forklift-safety-system/.

[6] «Bigmate,» [On Line]. Available: https://bigmate.com.au/technologies/computer-vision/.

[7] V. Staff, «VentureBeat,» 29 September 2020. [On Line]. Available: https://venturebeat.com/2020/09/29/applying-machine-learning-to-keep-employees-safe-and-save-lives/.

[8] «IBM,» [On Line]. Available: https://www.ibm.com/topics/computer-vision.

[9] B. D. Vijay Kotu, Data Science (Second Edition), 2019.

[10] «GeekforGeeks,» 29 July 2021. [On Line]. Available: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/.

[11] Deepanshi, «Analytics Vidhya,» 25 May 2021. [On Line]. Available: https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-your-first-machine-learning-model-linear-regression/.

[12] A. Saini, «Analytics Vidhya,» 3 August 2021. [On Line]. Available: https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/.

[13] 48saily, «Analytics Vidhya,» 25 February 2021. [On Line]. Available: https://www.analyticsvidhya.com/blog/2021/02/cost-function-is-no-rocket-science/.

[14] R. Kwiatkowski, 22 May 2021. [On Line]. Available: https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21.

[15] Z. Zhihua y J. Li, Big Data Mining for Climate Change, 2020.

[16] S. Sharma, «Towards Data science,» 6 September 2017. [On Line]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[17] V. Jain, «Towards Data Science,» 30 December 2019. [On Line]. Available: https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-

84ba9f82c253#:~:text=Simply%20put%2C%20an%20activation%20function,fired%20to%20the%20next%20neuron..

[18] R. Draelos, «Glass box,» 13 April 2019. [On Line]. Available: https://glassboxmedicine.com/2019/04/13/a-short-history-of-convolutional-neural-networks/.

[19] R. Girshick, I. Forrest, D. Trevor y M. Jitendra, «Deformable Parts Models are Convolutional Neural Networks,» 1 October 2014.

[20] U. J.R.R., K. van de Sande, T. Gevers y S. A. W.M., «Selective Search for Object Recognition,» 2012.

[21] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» 9 May 2016.

[22] J. Redmon y A. Farhadi, «YOLO9000: Better, Faster, Stronger,» 25 December 2016.

[23] K. He, X. Zhang, R. Shaoquing y J. Sun, «Deep Residual Learning for Image Recognition,» 10 December 2015.

[24] J. Redom y A. Farhadi, «YOLOv3: An Incremental Improvement,» 8 April 2018.

[25] A. Bochkovskiy, C.-Y. Wang y H.-Y. M. Liao, «YOLOv4: Optimal Speed and Accuracy of Object Detection,» 23 April 2020.

[26] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen y J.-W. Hsieh, «CSPNet: A New Backbone that can Enhance Learning Capability of CNN,» 27 November 2019.

[27] S. L. Lu Qi, H. Qin, J. Shi y J. Jia, «Path Aggregation Network for Instance Segmentation,» 18 September 2018.

[28] Y. Pei, D. S. Fussell, S. Biswas y K. Pingali, «An Elementary Introduction to Kalman Filtering,» 27 June 2019.

[29] A. Bewley, Z. Ge, L. Ott, F. Ramos y B. Upcroft, «SIMPLE ONLINE AND REALTIME TRACKING,» 2017.

[30] N. Wojke, A. Bewley y P. Drietrich, «Simple Online and Realtime Tracking with a Deep Association Metric,» 12 March 2017.

[31] «Xtion Pro Live,» [On Line]. Available: http://xtionprolive.com/asus-3d-depth-camera/asus-xtion-pro-live.

[32] «Python,» [On Line]. Available: https://www.python.org/doc/essays/blurb/.

[33] I. Orsolic, «Camera Calibration: Explaining Camera Distortions,» [En línea]. Available: https://ori.codes/artificial-intelligence/camera-calibration/camera-distortions/.

[34] «Open Source Computer Vision,» 2022. [On Line]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a.

[35] «Open Source Computer Vision,» 2022. [On Line]. Available: https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga354e0d7c86d0d9da75

de9b9701a9a87e.

[36] «Open Source Computer Vision,» 2022. [On Line]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d.

[37] «Open Source Computer Vision,» 2022. [On Line]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga7a6c4e032c97f03ba747966e6ad862b1.

[38] «Open Source Computer Vision,» 2022. [On Line]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga69f2545a8b62a6b0fc2ee060dc30559d.

[39] «Open Source Computer Vision,» 2022. [On Line]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga4abc2ece9fab9398f2e560d53c8c9780.

[40] «Open Source Computer Vision,» 2022. [On Line] Available: https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87.

Appendices

Appendix 1. Tracking file – Test 1.

```
3 1 388 475 260 391
4 1 384 474 256 400
5 1 380 476 250 408
6 1 376.5 477 233 415
7 1 374 478 220 424
8 1 371.5 478 205 437
9 1 367.5 479 201 448
10 1 363.5 477 199 451
11 1 360.5 476 197 455
12 1 357 464 200 455
13 1 359 451 194 449
14 1 361 448 194 448
15 1 363 446 192 446
16 1 368.5 445 191 445
17 1 375 445 190 445
18 1 385.5 444 189 444
19 1 394.5 440 187 440
20 1 401.5 429 185 429
21 1 405.5 419 183 419
22 1 407.5 415 179 415
23 1 408.5 414 175 414
24 1 405.5 412 169 412
25 1 402.5 412 167 412
26 1 399.5 412 161 412
27 1 396.5 412 157 412
28 1 390.5 412 157 412
29 1 371 410 162 410
30 1 357 412 160 411
31 1 342.5 412 163 412
32 1 328.5 415 163 414
33 1 304.5 417 201 417
34 1 284 415 198 415
35 1 268.5 414 195 414
36 1 247 439 156 439
37 1 228.5 457 175 457
38 1 220.5 463 183 462
39 1 206.5 464 213 461
40 1 197 464 232 459
41 1 190.5 464 241 457
42 1 181 466 256 455
43 1 170 465 274 448
44 1 152.5 463 287 445
45 1 142.5 465 283 447
46 1 135.5 467 271 448
47 1 123.5 469 247 445
48 1 113.5 469 227 443
49 1 103 467 206 437
50 1 94.5 465 189 431
51 1 93.5 469 187 419
52 1 93 468 186 413
53 1 92 469 184 415
54 1 92 469 184 418
55 1 89.5 464 179 418
56 1 83 467 164 425
57 1 80.5 464 161 423
58 1 73 460 146 422
```

```
59 1 62.5 455 121 424
60 1 59 451 118 430
61 1 59 443 118 432
62 1 59.5 436 119 429
63 1 59.5 435 119 429
64 1 63 433 126 431
65 1 65.5 429 131 427
66 1 66.5 430 133 430
67 1 66.5 428 133 428
68 1 66 424 132 424
69 1 63 398 126 398
70 1 63.5 370 127 370
71 1 65.5 347 131 347
72 1 65.5 333 131 333
73 1 66.5 327 133 327
74 1 73.5 327 147 327
75 1 75 327 150 327
76 1 80 325 160 324
77 1 81 322 162 322
78 1 82 313 164 313
79 1 83.5 302 159 302
80 1 85 294 156 294
81 1 88 279 154 279
82 1 90.5 266 149 266
83 1 93 255 146 255
84 1 94.5 245 141 245
85 1 98.5 242 131 242
86 1 102.5 240 125 240
87 1 105.5 239 121 239
88 1 108.5 236 123 236
89 1 111 234 124 234
90 1 113 232 126 232
91 1 116 229 124 229
92 1 120 224 122 224
93 1 121.5 220 121 220
94 1 124 217 120 217
95 1 125.5 200 121 200
96 1 130.5 188 115 188
97 1 138 186 120 186
98 1 145 186 122 186
99 1 149.5 185 119 185
100 1 157 185 116 185
101 1 164 183 108 183
102 1 168 179 110 179
103 1 170 175 110 175
104 1 171.5 171 107 171
105 1 173 167 102 167
106 1 175 159 98 159
107 1 184.5 151 83 151
108 1 189.5 149 81 149
109 1 194 148 78 148
110 1 198 148 72 148
111 1 201 148 70 148
112 1 212.5 147 95 147
113 1 223 147 96 147
114 1 226.5 147 89 147
115 1 233 147 100 147
116 1 236 145 98 145
117 1 239.5 141 93 140
118 1 242.5 139 89 138
119 1 250.5 133 81 132
```

```
120 1 260.5 128 65 128
121 1 267 133 60 133
122 1 283 133 88 133
123 1 293 133 98 133
124 1 299 131 106 131
125 1 305 130 116 130
126 1 309 130 114 130
127 1 312 129 114 129
128 1 316 130 104 130
129 1 322 129 90 129
130 1 326.5 129 83 129
131 1 330.5 130 77 130
132 1 336.5 129 75 129
133 1 353.5 129 75 129
134 1 364.5 129 79 129
135 1 370.5 130 83 130
136 1 374 129 88 129
137 1 379 130 98 130
138 1 380 130 96 130
139 1 381.5 131 101 131
140 1 383.5 131 105 130
141 1 385 130 104 129
142 1 394.5 127 73 127
143 1 403.5 120 61 120
144 1 409.5 125 53 125
145 1 417.5 133 49 133
146 1 426.5 138 67 138
147 1 433.5 143 77 143
148 1 437 148 78 148
149 1 438.5 153 79 153
150 1 440.5 158 75 158
151 1 442.5 160 77 160
152 1 445.5 160 77 160
153 1 450 161 80 161
154 1 453.5 160 79 160
155 1 456.5 167 77 167
156 1 459 177 78 177
157 1 460 182 78 182
158 1 461 188 76 188
159 1 464.5 195 85 195
160 1 467 202 88 202
161 1 468.5 207 91 207
162 1 472 210 94 210
163 1 477.5 212 105 212
164 1 479.5 212 109 212
165 1 484.5 212 117 212
166 1 486.5 211 117 210
167 1 490 231 118 231
168 1 492.5 245 119 245
169 1 494.5 253 121 253
170 1 496 263 118 263
171 1 497 269 118 269
172 1 498.5 279 121 279
173 1 499.5 283 123 283
174 1 502.5 284 125 284
175 1 505.5 284 127 284
176 1 509.5 285 125 285
177 1 512.5 288 123 288
178 1 515.5 310 121 310
179 1 520 329 126 329
180 1 525.5 350 135 350
```

```
181 1 535 359 160 359
182 1 543 363 172 362
183 1 546.5 365 171 364
184 1 547 366 174 366
185 1 550 365 178 365
186 1 552.5 366 173 366
187 1 554 371 170 371
188 1 556 399 166 399
189 1 556 408 166 408
190 1 555.5 411 167 411
191 1 554 415 170 415
192 1 550 414 178 414
193 1 549.5 430 179 430
194 1 548 442 182 442
195 1 547.5 444 183 444
196 1 548.5 444 181 444
197 1 550.5 443 175 443
198 1 549 447 180 447
199 1 526.5 446 169 446
200 1 510.5 445 185 445
201 1 496 448 188 448
202 1 488.5 448 187 448
203 1 486.5 446 193 446
204 1 474.5 444 167 444
205 1 462 443 156 443
206 1 456.5 443 149 443
207 1 453 439 144 439
208 1 449.5 431 137 431
209 1 440 409 136 409
210 1 427.5 379 141 379
211 1 416 360 142 360
212 1 405.5 346 137 346
213 1 389 335 150 335
214 1 373.5 331 153 331
215 1 360.5 329 149 329
216 1 350 320 138 320
217 1 342 310 128 310
218 1 331 301 126 301
219 1 317.5 280 129 280
220 1 305 269 130 269
221 1 292 269 128 269
222 1 285 267 128 267
223 1 281 266 124 266
224 1 273.5 265 119 265
225 1 269.5 264 117 264
226 1 266.5 263 117 263
227 1 258.5 258 109 258
228 1 251.5 247 107 247
229 1 245 237 104 237
230 1 237.5 221 105 221
231 1 234 213 100 213
232 1 232 209 102 209
233 1 229 208 94 208
234 1 224.5 207 97 207
235 1 222 207 96 207
236 1 218 206 100 205
237 1 216 205 104 204
238 1 212.5 202 107 201
239 1 207.5 198 111 197
240 1 202.5 194 113 193
241 1 196.5 186 113 186
```

```
242 1 185.5 175 107 175
243 1 177.5 171 95 171
244 1 174 170 94 170
245 1 178.5 170 73 170
246 1 180.5 170 69 170
247 1 174.5 169 89 169
248 1 173.5 168 91 168
249 1 171.5 166 85 166
250 1 171 164 86 164
251 1 173 161 94 161
252 1 174 155 94 155
253 1 174.5 150 91 150
254 1 176.5 146 89 146
255 1 179 145 82 145
256 1 181 142 80 142
257 1 182 141 80 141
258 1 181.5 141 77 141
259 1 180.5 141 73 141
260 1 180 139 70 139
261 1 181 137 74 137
262 1 192.5 135 87 135
263 1 200.5 134 89 134
264 1 210.5 134 103 134
265 1 216 134 106 134
266 1 218.5 134 107 134
267 1 221 134 110 134
268 1 223.5 133 107 133
269 1 226.5 132 103 132
270 1 230 133 96 133
271 1 232.5 132 91 131
272 1 237.5 133 79 132
273 1 249.5 133 65 132
274 1 255 133 64 132
275 1 268.5 134 81 133
276 1 273.5 135 83 134
277 1 277 136 76 135
278 1 281.5 136 71 135
279 1 283.5 137 81 136
280 1 286 139 92 138
281 1 291.5 143 81 142
282 1 295.5 147 79 147
283 1 296 159 80 158
284 1 296.5 163 81 162
285 1 294.5 170 79 169
286 1 293.5 175 81 174
287 1 293 181 86 181
288 1 292 185 82 185
289 1 292.5 187 83 187
290 1 293 188 92 188
291 1 288.5 188 107 188
292 1 286 189 106 189
293 1 285.5 202 103 202
294 1 285 213 104 213
295 1 285 225 108 225
296 1 285.5 239 109 239
297 1 285.5 246 111 246
298 1 285.5 248 115 247
299 1 285.5 249 117 248
300 1 285.5 249 117 248
301 1 285.5 249 117 248
302 1 285 257 114 256
```

```
303 1 284.5 269 113 269
304 1 282 285 116 285
305 1 280 300 118 300
306 1 278 314 118 314
307 1 275.5 319 121 319
308 1 273 322 128 322
309 1 270.5 323 135 323
310 1 267.5 322 143 322
311 1 263.5 330 145 330
312 1 261.5 347 145 347
313 1 261.5 362 147 362
314 1 261.5 374 149 374
315 1 262.5 382 149 382
316 1 264 386 150 386
317 1 266 388 154 388
318 1 269.5 388 159 388
319 1 271.5 386 163 386
320 1 273 385 166 384
321 1 274.5 401 165 401
322 1 275.5 406 165 406
323 1 276 409 164 409
324 1 275.5 410 161 410
325 1 275.5 411 151 411
326 1 276 412 140 412
327 1 277.5 411 131 411
328 1 280 413 130 413
329 1 280 415 126 415
330 1 278 414 126 414
331 1 275.5 416 123 416
332 1 275 415 128 415
333 1 278.5 417 135 417
334 1 284 417 144 417
335 1 289.5 415 147 415
336 1 296 413 150 413
337 1 301 413 152 413
338 1 306 411 150 411
339 1 309.5 408 149 408
340 1 311 407 148 407
341 1 312.5 403 143 403
342 1 314.5 387 141 387
343 1 316.5 367 137 367
344 1 318.5 344 133 344
345 1 322 333 126 333
346 1 325 331 122 331
347 1 327.5 328 121 328
348 1 332 328 122 327
349 1 337.5 327 117 326
350 1 343 324 118 323
351 1 348.5 318 119 317
352 1 355 308 118 308
353 1 359 301 118 301
354 1 361 294 118 294
355 1 364.5 276 121 276
356 1 367.5 263 119 263
357 1 369.5 253 117 253
358 1 373 249 114 249
359 1 376.5 249 115 249
360 1 378.5 249 115 249
361 1 381 249 112 249
362 1 383.5 248 109 247
363 1 384.5 241 109 240
```

```
364 1 385.5 236 107 235
365 1 386 231 106 230
366 1 386 227 106 226
367 1 385.5 220 109 220
368 1 389 200 102 200
369 1 391 192 96 192
370 1 393 190 92 190
371 1 394 190 90 190
372 1 397 189 88 189
373 1 400 189 88 189
374 1 403 188 88 188
375 1 407.5 187 87 187
376 1 409.5 183 85 183
377 1 408.5 179 77 179
378 1 410 175 82 175
379 1 411 169 82 169
380 1 411 159 78 159
381 1 414.5 152 85 152
382 1 417.5 150 85 150
383 1 417 149 72 149
384 1 419 149 68 149
385 1 422.5 148 65 148
386 1 423 147 66 147
387 1 421.5 145 61 145
388 1 421 144 60 143
389 1 418 144 54 143
390 1 413.5 144 47 143
391 1 412 144 46 143
392 1 417 145 66 145
393 1 417.5 146 71 146
394 1 412.5 145 63 145
395 1 412.5 145 93 145
396 1 406.5 145 77 145
397 1 409 144 96 144
398 1 403 144 78 144
399 1 403.5 144 85 144
400 1 402 145 80 144
401 1 402 146 74 145
402 1 403.5 146 73 145
403 1 405.5 146 73 145
404 1 407 147 72 146
405 1 407.5 148 77 147
406 1 409 154 76 153
407 1 411 160 86 160
408 1 413 166 94 166
409 1 414 171 92 171
410 1 417.5 177 87 177
411 1 420 180 86 180
412 1 420 181 88 181
413 1 420.5 182 93 182
414 1 422 183 94 183
415 1 423.5 183 95 183
416 1 424.5 185 97 185
417 1 426 196 94 196
418 1 431 208 94 208
419 1 435 217 96 216
420 1 439 230 98 229
421 1 444.5 238 107 238
422 1 448.5 241 109 241
423 1 452.5 241 117 241
424 1 457.5 242 125 242
```

```
425 1 460.5 243 127 243
426 1 470 269 122 269
427 1 476 285 128 285
428 1 478.5 295 131 295
429 1 481 308 132 308
430 1 482 320 134 320
431 1 484.5 325 141 325
432 1 486 326 144 326
433 1 488.5 326 145 326
434 1 489.5 325 143 325
435 1 490.5 326 143 326
436 1 493 331 144 331
437 1 497.5 367 141 367
438 1 502 392 148 392
439 1 507 410 156 410
440 1 510.5 421 157 421
441 1 515.5 432 163 432
442 1 524 436 186 436
443 1 532 440 204 440
444 1 536 443 206 443
445 1 537.5 444 203 444
446 1 539.5 452 199 452
447 1 539.5 470 199 467
448 1 538.5 479 201 472
449 1 534.5 479 209 461
450 1 530 479 218 448
451 1 525 479 228 440
452 1 518.5 479 241 429
453 1 508 479 262 416
454 1 503 479 272 405
455 1 500 479 278 395
456 1 495.5 479 287 385
457 1 486.5 479 305 366
458 1 484.5 479 309 354
459 1 474.5 479 329 318
460 1 472.5 478 333 283
461 1 471.5 479 335 267
462 1 469 479 340 250
463 1 467.5 479 339 242
464 1 464.5 479 345 238
465 1 461.5 479 347 241
```

