Hes·so VALAIS WALLIS

Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

BACHELOR THESIS

Tavernier Martin

VRRT - Video-based Realtime Race Timing

Informatique et Systèmes de communication (ISC)

ISC-ID-25-02

University of Applied Sciences and Arts Western Switzerland
Faculty of Engineering and Architecture
Haute École d'Ingénierie de Sion

Tavernier Martin

VRRT - Video-based Realtime Race Timing

Thesis submitted to the Bachelor degree programme
Informatique et Systèmes de communication (ISC) – Data engineering major
Haute École d'Ingénierie de Sion

Supervisor: **DR PIERRE-ANDRÉ MUDRY**

Expert: **DR JULIEN PILET**

Submitted on: July 25, 2025

Abstract

This bachelor's thesis was commissioned by SP-Timing. The objective was to create a system capable of timing participants in a race using a video-only solution. The timing accuracy had to be at least 1/10 second, the system had to be easy to deploy, as autonomous as possible.

The solution implemented in this work is composed as follows: first, two YOLOv11 models (one pre-trained and one fine-tuned) are responsible for detecting and locating people and bib numbers in an image. Once the bibs have been located on the image, an optical character recognition system (PaddleOCR) is tasked with reading the contents of the bib and extracting the participant's bib number. Next, a monocular depth estimation model, in our case DepthAnythingV2 Small, is used to estimate the distance of each person moving towards the camera, and compare it with the distance to the finish line. With this depth estimate, precise timing of each person can be established.

The current system can process an incoming video stream at 15 frames per second on a laptop (Lenovo Legion Y540-IRH15[1] - CPU : Intel i5 9300H[2] - GPU: Nvidia 1660Ti Mobile GPU[3].

The final system was used to time 269 runners at the Trail du Gramont held on 14.06.2025. Of the 269 runners, 188 were timed completely autonomously. Of the 81 remaining runners, 31 did not have a visible bib number, bringing the total number of runners down to 50. Of the remaining participants, the system failed to read the bib numbers of 47 participants, one participant had an incorrectly read bib number, and two were not detected at all by the system. This brings the critical errors count to 3 of 269. These results demonstrate the potential for professional use of a system of this type.

Keywords : engineering, vision, machine learning, deep learning, timing, real-time

[1] https://psref.lenovo.com/syspool/Sys/PDF/Legion/Lenovo_Legion_Y540_15IRH/Lenovo_Legion_Y540_15IRH_Spec.PDF

[2] https://www.intel.fr/content/www/fr/fr/products/sku/191075/intel-core-i59300h-processor-8m-cache-up-to-4-10-ghz/specifications.html

[3] https://www.nvidia.com/en-eu/geforce/gaming-laptops/gtx-1660-ti/

Résumé

Ce travail de bachelor prend place suite à la demande de l'entreprise SP-Timing. L'objectif était de créer un système capable de chronométrer des participants lors d'une course avec une solution basée sur de la vidéo uniquement. La précision du chronométrage doit atteindre au minimum 1/10 seconde, le système devait être facilement déployable, et avoir le plus grand niveau d'autonomie possible

La solution implémentée dans ce travail se compose comme suit : en premier, deux models YOLOv11 (un préentraîné et un ajusté) ont la responsabilité de détecter et localiser les personnes ainsi que les numéros de dossard sur une image. Une fois les dossard localisés sur l'image, un système de reconnaissance de caractères optique (PaddleOCR est chargé de lire le contenu du dossard et d'en extraire le numéro de dossard du participant. Ensuite, un modèle d'estimation de profondeur monoculaire, dans notre cas DepthAnythingV2 Small, est utilisé afin d'estimer la distance des chaque personne se déplaçant vers la caméra, et de la comparer à celle de la ligne d'arrivée. Avec cette estimation de profondeur, un chronométrage précis de chaque personne peut être établi.

Le système actuel arrive à traiter un flux vidéo entrant à 15 images par seconde sur un ordinateur portable (Lenovo Legion Y540-15IRH[4] - CPU : Intel i5 9300H[5] - GPU : Nvidia 1660Ti Mobile GPU[6]).

Le système final a été utilisé pour chronométrer 269 coureurs lors du trail du Gramont qui a eu lieu 14.06.2025. Sur les 269 coureurs, 188 ont pu être chronométrés en totale autonomie. Sur les 81 coureurs restants, 31 n'avaient pas un numéro de dossard visible, cela ramène le nombre de coureurs à 50. Sur les participants restants, le système n'a pas réussi à lire le numéro de dossard de 47 participants, un participant a eu un numéro de dossard erroné, et deux n'ont pas été détecté du tout par le système. cela amène donc le nombre d'erreurs critiques à 3 sur 269. Ces résultats démontrent la possibilité d'utilisation professionelle d'un système de ce type.

Keywords : engineering, vision, machine learning, deep learning, timing, real-time

[4] https://psref.lenovo.com/syspool/Sys/PDF/Legion/Lenovo_Legion_Y540_15IRH/Lenovo_Legion_Y540_15IRH_Spec.PDF

[5] https://www.intel.fr/content/www/fr/fr/products/sku/191075/intel-core-i59300h-processor-8m-cache-up-to-4-10-ghz/specifications.html

[6] https://www.nvidia.com/en-eu/geforce/gaming-laptops/gtx-1660-ti/

Acknowledgements

This work could not have taken place without the helps of the people around me.

Sebastien, who was here every day at the HES, I hope I were as motivating as you were for me.

My family, who supported my studies from the start, encouraged me to give the best of myself, and helped me in many ways during the long years of my studies.

And lastly Marta, who is a model I try to reach in motivation, courage and dedication.

Table of Contents

Chapter 1 – Introduction

This bachelor thesis is the result from a request from the enterprise SP-Timing. SP-Timing currently offers multiple solutions for the timing of multiple types of race (ski, trail, run, triathlon…). This project aim to research and develop a prototype for acquiring video from the finish line of a race, and determine automatically the finish time with a precision of a tenth of a second. The system need to be as cheap as possible, easy to install and reliable.

Currently, SP-Timing mainly uses a chip based solution, that consists of a chip and an (or multiples) antenna. The antenna is located at the finish line, and it detects the chip located on the bibs. The newly created system would be here to complete the currently used one, or as a standalone solution. This means that the system cannot rely on the existing solution to work.

The system would be placed in a way that ensures a good visibility of the finish line, but not too far from it in order to have enough details to ensure precision.

1.1 – Objectives

The objectives stated from the enterprise SP-Timing are the following :

- Literature review in the field (video based race timing)
- Propose and realize a solution for video acquisition from one or two cameras, allowing if possible an easy deployment on the field
- Propose, realize, validate and estimate the pricing of a race timing solution in real time based on video streams
- Test the system in controlled environment and on the field and document it

1.2 – Functionalities

From the objectives described above, the following functionalities were extracted :

- Detect persons and their position in a picture
- Identify, track and re-identify persons in a picture
- Detect bib and their position in a picture
- Read a bib number from a cropped picture
- Create a depth map from a picture
- Process if and when a person has crossed the finish line

1.3 – Constrains

There are multiples constrains that the system must take into account.

To start, the system must overcome some problems that occurs due to the usage of video. The most obvious one is the occlusion problem. This problem is unsolvable on its own, but some technics can be used to mitigate its effects.

Then, they are constrains added by the form of the race. The camera cannot be positioned too high above the ground, as it should be easy to deploy, and there is no structure present on every race. The finish line structure can change every time.

Chapter 2 – Existing solutions

In the race timing field, there is many existing solutions, with varying levels of precisions, costs and ease of use. But by restricting the solution to video based one greatly reduces the possibilities.

2.1 – Photofinish

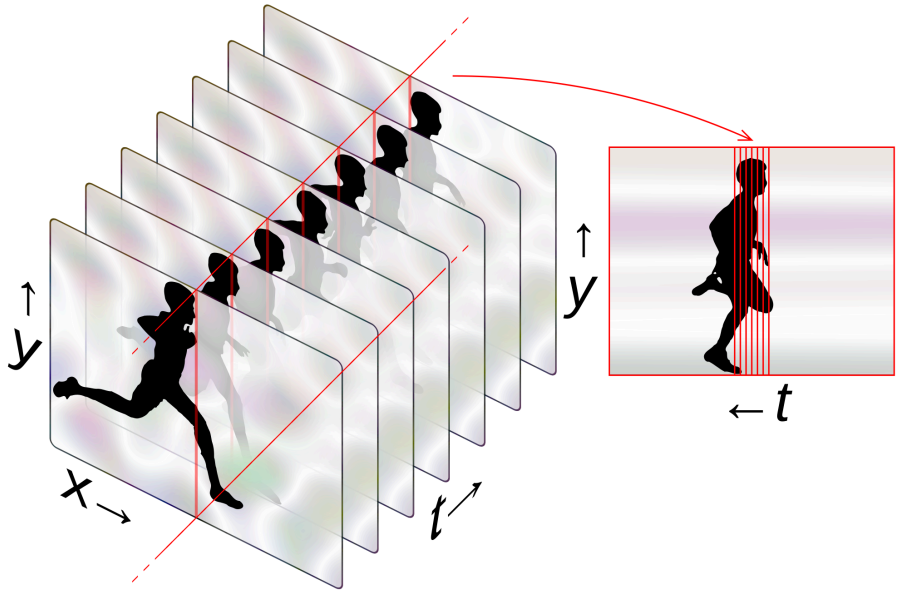


Figure 1 - Photofinish system - Source : Wikipedia

This system consist of a camera placed perpendicularly to the race finish line. It is a special camera capturing only a single vertical line of pixel at a time. This allow to replace the x-axis of the picture by the time, which allows for a precision dependant on the frame rate of the capture device.

This system is mainly used in races where each contestant have a predefined line in which to run, and with a limited number of participants. It is very precise ($1/1000_{th}$ of a second or less)[7].

A solution based on this could work well, but it would have some clear disadvantages :

- This system would need a second forward facing camera for bib identification in order to have a fully automatic solution
- On the type of races that SP-Timing currently works with, there is no fixed race line, people can pass the line as they wish. The system would need a flexibility that it does not have in order to work correctly.

2.2 – Non-professional solutions

There is a lot of non-professional solutions available for parts of this project. But everyone of them has either a less difficult problem (fixed runner path, single person at a time), or a less complete solution (no timing available, only bib detection). So those solutions are not meeting all the condition cited previously, but we cant take them as inspiration, to see what technology we could use for each part, when applicable.

[7] https://finishlynx.com/finishlynx-fully-automatic-timing-systems/

Chapter 3 – Methodology

This section will describe in details all the inner workings of the system, and the informations leading to the choice for its design.

3.1 – System architecture

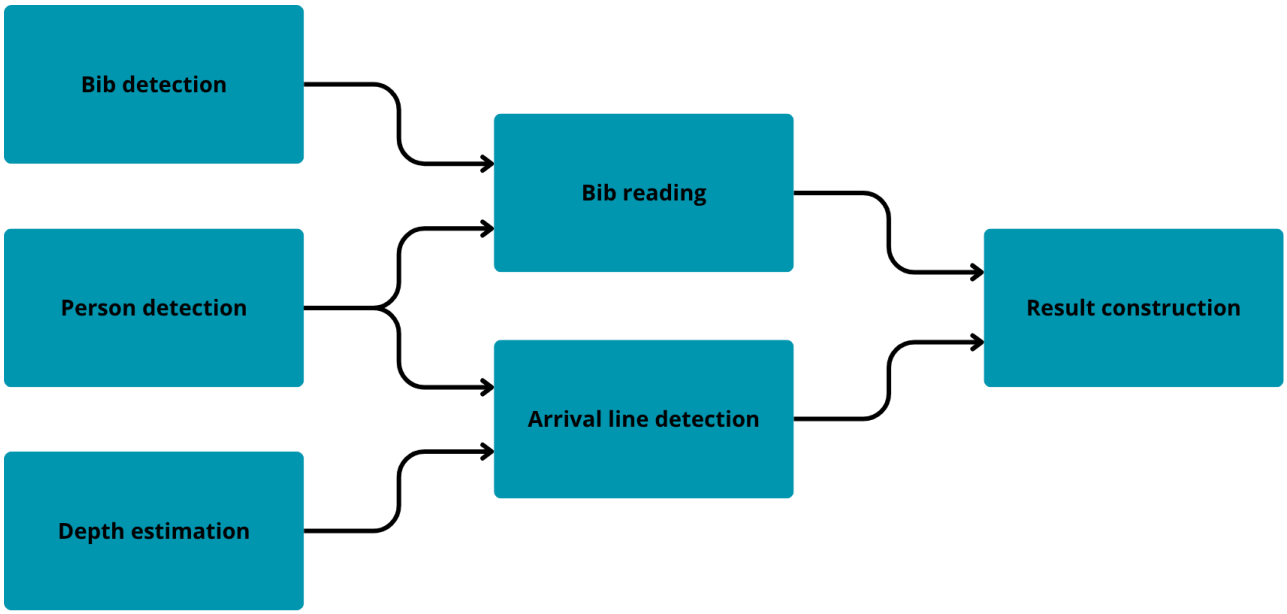


Figure 2 - System block diagram

As shown in the Figure 2, the system is composed of blocks. A granular system like this one allows for an easier technology swap if needed. For example the inner workings of the depth estimation block does not affects the other blocks, as long as the output stays the same. This enable the system to change and adapt to new technology, ensuring an up to date solution.

