sensors

MDPI

Article

Video based analysis and reporting of riding behavior in cyclocross segments

Jelle De Bock [1*], Steven Verstockt [2]

[1] Ghent University (IDLab – imec); jelle.debock@ugent.be
[2] Ghent University (IDLab – imec); steven.verstockt@ugent.be
* Correspondence: jelle.debock@ugent.be; Tel.: +32 9 33 14920

Abstract: In technical sports such as swimming or soccer video-based technique analysis is common practice, but in cycling disciplines such as cyclocross, road cycling or cross-country mountain biking it is far less common. In swimming the analysis is centered around the motion of the human body but cycling analysis should be more centered about riding lines and behavior. This paper presents an end-to-end solution to detect riders, collect metadata and analyze behaviors within a fenced area on a static camera video feed. The fence is a user-defined rectangular area (i.e. bounding box) on the video footage in which ride line analysis will be performed. First the riders are identified and tracked by an Alphapose skeleton detector and a spatiotemporally aware pose tracker. For each pose, rider modus (e.g. sitting or standing) and team jersey recognition is captured as extra meta-information. Finally, a post-processor analyses the riding lines and summarizes the metadata for all the riders that went through the defined fence on the video feed. This information can provide interesting insights in line choices based on the time riders spend in the fence with respect to the line that was taken and can be very valuable for performance analysis, storytelling and automatic summarization.

Keywords: pose estimation, sports, object detection, sports data analysis

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Sensors* **2021**, *21*, x. https://doi.org/10.3390/xxxxx

Academic Editor: Firstname Lastname

Received: date
Accepted: date
Published: date

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

In some sports such as cross-country mountain biking, marathon running or motor sports the race is often won or lost on a specific part of the racecourse. In Formula 1 racing, a prime example of a sport that is almost synonymous with advanced engineering and extensive data analysis, these sector times are available for the entire race lap. This information is available in real time and gives fans, race teams and commentators great insights in strengths/weaknesses, race tactics or vehicle capabilities of the riders that are on the track. Albeit lap timings in sport events are already very valuable and interesting it still provides only one part of global sector analysis. It is very good at illustrating who needed the least time to go from point A to point B. However, it is not only interesting to study how much time was needed, but also the path they followed and how exactly they went from point A to point B in a certain amount of time. Rather than just the raw timings, this extra meta-information might even help to understand why one athlete went faster than another. In this paper we will present a video-based approach providing an objective, data driven answer on the "why" part of the sector data analysis question. The remainder of this paper is organized as follows: in the related work section we will discuss relevant related work on sector timing and video based ride performance analysis; under methodology we will further elaborate on the video processing pipeline and the fencing rideline track data post processing; next, the results section will discuss some experiments we performed with the video pipeline and will point out interesting analysis use cases and visualizations of the produced rideline data; in the final section we will briefly summarize our main findings and contributions and point out some future work that is required to fine-tune the processing pipeline.

2. Related work

To answer the first part (i.e., the timing aspect) of the sector analysis question, several approaches have been suggested in literature. Sector timing is often performed by some kind of wireless communication protocol. Athletes or objects wearing an active or passive tag pass through a checkpoint which acts as a receiver or senses the tags at the checkpoint. Ultra-high-frequency RFID is one of the more popular technologies to use in sports such as mountain biking, running, skiing and road cycling or cyclocross. Kolaja and Ehlerova [1] performed a field study to test this technology in various sports and under various circumstances and found that this technology provides satisfactory results to accurately record checkpoint crossings. They also proposed an architecture with a backend database that (post)processes the raw data into actual checkpoint crossing timings. The big advantage of this technology is its reasonable cost and resource effectiveness and its relative easiness to set-up [2]. Another technology that can be used for the gate crossing problem is the Bluetooth Low Energy technology (BLE). Sun et al. [3] studied the accuracy of this technology for gate crossing. They used a high-speed camera to quantify the accuracy of the BLE technology in different running scenarios (BLE tags worn at different locations and emitting at different signal strengths) and found that the timing error is always less than 156 milliseconds. Timing information based on sensor information can also be very valuable for the video analysis methodology discussed in this paper. For instance, and as presented in our work, the gate-crossing and identification of riders within a segment can assist to further enhance the information extracted from the video footage (i.e., identified detected objects on the video stream).