Appendix 2. Tracking file – Test 2.

```
3 1 281.5 478 349 207
4 1 290 478 336 221
5 1 300 478 336 234
6 1 307 478 334 243
7 1 313 479 332 251
8 1 320.5 478 325 260
9 1 330 478 320 268
10 1 340.5 478 317 276
11 1 355 479 316 282
12 1 365.5 479 313 284
13 1 382 479 314 284
14 1 397 479 310 286
15 1 404 479 310 288
16 1 406.5 479 309 289
17 1 409.5 479 299 294
18 1 409 479 298 300
19 1 406 479 290 310
20 1 398.5 479 245 343
21 1 392 479 236 363
22 1 386 479 230 376
23 1 383 479 228 383
24 1 382 479 216 393
25 1 384 478 192 403
26 1 389.5 478 189 410
27 1 397 476 182 416
28 1 402 477 182 429
29 1 401 479 194 441
30 1 400.5 479 197 446
31 1 395.5 479 203 450
32 1 389 479 204 451
33 1 378.5 479 199 456
34 1 365.5 478 195 460
35 1 356.5 479 193 464
36 1 346.5 479 193 467
37 1 339.5 478 193 468
38 1 329.5 478 193 470
39 1 321 476 192 473
40 1 314.5 467 195 467
41 1 308.5 452 193 452
42 1 302.5 448 181 448
43 1 299 446 178 446
44 1 298 440 168 440
45 1 297.5 430 165 430
46 1 298 422 162 422
47 1 298.5 414 159 414
48 1 298.5 406 155 406
49 1 298.5 368 153 368
50 1 298.5 343 147 343
51 1 298 322 142 322
52 1 298 314 134 314
53 1 297 311 130 310
54 1 296 309 126 309
55 1 296 310 126 310
56 1 294.5 311 123 311
57 1 294 307 122 307
58 1 292.5 299 123 299
59 1 291 294 124 294
60 1 291 285 126 285
```

```
61 1 290 257 132 257
62 1 290 242 124 242
63 1 290 235 120 235
64 1 289.5 234 117 234
65 1 290.5 234 113 234
66 1 290.5 233 113 232
67 1 292 231 112 229
68 1 296 228 110 228
69 1 299 222 108 222
70 1 302 213 104 213
71 1 304 200 106 200
72 1 304.5 191 103 191
73 1 303 181 96 181
74 1 301.5 177 77 177
75 1 298.5 176 69 176
76 1 296.5 176 69 176
77 1 295.5 177 69 177
78 1 294.5 176 67 176
79 1 293 176 72 176
80 1 291.5 169 75 168
81 1 290 161 74 160
82 1 288.5 152 77 152
83 1 285.5 149 79 149
84 1 283.5 148 69 148
85 1 276.5 147 61 147
86 1 272 147 64 146
87 1 269 148 70 148
88 1 266 148 72 148
89 1 264.5 146 75 146
90 1 263 145 76 145
91 1 261.5 141 75 141
92 1 255.5 139 69 139
93 1 246 141 62 141
94 1 240 145 58 145
95 1 234 151 54 151
96 1 231.5 160 53 160
97 1 229 167 60 167
98 1 227.5 170 65 170
99 1 225 172 70 172
100 1 216 175 96 175
101 1 203 178 124 178
102 1 198 180 94 180
103 1 195 182 78 182
104 1 189.5 182 77 182
105 1 187 182 80 182
106 1 184 184 82 184
107 1 180.5 194 85 194
108 1 178.5 201 89 201
109 1 173.5 207 93 207
110 1 168.5 216 103 216
111 1 165.5 222 105 222
112 1 156.5 229 121 229
113 1 137.5 231 133 231
114 1 126 235 128 235
115 1 117.5 252 125 252
116 1 112.5 265 125 265
117 1 109 276 124 276
118 1 107 283 126 283
119 1 107 289 126 289
120 1 107.5 300 129 300
121 1 106.5 303 133 303
```

```
122 1 105.5 304 137 304
123 1 104.5 305 137 305
124 1 101 306 138 306
125 1 99.5 306 141 306
126 1 94 335 142 335
127 1 90 353 152 353
128 1 85.5 366 159 366
129 1 83.5 375 163 375
130 1 81 387 162 387
131 1 82 392 164 392
132 1 82 394 164 394
133 1 83.5 395 167 395
134 1 83 394 166 394
135 1 78.5 394 155 394
136 1 76.5 394 153 394
137 1 77.5 395 155 395
138 1 77.5 395 155 395
139 1 91 395 182 395
140 1 103 394 186 394
141 1 113.5 394 181 394
142 1 119.5 395 175 393
143 1 131.5 393 147 392
144 1 138.5 370 141 369
145 1 142 357 132 356
146 1 153.5 333 133 333
147 1 174.5 323 157 323
148 1 192.5 319 169 317
149 1 205 319 166 318
150 1 215.5 318 161 317
151 1 224 316 158 314
152 1 227.5 312 151 310
153 1 251.5 275 139 274
154 1 265.5 261 133 260
155 1 276 256 132 255
156 1 284.5 254 129 253
157 1 290 253 128 252
158 1 297.5 252 123 251
159 1 303.5 247 119 246
160 1 307 244 114 243
161 1 310 239 106 238
162 1 312.5 233 99 232
163 1 315 206 102 206
164 1 317 196 98 196
165 1 330 194 86 194
166 1 339.5 193 85 193
167 1 346 193 88 193
168 1 352 192 90 192
169 1 354.5 191 89 191
170 1 356 185 88 185
171 1 358.5 182 89 182
172 1 360.5 177 85 177
173 1 364 173 90 173
174 1 367.5 167 91 167
175 1 373 160 96 160
176 1 384 154 94 154
177 1 390 153 90 153
178 1 395.5 153 87 153
179 1 399 153 86 153
180 1 403.5 152 81 152
181 1 405 152 80 152
182 1 406 151 78 151
```

```
183 1 405 149 80 149
184 1 403.5 148 79 148
185 1 400 143 70 143
186 1 398.5 139 65 139
187 1 401 138 58 138
188 1 406 138 50 138
189 1 409.5 139 47 139
190 1 415 141 54 141
191 1 417 143 54 143
192 1 418 145 54 144
193 1 421.5 148 65 148
194 1 424.5 149 67 149
195 1 429 153 78 153
196 1 433 157 82 157
197 1 435.5 159 83 159
198 1 437.5 159 85 159
199 1 442 160 82 160
200 1 451 161 80 160
201 1 455 160 80 160
202 1 458 163 82 163
203 1 459.5 172 79 172
204 1 463 181 82 181
205 1 466.5 186 87 186
206 1 471 198 86 198
207 1 476.5 203 95 203
208 1 480 206 102 206
209 1 481.5 207 105 207
210 1 482.5 207 107 207
211 1 484.5 207 111 207
212 1 487 206 114 206
213 1 490.5 206 119 206
214 1 492.5 206 119 206
215 1 498 227 110 227
216 1 501 238 114 238
217 1 502 243 114 243
218 1 503 246 116 246
219 1 502 249 120 249
220 1 501.5 249 123 249
221 1 500.5 248 127 248
222 1 499.5 248 131 248
223 1 497.5 246 135 245
224 1 497 248 136 247
225 1 496.5 260 135 259
226 1 495.5 269 141 268
227 1 494 275 142 274
228 1 494.5 277 139 276
229 1 496 278 132 277
230 1 496.5 279 123 279
231 1 495 280 124 280
232 1 464 279 158 279
233 1 439.5 279 167 279
234 1 420.5 280 175 280
235 1 414 280 172 280
236 1 403 276 160 276
237 1 393.5 273 147 273
238 1 381 270 118 270
239 1 358.5 274 111 274
240 1 337.5 273 107 273
241 1 320 277 114 277
242 1 310.5 284 119 284
243 1 304 283 128 283
```

```
244 1 291.5 281 159 281
245 1 291 282 150 281
246 1 283.5 286 163 286
247 1 280.5 288 159 288
248 1 277 288 150 288
249 1 267 288 138 288
250 1 251.5 287 111 287
251 1 240 287 110 287
252 1 214.5 288 107 288
253 1 203 288 114 287
254 1 194.5 287 131 287
255 1 188 286 142 286
256 1 175 286 170 286
257 1 172.5 287 169 287
258 1 169 287 172 287
259 1 161.5 287 169 287
260 1 158 287 166 287
261 1 148.5 293 153 293
262 1 142.5 301 157 301
263 1 138 307 166 307
264 1 138 310 170 310
265 1 138.5 312 171 312
266 1 140.5 316 167 315
267 1 143.5 317 161 316
268 1 147 317 154 316
269 1 151.5 316 155 315
270 1 154.5 316 155 315
271 1 159.5 320 153 320
272 1 166.5 349 155 349
273 1 170 362 160 362
274 1 172.5 379 163 379
275 1 174 389 168 389
276 1 175 396 172 395
277 1 175.5 399 173 399
278 1 175.5 399 179 399
279 1 176.5 400 183 400
280 1 178 400 188 400
281 1 179 401 190 401
282 1 181.5 400 197 400
283 1 184.5 398 201 398
284 1 186.5 398 199 398
285 1 188.5 401 195 401
286 1 192 402 190 402
287 1 195 402 190 402
288 1 199 401 188 401
289 1 202 400 188 400
290 1 206 401 190 401
291 1 210.5 405 189 405
292 1 212 410 188 410
293 1 213.5 420 187 420
294 1 214.5 422 185 422
295 1 215 423 184 423
296 1 215.5 424 173 424
297 1 216 425 168 425
298 1 218 423 154 423
299 1 220 422 148 422
300 1 222.5 422 143 422
301 1 224.5 424 139 424
302 1 224 428 144 428
303 1 221.5 429 145 429
304 1 223 429 158 429
```

```
305 1 225.5 430 167 430
306 1 226.5 430 177 430
307 1 227.5 430 179 430
308 1 227 431 180 431
309 1 226.5 430 183 430
310 1 225 430 184 430
311 1 222 433 182 433