The Figure 2 shows the current state of the system, some changes discussed in Future work would imply a change in this graph.

The different blocks present in the Figure 2 and their respective input/output will be detailed below.

With all those blocks, every functionalities required to fulfill the requirements can be done.

3.2 – Person detection and tracking

The goal of this block is to detect persons present on a image, give each of the detection a unique Identifier (ID), track their position across multiple frames, and if possible, re-identify them after occlusion. This will allow the system to track every person in front of the camera, and associate data gathered by other blocks to the ID generated.

Figure 3 - Person detection block

The Figure 3 shows the input and outputs required for this part. As input, this block expect a colored pictures (of shape (y, x, 3), x and y being the dimension of the picture). This block needs to output detection boxes, with an ID per box. This means that this block needs to have identification, tracking and re-identification capabilities.

3.3 – Detection

Person detection is a well-studied problem with many solutions as this is a very active research topic. This implies that they are many different solutions to this problems, with varying accuracy results, inference speed, ease of use…

As this is a very competitive space, new models are released every month, with ever increasing accuracies and results. But those were not the most important metrics, as this task would not be the hardest in this project. The model can be just accurate enough for our task, but it needs to be fast. So the inference performance was a big factor in the choice of the solution for this task.

The solutions that were considered were :
- Ultralytics You Only Look Once (YOLO) models [1]
- Efficient DETR [2]
- RT-DETR [3]

The final choice was made to use the Ultralytics YOLO [1] implementation for multiples reasons. First, the inference implementation is very easy, only a few lines of code are needed. There is many models available (YOLOv5 to YOLOv11) with multiples model sizes (nano, small, base, large, extra-large) for varying tradeoffs between performance and accuracy. Lastly, the inference performance of the YOLO models are the best available for now, so it fits perfectly for our task. For the choice of the resolution at which to operate the model, as it is pre-trained with a 640x640 resolution, it is the one used for this project. Another one could be use, either to stretch out performances even more, but as this part of the solution is not the heaviest, the benefits vs costs of training this model to another resolution is out of scope for this Bachelor thesis.

3.4 – Tracking

Object (or person) tracking across multiple frames is a problem that can be decomposed in two main stages :
- Tracking
- Identification and re-identification

The tracking is the part that create a trace for each detection, and try to match it between frames to keep an history per ID. But this has a flaw, as if the trace is lost for too many frames, simple tracking is not sufficient.

Then comes the identification/re-identification, which generate a unique descriptor feature for each detection, and try to match each new detections with existing descriptor. In practice, this solution fixes the trace loss problem. As when the trace appears again, it will be matched with its descriptor.

They are some existing solutions for this problem, but as we need to keep the performance overhead low, the choices are pretty limited.

The Ultralytics YOLOv11 implementation comes with some trackers pre-implemented for us. Namely :

- BoT-SORT
- ByteTrack

So those two are the mains trackers on which I targeted my research. BoT-SORT is more computationally intensive, but with better results in the Re-ID task[8], so this is the one used in this project.

[8] As presented in https://github.com/NirAharon/BoT-SORT

Figure 4 - Tracking result

The Figure 4 depicts a typical visualization of tracking result. We can see that an ID and a trace is associated with every detection. This result will be sufficient for our system for now, but it is a target for future work.

3.5 – Bib detection

The goal of this system was to detect and locate every bib in a picture, and output each of their positions and size as boxes, in order to associate a bib to a participant and read its content.

Figure 5 - Person detection block

The Figure 5 depicts the input and output of this block. As input, a frame in color is expected (of dimension (y, x, 3), with x and y being the width and height of the picture). This part will output detection boxes for the bib detected in the pictures.

3.5.1 – Pretrained model

To detect bib, the use of a pre-trained model was quickly dismissed as the number publicly available pre-trained model were small, and none of them were using recent enough technology to be competitive. So the choice was quickly made to fine-tune a model based on some datasets found online. The choice was made to fine tune a YOLO model.

3.5.2 – Datasets

After some research, some datasets have been considered :

- https://people.csail.mit.edu/talidekel/RBNR.html
- https://universe.roboflow.com/gohar-w4jjb/bib_detect
- https://universe.roboflow.com/vrrt/bib-number-labeling-3z7gm-wekvr

After a quick analyze, the first dataset was quickly dismissed as it contained not enough pictures, with a label format that was not compatible with the YOLO model, and it would have taken too much time to convert them.

The second and third one are very similar, their quality are not that good, but as they are the only ones publicly available (and there is not enough time to label a dataset by hand).

But even the quality of the third dataset is not perfect, as when there is occlusion, it is not labelled as a detection.

3.5.3 – Training

With the technology stack made available by the Ultralytics YOLO implementation[9]. The training was done quickly and easily. This model was trained using pictures resized to 640x640. This choice was made to facilitate the merge of the two models for a later use, as described in Section 6.2

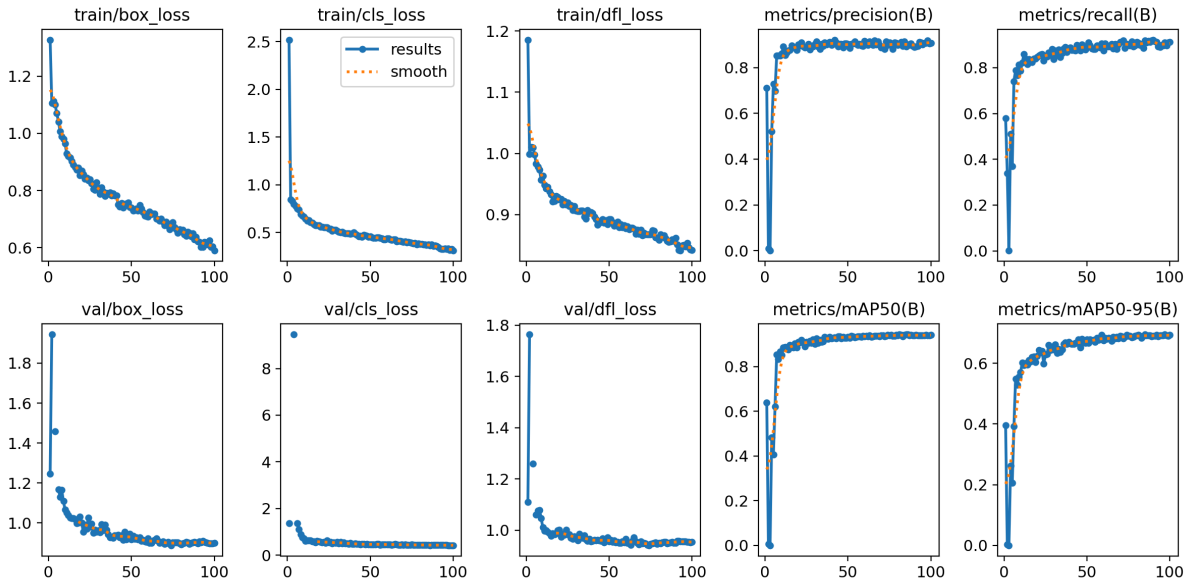


Figure 6 - YOLO fine-tuning statistics

As shown by the Figure 6, the training was completed without any issue, but we can observe that were are probably going to overfit if we continue the training any longer, as the validation curves are stagnating. This likely mean we have exhausted the value of the training data.

3.5.4 – Results

The result are as expected after analysis of the datasets, mitigated.

[9] https://docs.ultralytics.com/modes/train/

Figure 7 - Bib detection example 1

Figure 8 - Bib detection example 2

As seen in Figure 7 and Figure 8 in a unchallenging environment, the bib detection works as expected. But the confidence is already not ideal, as in such ideal conditions, the expected confidence should be between 0.85 - 1.0

Figure 9 - Bib detection example 3

Figure 10 - Bib detection example 4

But when the situation become more difficult, as in Figure 9 (low contrast between the t-shirt and the bib) and Figure 10 (bib far away, bib white on white t-shirt), the bib detector fails.

For now, as we will see in the Section 4, this result will suffice, and the work needed to create a better datasets put this solution outside of the project target, but this is clearly an area of improvement for future work.

Those are the results on a small datasets created form the videos taken at the “trail du Gramont” race[10].

Total	Bib detected	Bib missed	False positive
203	192 (94.5%)	11 (5.4%)	14

Table 1 - Bib detection benchmark result

[10] https://sptiming.ch/tdg25/

As shown in the Table 1, the results seems very good, but the dataset was put together by hand, and the situations were selected to be easy (no motion blur, no partially visible bib) so the results need to be taken while keeping that in mind. The number of false positive is pretty high, but this is not a problem, as the target was to have the highest recall possible. The false positive can be filtered out, by first allowing only a bib detection inside of a person detection box. This would effectively eliminate all detection outside of a person box. And then we can allow only a single detection per person box, to avoid reading numbers present on contestant t-shirt. As a last line of defense, we can put some filters on the result of the reading, but we will see those in the Section 3.8

3.6 – Depth estimation

The goal of this part is to allow for a precise timing of the participants. In order to do that, the best solution was to create a depth estimation from a single point of view (or two very similar ones for stereo vision). The depth estimation is what allows the system to find when a contestant would cross the line.

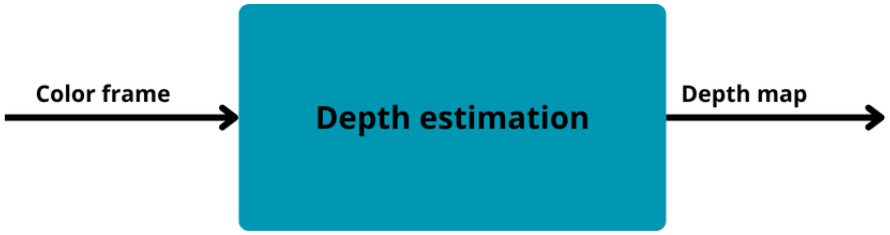


Figure 11 - Depth estimation block

The Figure 11 depicts the input and output expected for this block. As input, a frame in color is necessary. As output, a depth map is required. The depth map does not need any units as the next block supports relative depth estimation.

3.6.1 – Development

The challenging part of this problem was that they were two directions that are both justifiable : the monocular versus the stereo depth estimation. As monocular depth estimation depend greatly on the situation, the only way to know its precision was to benchmark it at a real race.

The number of tests in real situations were limited (two races were timed by SPTiming during the Bachelor thesis). The first race took place at the end of the second week of the Bachelor thesis, which was too early to have a system ready to test. The second race took place at the end of the fifth week of the Bachelor thesis. This race was the only real situation were the data that would be harvested would correspond to a real situation. Taking this calendar into account, the main goal was to reduce the risk as most as possible. And if the monocular depth estimation revealed to be not sufficient, the stereo depth estimation should be ready to take its place. This meant that, during the second race, data for both monocular and stereo depth estimation should be harvested at the same time.