Popular sports such as soccer, tennis, basketball or cycling are usually broadcasted on national television. Although video data cannot be directly used for performance analysis, several computer vision techniques can be utilized to extract performance data for further analysis. In basketball for instance, Arbués-Sangüesa et al. [4] proposed a methodology to extract and track visual features of basketball players using a combination of a pre-trained pose estimation model and an additional feature extraction network. Santhosh and Kaarthick [5] performed a similar workflow but used OpenCV algorithms such as HOG-descriptors to detect and track players. Additionally, they also defined a homography matrix to map the detected locations on the video frames on a top-view representation of the basketball field, allowing interesting visualizations such as heatmaps. Chakraborty and Meher [6] suggested a video-based ball detection and tracking methodology that facilitates extensive path analysis of the ball during basketball long shots.

In our rideline analysis we will adopt a video-based approach to extract the ride lines of cyclocross riders but as further explained in the discussion section of this paper, our approach might also benefit from additional sensor-based rider tracking.

3. Materials and Methods

The proposed methodology, shown in Figure 1, consists of several consecutive steps and ultimately produces a path that bicycle riders have followed through the defined fence. In this section we will further elaborate on each of the steps that are required to produce this path info.

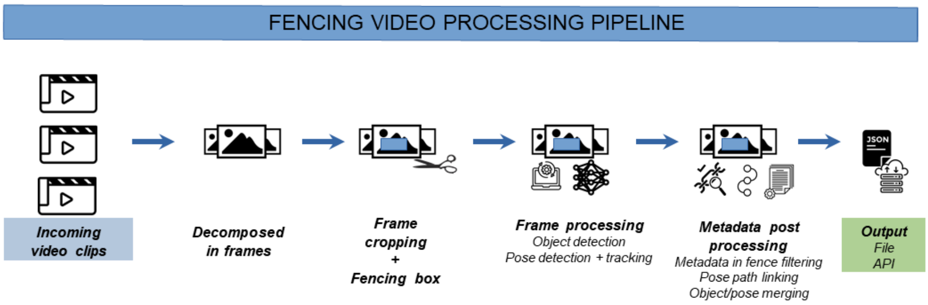

Figure 1: Different steps in the video processing pipeline to produce segment path data

The first step of the analysis consists of the decomposition and initial preprocessing of the incoming video source. The pipeline accepts both recorded video clips and live video data delivered by popular streaming formats such as HTTP Live Streaming (HLS) and the Newtek Network Device Interface (NDI) protocols. The video data is decomposed into frames and based on the framerate of the recorded video some frames are periodically skipped for an optimal balance between accuracy and processing time. Experiments with high-definition footage (1920x1080 pixels) at a framerate of 30 frames per second show that processing every third frame gives the best balance between pipeline detection accuracy and processing speed. By processing every third frame the video is processed in (near) real time. For preprocessing, the frame can be cropped to leave out irrelevant background information for further analysis (and further speed up processing times). Next, a rectangular fence is defined within the cropped region. The fence is defined as the region of video in which movements and behavior of riders are analyzed as illustrated by the *measurement zone rectangle* in Figure 2. Only detections within this region will be considered in further analysis (see post processing subsection).

Figure 2: Illustration of the fence principle on a video frame of a cyclocross training session. The red rectangle is the region of interest. Results of pose detection, tracking and post processing are illustrated by the yellow path within the fence.