312 1 219.5 430 179 430
313 1 217 421 178 421
314 1 215 406 174 406
315 1 214.5 402 169 402
316 1 214.5 401 167 401
317 1 216.5 399 165 399
318 1 218.5 395 161 395
319 1 220.5 394 159 394
320 1 230 384 154 384
321 1 237 373 146 373
322 1 241 366 144 366
323 1 244.5 354 143 354
324 1 247.5 339 141 339
325 1 253.5 302 137 302
326 1 256.5 290 129 290
327 1 258 285 124 285
328 1 258.5 283 121 283
329 1 258.5 281 119 281
330 1 258.5 273 117 273
331 1 258.5 263 117 263
332 1 259 247 118 247
333 1 259 235 118 235
334 1 258.5 223 117 223
335 1 258.5 214 113 214
336 1 258 211 108 211
337 1 258.5 210 105 210
338 1 259 210 104 210
339 1 259 210 104 210
340 1 261.5 194 99 194
341 1 262.5 181 95 181
342 1 263.5 173 93 173
343 1 264.5 165 93 164
344 1 265 162 88 161
345 1 263 162 84 160
346 1 263.5 162 83 160
347 1 265 161 84 159
348 1 263 161 76 161
349 1 265.5 157 81 157
350 1 266.5 154 75 154
351 1 266.5 151 67 151
352 1 266 149 66 149
353 1 264.5 147 65 147
354 1 263.5 144 61 144
355 1 263.5 143 59 143
356 1 263.5 142 55 142
357 1 264 142 56 142
358 1 264 142 56 142
359 1 264.5 142 55 142
360 1 266.5 143 55 143
361 1 269.5 143 59 143
362 1 274 142 62 142
363 1 275.5 142 61 142
364 1 277 142 60 142
365 1 278.5 142 55 142
```

```
366 1 280 142 56 142
367 1 282 143 56 143
368 1 284.5 143 55 142
369 1 295 142 72 142
370 1 303 143 80 143
371 1 311.5 143 95 143
372 1 318 144 102 144
373 1 322 143 112 143
374 1 324.5 143 113 143
375 1 326.5 142 113 142
376 1 327.5 143 111 143
377 1 330.5 143 103 143
378 1 333.5 142 97 142
379 1 337.5 138 81 138
380 1 344 131 68 131
381 1 359 136 56 136
382 1 375.5 137 79 137
383 1 385 139 86 139
384 1 389 140 90 140
385 1 391.5 140 89 140
386 1 391.5 141 89 141
387 1 394 141 82 141
388 1 397.5 142 75 142
389 1 410 142 60 142
390 1 419.5 142 61 142
391 1 423.5 142 61 142
392 1 428 142 64 142
393 1 429 141 64 141
394 1 428.5 141 61 141
395 1 429.5 141 63 141
396 1 432.5 141 69 141
397 1 437.5 143 73 143
398 1 440.5 145 79 144
399 1 442 150 84 149
400 1 443.5 158 85 158
401 1 443.5 166 85 166
402 1 444 172 80 172
403 1 447.5 176 87 176
404 1 451 181 84 181
405 1 452 185 84 185
406 1 453 187 80 186
407 1 454.5 187 91 186
408 1 456 187 90 186
409 1 456 187 94 186
410 1 457 187 94 186
411 1 456.5 187 93 186
412 1 460.5 204 93 204
413 1 463 214 96 214
414 1 469 226 98 226
415 1 472 236 102 236
416 1 475.5 246 107 246
417 1 477 248 110 248
418 1 487 251 132 251
419 1 493 252 134 252
420 1 499 267 132 267
421 1 502.5 279 135 279
422 1 507 296 132 296
423 1 509.5 311 131 311
424 1 511.5 326 135 326
425 1 512.5 332 137 332
426 1 514 335 142 335
```

```
427 1 515.5 335 145 334
428 1 518.5 334 147 333
429 1 520.5 335 147 334
430 1 523 335 148 334
431 1 525.5 350 145 350
432 1 533 379 150 379
433 1 540 398 160 398
434 1 543.5 407 163 407
435 1 545 413 166 413
436 1 547.5 418 169 418
437 1 551 425 176 425
438 1 551.5 428 175 428
439 1 550 431 178 431
440 1 549.5 432 179 432
441 1 549 432 180 432
442 1 548.5 429 181 429
443 1 549 429 180 429
444 1 548 429 182 429
445 1 546.5 431 185 431
446 1 542 432 194 432
447 1 537 430 204 428
448 1 533 429 212 427
449 1 529 426 220 423
450 1 522.5 425 219 422
451 1 513.5 424 215 421
452 1 502 423 220 420
453 1 488 420 222 415
454 1 463 415 214 408
455 1 447 413 192 405
456 1 434 412 184 406
457 1 427.5 413 181 408
458 1 415.5 407 159 403
459 1 405.5 405 151 404
460 1 380 413 148 413
461 1 364 431 152 431
462 1 346 425 162 425
463 1 328 428 162 428
464 1 314.5 422 169 422
465 1 296 421 202 420
466 1 289 421 200 420
467 1 284 425 206 424
468 1 274 427 216 426
469 1 257.5 429 229 429
470 1 239 428 230 428
471 1 215.5 427 233 426
472 1 195.5 430 215 429
473 1 176 428 200 427
474 1 159 431 202 430
475 1 151 431 210 429
476 1 141 430 230 428
477 1 136.5 429 233 427
478 1 131.5 431 249 430
479 1 129.5 434 251 434
480 1 115 429 224 429
481 1 103.5 429 203 429
482 1 87 431 166 429
483 1 83 431 166 428
484 1 83 430 166 425
485 1 83.5 423 167 417
486 1 83 418 166 413
487 1 83 416 166 413
```

```
488 1 83 415 166 412
489 1 82.5 414 165 410
490 1 81.5 415 163 411
491 1 78 418 156 416
492 1 74.5 417 149 416
493 1 73 419 146 419
494 1 73.5 418 147 418
495 1 76 418 152 418
496 1 76.5 416 153 416
497 1 81.5 416 163 416
498 1 84.5 419 169 419
499 1 86.5 422 173 422
500 1 91 421 182 421
501 1 95 419 190 419
502 1 97.5 417 195 417
503 1 98 415 196 415
504 1 100 411 200 411
505 1 100.5 401 197 401
506 1 106.5 389 183 389
507 1 112 368 174 368
508 1 117 355 170 355
509 1 126.5 334 171 334
510 1 134 326 162 326
511 1 142.5 322 155 322
512 1 149.5 320 153 320
513 1 155 315 146 314
514 1 162.5 305 145 305
515 1 166.5 297 145 297
516 1 169.5 282 145 282
517 1 174.5 265 137 265
518 1 179.5 249 133 249
519 1 182 245 128 245
520 1 184.5 244 131 244
521 1 192 245 130 245
522 1 198.5 245 125 245
523 1 203.5 245 119 245
524 1 211 235 108 235
525 1 216 227 108 227
526 1 221.5 215 109 215
527 1 228 196 114 196
528 1 233 190 110 190
529 1 236 187 108 187
530 1 238.5 186 103 186
531 1 242.5 181 97 181
532 1 248.5 176 85 176
533 1 250.5 164 95 164
534 1 255.5 155 89 155
535 1 254.5 152 87 152
536 1 256 151 82 151
537 1 255.5 151 85 151
538 1 255 150 88 150
539 1 255 150 90 150
540 1 256 149 88 149
541 1 257 148 86 148
542 1 260.5 147 75 147
543 1 265.5 145 67 145
544 1 271 143 54 142
545 1 275.5 142 47 141
546 1 278 142 48 142
547 1 279 142 50 142
548 1 280 142 46 142
```

```
549 1 280 142 60 142
550 1 279 142 66 142
551 1 278.5 143 73 143
552 1 278.5 143 73 143
553 1 288 143 76 143
554 1 291 143 76 143
555 1 293.5 142 79 141
556 1 295 144 72 143
557 1 293.5 148 73 147
558 1 291.5 154 77 154
559 1 289.5 157 73 157
560 1 287 161 74 161
561 1 280.5 171 77 171
562 1 277.5 175 81 175
563 1 273 177 88 177
564 1 270.5 177 81 176
565 1 263.5 178 95 177
566 1 260.5 178 97 177
567 1 258.5 179 97 179
568 1 257 183 96 183
569 1 254.5 194 97 194
570 1 252.5 202 101 202
571 1 252 209 100 209
572 1 252 212 104 212
573 1 252.5 217 103 217
574 1 252.5 228 105 228
575 1 252 234 106 234
576 1 253 237 110 237
577 1 253 239 114 239
578 1 253.5 239 113 239
579 1 254.5 239 115 239
580 1 253.5 241 115 241
581 1 253 260 112 260
582 1 252.5 272 115 272
583 1 250.5 285 117 285
584 1 249.5 294 117 294
585 1 247.5 308 121 308
586 1 242.5 315 131 315
587 1 240 317 138 317
588 1 239 316 144 316
589 1 238.5 316 145 316
590 1 240 362 144 362
591 1 240.5 380 151 380
592 1 244 400 152 400
593 1 248.5 408 159 408
594 1 253.5 413 167 413
595 1 260 415 176 415
596 1 266 414 184 414
597 1 270 412 188 412
598 1 273 425 188 425
599 1 275 445 190 445
600 1 277 465 192 465
601 1 278.5 477 195 475
602 1 280.5 479 201 472
603 1 282 479 206 468
604 1 282.5 479 211 464
605 1 283.5 479 229 459
606 1 284.5 479 233 454
607 1 294 479 284 429
608 1 297.5 479 279 415
609 1 305.5 479 287 396
```

```
610 1 309.5 479 281 385
611 1 327 479 298 370
612 1 344 479 308 357
613 1 359 479 322 345
614 1 371.5 479 327 336
615 1 383 479 342 326
616 1 388.5 479 343 320
617 1 392.5 479 347 312
```