3.6.2 – Stereo depth estimation

Stereo depth estimation was considered during half of the project, as the results from monocular depth estimation could not be asserted to match the precision needed for the timing solution until some data have been gathered. In theory, stereo depth estimation was a good solution, as it should have superior precision and be less subjective to the scene situation (as the race take places mostly outdoor, picture texture would not be a problem). With multiples matching algorithm available, ranging from the simple Block Matching[11] to some deep learning alternative like FoundationStereo [4], there would be multiple choices, with some compromise between precision and performance.

In order to harvest stereo video, some preparation was needed. Namely create a stereo bar to hold both cameras in place, create the tools to calibrate the setup and create the solution to record some synchronized video with both cameras at the same time. Doing this without any specialized hardware is very complicated

[11] https://en.wikipedia.org/wiki/Block-matching_algorithm

(if not impossible). In my tests, achieving sub millisecond precision is possible, but only at half the designed operating frame-rate of the cameras. So for the race, the choice has been made to record the videos in parallel with a timestamp for each frame, and to synchronize them afterward (allowing for a maximum error of Frame-per-second (FPS)/2. In our case 15ms). This allowed to record monocular and stereo video both at the same time, as the timestamps recorded could allow us to test the precision of the timing system after the race.

Once the stereo videos have been harvested, some tests have been led to assert the capabilities of the different solutions.

Figure 12 - FoundationStereo depth estimation

The results gathered from FoundationStereo have not been compelling, as they are worse that they should be, as shown in Figure 12. My analysis is that the baseline (distance between where the right and left images are taken) that I choose to record the data was way bigger than the baseline used to train this model. Or the calibration and rectification done on both the videos was not precise enough.

3.6.3 – Monocular depth estimation

The monocular depth estimation was a very good choice for this project, as it requires no calibration and no additional hardware. So this solution is the easiest to use.

But as there is no free lunch, it comes with disadvantages too. The precision depends greatly of the scene and there is no absolute scale. This means that the system need to work with relative scale too, this will be detailed in the next chapter.

For the monocular depth estimation model choice, after some research, some models were envisaged. Mainly the new DepthPro [5], DepthAnythingV2 [6] and its derivative VideoDepthAnything [7].

The tests were made on the entire dataset created from race videos, but the example shown here will depict only one picture of this dataset.

Figure 13 - This is the picture taken as example of the dataset

The Figure 13 depict the original picture on which depth estimation was made using different models. It is taken as an example from the dataset created from the “trail du Gramont” race.

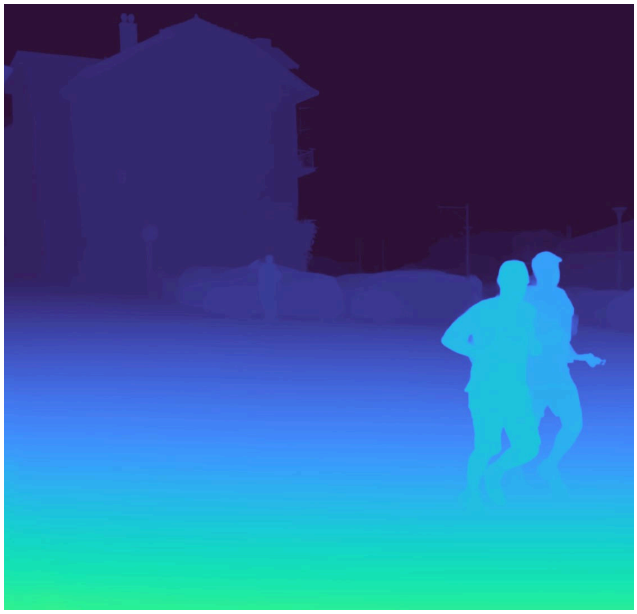


Figure 14 - DepthPro depth estimation

Figure 15 - DepthAnythingV2 Base depth estimation

As shown in the Figure 14 and Figure 15, the DepthPro and DepthAnythingV2 model are small, but DepthPro is still dominating precision-wise on finer details. But this come at a cost, as during the benchmarks, DepthPro was running 3-4 times slower that DepthAnythingV2. So as we don’t need this much precision on the finer details, DepthPro was set aside.

VideoDepthAnything is based on DepthAnythingV2, but with additional complexity to ensure temporal consistency. As this system can work well with relative depth, we don’t need this functionality and we can’t afford the hit in performance, so this solution was set aside too.

To choose the right version of the DepthAnythingV2 model, and the right picture size to find the right balance between performance and precision, I conducted a performance benchmark.

Input size	DepthAnythingV2 Base	DepthAnythingV2 Small
128x72	88.0 FPS	102.3 FPS
256x144	32.9 FPS	74.1 FPS
384x216	18.6 FPS	50.7 FPS
512, 288	9.4 FPS	27.6 FPS
1280, 720	0.7 FPS	2.0 FPS

Table 2 - DepthAnythingV2 model performance benchmark

The Table 2 shows the results of the benchmark. This benchmark was performed on a Lenovo Legion Y540-IRH15[12] laptop. This laptop is equipped with an Intel i5 9300H[13] CPU and a Nvidia 1660Ti Mobile GPU[14].

Figure 16 - DepthAnythingV2 Base (384x216)

Figure 17 - DepthAnythingV2 Small (384x216)

The Figure 16 and Figure 17 shows us that the resolution 384x216 is not sufficient to get enough details for race timing. As the distinction between the two runners got too much error for both models, with DepthAnythingV2 Base being slightly better.

[12] https://psref.lenovo.com/syspool/Sys/PDF/Legion/Lenovo_Legion_Y540_15IRH/Lenovo_Legion_Y540_15IRH_Spec.PDF

[13] https://www.intel.fr/content/www/fr/fr/products/sku/191075/intel-core-i59300h-processor-8m-cache-up-to-4-10-ghz/specifications.html

[14] (link("https://www.nvidia.com/en-eu/geforce/gaming-laptops/gtx-1660-ti/"))

Figure 18 - DepthAnythingV2 Base (512x288)

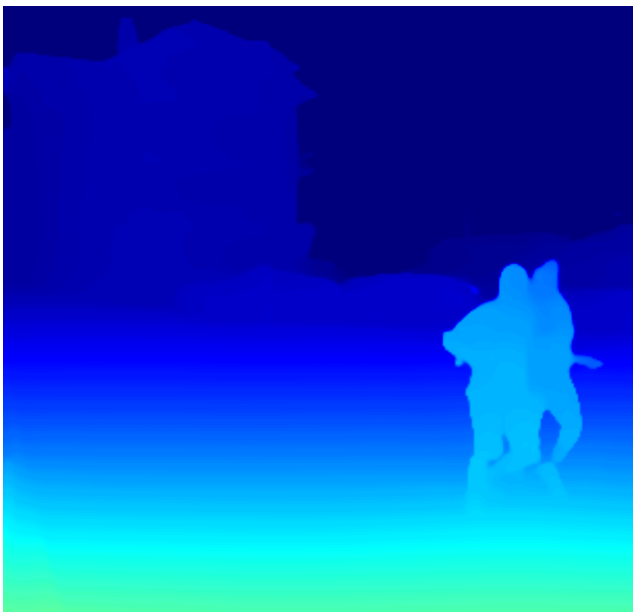


Figure 19 - DepthAnythingV2 Small (512x288)

The Figure 18 and Figure 19 depict the results taken from a picture of resolution 512x288. The distinction between the two runners is now clear, with relative depth between them being correct. As there is not a big enough difference between the Base and the Small models and as the hit in performance would cancels any benefit given by the Base model. The small model with a resolution of 512x288 was chosen.

One way to increase the quality of the results would be to crop out useful parts of the pictures.

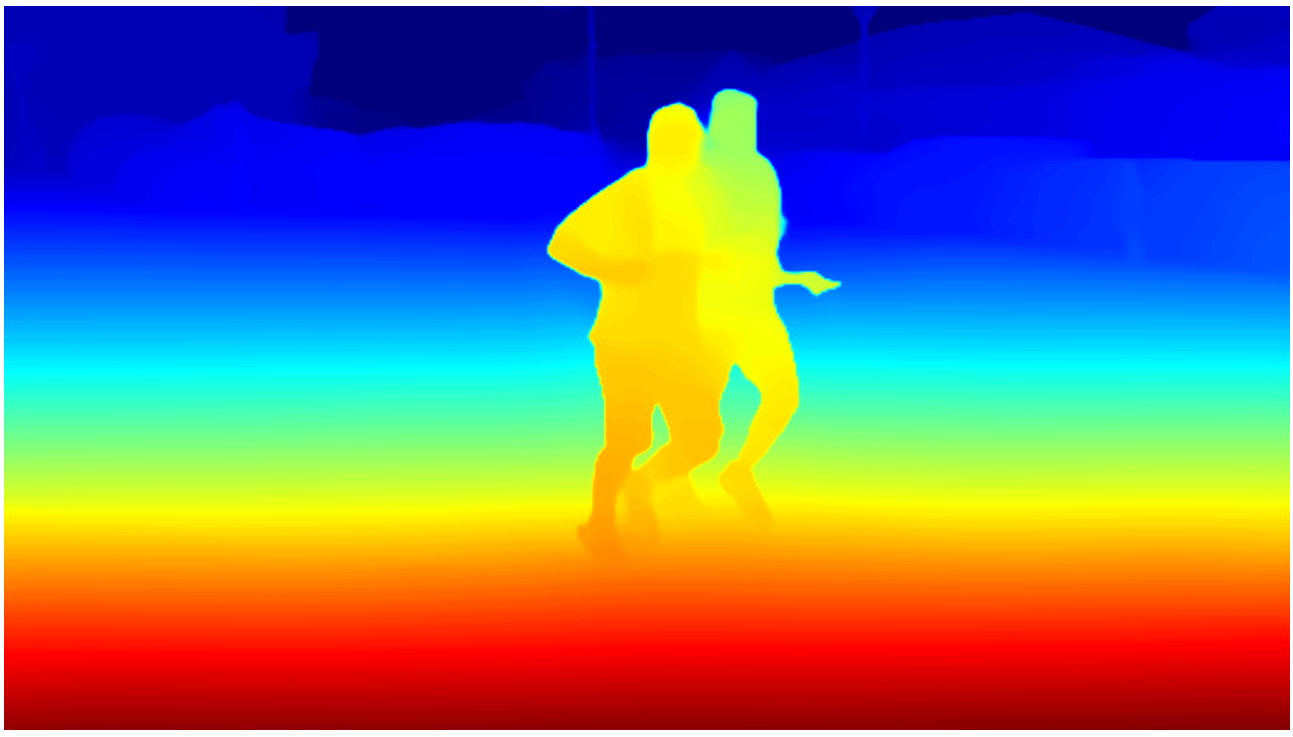


Figure 20 - DepthAnythingV2 Small, 512x288 cropped to subject only

As shown in the Figure 20, by cropping the picture to match the region of interest, we can now have level of details sufficient for race timing. The distinction between the runner is now clear, with one being behind the other. And as the performance of the DepthAnythingV2 Small model with a resolution of 512x288 is of a little less than 30 FPS, this fits perfectly our use case.

3.7 – Arrival line detection

This sub-system is responsible for detecting when a person cross the arrival line, filtering out false positive like a pedestrian, or a person passing the line in the wrong direction, and outputting the precise frame on which a runner crosses the arrival line.

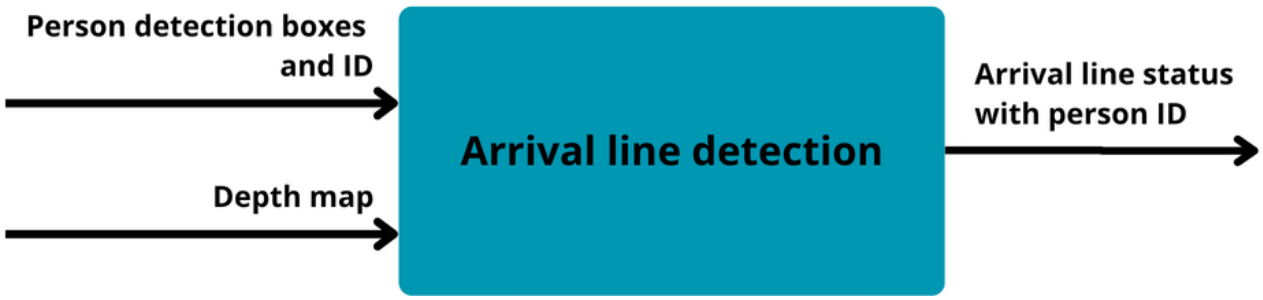


Figure 21 - Arrival line detection block

The Figure 21 depicts the input and outputs of the block. As input, a list of person detection boxes and their corresponding ID, in pair with a depth map corresponding to the picture are expected. As output, this blocks need to return a list of ID which have crossed the line on this frame.