In the following two steps a combination of computer vision and machine learning algorithms will gather more information about the riders that are present in the fence. As a start, an Alphapose pose estimator [7] is run on the frame to detect the riders and their body keypoints. To track riders through the frames, Alphapose offers various pose tracking implementations (e.g., PoseFlow, Human ReID or detector based). However, after thorough experimentation with the different trackers and its parameter configurations we were not able to achieve satisfiable results. The trackers work great on pedestrians, but on skeletons that are pedaling a bicycle the tracking often fails. The tracker can fail in two different ways: a first being a tracking identifier swap, especially after partial occlusion of one skeleton behind the other. Another tracking failure occurs when a new tracking identifier is assigned to an *already seen* skeleton. However, for further path analysis we decided to position the camera in such a way that it is filming the fence from a frontal but overhead camera angle. This has the advantages that the fence is more accurately representing the real-life coordinates and that skeleton swaps are less likely to occur as there is a better unobstructed view on the riders (e.g., riders will not *be hidden after each other*). With this extra prerequisite in mind, we implemented a more straightforward, yet purpose tailored spatiotemporally aware tracking mechanism that mostly circumvents the mentioned shortcomings of the trackers included within Alphapose. Full details of the tracking methodology can be found in Appendix 1, but we will briefly discuss the main working principles of the technique. The technique keeps track of the skeletons seen in the last 5 frames with its last known coordinates within the fence. When a new frame is processed, the distance matrix between the old poses center locations and the new pose centers is calculated. The new pose matches with an older pose if it has the minimum distance to that old pose and the distance is smaller than 25 percent of its diagonal size of the bounding box around the new pose. Each time a new frame is processed, the poses older than 5 frames ago are also removed from the pose match dictionary. This approach works very well in cycling as cyclists travel from a starting to an end point within the fence, so the corresponding bounding boxes are also moving similarly through the fence over time.

In the next step of the video processing pipeline, a clear distinction between each of the *ridemodi* a rider can adopt is made. In cyclocross, riders ride on or run with their bikes, based on the technicality and surface conditions. Technical sectors might for instance be perfectly rideable for a rider with great technical prowess but might be completely unrideable for another less technical rider. The barriers are a great example of such a technical sector. If the barriers are relatively high and are placed at a challenging part of the course (e.g., uphill or after a corner), some less technical riders will be forced to dismount their bike and run over this course feature. Monitoring these differences among riders within the video fence can be for instance very valuable in helping to understand why, how and where riders are taking a certain line and explain why one rider is slower than another. To answer the riding mode, question a neural network was trained to detect cyclists that are either running or cycling and spectators. To train such a model a training dataset of 869 images was constructed, with 747 cyclists that are riding, 457 running, 116 crashing and 1038 spectators. The dataset was split uniformly across the categories in 75% for the training data, 20% for test and 5% for validation. The data was used to train a YoloV5 model (*yolov5s* variant) for 100 epochs and achieved a mean Average Precision (mAP) of 68%. As can be seen in Figure 3, it is the spectator class that is degrading the overall model's performance quite a bit as it classifies most spectators as background.