Appendix 3. Tracking file – Test 3.

```
3 1 167.5 478 311 302
3 2 587.5 356 103 355
4 1 166.5 478 311 306
4 2 583 351 112 350
5 1 166 479 312 308
5 2 578.5 344 121 343
6 1 166 479 308 307
6 2 575.5 340 127 340
7 1 164.5 479 301 307
7 2 572.5 338 133 338
8 1 164.5 479 293 306
8 2 570 336 138 335
9 1 164 479 280 308
9 2 568 335 142 333
10 1 162 479 268 312
10 2 567.5 333 143 331
11 1 161.5 479 253 319
11 2 567.5 332 141 329
12 1 160.5 479 247 327
12 2 568 332 140 329
13 1 158.5 478 231 336
13 2 568.5 331 139 328
14 1 154.5 479 207 354
14 2 568.5 332 141 328
15 1 152 478 204 363
15 2 568 332 142 326
16 1 150 478 186 375
16 2 568.5 333 141 324
17 1 148 478 184 388
17 2 568.5 334 139 328
18 1 146 479 190 396
18 2 568.5 335 141 333
19 1 145.5 479 267 421
19 2 568 335 142 329
20 1 139 479 270 432
20 2 568.5 335 139 332
21 1 131 479 258 438
21 2 568.5 335 141 330
22 1 129 479 258 445
22 2 568 334 142 332
23 1 132 479 264 451
23 2 567 333 144 333
24 1 125 479 246 458
24 2 555 333 168 333
25 1 122 479 244 461
25 2 542.5 331 193 331
26 1 121 479 242 462
26 2 536 330 206 330
27 1 120.5 479 241 463
27 2 532.5 330 211 330
28 1 119.5 479 239 464
28 2 530 330 214 330
29 1 118 479 236 465
29 2 525 331 208 331
30 1 115 479 228 468
30 2 519.5 333 195 333
31 1 114.5 479 223 471
31 2 515 333 188 333
```

```
32 1 113.5 479 217 473
32 2 511 333 180 333
33 1 112.5 479 211 478
33 2 507.5 333 177 332
34 1 112 467 208 467
34 2 504 333 170 332
35 1 112 456 200 456
35 2 502 333 170 332
36 1 111.5 429 197 429
36 2 499 339 160 339
37 1 111.5 416 191 416
37 2 491.5 345 175 344
38 1 111 406 182 406
38 2 482 361 168 360
39 1 111 401 178 401
39 2 475.5 379 167 379
40 1 110.5 399 173 399
40 2 472 390 166 390
41 1 110 397 174 397
41 2 471 397 166 397
42 1 109.5 390 173 390
42 2 470 401 166 401
43 1 108.5 380 175 380
43 2 466.5 404 175 404
44 1 106 365 172 365
44 2 462 407 178 407
45 1 106 354 174 354
45 2 457 411 170 411
46 1 106 320 176 320
46 2 454.5 412 165 412
47 1 107 309 168 309
47 2 451.5 413 165 413
48 1 109 304 162 304
48 2 443 417 174 417
49 1 112.5 303 157 303
49 2 432 421 182 421
50 1 113.5 302 151 302
50 2 425 421 188 421
51 1 117 301 148 301
51 2 418.5 422 189 422
52 1 120.5 298 139 298
52 2 412 421 188 421
53 1 124.5 290 135 290
53 2 405 420 186 420
54 1 128 281 134 281
54 2 391 421 206 421
55 1 131.5 267 133 267
55 2 378.5 420 211 420
56 1 136.5 248 135 248
56 2 364 421 212 421
57 1 139.5 236 133 236
57 2 354.5 422 205 422
58 1 142.5 232 129 232
58 2 346.5 422 195 422
59 1 144.5 231 127 231
59 2 339 426 178 426
60 1 144.5 231 125 231
60 2 334.5 430 173 430
61 1 146 231 122 231
61 2 330.5 431 169 431
62 1 147.5 233 123 233
```

```
62 2 328.5 433 177 433
63 1 147.5 231 121 231
63 2 325 432 184 432
64 1 149.5 228 121 228
64 2 319.5 430 189 430
65 1 150 224 120 224
65 2 309.5 430 199 430
66 1 150.5 217 113 217
66 2 293.5 428 197 428
67 1 150 210 108 210
67 2 277 439 196 439
68 1 151 194 100 194
68 2 260 452 184 446
69 1 153 184 96 184
69 2 247.5 462 169 451
70 1 156 179 90 179
70 2 238.5 462 157 450
71 1 159.5 177 91 177
71 2 236 459 146 452
72 1 161.5 178 89 177
72 2 233 453 148 448
73 1 159 178 68 177
73 2 231 449 152 447
74 1 160.5 174 65 173
74 2 219 440 182 440
75 2 213.5 435 189 435
76 2 201 431 214 431
77 2 196.5 432 217 432
78 2 187.5 429 227 429
79 2 177.5 427 249 427
80 2 169.5 427 253 427
80 3 221 144 74 144
81 2 157 431 264 431
81 3 229.5 149 77 149
82 2 150 433 268 433
82 3 235.5 151 81 151
83 2 139 430 272 430
83 3 243.5 151 85 151
84 2 132 429 264 429
84 3 249.5 151 87 151
85 2 124.5 433 247 433
85 3 257 152 96 152
86 2 115 432 230 432
86 3 262 150 100 150
87 2 105.5 429 211 429
87 3 264.5 148 101 147
88 2 99 429 198 429
88 3 267 146 96 145
89 2 98.5 431 197 431
89 3 272 139 82 138
90 2 98 435 196 435
90 3 279 132 66 130
91 2 98 439 196 437
91 3 288.5 127 59 127
92 2 98 438 196 435
92 3 297 131 56 131
93 2 97.5 436 195 433
93 3 304 133 66 133
94 2 97 435 194 432
94 3 314 133 86 133
95 2 96.5 436 193 434
```

```
95 3 323.5 131 101 131
96 2 96.5 436 193 435
96 3 327.5 129 105 129
97 2 95.5 437 191 437
97 3 330.5 130 105 130
98 2 93.5 438 187 438
98 3 332.5 132 107 132
99 2 93 436 186 436
99 3 334 132 106 132
100 2 92 434 184 434
100 3 336 132 98 132
101 2 91.5 427 183 427
101 3 340.5 133 89 133
102 2 91.5 412 183 412
102 3 359 133 64 133
103 2 91.5 399 183 399
103 3 371 132 60 132
104 2 94 367 188 367
104 3 387.5 132 89 132
105 2 95.5 356 183 356
105 3 394.5 132 93 132
106 2 98 351 180 351
106 3 396.5 136 91 136
107 2 100.5 348 179 348
107 3 398 138 92 138
108 2 104 348 178 348
108 3 399.5 139 89 138
109 2 107 345 174 344
109 3 406.5 140 85 139
110 2 107.5 341 173 340
110 3 411 140 82 139
111 2 105 327 166 327
111 3 424.5 144 87 144
112 2 103 314 162 314
112 3 433 152 92 152
113 2 103 297 160 297
113 3 439 161 94 161
114 2 103.5 284 159 284
114 3 442 165 96 165
115 2 104 277 158 277
115 3 443 168 94 168
116 2 106.5 270 161 270
116 3 439.5 170 75 170
117 2 111.5 268 165 267
117 3 442.5 179 89 179
118 2 113 267 162 266
118 3 444 184 90 184
119 2 117 267 160 266
119 3 444.5 186 87 186
120 2 120 268 158 267
120 3 444 186 90 186
121 2 124.5 267 151 266
121 3 444.5 186 91 186
122 2 129 266 144 265
122 3 447 187 94 187
123 2 130.5 264 145 263
123 3 448 187 92 187
124 2 132.5 258 143 257
124 3 450 188 96 188
125 2 136 252 134 252
125 3 451 199 92 199
```

```
126 2 138 247 132 247
126 3 453 207 98 207
127 2 139.5 243 133 243
127 3 455.5 214 99 214
128 2 143 235 138 235
128 3 457 219 96 219
129 2 148 223 140 223
129 3 460.5 226 99 225
130 2 153.5 216 135 216
130 3 465.5 237 101 237
131 2 158.5 213 131 213
131 3 471 244 108 244
132 2 160.5 212 131 212
132 3 476.5 247 119 247
133 2 162.5 212 127 212
133 3 480.5 249 125 249
134 2 163.5 213 129 213
134 3 483 250 130 250
135 2 165.5 213 127 213
135 3 486.5 249 135 249
136 2 167 214 122 214
136 3 489.5 247 135 247
137 2 167.5 213 117 213
137 3 493.5 247 135 247
138 2 166.5 213 113 213
138 3 495.5 254 133 254
139 2 167.5 206 87 206
139 3 502.5 283 125 283
140 2 169 201 82 201
140 3 506 302 130 302
141 2 170.5 196 87 196
141 3 508 319 134 319
142 2 172 191 86 191
142 3 509.5 326 143 326
143 2 172 188 88 188
143 3 510.5 329 145 329
144 2 175.5 187 97 187
144 3 513 331 150 331
145 2 180.5 188 109 188
145 3 516 331 152 331
146 2 185.5 189 113 189
146 3 519 330 154 329
147 2 191 192 116 192
147 3 523.5 354 151 354
148 2 195.5 194 115 194
148 3 528.5 379 153 379
149 2 201.5 198 105 198
149 3 537 406 164 406
150 2 204.5 199 103 199
150 3 543 425 176 425
151 2 207.5 201 97 201
151 3 545.5 431 187 431
152 2 210.5 201 95 201
152 3 546 431 186 431
153 2 211.5 201 93 201
153 3 547 434 184 434
154 2 212.5 201 91 201
154 3 548 435 182 435
155 2 214 201 90 201
155 3 549 434 180 434
156 2 218.5 200 85 200
```

```
156 3 549.5 432 179 432
157 2 221 200 84 200
157 3 545 429 188 429
158 2 233.5 200 109 200
158 3 532.5 428 213 428
159 2 246.5 200 123 200
159 3 521.5 431 235 431
160 2 257 200 130 200
160 3 515.5 430 247 430
161 2 268 203 126 202
161 3 492 432 254 432
162 2 275 205 120 204
162 3 475 431 240 431
163 2 279.5 207 115 206
163 3 460 431 220 431
164 2 283.5 207 111 206
164 3 451.5 432 191 432
165 2 287 207 104 206
165 3 442 436 174 436
166 2 294.5 208 91 207
166 3 429.5 438 153 438
167 2 305 211 106 210
167 3 418.5 441 157 441
168 2 305 213 80 212
168 3 407 447 160 447
169 2 307.5 218 79 217
169 3 397.5 454 165 454
170 2 307.5 219 77 218
170 3 389 455 166 455
171 2 312.5 210 71 209
171 3 376.5 456 167 455
172 2 316 206 66 205
172 3 368 453 166 451
173 2 320.5 205 65 204
173 3 351.5 451 157 451
174 3 341.5 447 157 447
175 3 334.5 441 157 441
176 3 331.5 438 153 438
177 3 333 438 144 438
178 3 329.5 439 155 439
179 3 323 439 160 439
180 3 316.5 440 161 440
181 2 381 230 84 230
181 3 300 440 170 440
182 2 390 231 80 231
182 3 279 440 180 440
183 2 398.5 231 89 231
183 3 268.5 439 183 439
184 2 406 232 108 232
184 3 262 439 182 439
185 2 415.5 232 113 232
185 3 257.5 439 183 439
186 2 419.5 232 117 231
186 3 255 437 180 437
187 2 422 232 118 231
187 3 251.5 437 175 437
188 2 421.5 232 117 232
188 3 246.5 436 169 436
189 2 420.5 231 111 231
189 3 242.5 432 165 432
190 2 418 231 110 231
```

```
190 3 240 425 162 425
191 2 417.5 231 111 231
191 3 238 419 160 419
192 2 417.5 232 109 232
192 3 232.5 396 155 396
193 2 417.5 232 113 231
193 3 229.5 378 151 378
194 2 419 232 122 231
194 3 224 344 148 344
195 2 421.5 231 125 230
195 3 219.5 333 139 333
196 2 423.5 225 129 224
196 3 216 328 136 325
197 2 424 217 126 216
197 3 212 326 132 324
198 2 423 212 126 211
198 3 209 327 134 326
199 2 422.5 208 119 207
199 3 205.5 325 133 324
200 2 420.5 205 113 204
200 3 202 321 134 320
201 2 420 203 110 202
201 3 199.5 313 139 312
202 2 418.5 203 111 203
202 3 195.5 306 141 306
203 2 419 203 112 203
203 3 191 300 138 300
204 2 418 203 108 203
204 3 185.5 283 141 283
205 2 416 204 112 204
205 3 180 260 140 260
206 2 415.5 203 113 203
206 3 175.5 252 135 252
207 2 415 200 114 200
207 3 173 249 132 249
208 2 416.5 195 109 195
208 3 171.5 247 129 247
209 2 414.5 191 99 190
209 3 169.5 248 125 247
210 2 414 189 92 189
210 3 168.5 247 119 247
211 2 414 185 86 185
211 3 167.5 243 115 243
212 2 414.5 182 79 182
212 3 166 240 112 240
213 2 414.5 180 75 180
213 3 166 231 110 231
214 2 415 175 74 175
214 3 165.5 219 111 219
215 2 414 172 72 172
215 3 165.5 206 111 206
216 2 414 170 70 170
216 3 165.5 200 109 200
217 2 415.5 169 65 169
217 3 166 194 110 194
218 2 414 168 72 168
218 3 165.5 191 107 191
219 2 411 169 84 169
219 3 163 190 104 190
220 2 410.5 171 85 171
220 3 162.5 189 101 189
```

```
221 2 412.5 174 79 174
221 3 160 186 106 186
222 2 411.5 176 85 176
222 3 161.5 182 95 182
223 2 414.5 179 77 179
223 3 159.5 179 101 179
224 2 414.5 181 79 181
224 3 158.5 175 103 175
225 2 415 183 76 183
225 3 160 170 98 170
226 2 411 187 84 187
226 3 159 162 98 162
227 2 410 189 86 189
227 3 157.5 159 101 159
228 2 409.5 192 75 192
228 3 159.5 158 103 158
229 2 409.5 193 71 193
229 3 164 157 106 157
230 2 408.5 193 73 193
230 3 169 157 100 157
231 2 404.5 193 71 192
231 3 172 157 98 157
232 2 401 193 72 192
232 3 174.5 157 95 157
233 2 397.5 193 77 193
233 3 176 158 96 158
234 2 393.5 193 81 193
234 3 178 157 90 157
235 2 390 193 82 193
235 3 179 154 88 154
236 2 382 195 92 195
236 3 180 148 84 148
237 2 376 199 98 199
237 3 180.5 143 83 143
238 2 363 208 122 208