This system is not very complicated, as we can take a point for each contestant (with the most replicability possible), and compare its depth with a point on the same X coordinate of the arrival line.

The point is positioned in the middle of the person detection box, this correspond to the hips/lower torso of a person. But this point can be moved, or even duplicated. But in the tests conducted, the performance hit of running a full pose estimation model in order to have precise body point for each people does not translate to a big difference in precision.

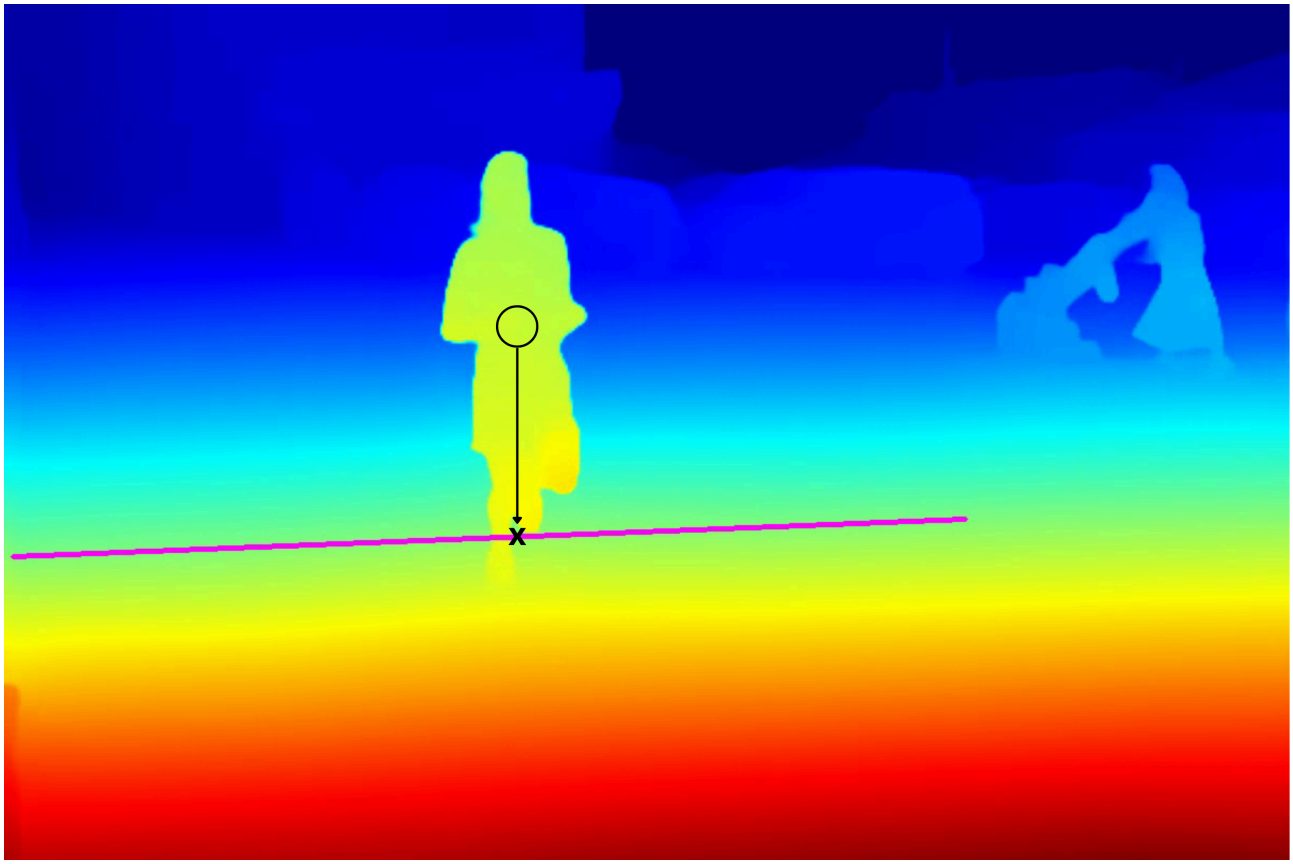


Figure 22 - Arrival line system

This system has a flaw, it is very vulnerable to occlusion. To reduce this flaw to the minimum, we can take depth value along the arrival line, filter out the values that are in a person detection box, and then use them

to fit a linear regression. This regression will allow the system to average out little errors due to noise, and generate missing data (like depth behind a leg) in order to be more resistant to occlusion.

This fixes the problem of monocular depth estimation model, which is that as the depth predictions are relative, the depth value can change from frame to frame. As this solution can be run on a frame basis, this problem won't have any impact.

To avoid line cross false positive, we can add a safeguard, we can consider valid only people going in the right direction (in our case, coming closer to the camera), and we enforce that the point on the runner body should be between some horizontal bounds (x coordinates) as the arrival line. To detect if a runner is going in the right direction or not, we can fit a line to its depth values. This will allow the system to detect the slope of the depth evolution for this contestant and filter this person out if the it goes in the wrong direction.

3.8 – Bib reading

This system is the one responsible to read the content of the detected bib. It is expected of this system to already have some sort of filters to remove false positive, and have the highest possible accuracy. It is the only block to work on the frame at its full quality.

Figure 23 - Bib reading block

The Figure 23 depicts the input and output of this block. As input, this sub-system needs to get the person detection boxes with their corresponding ID, and the bib detection boxes. As output, it is expected to returns, a list of person ID and new bib read pairs.

For this part, multiples OCR engines were envisaged. The engines that were tested were :

- Tesseract [8]
- EasyOCR [9]
- PaddleOCR [10]

From those three engines, pytesseract performed the worse (< 10%) of success, so it was quickly set aside.

The last two (EasyOCR and PaddleOCR) were both quite capable.

The dataset on which the benchmark was made was constituted of pictures taken from videos of the "trail du Gramont" race.

OCR type	Total	Full read	Partial read	Wrong read	no read
PaddleOCR	195 (100.0%)	92 (47.2%)	24 (12.3%)	2 (1.0%)	77 (39.4%)
EasyOCR	195 (100.0%)	69 (35.4%)	9 (4.6%)	5 (2.6%)	112 (57.4%)

Table 3 - Bib reading benchmark result

As shown in Table 3, the two OCR engines are good, but they are some criteria that distinguish the two. As we can see, PaddleOCR achieve a noticeable increase in accuracy . But another important feature is the ratio partial read vs wrong read. As PaddleOCR is better in those two category, this will allow us to run bib number reconstruction when multiples observations of the same bib are made.

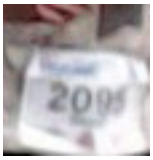
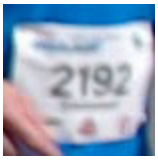
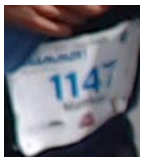
Input	Ground truth	EasyOCR (confidence)	PaddleOCR (confidence)
	2095	None	**2095** (0.858)
	2192	None	**2192** (0.995)
	1147	**1147** (0.973)	None
	2188	2186 (0.974)	**2188** (0.992)

Table 4 - OCR result comparison

The Table 4 shows some example of the dataset and the corresponding result from the two different OCR engine. We can see from the two firsts examples that PaddleOCR [10] is more tolerant to blur. But in the third row, we can see that PaddleOCR [10] gives no results back, whereas EasyOCR [9] succeed in reading the correct number. But the fourth example is why PaddleOCR [10] will be better for our task. As both OCR gives a result back with high confidence, but EasyOCR [9] give a wrong result back. The task is not easy, as the lighting conditions make the number appear as 2186. But for this task, we want the highest precision possible, event if we cannot read the number on some frames. As a human revision would be possible, it is better to not have any number read than a wrong number.

3.9 – Result construction

This block is responsible of constructing the result, and keeping the data in between frames. It is expected of this block to do some processing on the results in input, in order to filter out false positive detectable only on a multi-frame basis. This block can reconstruct bib detection per person in order to increase the precision on the detection.

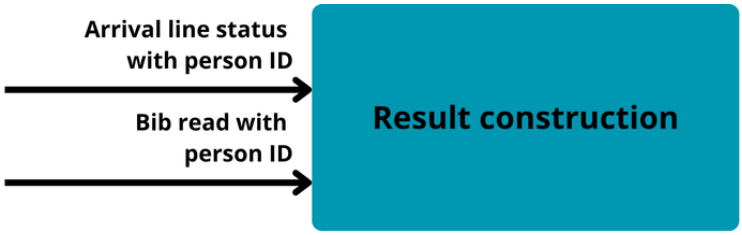


Figure 24 - Result construction block

The Figure 24 shows the input of the result construction block. This block takes as input two list. The first one contains the ID of the persons who have crossed the line on this frame, and the second list contains each new bib read, associated with their person ID on this frame.

This block is the simplest one, as his main role is to collect and construct a result by taking the results of the preceding blocks frame by frame, and merging them into the final result.

It is in this block that the number reconstruction processing is done, as it is a task that need temporal resolution in order to work. When a new bib is detected for a person, we try to reconstruct a full number by assembling small detections. This means that if a detection is made, and is present (fully) in another one, for example “204” is fully contained in “2049”. So the confidence associated with “2049” benefits from new detection of “204”. This is the simplest form of reconstruction, and some more complicated ones could be implemented under some conditions, this will be detailed in the Section 5.2.

3.10 – Complete pipeline

The complete system works as follows.

At first, the picture is processed in parallel by :

- A YOLOv11 model (pre-trained on the COCO dataset) **TODO** cite coco dataset !!!
- A YOLOv11 model fine-tuned on a bib detection dataset
- A DepthAnythingV2 Small model

Then, from the different results of the precedent step, for each person, we check if a new bib detection has happened and we check if a person has passed the arrival line (in the right direction).

If a detected bib is located inside a single person boxes, the system then try to read the content of the bib. If a bib is correctly read, it is then passed as output.

When the system detect that a new person has crossed the arrival line, we check if it is associated with a bib number or not, and if it is not (or if the confidence is not high enough), we can save a video clip for a human reviewer to check and add manually the bib number.

The last step is to construct the results with each person that has crossed the line and each person that has at least one bib read with enough confidence.

Chapter 4 – Results

This section will discuss the mains results obtained in this project. Those were computed on two video clip from the “trail du Gramont” race. The race took place on the 14.06.2025, the race results are available online[15]

As the bib numbers for a particular race are known before hand, a control of the bib read while computing the results is possible. This means that if the system read a bib number that does not exists, it is discarded immediately.

The first video clip span from 10h15 to 11h10. This video will from now on be referenced as video one.

The second video clip span from 12h20 to 13h45. This video will from now on be referenced as video two.

Both videos were recorded at a 1920x1080 resolution (even if some processing downscale the pictures before computing) and at 30 FPS

Those videos were recorded using a Logitech C920 Pro[16] camera.

The point of view between the two video clips differ slightly, as shown in the Figure 25 and the Figure 26

Figure 25 - First video point-of-view

Figure 26 - Second video point-of-view

[15] https://sptiming.ch/tdg25/
[16] https://www.logitech.com/fr-ch/shop/p/c920-pro-hd-webcam

4.1 – Caveat

As the position available for the camera did not allow me to film the finish line and the bib numbers, the point of view selected include only the bib numbers, so the timings cannot be compared directly with the officials, as there will be a difference of 5-10 seconds to take into account. In order to test the accuracy of the timing solution, some data have been labeled by hand.