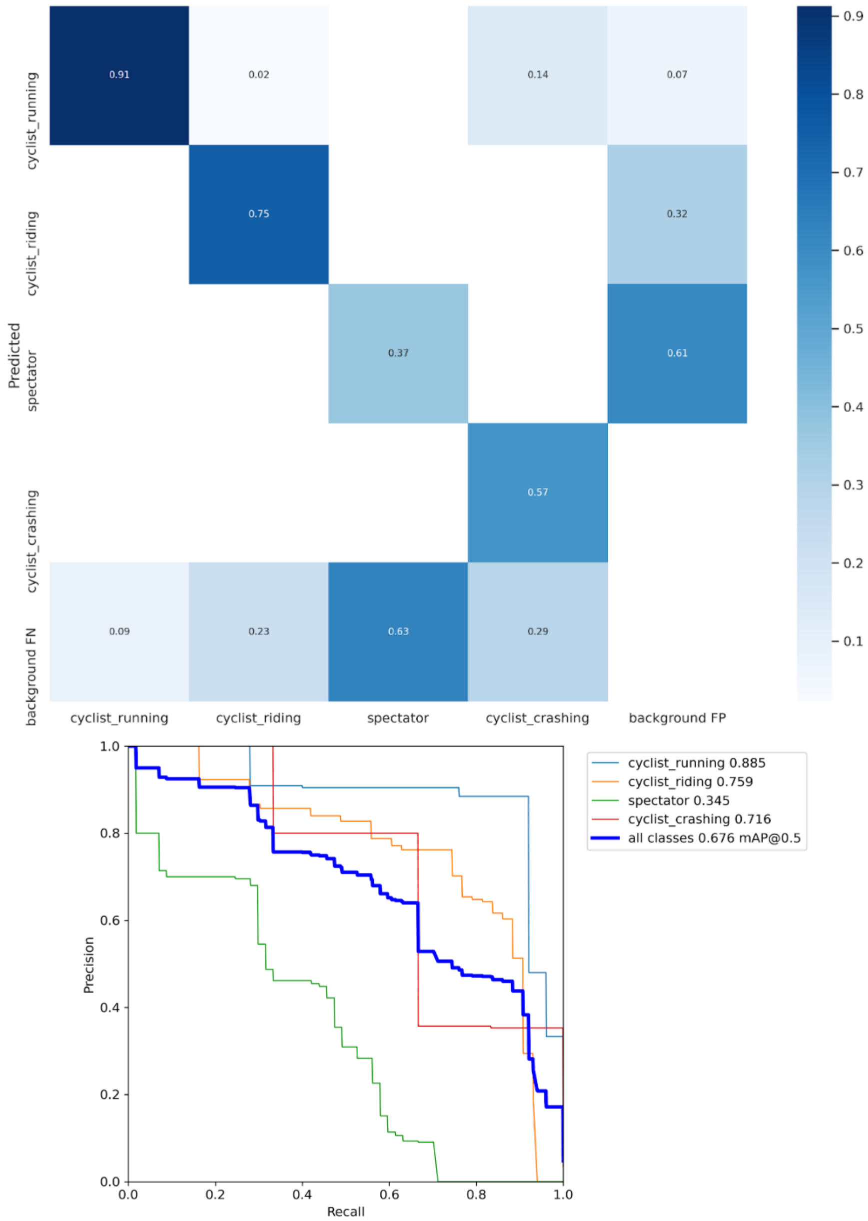

Figure 3: Confusion matrix and precision recall curve for the rider mode detector. As can be seen, the green line of spectators is performing significantly worse than the other classes.

Another interesting metadata generator within the pipeline is the team (jersey) recognition module. The team jersey recognition should be capable of detecting to which team a rider belongs based on the team jersey they are wearing. Team jerseys usually have distinct patterns with some sponsors on them. If team jerseys wouldn't change over the years, it might be perfectly feasible to train a *state-of-the-art object recognition model that* has a couple of hundreds of images for each team. However, in practice team jersey designs, sponsors and even colors usually change every year (or sometimes even faster), which makes this approach rather unfeasible. To overcome this limitation, a methodology that only uses relatively few examples for each jersey class should be implemented. For this purpose, a transfer learning approach was used. In neural network transfer learning, the trained knowledge of an existing neural network is reused to do the classification or detection task for another unseen, but related problem [8]. This is usually done by removing the last output layer of the network and adding another one instead. All weights of the original network are *frozen* (i.e., are not trained any further), but the last layer's weights are based on the (limited amount) of provided problem specific training data. For our team classification module, we trained a RESNET18 model, with its last fully connected layer replaced by a linear layer that was retrained. In the first attempt of preparing the training data, the team jerseys were rectangularly cropped from a larger image. This introduced a lot of background noise in the image, which had a negative impact on the trained predictor's accuracy. This shortcoming was mostly solved by an extra model that crops the relevant body parts from an image with background information [9]. For our model's training data, we retrieved the torso from the humanparser's generated body part segmentation output. An example of a before (with background information) and after (without) can be consulted in Figure 4. This significantly improved the model's capability to accurately classify teams based on their jerseys. With this approach, a model was trained on five teams (see Figure 6 for an overview of the corresponding jerseys). For each team a total of 8 images were used to train the final output layer of the RESNET18 model. The training images were preprocessed using a set of (random) image transformations randomly performing light condition changes, horizontal flips, slight rotations and/or cropping. The other 3 images were used for testing purposes. The model achieved a *96% validation accuracy after 10 epochs* of training time. The model was further validated on a number of unseen images for each team. The confusion matrix of this extra validation data is shown in Figure 5. As could be seen, some teams are still mistaken one for another, but the provided shots are from multiple camera angles and zoom levels, so additional more clever cropping might still improve the model's prediction.