238 3 183.5 143 79 143
239 2 358 214 120 214
239 3 186 142 78 142
240 2 349.5 227 109 227
240 3 188 142 78 141
241 2 344.5 232 107 232
241 3 189 142 72 141
242 2 340 234 104 234
242 3 190.5 143 71 142
243 2 339 235 102 234
243 3 191.5 143 69 142
244 2 330 234 118 233
244 3 191.5 143 65 142
245 2 326 235 108 234
245 3 192.5 143 63 142
246 2 314.5 251 95 251
246 3 193 141 62 140
247 2 291.5 272 97 272
247 3 192 142 54 141
248 2 282 284 96 284
248 3 192 142 52 141
249 2 268 290 104 290
249 3 190.5 142 59 142
250 2 258.5 293 99 293
250 3 189 142 60 142
251 2 250 293 98 293
```

```
251 3 189.5 142 59 142
252 2 242 293 102 293
253 2 231 292 110 292
254 2 220.5 300 113 300
255 2 214.5 310 117 310
256 2 209.5 317 125 317
257 2 203.5 329 133 329
258 2 185 342 180 342
259 2 179.5 350 179 350
260 2 174.5 353 181 353
261 2 174.5 354 177 354
262 2 171.5 356 173 356
262 3 221.5 158 77 158
263 2 160.5 356 179 355
263 3 224.5 156 91 156
264 2 138.5 358 199 357
264 3 232 161 114 161
265 2 124.5 360 199 359
265 3 240 166 112 166
266 2 121 367 200 366
266 3 243.5 169 113 169
267 2 121 376 206 375
267 3 246.5 179 105 179
268 2 126 379 220 379
268 3 248 183 102 183
269 2 130.5 383 219 383
269 3 251 184 98 183
270 2 133.5 387 213 386
270 3 255.5 185 95 185
271 2 135.5 389 211 387
271 3 260 186 104 186
272 2 138 389 206 386
272 3 266.5 187 111 187
273 2 141 388 200 385
273 3 271.5 190 109 190
274 2 143.5 387 197 386
274 3 276 202 108 202
275 2 146 386 196 386
275 3 280 214 108 214
276 2 148 393 192 393
276 3 284.5 220 111 220
277 2 150 401 190 401
277 3 287 225 110 225
278 2 152 407 186 407
278 3 289.5 229 111 229
279 2 152.5 410 189 410
279 3 291 234 110 234
280 2 152.5 414 187 414
280 3 293.5 238 109 236
281 2 154 416 188 416
281 3 296 239 112 235
282 2 156.5 423 175 423
282 3 304.5 240 113 239
283 2 157.5 427 173 427
283 3 310.5 239 113 238
284 2 157.5 427 175 427
284 3 317 239 116 238
285 2 159 426 182 426
285 3 323.5 240 113 239
286 2 162 427 192 427
286 3 334.5 251 125 251
```

```
287 2 164 427 194 427
287 3 342.5 259 127 259
288 2 164.5 426 195 426
288 3 345.5 264 121 264
289 2 164.5 425 193 425
289 3 354.5 278 113 278
290 2 161 426 190 426
290 3 360 283 112 283
291 2 158.5 426 187 426
291 3 364 286 114 286
292 2 153 426 186 426
292 3 374 287 122 287
293 2 150 427 188 427
293 3 386.5 287 121 287
294 2 148 426 188 426
294 3 394 286 124 286
295 2 146.5 425 191 425
295 3 406 296 130 296
296 2 145.5 427 193 427
296 3 419 322 132 322
297 2 145 428 194 428
297 3 426 335 136 335
298 2 144 428 196 428
298 3 431 344 136 344
299 2 144 427 198 427
299 3 435 352 138 351
300 2 142 427 200 427
300 3 441 362 140 362
301 2 140 427 202 427
301 3 444 369 142 369
302 2 136.5 427 201 427
302 3 450 373 144 373
303 2 133.5 428 199 428
303 3 457 372 142 372
304 2 130.5 427 203 427
304 3 470 373 136 373
305 2 128 423 204 423
305 3 477.5 374 137 374
306 2 120 395 204 395
306 3 493 374 148 374
307 2 115 375 200 375
307 3 503 375 152 375
308 2 113 364 196 364
308 3 511 377 158 377
309 2 113.5 360 193 360
309 3 517.5 388 157 388
310 2 115.5 358 187 358
310 3 525 413 164 413
311 2 119 358 182 358
311 3 537 433 190 433
312 2 122 359 178 359
312 3 541.5 438 195 438
313 2 124 359 178 359
313 3 542.5 441 193 441
314 2 128 353 172 353
314 3 545.5 441 187 441
315 2 131 341 168 341
315 3 546 442 186 442
316 2 133 328 168 328
316 3 545.5 442 187 442
317 2 136 315 166 315
```

```
317 3 546 446 186 446
318 2 140.5 292 165 292
318 3 552.5 446 173 443
319 2 142 278 162 278
319 3 556.5 445 165 439
320 2 145 273 152 273
320 3 557.5 443 163 434
321 2 146.5 272 147 272
321 3 557 439 164 430
322 2 147.5 272 145 272
322 3 558 441 162 432
323 2 147 272 144 272
323 3 555.5 433 167 421
324 2 147 272 144 272
324 3 556.5 439 165 429
325 2 147 272 144 272
325 3 554.5 437 169 427
326 2 145 264 144 264
326 3 547.5 439 183 432
327 2 142 255 144 255
327 3 534 429 210 423
328 2 140 247 138 247
328 3 535 431 208 427
329 2 138.5 232 135 232
329 3 526.5 434 225 431
330 2 138 223 130 223
330 3 522.5 435 233 432
331 2 139 219 124 219
331 3 525 436 226 433
332 2 140 217 124 217
332 3 517.5 433 239 432
333 2 140.5 216 119 216
333 3 508.5 431 217 431
334 2 141 216 118 216
334 3 505.5 425 209 425
335 2 141.5 216 115 216
335 3 498.5 425 197 425
336 2 143 216 112 215
336 3 493.5 419 179 419
337 2 143 217 114 216
337 3 487.5 416 169 416
338 2 144 216 110 215
338 3 472.5 422 173 422
339 2 145 215 110 214
339 3 461 422 180 422
340 2 146 210 108 209
340 3 445.5 439 177 439
341 2 149.5 195 111 195
341 3 408.5 437 193 437
342 2 152.5 186 109 186
342 3 388 433 194 433
343 2 155 176 106 176
343 3 369.5 430 193 430
344 2 156.5 173 107 173
344 3 362.5 427 191 427
345 2 162.5 172 99 172
345 3 357.5 430 185 430
346 2 165 172 104 172
346 3 355 430 176 430
347 2 165.5 172 103 172
347 3 354 430 168 430
```

```
348 2 168 173 98 173
348 3 346.5 430 159 430
349 2 170 172 90 172
349 3 317 430 150 430
350 2 170.5 172 85 172
350 3 293.5 430 141 430
351 2 170 170 78 170
351 3 271.5 433 141 433
352 2 169.5 167 77 167
352 3 261.5 434 147 434
353 2 171 162 76 162
353 3 253.5 436 159 436
354 2 175 156 60 156
354 3 246.5 438 171 438
355 2 176.5 153 57 153
355 3 241.5 439 185 439
356 3 237.5 438 193 438
357 3 233.5 438 203 438
358 3 229 437 210 437
359 3 222 433 214 433
360 3 212.5 430 223 430
361 3 201 421 204 420
362 3 179.5 419 183 417
363 3 159 419 176 417
364 3 140 420 154 418
364 6 218.5 148 101 148
365 3 132.5 417 159 414
365 6 220 147 100 147
366 3 125 411 176 409
366 6 224.5 147 83 146
367 3 118.5 409 189 407
367 6 227 147 86 146
368 3 111 410 202 409
368 6 230.5 148 81 147
369 3 107.5 410 207 410
369 6 233.5 149 75 148
370 3 105.5 409 211 409
370 6 234.5 149 71 148
371 3 104.5 408 209 408
371 6 241.5 149 69 148
372 3 101.5 409 203 409
372 6 247.5 149 73 148
373 3 95 405 190 405
373 6 261.5 150 91 149
374 3 91 403 182 403
374 6 272 150 100 149
375 3 88 402 176 402
375 6 279 151 100 150
376 3 87.5 403 175 403
376 6 283 151 100 150
377 3 87.5 403 175 403
377 6 284 151 100 150
378 3 87 406 174 406
378 6 285 149 98 148
379 3 87.5 408 175 408
379 6 286.5 147 97 146
380 3 90 405 180 405
380 6 294 141 78 140
381 3 90.5 402 181 402
381 6 301.5 138 69 137
382 3 91.5 394 183 394
```

```
382 6 308 139 62 138
383 3 92 385 182 385
383 6 315 143 64 142
384 3 97 366 174 366
384 6 330 141 86 141
385 3 101 347 170 347
385 6 338 139 90 139
386 3 105 328 164 328
386 6 344.5 139 97 139
387 3 108.5 312 159 312
387 6 349 141 102 141
388 3 110.5 306 151 306
388 6 350.5 143 103 143
389 3 112.5 303 145 303
389 6 353 144 104 144
390 3 113.5 302 141 302
390 6 354.5 144 97 144
391 3 115 300 140 300
391 6 357 144 92 144
392 3 117 298 138 298
392 6 360 144 84 144
393 3 119 296 138 296
393 6 364 144 78 143
394 3 119.5 293 139 293
394 6 368 144 72 143
395 3 122 284 142 284
395 6 381.5 144 59 144
396 3 123.5 277 141 277
396 6 388.5 144 59 144
397 3 125 265 140 265
397 6 396.5 143 75 143
398 3 126.5 246 139 246
398 6 404 143 88 143
399 3 131.5 236 145 236
399 6 410 145 92 145
400 3 135 235 144 235
400 6 411 146 90 146
401 3 137.5 234 139 234
401 6 412 149 92 149
402 3 139.5 234 139 234
402 6 413.5 150 93 150
403 3 141 234 136 234
403 6 414.5 151 93 151
404 3 142.5 234 129 234
404 6 420 151 106 151
405 3 145 234 130 234
405 6 422 151 98 151
406 3 146 234 128 234
406 6 425.5 151 101 151
407 3 148 231 120 231
407 6 428 151 92 151
408 3 150.5 226 121 226
408 6 434 151 88 151
409 3 152.5 221 117 221
409 6 442 155 92 155
410 3 156 213 112 213
410 6 450 161 94 161
411 3 160.5 204 109 204
411 6 453.5 169 89 169
412 3 162.5 197 109 197
412 6 456 174 96 174
```

```
413 3 163 186 104 186
413 6 456.5 176 97 176
414 3 166 180 102 180
414 6 456 178 98 178
415 3 173 179 92 179
415 6 456.5 182 97 182
416 3 176.5 178 89 178
416 6 456 185 100 185
417 3 178.5 178 87 178
417 6 455.5 185 101 185
418 3 181.5 179 81 179
418 6 455 186 104 186
419 3 184.5 178 77 178
419 6 455.5 186 107 186
420 3 185 178 78 178
420 6 455.5 185 107 185
421 3 186 175 76 175
421 6 455.5 185 107 185
422 3 186 172 70 172
422 6 455.5 185 105 185
423 3 186 170 68 170
423 6 455 185 102 185
424 3 185 170 68 170
424 6 456.5 195 99 195
425 3 183.5 170 65 170
425 6 459.5 204 101 204
426 3 183 167 70 167
426 6 461 211 108 211
427 3 184.5 165 73 165
427 6 463 216 108 216
428 3 184.5 166 77 166
428 6 466.5 225 107 225
429 3 184 166 80 166
429 6 469.5 232 109 232
430 3 183 167 82 167
430 6 473.5 235 113 235
431 3 182 166 88 166
431 6 478.5 236 119 236
432 3 181 166 90 166
432 6 483 236 126 236
433 3 180.5 167 93 167
433 6 486.5 236 131 235
434 3 180 166 94 166
434 6 489 235 130 234
435 3 179.5 167 93 167
435 6 488.5 236 129 235
436 3 180.5 168 93 168
436 6 491 236 134 235
437 3 181 170 94 170
437 6 493.5 244 135 244
438 3 182 170 92 170
438 6 495.5 251 135 251
439 3 185 170 90 170
439 6 497 265 136 265
440 3 187.5 170 89 170
440 6 498.5 273 135 273
441 3 188.5 170 89 170
441 6 500.5 281 137 281
442 3 191.5 171 107 171
442 6 504.5 291 141 291
443 3 198 171 122 171
```

```
443 6 507.5 297 141 297
444 3 203 171 128 171
444 6 509 299 140 299
445 3 210 175 132 175
445 6 514 299 148 299
446 3 215 179 128 179
446 6 518 300 148 300
447 3 218.5 183 121 183
447 6 519.5 301 149 301
448 3 221.5 190 117 189
448 6 517.5 301 139 301
449 3 226 193 112 193
449 6 523 301 156 301
450 3 227.5 194 107 194
450 6 525.5 312 153 312
451 3 230 195 104 195
451 6 528.5 323 159 323
452 3 236 195 104 195
452 6 534.5 345 171 345
453 3 239.5 195 105 195
453 6 537.5 357 173 357
454 3 244.5 196 107 196
454 6 540 364 178 364
455 3 248 200 110 200
455 6 541.5 366 181 366
456 3 251.5 211 111 211
456 6 546.5 367 183 366
457 3 257.5 222 115 222
457 6 551 368 176 367
458 3 263 228 116 228
458 6 554 366 170 364
459 3 267.5 235 119 235
459 6 555.5 367 167 365
460 3 271 239 120 238
460 6 556 366 166 364
461 3 275.5 242 119 241
461 6 557 368 164 367
462 3 279.5 243 115 242
462 6 557 366 164 365
463 3 282.5 244 117 243
463 6 557 366 164 366
464 3 285.5 244 117 243
464 6 557 367 164 367
465 3 292 244 112 244
465 6 558 367 162 367
466 3 297 243 116 243
466 6 558.5 366 161 365
467 3 302 242 114 242
467 6 559.5 368 159 367
468 3 306 242 112 242
468 6 559 370 160 370
469 3 309 242 114 242
469 6 559 370 160 370
470 3 314 242 112 242
470 6 555.5 367 167 367
471 3 319.5 241 109 241
471 6 545.5 365 187 365
472 3 323 241 106 241
472 6 540 363 198 363
473 3 330 240 102 239
473 6 527.5 365 223 365
```

```