This means that for a time recorded by the camera to be considered correct, a difference needs to be accounted. This difference is acceptable between 0-10 seconds (as the time to finish the race from the camera to the official finish line vary from runner to runner). But this difference needs to be in the right way, this means that the camera time needs to be triggered before the official one.

All this was only done to compensate for the bad camera point of view. In a real use of the system, this would not be necessary.

4.2 – Annotation

To detail some of the results, some annotated pictures will be shown. The annotation are the following :

- Each detected person is in a square, with some text above it. The text shows the person ID, and its bib number (if one has been detected). The bib number detection builds up during the time the person is visible.
- The arrival line created for the clip is drawn in light blue
- A circle located on the hips/lower torso of each person is drawn, this circle represent the point taken to make the depth comparison between the person and the arrival line. This circle can change color depending on the situation.

Figure 27 - Non runner person detected

The Figure 27 shows a person that has been detected, but no bib has been detected on it, and this person is not going in the right direction (fast enough), as shown by the red circle. This is the typical pedestrian detection. It is filtered out by the system, and does not interfere with the race timing.

Figure 28 - A participant evolution over time

The Figure 28 depicts the typical life-cycle of a participant in the system. From left to right, at first, a person is detected, with no bib. Then, the first bib detections happens (pink rectangle), we can observe that at first, only a partial bib number is detected ('210' in place of '2102'). After some time, the full bib of the participant is read and linked to the person ID. And lastly, the person passes the arrival line (draw in light blue), represented by the green circle on the person lower torso, and the person detection boxes going from red to green. This example will help the understanding of the rest of the analysis.

4.3 – Result category

In order to make some statistics on the results, some categories need to be defined :

- **Correct detection** : This means that the runner got his bib number detected correctly, and his time needs to fit in the conditions mentioned in Section 4.1
- **Time not detected** : The depth estimation system could not generate a precise timing, but the bib number was correctly detected.
- **Bib not found** : The bib reading system could not read a bib for this person, but the depth estimation system has generated a correct timing for this person.
- **Wrong time** : The bib has been detected and it exists in the current race, but it was outside of the bounds described in Section 4.1.
- **Nothing found**: Nor the bib reading or the depth estimation system could detect this participant.

From all those categories, the two we want to avoid the most are : Wrong time and Nothing found, as those two cases means that we have no way of detecting that something went wrong in a real race.

4.4 – Statistics

Video	Total participant	Correct	No time	No bib	Wrong time	Nothing found
1	**141**	105 (74.4%)	3 (2.1%)	32 (22.6%)	0 (0.0%)	1 (0.7%)
2	**128**	78 (60.9%)	0 (0.0%)	46 (35.9%)	3 (2.3%)	1 (0.8%)

Table 5 - Results statistics

As shown in Table 5, the results seems pretty good, but some supplementary analysis is needed.

By looking closer to the missed bibs, we can see that some of them are not readable, even with more cameras, or better software.

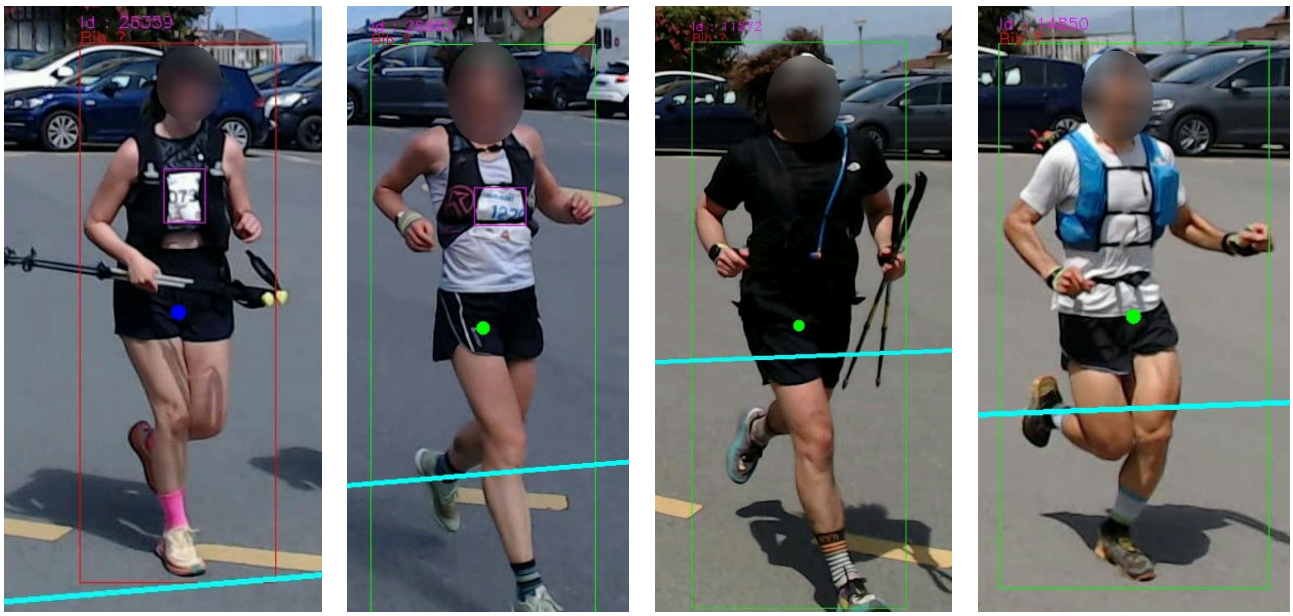


Figure 29 - Some examples of unreadable bib

The Figure 29 depicts some example of unreadable bib. From left to right : **2073**, **1229**, **1166**, **3**. Those examples are not representative of the quality of the system. In order to have a better representation, we will remove them from now on.

Video	Total participant	Correct	No time	No bib	Wrong time	Nothing found
1	**134 (-7)**	105 (78.3%)	3 (2.2%)	25 (18.6%)	0 (0.0%)	1 (0.7%)
2	**104 (-24)**	78 (75.0%)	0 (0.0%)	22 (21.1%)	3 (2.8%)	1 (0.9%)

Table 6 - Results statistics - filtered

As we can see in Table 6, the impact of the filter is more significant on the second video, as there was more people with their bib under some clothing/equipment.

4.5 – Result analysis

This section will analyze the different cases where the system failed, to try to assert its capabilities and its limitations.

4.5.1 – Wrong timing

Now we will focus on the three wrong timing to analyze them and find the reason of failure.

Case 1

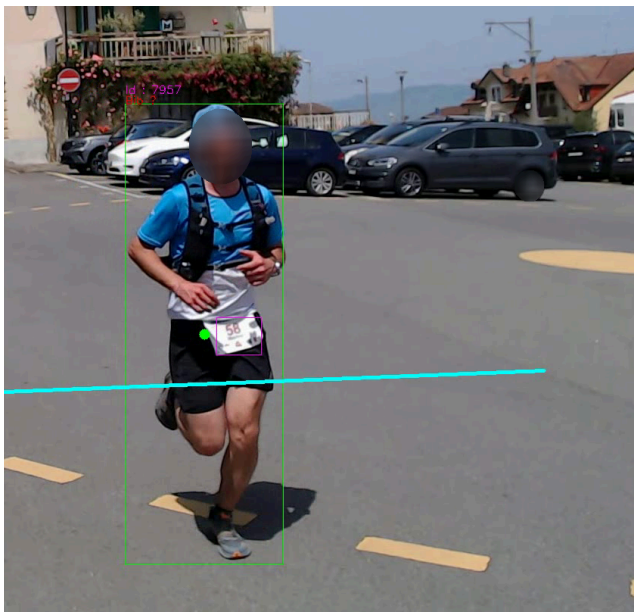


Figure 30 - Participant 58 detected by the camera

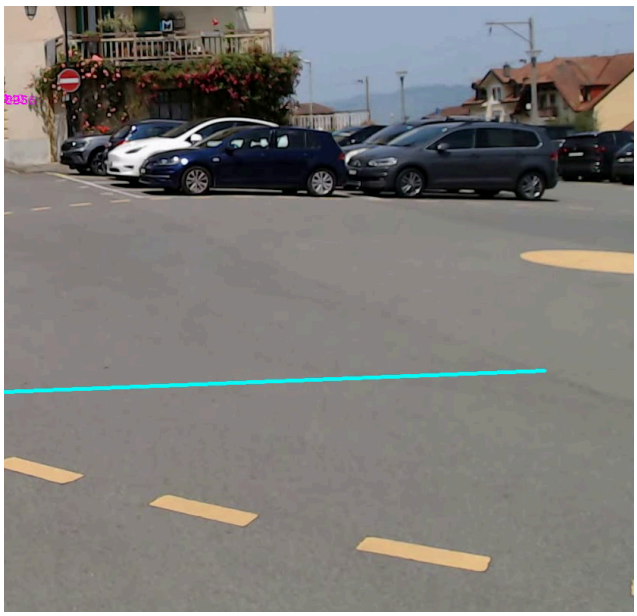
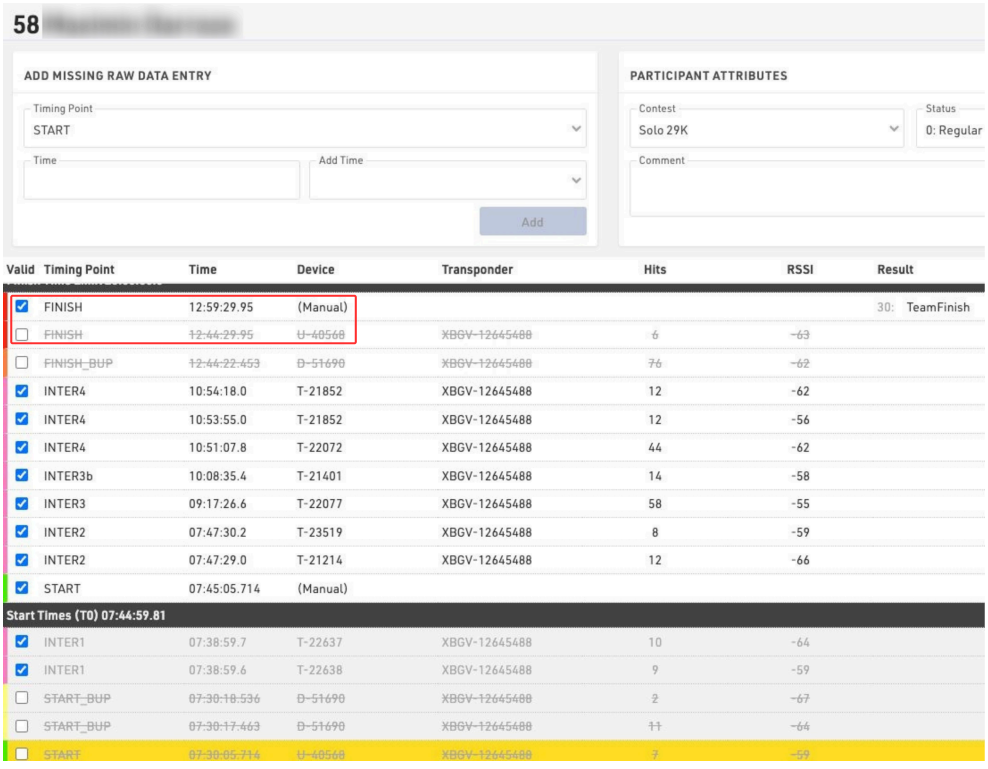


Figure 31 - Participant 58 official time

As depicted in the Figure 30 and the Figure 31, the camera system has detected the contestant 58 correctly, but the picture taken from the official timing shows no one. By looking into both system timing, we can see that the camera system detected the contestant 15 minutes before his official time. This might be a sign of a manual time correction.