Original image | **Humanparser segmentation masked output**

Figure 4: Illustration of the cropping and masking of the torso of a cyclist.

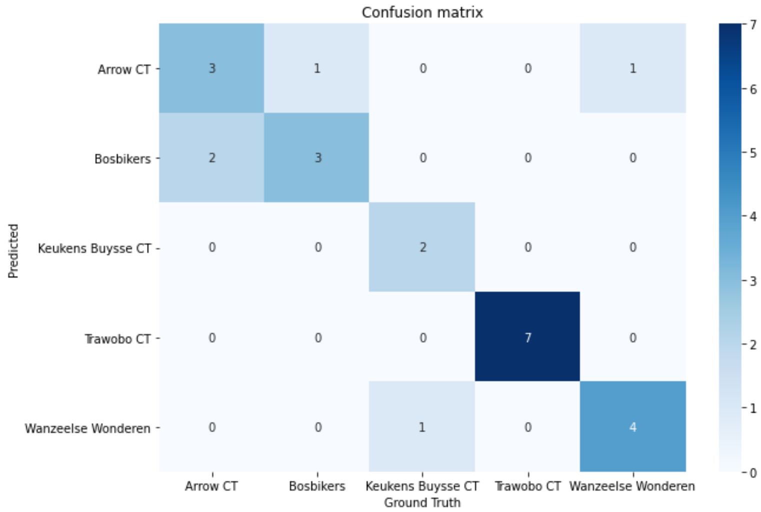

Figure 5: Confusion matrix of team predictions on unseen validation images trained on five team jersey classes.

Figure 6: Overview of the five team jerseys that were used to train the team jersey recognition model.

The next link in the video processing pipeline is the check if a rider is in the fence. This is determined by the center of the bounding box (bbox) around the joints that are detected by the Alphapose detector. If the center of the rider's bbox is within the fence's box, the rider is considered *in the fence*.

After the previously mentioned pipeline elements have run, the post-processing step can now be performed. The difference with the previous steps and the post processing step is that the previous steps take place on a *per frame level*, but the post processing happens across multiple frames. In the real-time path analysis scenario, the post processing is initiated if the fence remains empty for 10 consecutive analyzed frames. In post-processing the actual paths travelled by the tracked skeletons are determined. If desired, the coordinates of the paths can also be transformed into real life coordinates using a homography perspective transformation [10]. The main challenge within this post-processing step is the handling of the re-identification of the pose tracker. A pose is re-identified if the pose tracker assigns a new tracking identifier to a pose that was already seen in a previous frame. The possible causes for re-identification can be usually reduced to two different categories. The first, is when the Alphapose estimator fails to map the skeletons for one or more consecutive frames, which causes a *jump* in the subsequent positions which is too high for our *simple geospatial pose tracker* to link it to a previous pose instance. Another culprit can be when two poses are basically overlapping each other and both identifiers are mistaken for each other. With these limitations in mind, a post processing strategy can now be implemented to search and solve the re-identifications. The strategy exploits the fact that in our *fenced approach* the skeletons will always travel in a consistent direction within the fence (e.g., left to right, right to left, top to bottom or bottom to top). With this added constraint, a straightforward, yet powerful geospatially aware pose path merger can be implemented. The merging process is illustrated in Figure 7 and will be briefly discussed in the next paragraph. In summary, the merging process consists of three steps. In the first step, the paths are split based on jumps in frame numbers of the different tracks. A track is defined as a pose that was tracked over time in the fence and was assigned a tracking identifier. The criterion to split a track is that tracks that have a non-subsequent frame sequence are split into two separate tracks. This prepares the tracks for step two of the merging process where the tracks are attempted to be merged again based on a spatiotemporal weighting function. As illustrated in Figure 7, a track has a number of match candidates that can be matched. The selected candidate is the one with minimum spatiotemporal distance and below a certain threshold that is set based on the fence's dimensions. In the last step, the spatiotemporally linked tracks are iteratively matched based on the index lists of frame numbers within a track. This process stops if all paths have index lists that are non-overlapping.