474 3 333.5 240 103 239
474 6 522.5 363 233 362
475 3 338 240 106 239
475 6 516.5 363 245 362
476 3 341 240 106 239
476 6 510 362 252 361
477 3 346.5 240 107 239
477 6 500.5 364 249 362
478 3 356 243 106 243
478 6 485 365 224 365
479 3 361 251 108 251
479 6 475.5 366 221 364
480 3 364.5 258 107 258
480 6 465.5 367 197 367
481 3 367.5 263 107 263
481 6 460 368 190 368
482 3 366.5 269 87 269
482 6 449.5 366 167 366
483 3 366.5 263 85 263
483 6 443 366 154 366
484 3 366.5 240 69 240
484 6 433 371 132 371
485 3 363.5 233 55 233
485 6 418.5 373 151 373
486 3 366 231 50 231
486 6 393 372 156 372
487 6 381.5 383 159 381
488 6 363 383 160 379
489 3 423.5 287 95 287
489 6 348.5 380 169 378
490 3 436.5 294 107 294
490 6 342.5 382 189 382
491 3 443 297 112 297
491 6 335.5 384 165 384
492 3 448.5 302 111 302
492 6 326.5 384 163 384
493 3 454 316 114 316
493 6 313.5 384 159 384
494 3 458 324 122 324
494 6 305 384 154 384
495 3 461 329 130 329
495 6 296.5 385 153 385
496 3 471.5 347 151 347
496 6 264 385 138 385
497 3 483 354 168 354
497 6 241 384 150 384
498 3 494.5 357 183 357
498 6 230 385 154 385
499 3 504 357 186 356
499 6 221.5 386 169 386
500 3 508 357 184 356
500 6 217.5 387 177 387
501 3 534 383 174 383
501 6 202 384 196 383
502 3 545 399 174 399
502 6 193.5 379 185 379
503 3 549.5 405 179 405
503 6 186.5 371 181 371
504 3 550.5 408 177 408
504 6 180.5 368 175 368
505 3 551 408 176 408
```

```
505 6 163.5 367 159 367
506 3 551 410 176 409
506 6 152 366 150 366
507 3 550.5 412 177 410
507 6 131 361 172 361
508 3 551 415 176 414
508 6 122 356 180 356
509 3 551 420 176 419
509 6 119 353 190 353
510 3 550.5 430 177 430
510 6 118 353 196 353
511 3 550 438 178 438
511 6 118 352 204 352
512 3 550 443 178 443
512 6 110 352 214 352
513 3 549.5 444 179 444
513 6 103 352 204 352
514 3 549.5 446 179 446
514 6 99.5 352 199 352
515 3 550 445 178 445
515 6 96 350 192 350
516 3 550.5 444 177 444
516 6 95 347 190 347
517 3 553 444 172 444
517 6 92.5 343 185 343
518 3 555 441 168 441
518 6 90 339 180 339
519 3 562 441 150 441
519 6 89.5 339 179 339
520 3 557.5 443 163 443
520 6 87.5 338 175 338
521 3 554 445 170 445
521 6 88 338 176 338
522 3 551.5 445 175 445
522 6 89 338 178 338
523 3 547.5 445 183 445
523 6 90 338 180 338
524 3 544 451 190 451
524 6 91 339 182 339
525 3 526.5 451 225 451
525 6 92.5 338 185 338
526 3 521 449 236 449
526 6 96.5 336 179 336
527 3 510.5 451 231 451
527 6 100 331 174 331
528 3 499.5 448 197 448
528 6 105 321 164 321
529 3 493 449 184 449
529 6 108 313 162 312
530 3 490 449 184 449
530 6 111 307 160 307
531 3 478 447 156 447
531 6 118 288 168 288
532 3 469 443 148 443
532 6 122 278 166 278
533 3 457 422 150 422
533 6 126 272 166 272
534 3 444.5 409 143 409
534 6 129 271 164 271
535 3 433.5 404 139 404
535 6 131 271 162 271
```

```
536 3 426.5 398 135 398
536 6 132 270 160 270
537 3 419.5 393 135 393
537 6 132.5 270 159 270
538 3 399 389 166 389
538 6 133 271 152 271
539 3 389.5 385 161 385
539 6 133.5 272 151 272
540 3 368 380 184 380
540 6 131 268 146 268
541 3 359.5 378 179 378
541 6 130.5 265 145 265
542 3 354 376 184 376
542 6 133 258 132 258
543 3 351.5 375 179 375
543 6 134 253 132 253
544 3 347 372 172 372
544 6 135 241 130 241
545 3 335 367 170 367
545 6 137.5 225 129 225
546 3 327 364 166 364
546 6 138.5 216 125 216
547 3 319 363 164 363
547 6 138.5 213 123 213
548 3 309.5 364 157 364
548 6 140.5 211 125 211
549 3 265 368 144 368
549 6 145 210 110 210
550 3 247.5 369 151 369
550 6 147.5 210 111 210
551 3 238.5 367 165 367
551 6 150 208 108 208
552 3 235 366 166 366
552 6 145 206 88 206
553 3 226.5 361 173 361
553 6 146 203 84 203
554 3 216.5 357 165 357
555 3 208 354 158 354
556 3 196 344 152 344
557 3 179 334 138 334
558 3 166.5 332 127 332
559 3 160 325 136 325
560 3 158.5 319 135 319
561 3 158 317 136 317
562 3 155 312 142 311
563 3 152 311 146 310
564 3 149 310 152 309
564 8 211.5 166 81 165
565 3 146.5 310 157 309
565 8 211 169 90 168
566 3 141.5 307 171 306
566 8 213.5 170 93 170
567 3 136.5 304 179 303
567 8 215 170 90 170
568 3 133 302 174 302
568 8 214 169 84 169
569 3 130 299 180 299
569 8 217.5 168 85 168
570 3 118 295 182 295
570 8 220.5 158 81 157
571 3 108.5 292 181 292
```

```
571 8 222 153 76 153
572 3 98.5 289 177 289
572 8 224.5 149 73 149
573 3 89.5 288 171 288
573 8 226 145 72 144
574 3 84.5 288 167 288
574 8 227.5 143 69 142
575 3 77.5 288 153 288
575 8 235.5 145 83 144
576 3 74 288 148 288
576 8 241 144 86 142
577 3 73 288 146 288
577 8 247 145 94 145
578 3 73 289 146 289
578 8 252.5 143 99 143
579 3 73.5 289 147 289
579 8 257.5 142 105 142
580 3 73.5 289 147 289
580 8 263 141 112 141
581 3 74 290 146 290
581 8 270 142 108 142
582 3 74 290 146 290
582 8 274.5 143 105 143
583 3 75 288 150 288
583 8 279 145 104 145
584 3 75.5 287 151 287
584 8 281.5 146 103 146
585 3 75.5 288 151 288
585 8 283.5 145 99 144
586 3 74 288 148 288
586 8 287 145 86 144
587 3 72.5 289 145 289
587 8 294.5 145 75 144
588 3 72 288 144 288
588 8 302 145 74 145
589 3 74.5 285 149 285
589 8 311.5 145 73 145
590 3 75 281 150 281
590 8 317 145 72 145
591 3 75.5 276 151 276
591 8 322 145 76 145
592 3 80.5 263 161 263
592 8 333.5 145 93 145
593 3 81 258 162 258
593 8 338 144 94 144
594 3 81.5 257 163 256
594 8 340 145 94 145
595 3 82.5 257 163 256
595 8 342 145 94 145
596 3 83 258 164 257
596 8 343.5 144 89 144
597 3 83 257 166 256
597 8 346.5 143 83 143
598 3 82.5 257 165 256
598 8 351.5 141 75 141
599 3 83 257 164 257
599 8 366 142 70 142
600 3 83 258 164 258
600 8 375 143 72 143
601 3 83.5 257 161 257
601 8 380.5 145 73 145
```

```
602 3 86 258 154 258
602 8 386.5 145 79 145
603 3 87.5 259 153 259
603 8 390 146 80 146
604 3 89.5 260 149 260
604 8 392 148 78 148
605 3 92.5 257 141 257
605 8 389 148 70 148
606 3 94 255 134 254
606 8 389 149 72 149
607 3 98 252 130 252
607 8 391 149 70 149
608 3 105 250 134 250
608 8 394 148 72 148
609 3 117.5 248 153 248
609 8 399.5 147 65 147
610 3 132.5 245 165 245
610 8 405.5 148 61 148
611 3 142.5 243 167 243
611 8 409.5 148 61 148
612 3 148 242 168 242
612 8 410.5 149 61 149
613 3 158 242 160 242
613 8 413 148 78 147
614 3 166 244 154 244
614 8 418.5 148 87 148
615 3 171.5 245 151 245
615 8 422 148 92 148
616 3 178 245 138 245
616 8 423 148 106 148
617 3 181.5 245 137 245
617 8 424 148 108 148
618 3 187 245 124 245
618 8 425.5 148 113 148
619 3 190.5 246 119 246
619 8 426 148 114 148
620 3 204.5 246 99 246
620 8 429.5 147 93 147
621 3 221.5 246 123 246
621 8 432 145 100 145
622 3 241.5 245 135 245
622 8 427 145 100 145
623 3 254.5 245 141 245
623 8 421.5 144 93 144
624 3 267.5 245 145 245
624 8 422.5 144 109 144
625 3 273.5 245 149 245
625 8 420.5 145 109 145
626 3 278.5 243 139 243
626 8 417 145 102 145
627 3 283 242 134 242
627 8 413.5 145 97 145
628 3 291.5 240 119 240
628 8 410 146 92 146
629 3 304 238 88 238
629 8 406.5 146 83 146
630 3 319 238 74 238
630 8 403.5 146 79 146
631 3 329 243 70 243
631 8 402.5 144 77 144
632 3 339 244 76 244
```

```
632 8 401 144 62 144
633 3 365 240 146 240
633 8 399.5 143 59 143
634 3 378 242 152 242
635 3 382 245 146 245
636 3 387 246 152 246
636 8 343 140 64 140
637 3 392.5 246 135 246
637 8 340.5 141 85 141
638 3 396.5 247 129 247
638 8 342 143 90 143
639 3 406 247 128 246
639 8 343.5 145 99 145
640 3 426 246 124 245
640 8 341.5 145 89 145
641 3 446.5 246 107 245
641 8 337 145 78 145
642 3 459.5 246 107 245
642 8 318.5 145 63 145
643 3 464.5 246 107 245
643 8 307 145 68 145
644 3 471 247 122 247
644 8 298 144 76 144
645 3 471.5 246 113 246
645 8 289.5 144 91 144
646 3 471.5 246 115 246
646 8 285 144 96 143
647 3 473 248 120 248
647 8 283 143 98 142
648 3 476 252 130 252
648 8 282.5 145 95 144
649 3 485 253 146 253
649 8 283 147 94 146
650 3 489.5 254 143 254
650 8 282 147 94 146
651 3 497 255 142 255
651 8 280.5 149 89 148
652 3 506.5 255 141 255
652 8 275 149 92 149
653 3 515.5 256 135 256
653 8 269 150 84 150
654 3 521.5 257 139 257
654 8 263.5 150 81 150
655 3 524.5 256 143 256
655 8 261.5 152 79 152
656 3 525.5 256 145 256
656 8 259 155 78 155
657 3 528 256 152 256
657 8 254.5 166 75 166
658 3 529.5 258 153 258
658 8 251.5 175 75 175
659 3 530.5 260 157 260
659 8 249 179 82 179
660 3 529 267 162 267
660 8 247.5 183 83 183
661 3 524.5 275 171 275
661 8 247.5 184 85 184
662 3 521.5 279 173 279
662 8 245.5 183 89 183
663 3 518 284 174 284
663 8 238.5 184 107 184
```

```
664 3 514.5 288 173 288
664 8 235.5 184 107 184
665 3 513 292 170 292
665 8 234 184 108 184
666 3 510 295 164 295
666 8 234 183 104 183
667 3 508.5 296 161 296
667 8 234.5 183 103 183
668 3 504.5 296 151 296
668 8 237 183 94 183
669 3 502 296 148 296
669 8 236 184 100 184
670 3 492.5 296 155 296
670 8 236 194 96 194
671 3 484.5 296 161 296
671 8 234 205 100 205
672 3 480.5 297 161 297
672 8 233.5 210 103 210
673 3 478 300 162 300
673 8 233 211 106 211
674 3 475.5 319 155 319
674 8 233.5 215 101 215
675 3 473.5 341 153 341
675 8 232.5 220 107 220
676 3 472 359 154 359
676 8 232 221 110 221
677 3 470.5 369 157 369
677 8 232 223 112 223
678 3 470 376 158 376
678 8 233 222 112 221
679 3 468.5 381 159 381
679 8 232.5 222 117 221
680 3 468 387 160 387
680 8 233 222 114 221
681 3 467 392 166 392
681 8 235.5 222 109 221
682 3 466.5 394 173 394
682 8 229.5 223 127 223
683 3 465.5 395 177 395
683 8 227 223 128 223
684 3 463 395 186 395
684 8 225.5 223 127 223
685 3 462 395 188 395
685 8 226.5 223 129 223
686 3 459.5 398 187 398
686 8 231.5 224 133 224
687 3 458.5 398 187 398
687 8 235.5 225 133 224
688 3 457 400 188 400
688 8 240 226 142 225
689 3 452.5 412 193 412
689 8 242.5 225 145 225
690 3 434.5 457 203 456
690 8 249.5 228 137 228
691 3 428.5 474 195 470
691 8 254.5 231 131 231
692 3 423 479 196 473
692 8 258 231 126 231
693 3 419 479 196 472
693 8 260.5 231 121 230
694 3 412 479 206 469
```

```
694 8 263.5 230 113 229
695 3 407.5 479 213 466
695 8 266 230 112 229
696 3 403 479 218 464
696 8 268 230 106 229
697 3 393.5 479 243 455
697 8 278 235 108 235
698 3 387 479 266 440
698 8 288.5 239 99 239
699 3 384 479 278 419
699 8 295.5 234 89 234
700 3 385 479 284 405
700 8 300 231 86 231
701 3 388 479 294 393
701 8 310 231 96 231
702 3 391.5 479 295 384
702 8 314.5 231 93 230
703 3 400.5 479 301 372
703 8 316 231 76 230
704 3 408.5 479 303 365
704 8 317.5 231 81 229
705 3 419.5 479 321 351
705 8 323 232 94 231
706 3 423.5 479 323 340
706 8 328.5 233 99 232
707 3 425 479 330 329
707 8 337.5 232 109 231
```