58 [redacted]

ADD MISSING RAW DATA ENTRY

Timing Point: START

Time:

Add Time:

Add

PARTICIPANT ATTRIBUTES

Contest: Solo 29K

Status: 0: Regular

Comment:

Valid	Timing Point	Time	Device	Transponder	Hits	RSSI	Result
☑	FINISH	12:59:29.95	(Manual)				30: TeamFinish
☐	~~FINISH~~	~~12:44:29.95~~	~~U-40568~~	~~XBGV-12645488~~	~~6~~	~~-63~~	
☐	~~FINISH_BUP~~	~~12:44:22.453~~	~~D-51690~~	~~XBGV-12645488~~	~~76~~	~~-62~~	
☑	INTER4	10:54:18.0	T-21852	XBGV-12645488	12	-62	
☑	INTER4	10:53:55.0	T-21852	XBGV-12645488	12	-56	
☑	INTER4	10:51:07.8	T-22072	XBGV-12645488	44	-62	
☑	INTER3b	10:08:35.4	T-21401	XBGV-12645488	14	-58	
☑	INTER3	09:17:26.6	T-22077	XBGV-12645488	58	-55	
☑	INTER2	07:47:30.2	T-23519	XBGV-12645488	8	-59	
☑	INTER2	07:47:29.0	T-21214	XBGV-12645488	12	-66	
☑	START	07:45:05.714	(Manual)				
Start Times (T0) 07:44:59.81							
☑	INTER1	07:38:59.7	T-22637	XBGV-12645488	10	-64	
☑	INTER1	07:38:59.6	T-22638	XBGV-12645488	9	-59	
☐	~~START_BUP~~	~~07:30:18.536~~	~~D-51690~~	~~XBGV-12645488~~	~~2~~	~~-67~~	
☐	~~START_BUP~~	~~07:30:17.463~~	~~D-51690~~	~~XBGV-12645488~~	~~11~~	~~-64~~	
☐	~~START~~	~~07:30:05.714~~	~~U-40568~~	~~XBGV-12645488~~	~~7~~	~~-59~~	

Figure 32 - The detailed results of the contestant 58

After asking SP Timing, they confirmed that the participant started at the wrong time, and got his time corrected manually afterward, as shown by Figure 32. In the red rectangle, we can see that the automatic device recorded the same finish time as the camera system, but a manual override happened after. And in the bottom of Figure 32, we can see that the runner started the race 15 minutes early. As they were multiples starts, this is an easy mistake to make. This situation shows that the camera system worked correctly for this participant.

Case 2

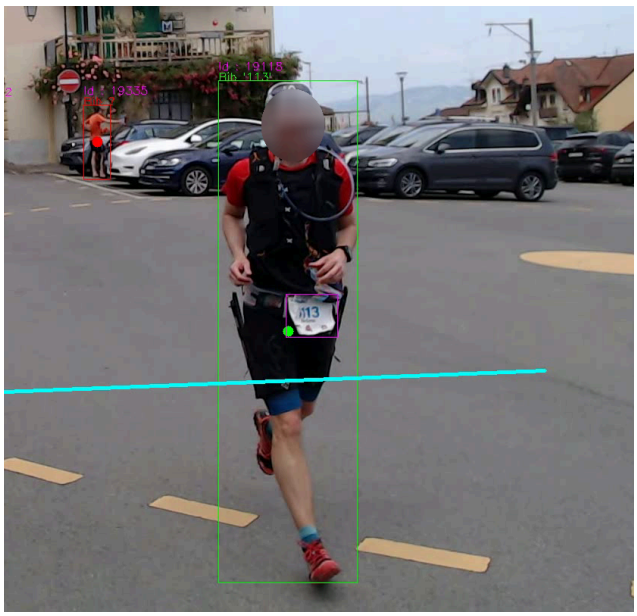


Figure 33 - Participant 113 detected by the camera

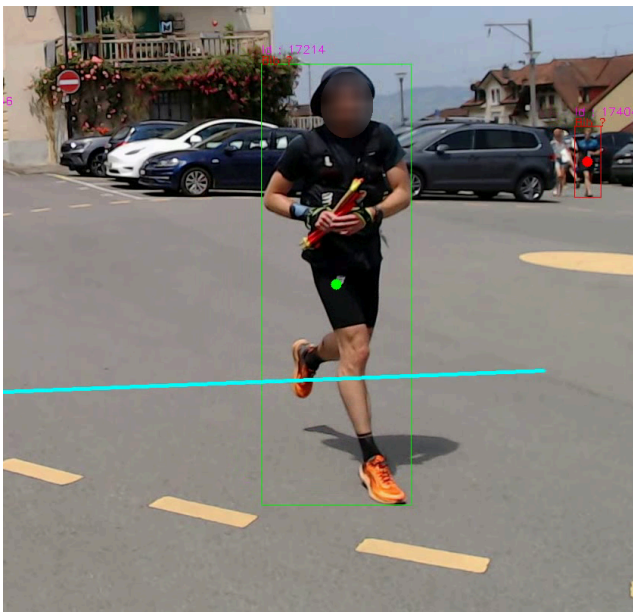


Figure 34 - Participant 113 official time

As we can observe from the Figure 33, the system failed with this participant. As the bib from the runner “1113” was not displayed very well, the first number was not visible to the camera, this means that he has been read as “113”.

The real “113”, as shown in Figure 34, did not have a bib at all , this means that it removed the only way for the system to correct itself, by detecting “113” a second time with more confidence, and tagging the first as a wrong detection.

Both those situations lead to a wrong time recorded for the contestant “113”, which is a critical error of the system. This could however be avoided if constant length numbers would be printed, as this would allow for the system to classify a partial or a full detection based on the length of the bib, and could even enable the system to do more complicated number reconstruction.

Case 3

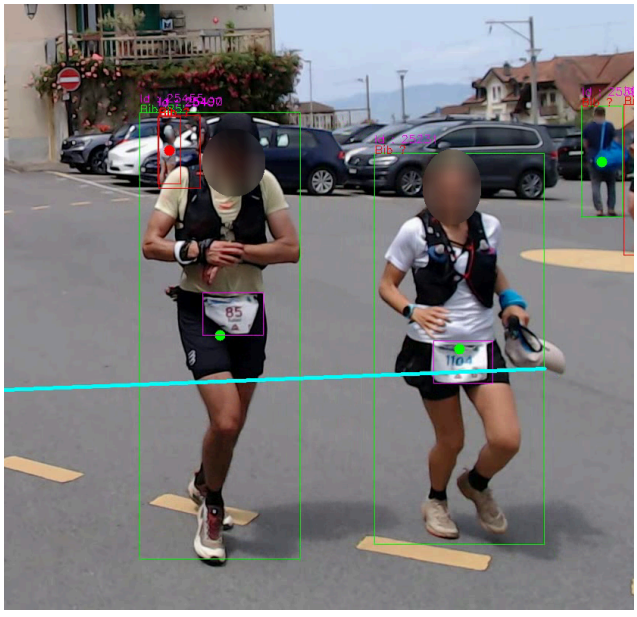


Figure 35 - Participant 1104 detected by the camera

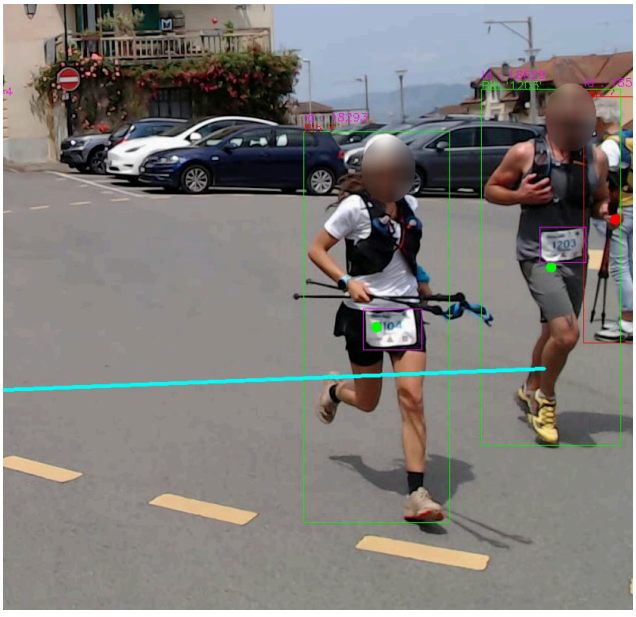


Figure 36 - Participant 1104 official time

On the Figure 35, we can see on the right that the participant 1104 is correctly detected by the camera system. But on the Figure 36, we can see that the participant 1104 is detected a second time, those two detections are 30 minutes apart. From the two different timings, the system took the one with the most confidence on the detection, in this case the second detection, whereas the official race timing was the first detection, so this explain the differences between the two detections.

4.5.2 – No time

This section will analyze in details the case where no time could be registered for a runner.

Case 1 and 2

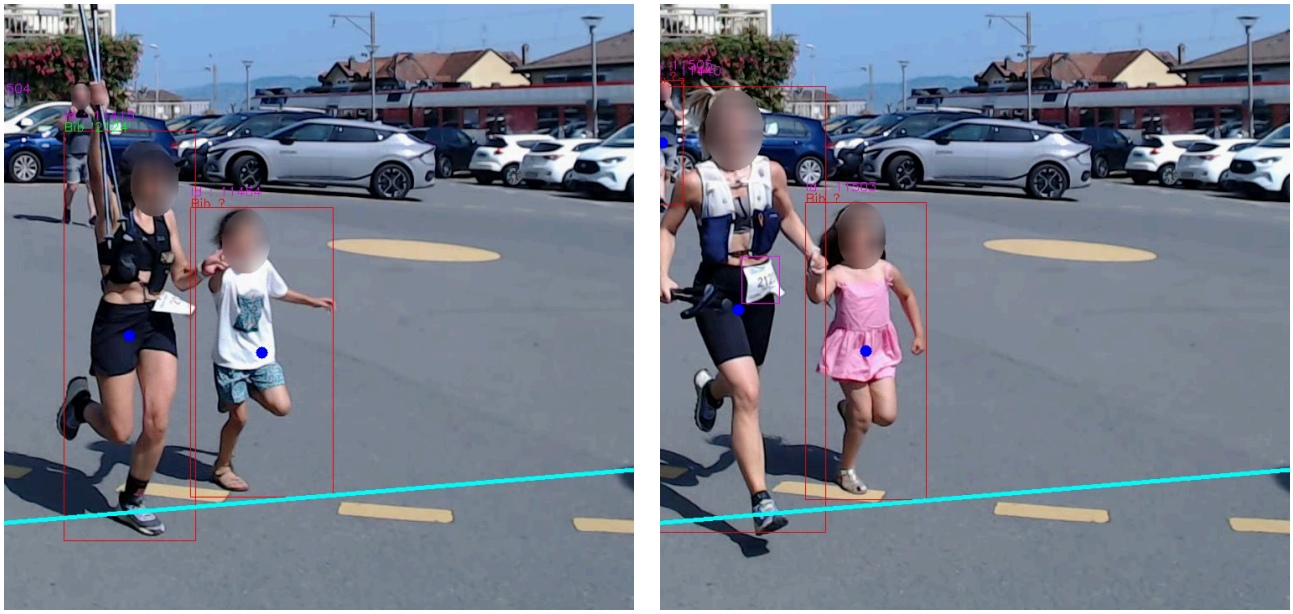


Figure 37 - From left to right, runner n°2124 and n°2123

The pictures shown in Figure 37 depict the pictures taken at the moment of the official race timing.

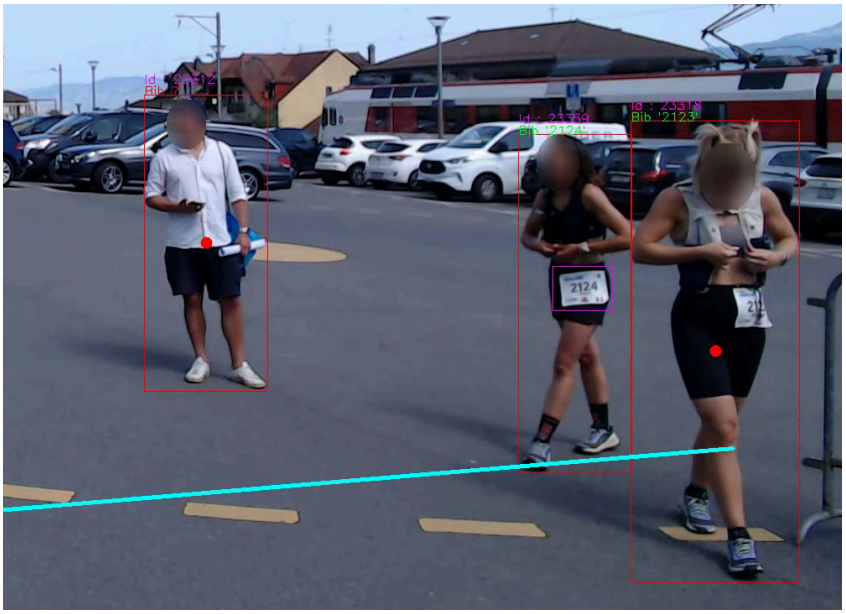


Figure 38 - Runners second passage

The Figure 38 shows us why those two runners got no precise timing, as they were detected a first time (from their first passage) in Figure 37, but then they were detected a second time in Figure 38, which erased their first detection as the detection got more total confidence the second time, as they were greatly positioned relative to the camera to be detected correctly.