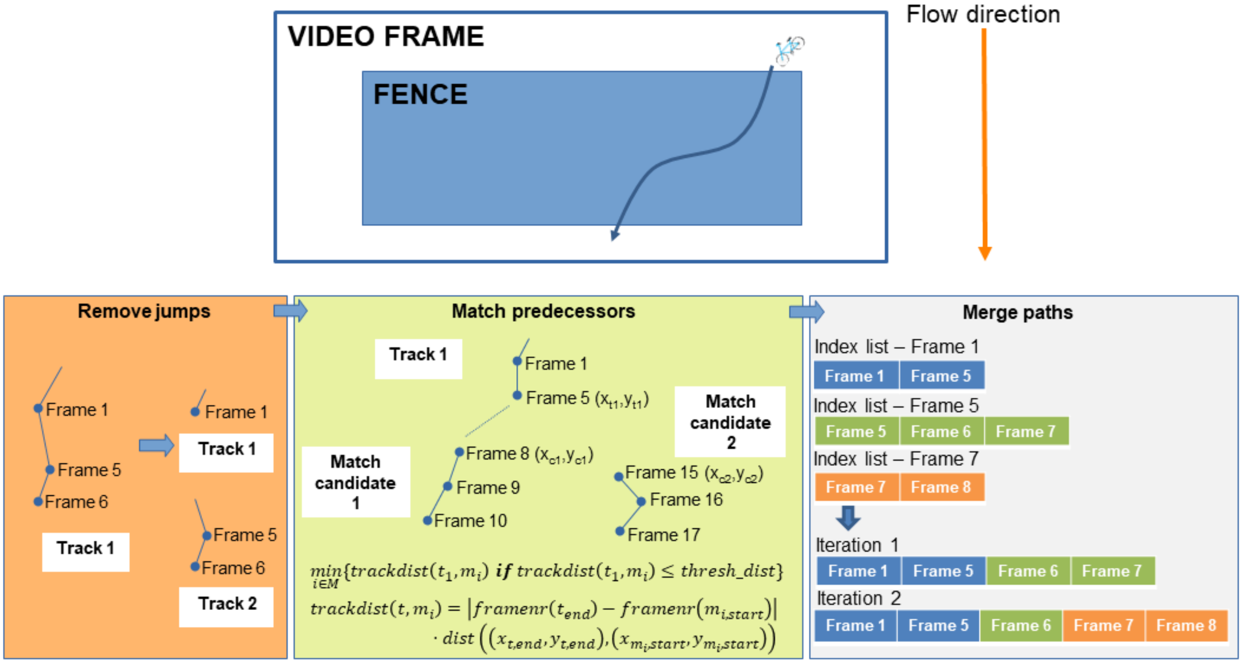

Figure 7: A schematic overview of the three-step pose tracking merging strategy.

The proposed pose tracking methodology combined with the merging strategy produces a set of pose tracks that can be directly used in the following steps in the video pipeline. In this next step the pose tracks (or parts of the tracks) that are within the boundaries of the desired fence's coordinates are extracted. To check if a skeleton of a path is within the fence, the center of the bounding box around its joints was used. Using this approach, the tracked skeletons' coordinates stay much more consistent and less spikey as when a joint such as the knee or foot were used.

The skeleton is considered within the fence if this center coordinate is within the fences bounding box coordinates (see Figure 8). This *check* is performed for every frame in which the skeleton in the track was detected. A valid path in the fence is defined as a rider that is entering and exiting the defined bounding box (and has multiple detections within the fence).