Appendix 4. Tracking file – Test 4.

```
3 1 136 479 264 336
3 2 540 308 144 307
4 1 127.5 479 247 350
4 2 544.5 304 151 303
5 1 116.5 479 227 364
5 2 547 303 154 301
6 1 108 479 216 372
6 2 548.5 302 157 300
7 1 95.5 479 187 381
7 2 548.5 300 157 298
8 1 96 479 192 391
8 2 548 298 154 297
9 1 93.5 479 187 400
9 2 545.5 297 147 296
10 1 95 479 190 406
10 2 543.5 298 145 297
11 1 97 479 194 411
11 2 542.5 298 145 298
12 1 100.5 478 201 419
12 2 542 298 144 298
13 1 102.5 478 205 428
13 2 542 297 142 297
14 1 103.5 479 207 434
14 2 541 298 140 298
15 1 110.5 479 221 436
15 2 541.5 299 143 299
16 1 117 479 234 438
16 2 542.5 301 147 301
17 1 120.5 479 241 439
17 2 543 303 146 303
18 1 123 479 246 441
18 2 543.5 304 147 304
19 1 125 478 250 450
19 2 545 304 152 304
20 1 126 478 252 454
20 2 546.5 305 153 305
21 1 125 479 246 459
21 2 548 306 156 306
22 1 121.5 479 241 463
22 2 550.5 306 159 306
23 1 119.5 479 239 465
23 2 552 307 162 307
24 1 117.5 479 235 465
24 2 553 307 164 307
25 1 114.5 479 229 466
25 2 554.5 307 165 307
26 1 112.5 479 225 469
26 2 556 306 164 306
27 1 110 479 220 473
27 2 560 306 156 306
28 1 108.5 479 217 475
28 2 563.5 308 149 308
29 1 107 478 214 475
29 2 567 306 144 306
30 1 108 478 216 477
30 2 570.5 307 137 307
31 1 106.5 474 213 474
31 2 571 306 136 306
```

```
32 1 104 458 208 458
32 2 571 313 136 313
33 1 100.5 448 201 448
33 2 571 315 136 315
34 1 91.5 428 179 428
34 2 573 314 132 314
35 1 87 414 174 414
35 2 576.5 319 119 319
36 1 85.5 410 171 410
36 2 577.5 326 123 326
37 1 84.5 409 169 409
37 2 574.5 338 129 338
38 1 84 408 168 408
38 2 568.5 348 141 346
39 1 84 406 168 406
39 2 560.5 357 157 354
40 1 83.5 405 167 404
40 2 559 363 160 359
41 1 83.5 401 167 401
41 2 559 368 160 358
42 1 83.5 396 167 396
42 2 559 371 160 364
43 1 84 392 168 392
43 2 558.5 373 159 367
44 1 85 384 170 384
44 2 558.5 371 161 365
45 1 89.5 360 179 360
45 2 554 375 170 372
46 1 93 337 174 337
46 2 548.5 373 181 372
47 1 97 321 170 321
47 2 546 374 186 373
48 1 99 315 166 315
48 2 544 381 188 379
49 1 102.5 313 161 313
49 2 542 402 188 399
50 1 104.5 309 157 309
50 2 539.5 422 189 421
51 1 108.5 306 151 306
51 2 538 442 196 442
52 1 110.5 304 147 304
52 2 536.5 450 201 450
53 1 113.5 300 147 300
53 2 533.5 454 211 454
54 1 117 294 148 294
54 2 528.5 455 211 455
55 1 118 290 146 290
55 2 526 456 208 456
56 1 118.5 281 143 281
56 2 521.5 457 197 457
57 1 119 276 140 276
57 2 518.5 458 191 458
58 1 121.5 244 137 244
58 2 503 453 196 453
59 1 123.5 234 129 234
59 2 492.5 456 197 456
60 1 131 231 130 231
60 2 462 456 226 456
61 1 136.5 229 123 229
61 2 430 452 236 452
62 1 139.5 228 119 228
```

```
62 2 413 451 232 451
63 1 143.5 224 113 224
63 2 398 449 212 449
64 1 146.5 218 111 218
64 2 388.5 453 197 453
65 1 149.5 214 107 214
65 2 384.5 451 197 451
66 1 152.5 206 105 206
66 2 377.5 445 187 445
67 1 157 191 110 191
67 2 364.5 440 165 440
68 1 162.5 181 105 181
68 2 340.5 446 155 446
69 1 167 178 102 178
69 2 324.5 450 155 450
70 1 166 177 82 177
70 2 301.5 447 151 445
71 1 169.5 176 73 176
71 2 277 451 164 451
72 1 172 175 72 175
72 2 263.5 451 183 451
73 1 175 171 68 171
73 2 256.5 450 191 450
74 2 238.5 451 225 451
75 2 228.5 453 239 453
76 2 207.5 456 249 456
77 2 183.5 460 255 460
78 2 152.5 460 251 460
78 3 221.5 127 67 127
79 2 136 460 248 460
79 3 229.5 138 77 138
80 2 118.5 465 229 465
80 3 244.5 142 103 142
81 2 116 466 232 466
81 3 256.5 142 123 142
82 2 116 466 232 466
82 3 262 139 126 139
83 2 116 463 232 463
83 3 266 138 122 138
84 2 117 463 234 459
84 3 270 138 112 137
85 2 111 464 222 462
85 3 279 137 92 136
86 2 105 464 210 463
86 3 292 137 84 137
87 2 100.5 461 201 460
87 3 305.5 136 79 136
88 2 92.5 450 185 448
88 3 314 135 72 135
89 2 80.5 444 159 442
89 3 316 135 64 134
90 2 75 440 150 437
90 3 316.5 134 59 133
91 2 72.5 432 145 426
91 3 327.5 133 91 133
92 2 73 426 146 396
92 3 335.5 133 103 133
93 2 71.5 411 143 369
93 3 344 133 116 133
94 2 71 406 142 360
94 3 347.5 133 113 133
```

```
95 2 69.5 401 139 353
95 3 355.5 131 97 131
96 2 69 398 138 348
96 3 359.5 131 91 131
97 2 67.5 395 135 345
97 3 362.5 131 91 131
98 2 64.5 391 127 341
98 3 371.5 132 89 132
99 2 62.5 389 125 339
99 3 376.5 133 83 133
100 2 63.5 387 127 334
100 3 378.5 136 57 136
101 2 63.5 390 127 335
101 3 381.5 142 53 142
102 2 66 391 132 340
102 3 385 149 52 149
103 2 68.5 393 137 346
103 3 386 154 60 154
104 2 71.5 391 143 351
104 3 387.5 155 63 155
105 2 73 387 144 372
105 3 391 156 78 156
106 2 73.5 380 145 376
106 3 390.5 156 79 156
107 2 74 370 148 369
107 3 392 156 82 156
108 2 74 358 148 358
108 3 387 157 82 157
109 2 74.5 350 149 350
109 3 386 162 84 162
110 2 75 338 150 338
110 3 383.5 169 85 169
111 2 76.5 324 153 324
111 3 381 179 90 179
112 2 77 316 154 316
112 3 381.5 183 97 183
113 2 77.5 308 155 308
113 3 380 188 98 188
114 2 76.5 305 153 305
114 3 376.5 197 101 197
115 2 76 304 152 304
115 3 374 201 102 201
116 2 76 304 152 304
116 3 372 203 102 203
117 2 75.5 305 151 305
117 3 371 203 100 203
118 2 75 303 150 303
118 3 369 202 98 202
119 2 75 287 150 287
119 3 356.5 207 101 207
120 2 74 269 148 269
120 3 341.5 215 113 215
121 2 74.5 254 149 254
121 3 333 222 112 222
122 2 77 250 154 250
122 3 327 235 106 235
123 2 79 248 158 248
123 3 323.5 242 107 242
124 2 81 248 154 248
124 3 319 244 104 244
125 2 83.5 248 147 248
```

```
125 3 309 245 106 245
126 2 85.5 248 143 248
126 3 300 245 106 245
127 2 90 246 140 246
127 3 286 245 114 245
128 2 93.5 243 139 243
128 3 279.5 249 111 249
129 2 97 236 132 236
129 3 266.5 261 123 261
130 2 99 232 130 232
130 3 260.5 278 121 278
131 2 101.5 223 125 223
131 3 249 293 136 293
132 2 105 210 126 210
132 3 240 301 144 301
133 2 107.5 195 127 195
133 3 234.5 315 147 315
134 2 109.5 187 129 187
134 3 231.5 322 141 322
135 2 111.5 186 123 186
135 3 228.5 324 127 324
136 2 112.5 186 117 186
136 3 224.5 325 121 325
137 2 115 186 106 186
137 3 216 324 108 323
138 2 115 186 96 186
138 3 206.5 326 109 325
139 2 116.5 186 89 186
139 3 201 324 106 323
140 2 118.5 183 85 183
140 3 190 325 112 324
141 3 179 325 126 324
142 3 165 340 152 340
143 3 159 353 162 353
144 3 156.5 358 169 358
145 3 145.5 378 183 377
146 3 134 386 200 385
147 3 119 389 214 387
147 4 135 83 68 83
148 3 108.5 388 217 384
148 4 138.5 121 81 121
149 3 100 389 200 383
149 4 139 140 98 140
150 3 95 389 190 384
150 4 139.5 148 111 147
151 3 91 392 182 387
151 4 140.5 152 117 151
152 3 89 395 178 390
152 4 141 165 110 165
153 3 91 403 182 398
153 4 142 171 118 171
154 3 92 412 184 409
154 4 144.5 172 107 172
155 3 92 410 184 408
155 4 145 172 100 172
156 3 93.5 414 187 414
156 4 150 167 76 167
157 3 93.5 413 187 413
157 4 152 158 74 158
158 3 94 417 188 416
158 4 150.5 150 79 150
```

```
159 3 97.5 435 195 434
159 4 150 145 80 145
160 3 100 444 200 441
160 4 151 145 74 145
161 3 103.5 444 207 441
161 4 155 150 64 150
162 3 114.5 451 211 447
162 4 156.5 156 63 156
163 3 138 456 238 452
164 3 158 457 238 452
165 3 180.5 464 225 459
166 3 196.5 468 203 461
167 3 207.5 470 195 463
168 3 214 471 182 462
169 3 224 467 166 458
170 3 238.5 469 161 462
171 3 258.5 468 183 462
171 5 125.5 214 99 214
172 3 288 467 208 463
172 5 124 216 98 216
173 3 305 467 212 464
173 5 121 216 100 216
174 3 317.5 467 211 462
174 5 118 216 106 216
175 3 330.5 466 209 460
175 5 113.5 217 113 217
176 3 339.5 465 211 459
176 5 114 232 104 231
177 3 355 453 154 451
177 5 108 251 122 251
178 3 383.5 457 127 457
178 5 96.5 267 147 267
179 3 399.5 448 125 447
179 5 91.5 272 155 272
180 3 417.5 437 129 437
180 5 86.5 273 161 273
181 3 428.5 434 133 434
181 5 83.5 274 161 274
182 3 441.5 426 147 426
182 5 81 274 160 274
183 3 457.5 414 175 414
183 5 77 275 154 275
184 3 470.5 409 193 409
184 5 73 275 146 275
185 3 479 411 200 411
185 5 69.5 278 139 278
186 3 503 417 226 417
186 5 62 299 116 299
187 3 519 417 232 417
187 5 59.5 314 119 314
188 3 523.5 416 231 416
188 5 60 324 120 324
189 3 528 412 222 412
189 5 60 330 120 330
190 3 534 412 210 412
190 5 58.5 332 117 332
191 3 537 413 204 413
191 5 58 332 116 332
192 3 536.5 412 193 412
192 5 56.5 331 113 331
193 3 531 411 186 411
```

```
193 5 54 330 108 330
194 3 518 404 200 404
194 5 52.5 333 105 333
195 3 508 400 206 400
195 5 51.5 346 103 346
196 3 500.5 398 207 398
196 5 51.5 359 103 358
197 3 493 397 204 397
197 5 52 369 104 366
198 3 484 399 200 399
198 5 52.5 381 105 373
199 3 475 399 192 399
199 5 51.5 387 103 375
200 3 468.5 399 191 399
200 5 52 390 104 375
201 3 460 390 166 390
201 5 52.5 393 105 370
202 3 455.5 380 143 380
202 5 52.5 393 105 362
203 3 448 374 130 374
203 5 52 393 104 358
204 3 440 373 134 373
204 5 52 392 104 352
205 3 432 371 138 371
205 5 52 390 104 347
206 3 400.5 373 149 373
206 5 56.5 390 113 335
207 3 378.5 375 151 375
207 5 65 390 130 330
208 3 363.5 375 147 375
208 5 74.5 389 149 327
209 3 354 372 150 372
209 5 82.5 390 165 331
210 3 346 366 150 366
210 5 86 394 168 368
211 3 327 348 132 348
211 5 92.5 398 185 362
212 3 316.5 332 127 332
212 5 93 397 182 385
213 3 300 320 120 319
213 5 94.5 398 187 395
214 3 287.5 307 109 305