Case 3

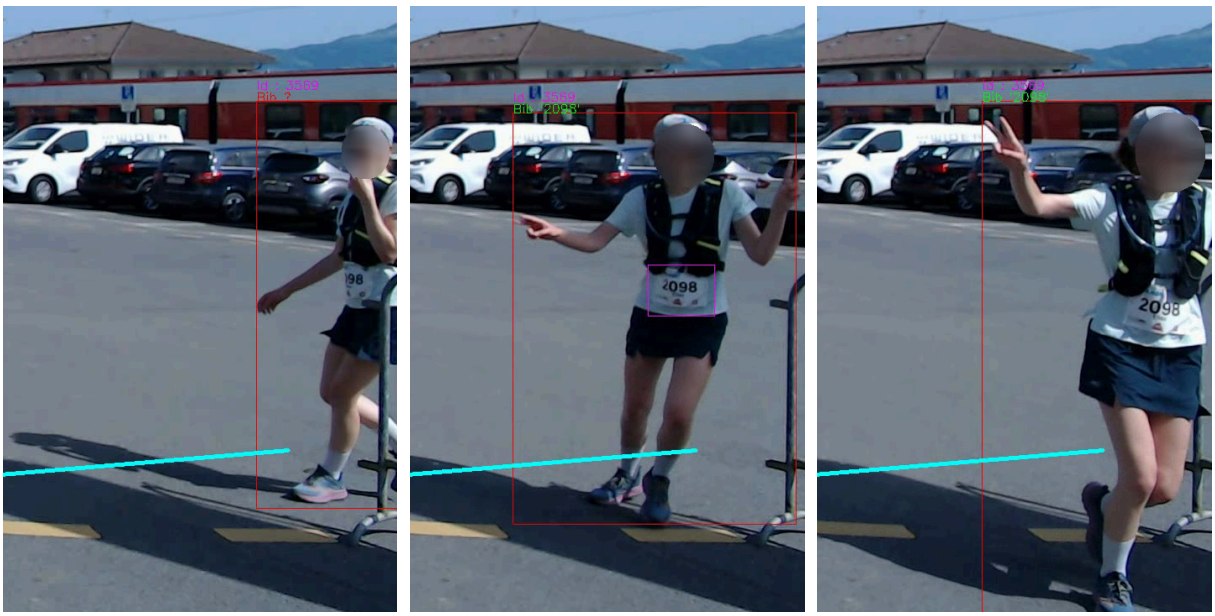


Figure 39 - Passage of runner n°2098

As we can see in Figure 39, the runner n°2098 did not take a good path to the finish line, this mean that the system did not consider it a correct passage and could not get any precise timing for it, it has however successfully detected the bib number, so a trace of this passage was kept in the results.

4.5.3 – Nothing detected

There was two cases were no time and no bib was detected. This is considered a critical failure as the system has no way of telling it has failed. This section will details those two cases

Case 1

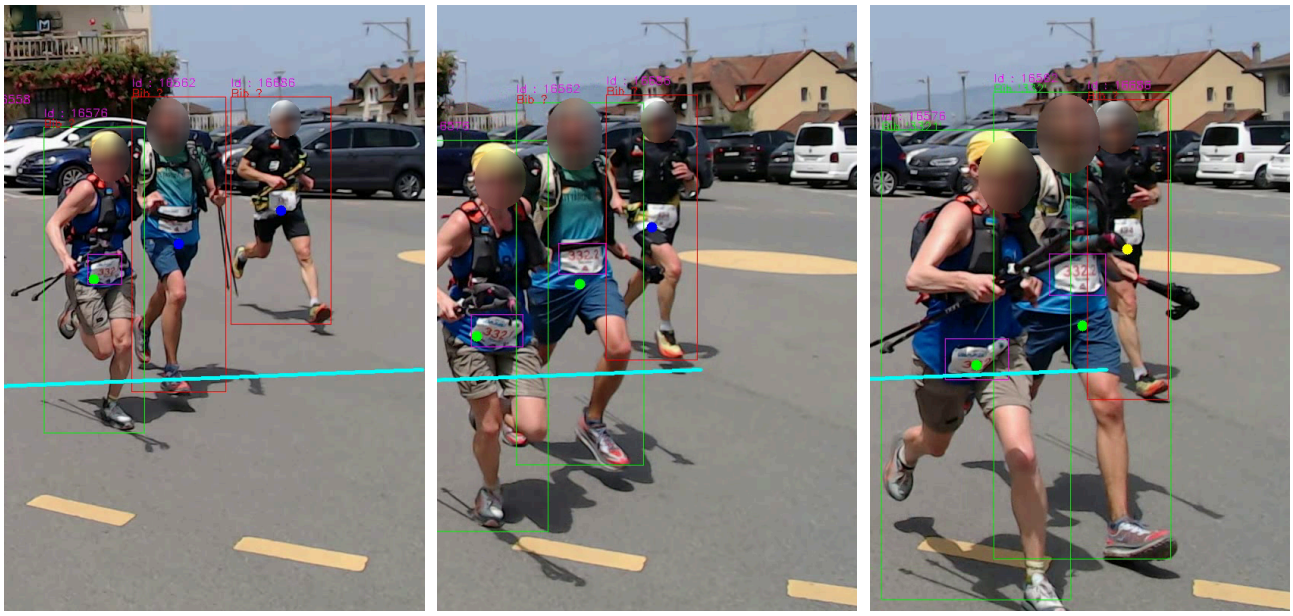


Figure 40 - Passage of runner n°134

The Figure 40 shows the passage of the finish line by the runner n°134 (rightmost person in the pictures). As we can see, the combined facts that there is some occlusion and that the runner passed at the edge of the line makes that the system could not see that person pass the arrival line. A wider arrival line could have detected his passage, but this solution just reduce the number of times where occlusion is too big to detect a person, it would not eliminate it.

Case 2

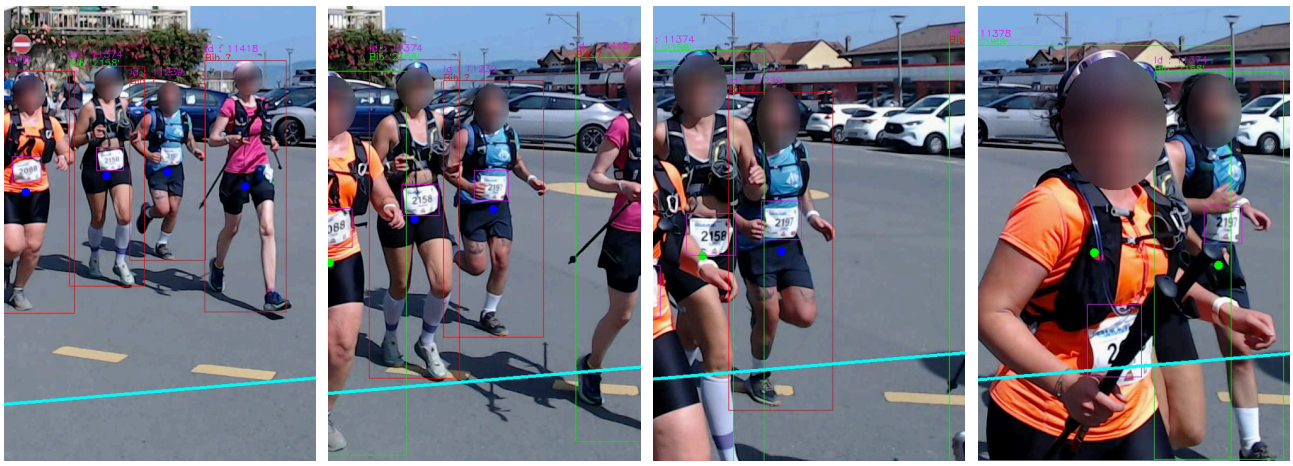


Figure 41 - Passage of runner n°2197 (blue t-shirt)

This situation is quite similar to Case 1, there is occlusion happening at the wrong time, which lead to the system being unable to distinguish the two person (and their corresponding bibs).

There is no real solution to the problem of occlusion, as we can only mitigates its consequences, but we have no way of eliminating this problem altogether. We will see the strategies to reduce the occlusion problem in the Section 5.6.

4.6 – Timing precision

As mentioned in Section 4.1, an comparison between the official timing system and the camera timing would be impossible, so a small subset (50 runners) of video two has been hand labeled to estimate the precision of the timing system. To do so, I labeled the frames at which I estimate that the runner passes the arrival line, and then we can compare that to the timing made automatically by the video system.

As each frame recorded has an associated timestamp, we can asses the difference in milliseconds.

Metric	Value
Median error	67.99 ms
Mean Absolute Error (MAE)	64.87 ms
Standard deviation	37.82 ms
Maximum error	163.90 ms
Minimum error	0.0 ms[17]

Table 7 - Metrics on timing error (in milliseconds)

The Table 7 shows us that the system achieve great precision in timing. As considering a frame-rate of 30, this means that in average, the system is wrong by 2 frames. We can observe too that the results have an offset, as told by the difference between the MAE and the standard deviation.

[17] This results is perfect because the hand-labeling was done on a frame basis. So this result tells us that the system choosed the same frame as the annotator

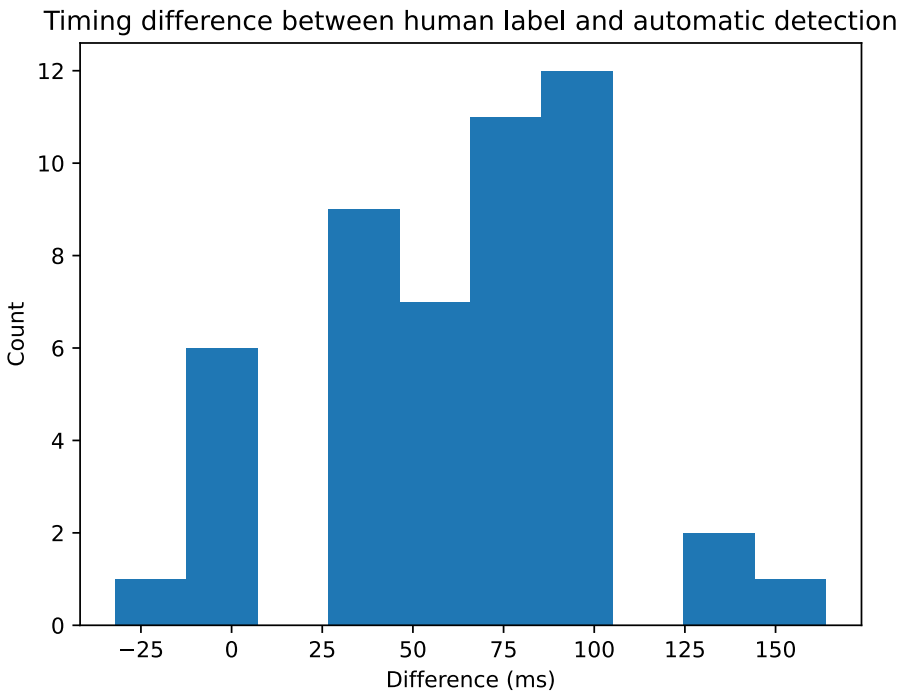
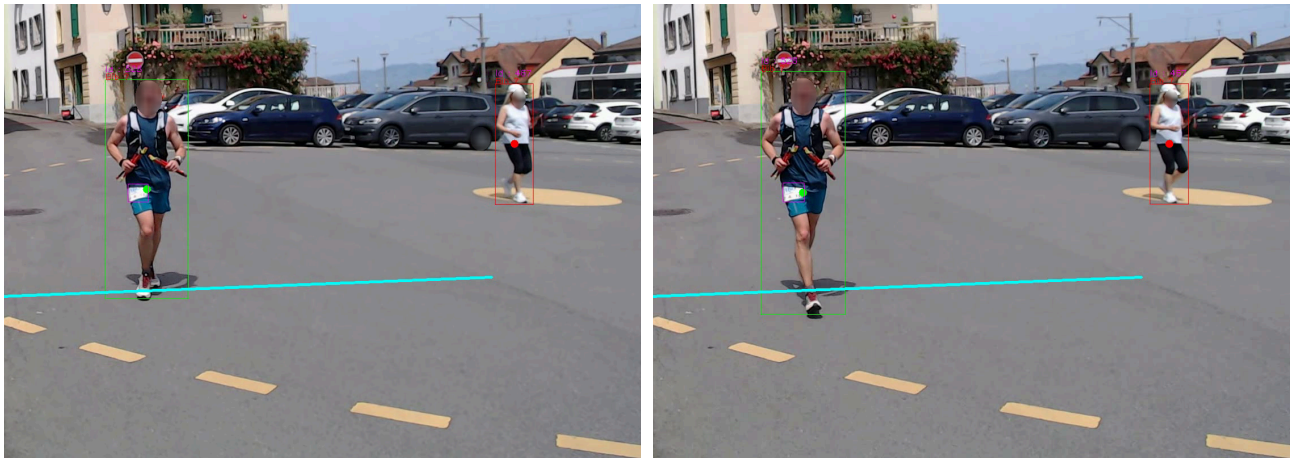