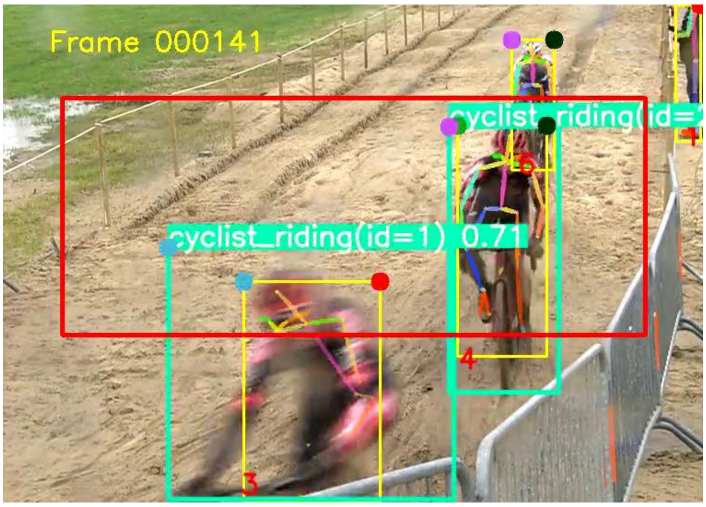

Figure 8: Rider entering (id = 6) and rider exiting (id = 3) the sand pit fence (red rectangle)

Once the valid paths have been detected, all information is available to create insightful statistics of the path a rider did follow within the fence. The extra metadata such as the rider modi and the team jersey recognition results are used to annotate the path with a major riding mode and the team probability scores of the rider. The combination of video (stream) frame rate and the start frame and end frame when the rider entered or exited the fence also allows to give an estimate of how much time the rider did spend in that zone. In summary, in the final step, the data of the tracked skeleton across different frames within the fence is brought together and summarized, which on its turn can be published. In our video pipeline the fence data is published to a REST API, allowing easy retrieval for other stakeholders within the cyclocross broadcasting world.

3. Results

In the previous section we introduced the full video processing pipeline. In this section we will discuss some applications and results that can be achieved with the video processing pipeline. A first application is the direct application and analysis of the riding lines within the fence. Figure 9 shows an infographic of how this data can be visualized. As mentioned in the previous section, the coordinates of the poses were first mapped on real-life coordinates to optimally represent the true shape of the fence and paths that were followed. In the rideline graph presented in Figure 9 we can see that the blue and orange rider follow a similar path on the track, but the green rider deviates from that similar path near the end of the path. This specific rider made a technical mistake in the sand, causing a deviation from the other riders' lines.

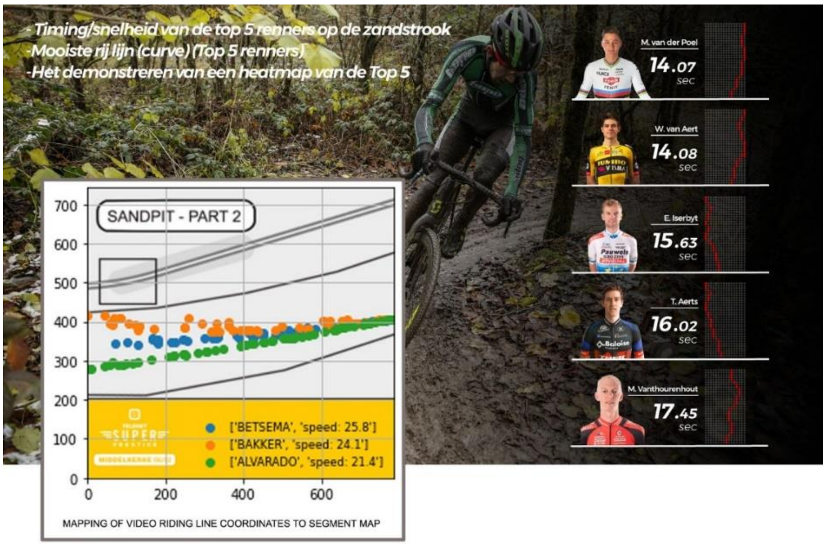

Figure 9: Schematic overview of the produced fence paths of the various riders that went through the fence. Left graph is the actual ride line data (riders go through the fence from the right to the left), the central infographic image is the idea for visualization during video broadcasts.