214 5 94 397 186 397
215 3 279.5 305 107 303
215 5 93.5 394 187 392
216 3 273.5 306 105 305
216 5 94.5 398 189 398
217 3 268.5 305 105 305
217 5 95 399 188 399
218 3 256.5 303 131 303
218 5 97 408 192 408
219 3 251 302 138 302
219 5 102.5 416 205 416
220 3 248.5 298 137 298
220 5 113.5 424 223 424
221 3 243 294 146 294
221 5 140 436 252 436
222 3 236 287 140 287
222 5 162 442 254 442
223 3 230 279 114 279
223 5 181 445 246 445
```

```
224 3 226.5 262 89 262
224 5 196 447 236 447
225 3 219.5 255 85 255
225 5 206 448 224 448
226 5 215 446 204 446
227 5 220.5 447 197 447
228 3 152.5 204 101 204
228 5 229 445 170 445
229 3 150 224 110 224
229 5 244 446 184 446
230 3 149 253 114 253
230 5 261 443 200 443
231 3 148 263 116 263
231 5 272.5 445 203 445
232 3 149 268 128 268
232 5 286 442 202 442
233 3 144 265 124 265
233 5 300.5 444 201 444
234 3 138.5 255 117 255
234 5 307.5 443 189 443
235 3 132.5 245 117 245
235 5 313 442 170 442
236 3 130.5 241 115 241
236 5 316.5 441 167 441
237 3 128.5 237 117 237
237 5 320.5 439 159 439
238 3 126.5 232 121 232
238 5 328.5 438 151 438
239 3 124.5 231 119 231
239 5 337 437 152 437
240 3 123.5 231 117 231
240 5 345 415 158 415
241 3 125 229 114 229
241 5 356.5 381 159 381
242 3 126 229 112 229
242 5 362.5 368 155 368
243 3 129 229 110 229
243 5 370 361 152 361
244 3 132 227 112 227
244 5 376 361 150 361
245 3 134.5 224 111 224
245 5 380 361 148 361
246 3 136.5 222 111 222
246 5 382.5 360 149 360
247 3 138 219 110 219
247 5 385.5 360 149 360
248 3 140.5 216 103 216
248 5 393.5 358 147 358
249 3 142 214 100 214
249 5 401 350 148 350
250 3 145 214 98 214
250 5 408.5 342 153 342
251 3 152.5 214 107 214
251 5 418.5 314 163 314
252 3 158 212 114 212
252 5 425.5 299 159 299
253 3 161 212 118 212
253 5 429 294 154 294
254 3 163.5 211 123 211
254 5 434.5 293 151 293
255 3 168.5 212 123 212
```

```
255 5 439.5 293 145 293
256 3 170.5 212 121 212
256 5 441.5 293 141 293
257 3 172 213 120 213
257 5 442.5 293 139 293
258 3 174 214 120 214
258 5 440.5 291 131 291
259 3 175.5 215 119 215
259 5 439 285 128 285
260 3 177 215 114 215
260 5 438.5 277 125 277
261 3 180 215 108 215
261 5 437.5 267 125 266
262 3 182.5 215 103 215
262 5 437.5 256 125 255
263 3 187 215 94 215
263 5 437 236 126 235
264 3 188 215 90 215
264 5 437.5 228 119 227
265 3 193 214 90 214
265 5 437 225 118 224
266 3 203 214 104 214
266 5 439 225 112 224
267 3 218 214 124 214
267 5 439.5 225 111 224
268 3 233.5 214 145 214
268 5 439.5 225 113 224
269 3 244.5 215 137 215
269 5 440 224 116 223
270 3 252.5 215 131 215
270 5 440 217 116 216
271 3 256 215 128 215
271 5 439.5 211 115 211
272 3 260.5 215 119 215
272 5 440 206 116 206
273 3 274.5 216 81 216
273 5 439 189 110 189
274 3 284.5 225 79 225
274 5 437 180 102 180
275 3 297 227 90 227
275 5 433 177 100 177
276 3 316 232 116 232
276 5 425.5 176 105 176
277 3 328 233 126 233
277 5 420.5 175 107 175
278 3 337.5 231 135 231
278 5 415.5 176 107 176
279 3 343 231 138 231
279 5 413 176 104 176
280 3 345 232 140 232
280 5 412.5 177 99 177
281 3 347.5 233 141 233
281 5 413 176 86 176
282 3 356.5 233 115 233
282 5 413.5 166 57 166
283 3 370.5 233 111 232
283 5 412.5 162 53 162
284 3 380 234 100 233
285 3 394.5 234 109 234
286 3 402.5 234 133 234
287 3 409 233 130 233
```

```
288 3 419.5 233 113 233
289 3 423 233 108 233
290 3 423 234 104 233
290 5 358 153 80 153
291 3 423 234 104 233
291 5 341.5 144 43 144
292 3 426.5 234 101 233
292 5 349 150 78 150
293 3 429.5 235 101 234
293 5 350 152 86 152
294 3 436 238 92 238
294 5 350 152 80 152
295 3 442 236 90 236
295 5 349 152 84 152
296 3 451.5 246 79 246
296 5 343.5 152 89 152
297 3 458 250 74 250
297 5 337.5 151 93 151
298 3 463 253 80 253
298 5 333 152 102 152
299 3 465.5 255 83 255
299 5 332.5 154 101 154
300 3 468.5 258 87 258
300 5 340.5 156 71 156
301 3 483 262 124 261
301 5 333 160 88 160
302 3 491 264 136 263
302 5 327.5 164 89 164
303 3 496 264 140 263
303 5 324.5 169 87 169
304 3 501.5 265 147 264
304 5 322.5 174 99 174
305 3 502 265 154 264
305 5 318.5 176 103 176
306 3 499.5 265 153 264
306 5 307 178 84 178
307 3 497 266 148 265
307 5 306.5 178 101 178
308 3 494.5 268 145 267
308 5 304 183 100 183
309 3 492.5 268 145 267
309 5 304 190 100 190
310 3 489.5 268 139 268
310 5 301.5 203 95 203
311 3 483.5 267 135 267
311 5 300 219 90 219
312 3 480.5 267 135 267
312 5 295 231 100 231
313 3 477 267 132 267
313 5 290 236 104 236
314 3 475 266 130 266
314 5 286.5 237 111 237
315 3 473 267 130 267
315 5 279.5 239 125 239
316 3 471 267 126 267
316 5 275 239 128 239
317 3 469.5 266 125 266
317 5 268.5 238 137 238
318 3 468.5 266 123 266
318 5 264 240 138 240
319 3 467 266 122 266
```

```
319 5 261.5 244 141 244
320 3 464.5 261 119 261
320 5 257 262 132 262
321 3 463 255 120 255
321 5 257.5 275 133 275
322 3 458 245 114 245
322 5 259.5 288 133 288
323 3 457.5 240 119 240
323 5 261.5 295 135 295
324 3 453.5 218 119 218
324 5 265 299 138 299
325 3 451 207 114 207
325 5 267 300 142 300
326 3 448 204 114 204
326 5 268.5 300 151 300
327 3 446.5 203 115 203
327 5 268.5 300 153 300
328 3 442 201 104 201
328 5 269.5 311 155 311
329 3 438 199 98 199
329 5 269.5 332 153 332
330 3 434.5 193 95 193
330 5 269.5 347 153 347
331 3 432.5 187 95 187
331 5 268.5 355 151 355
332 3 430.5 178 99 178
332 5 266 362 150 362
333 3 429 167 96 167
333 5 262.5 368 149 368
334 3 425.5 161 91 161
334 5 255 371 162 371
335 3 424.5 159 89 159
335 5 252 372 162 372
336 3 422.5 158 85 158
336 5 247.5 373 167 372
337 3 419 159 80 159
337 5 242 371 174 370
338 3 417.5 159 81 159
338 5 238.5 371 175 370
339 3 414.5 159 79 159
339 5 233 373 174 372
340 3 413.5 160 77 160
340 5 231 372 172 371
341 3 409.5 159 71 159
341 5 216 385 164 385
342 3 406.5 160 69 160
342 5 201 398 174 398
343 3 405 159 66 159
343 5 195.5 413 175 413
344 3 405 156 62 156
344 5 186 417 204 417
345 3 406 153 58 153
345 5 179 414 214 414
346 3 406.5 152 55 152
346 5 176 413 216 413
347 3 407.5 152 55 152
347 5 173.5 416 221 416
348 3 407.5 152 53 152
348 5 154.5 415 261 415
349 3 408 152 54 152
349 5 136.5 415 259 415
```

```
350 3 406.5 153 61 153
350 5 126 419 246 419
351 3 402.5 154 73 154
351 5 115.5 420 227 420
352 3 400 155 76 155
352 5 108 420 216 420
353 3 396 156 84 156
353 5 94.5 417 187 417
354 3 394 157 88 157
354 5 93 418 186 418
355 3 391.5 158 93 158
355 5 93 420 186 420
356 3 390 158 98 158
356 5 93 418 186 418
357 3 389.5 162 97 162
357 5 92.5 418 185 417
358 3 390 164 98 164
358 5 93 417 186 415
359 3 393 164 92 164
359 5 90.5 415 181 412
360 3 394.5 165 91 165
360 5 88.5 414 177 411
361 3 395.5 165 91 165
361 5 79 411 154 409
362 3 391.5 165 95 165
362 5 68 397 134 395
363 3 386 169 102 169
363 5 65.5 392 131 391
364 3 382 175 104 175
364 5 65 390 130 390
365 3 378 182 102 182
365 5 66.5 386 133 386
366 3 376.5 186 105 186
366 5 67.5 384 135 384
367 3 374 194 100 194
367 5 69.5 384 139 384
368 3 372 202 96 202
368 5 70.5 385 141 385
369 3 370 206 96 206
369 5 71 383 142 383
370 3 364 209 96 208
370 5 69.5 377 139 377
371 3 357.5 209 101 209
371 5 68.5 365 137 361
372 3 352 209 104 209
372 5 66.5 353 133 349
373 3 346 208 110 208
373 5 67 329 134 327
374 3 342 214 110 214
374 5 67 311 134 310
375 3 335 234 112 234
375 5 67 306 134 306
376 3 331.5 245 115 245
376 5 68 305 136 305
377 3 329 250 118 250
377 5 71 305 142 305
378 3 327 253 120 253
378 5 72.5 306 145 306
379 3 325 258 120 258
379 5 73.5 305 147 305
380 3 322 268 118 268
```

```
380 5 74 306 148 306
381 3 320.5 274 119 274
381 5 74.5 306 149 306
382 3 319 275 118 275
382 5 76 302 152 302
383 3 317 276 116 276
383 5 77 295 154 295
384 3 315.5 276 115 276
384 5 77 288 154 288
385 3 312.5 275 113 275
385 5 79 277 154 277
386 3 308.5 275 111 275
386 5 83 260 154 260
387 3 301 291 120 291
387 5 88 242 156 242
388 3 295 300 128 300
388 5 91 236 150 235
389 3 287 312 126 311
389 5 95 234 144 233
390 3 282 328 118 328
390 5 100 233 142 232
391 3 275.5 340 119 340
391 5 103.5 234 141 233
392 3 266.5 346 129 346
392 5 106.5 233 137 232
393 3 260.5 347 133 347
393 5 109.5 233 135 232
394 3 248 348 154 348
394 5 110 225 130 224
395 3 237 350 166 349
395 5 107.5 218 119 217
396 3 229 360 172 358
396 5 112 202 104 202
397 3 224.5 390 171 389
397 5 117 191 100 191
398 3 221.5 426 173 426
398 5 119 185 96 185
399 3 220.5 452 181 451
399 5 131 169 94 169
400 3 222 467 184 466
400 5 135 165 92 165
401 3 226.5 474 205 470
401 5 140.5 137 79 137
402 3 228.5 477 213 471
402 5 144 128 64 128
403 3 229.5 479 221 469
403 5 147.5 118 59 118
404 3 230 479 224 468
405 3 231 479 234 464
406 3 231.5 479 249 451
407 3 231.5 479 251 442
408 3 231 479 258 431
409 3 225.5 479 287 412
410 3 222 479 302 397
411 3 220 479 314 384
411 9 225 158 114 158
412 3 223 479 328 373
412 9 237 157 112 157
413 3 225.5 479 331 363
413 9 242.5 157 111 157
414 3 237.5 479 353 347
```

```
414 9 248 157 108 157
415 3 246.5 479 351 339
415 9 249 157 108 157
416 3 260 479 358 329
416 9 252.5 155 99 154
417 3 286.5 479 367 318
417 9 260 150 82 150
418 3 301 479 364 312
418 9 264 147 76 147
419 3 316 479 370 305
419 9 269.5 146 75 145
```