Figure 42 - Histogram of timing difference

The Figure 42 depicts, as expected, a Gauss curve. But as we can see, it is not centered on 0. This means that the system timing detects the line passage later than the human label. This indicate that the system is more precise than what its average suggest us. And as confirmed by the standard deviation in Table 7, the standard deviation is lower that the average (or the median).



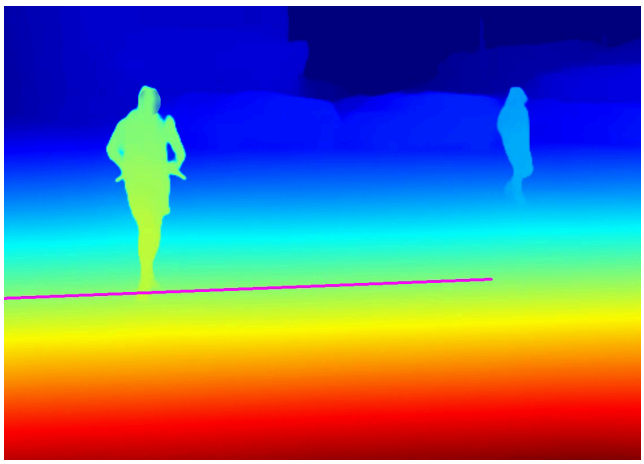


Figure 43 - Frame tagged by the system as finish

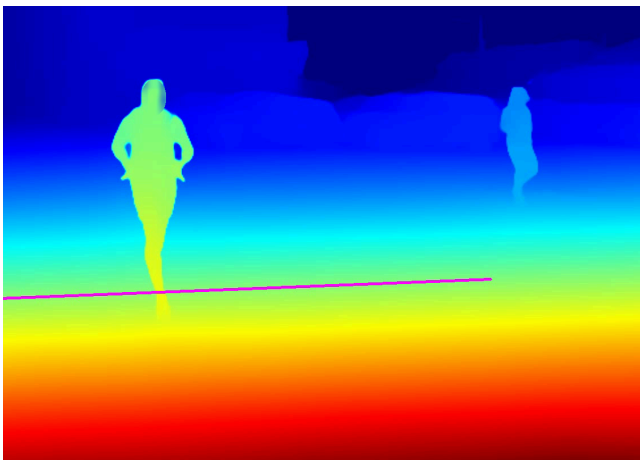


Figure 44 - Frame hand-labeled as finish

The Figure 43 depict the frames on which the depth estimation saw the participant cross the finish line. The Figure 44. There is a difference of two frames between the two. As we can see, the system triggers a little early relative to the human label, but the difference is very small to the eye.

Chapter 5 – Discussion

This chapter will discuss in detail the limitation of the system, its weaknesses and how to mitigate/eliminate them

5.1 – Bib detection

For now, the bib detection is tuned to have the highest recall possible, without much care for the accuracy. This means that false positive are common. This is not a problem in our case as the probability that a false positive occurs, in a person box, with a valid bib number on it is very low (if not zero). But as we allow for only one bib detection at a time in a person box, some false positive are effectively "hiding" the true positive.

To upgrade this part, some better labeled data would be needed. One could either complete and verify the open datasets used for training. Or create one from scratch

5.2 – Bib reading

The biggest improvement this part would benefit from would be to fix all the bib number to a precise length, for example 4 digits. As wrong read are very rare, partial ones pretty common, this would ensure the system detects every partial read, and let a human reviewer intervene and write manually the bib number.

A new bib reconstruction system could even be created, as by fixing the length of the bib numbers, some more complex reconstruction is possible (without increasing the number of false positives). Something based on the Bayesian inference[18]

5.3 – Timing precision

The current timing solution (monocular depth estimation) is very dependant on the situation. This means that for a good camera and arrival line position,the results are what is described in Section 4.6, but, according to my tests, the standard deviation can go up to about 250ms in the worst cases (arrival line too close to camera, person right in front of the cameras). This imply that the camera position and point of view is very important, as a bad situation will decrease the precision of the timing even when no occlusion is happening.

5.4 – Performances and optimizations

For now, the entire systems run at an average of 15 FPS on a Intel i5-9300H CPU and a Nvidia 1660Ti Mobile GPU. This is sufficient for now, but a faster processing means a more precise timing solution.

The first step to optimize this solution, would be to merge the two YOLO models into a single one. This has not been done for now as this would need a dataset containing both person and bib number labeled. This dataset does not exists for now. This merges with the point of Section 5.1, which is to create a good quality dataset containing labels for both person and bib classes.

The second (more demanding) step would be to dig into the implementation of the YOLO model, to allow for sending a single time the frame to analyze to the GPU memory, and doing a single time the preprocessing (if applicable). As for now, the systems sends three copy of each frame to the GPU.

5.5 – Frame-rate impact

As mentioned in Section 5.4, the current systems run at an average of 15 FPS. This means that by computing the results on a 30 FPS video, we are cheating by considering the system could run at a higher frame rate than it really does. To asses the difference that it would make, the results shown in this section depict the computing done on 1/2 frames of the video 1

[18] https://en.wikipedia.org/wiki/Bayesian_inference

Framerate	Total participant	Correct	No time	No bib	Wrong time	Nothing found
30	**134**	105 (78.3%)	3 (2.2%)	25 (18.6%)	0 (0.0%)	1 (0.7%)
15	**134**	97 (72.4%)	4 (2.9%)	32 (23.9%)	0 (0.0%)	1 (0.7%)

Table 8 - Results at 15 FPS - Filtered

The Table 8 shows the results of the comparison between a recording running at 30 FPS and one running at 15 FPS.

As expected, we can see that the drop in temporal resolution translate to a drop in the number of bib read and time recorded. Some parameters of the system could be optimized to work best at 15 FPS (number of frame recorded before line, slope of the depth evolution etc…). But this comparison gives a good idea of the impact of this change.

This shows us a great strength of our current system, the bib reading system has a great accuracy, as even with less frame, we see a drop in recall, but none in accuracy.

5.6 – Occlusion mitigation

To mitigate occlusion problems, the first step would be to take a better position for the camera, and enforce every runner to have the bib located in the same location (with no equipment/clothing on it). This would increase the performance of the video based bib reading without any major change to the system itself.

The second solution would be to add a copy of the system 50 meters before the final line, with another point of view. This would ensure that participants hiding behind others could be detected.

The third solution would imply a greater effort, it would be to add another point of view. This would mean that we would run the system on two videos in parallel. With this, a calibration would be needed to ensure that every detection is shared between the two cameras.

Chapter 6 – Conclusion

6.1 – Current state

The system, as of now, works as expected or better. As there is some cases where it exceed expectations, and others where it could do better. But even with all those flaws, it has succeed in, from 269 runners, have only 2 critical failure where no data could be registered for the runners. This is a success as even for cases where the bib number could not be read, or the precise timing not be deducted, a human intervention is still possible.

As the system is designed in blocks, changing the inner working of some parts (for example switching the monocular depth estimation for a stereo system) would not have any impact outside of this module. This allows for independent upgrade going along with the technology (as some solutions used are at the center of many research).

This solution is almost good enough to be used commercially, but it still needs some work in some area to be viable. In particular the bib detection and reading could be improved, as for many cases with partial occlusion, a human can, by looking at the video, reconstruct the full bib number. But for now, the system cannot reconstruct it if a number is partially obstructed during all the frames.

Performance-wise, the system is currently usable, but it could be better if some data with person and bib labeled would be found or created. This would allow to merge the two detection models into a single one, reducing performance overhead.

6.2 – Future work

A considerable upgrade that the system could benefit from would be to upgrade the bib reading system. In order to do so, one could either fine-tune an available OCR to our situation with data taken from the video of the race. Or create and train a model from scratch on it. The results from this approach can only be better if the data is good, and with the recorded videos, there is a consequent volume of dat available.

Another possible upgrade is concerning the number reconstruction. For now it is in a quite simple form, but if the bib number length is fixed for a race, a complex solution could be built to reconstruct a full number from partial only detections.

As a last resort, a multi camera system could be envisaged, with two different point of view to ensure reliable detections.

Bibliography

[1] G. Jocher and J. Qiu, "Ultralytics YOLO11." [Online]. Available: https://github.com/ultralytics/ultralytics

[2] Y. Zhao *et al.*, "DETRs Beat YOLOs on Real-time Object Detection." 2023.

[3] W. Lv, Y. Zhao, Q. Chang, K. Huang, G. Wang, and Y. Liu, "RT-DETRv2: Improved Baseline with Bag-of-Freebies for Real-Time Detection Transformer." [Online]. Available: https://arxiv.org/abs/2407.17140

[4] B. Wen, M. Trepte, J. Aribido, J. Kautz, O. Gallo, and S. Birchfield, "FoundationStereo: Zero-Shot Stereo Matching," *arXiv*, 2025.

[5] A. Bochkovskii *et al.*, "Depth Pro: Sharp Monocular Metric Depth in Less Than a Second," in *International Conference on Learning Representations*, 2025. [Online]. Available: https://arxiv.org/abs/2410.02073

[6] L. Yang *et al.*, "Depth Anything V2," *arXiv:2406.09414*, 2024.

[7] S. Chen *et al.*, "Video Depth Anything: Consistent Depth Estimation for Super-Long Videos," *arXiv:2501.12375*, 2025.

[8] T. O. Team, "Tesseract OCR: Open Source Optical Character Recognition Engine." [Online]. Available: https://github.com/tesseract-ocr/tesseract

[9] JaidedAI, "EasyOCR: Ready-to-use OCR with 80+ Languages Supported." [Online]. Available: https://github.com/JaidedAI/EasyOCR

[10] P. Authors, "PaddleOCR, Awesome multilingual OCR toolkits based on PaddlePaddle.." [Online]. Available: https://github.com/PaddlePaddle/PaddleOCR

[11] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "BoT-SORT: Robust Associations Multi-Pedestrian Tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[12] Y. Zhang *et al.*, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," 2022.

Appendix

Table of Acronyms

FPS :	Frame-per-second
ID :	Identifier
MAE :	Mean Absolute Error
OCR :	Optical Character Recognition
YOLO :	You Only Look Once

List of Figures

Code Appendix