As mentioned, this raw rideline data is also published to an API, so this information could also be used by video broadcasters to directly incorporate these near-real-time stats in the live video feed (or in race summaries or recaps afterwards). This principle is shown in the infographic in Figure 9, where the ridelines of the top 5 riders are shown on top of the video broadcast.

Lastly, it is also worth mentioning that the application of the introduced principles is not limited to cyclocross and other cycling disciplines only. As a side project the video processing pipeline was reused to analyze a filmed ski downhill run. The ride mode detector was changed by a ski pole and flag detector. The fencing module was also removed from the pipeline but was replaced by a pipeline post-processing element that looked at when the skier's bounding box overlapped with the detected ski flags bounding boxes (see Figure 10). With this information the path the skiers took could be recreated as well as the timings of the segments between consecutive flags that were slalomed.

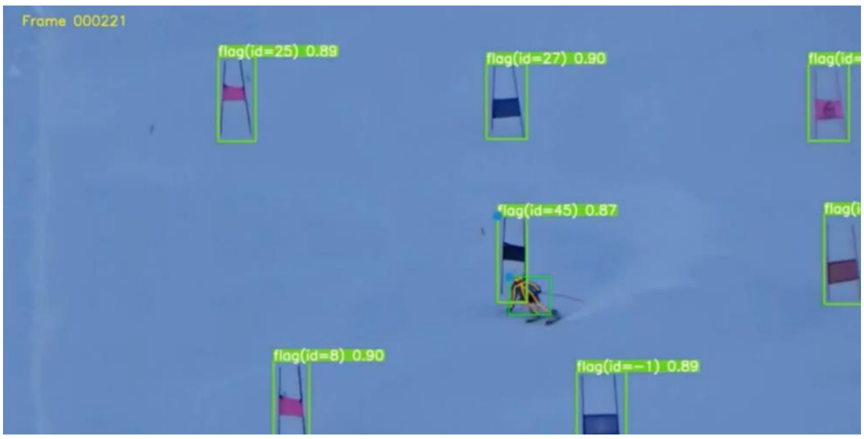

Figure 10: Skier detected within proximity of the slalom flag.

5. Conclusions

In this paper we presented a *video based end-to-end modular and stepwise solution for detection and analysis of cyclocross riding lines*. Human poses, ride modi and team detection analyses were performed on a frame-based level. These pipeline results are merged and post-processed by the pose tracker and were further processed into ridelines by line analysis post processing. The pipeline outputs both the raw metadata output (e.g., tracked skeletons and ride modi) and the processed rideline data. As the pipeline is fully modular, pipeline elements can be added or removed as required. As an example, in sports such as cyclocross and motocross a MyLaps gate timing solution is often used. When the riders wearing a transponder ride over a measurement loop, he/she is registered, and the time of crossing is recorded on the MyLaps system. Integration of these measurement loops at the start and endpoints of the analyzed fences might not only give a conclusive answer about the team of a detected rider in video, but it will also automatically and correctly identify the rider's identity. As a final note it is also worth mentioning that stitching multiple fences that are individually analyzed can yield an even bigger region to analyze.

Funding: This research is funded by the imec ICON project DAIQUIRI (Agentschap Innoveren en Ondernemen project nr. HBC.2019.0053).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

```
last_id ← 0         // When a new skeleton is found it gets last id + 1 as its id
tracked_skeletons ← []      // skeletons that were recently seen in video
n_frames_in_history ← 5

for frame_nr in video_frames do
    poses ← frame_resuls[frame_nr][‘poses’]
    pose_center_xy ← [ [x,y] for pose[‘coordinates’] in poses ]
pose_ids, tracked_skeletons ← map_poses(pose_center_xy, tracked_skeletons)
    tracked_skeletons ← cleanup_old_poses(tracked_skeletons)
    //Distance between tracked and new pose skeletons (omitted for readability)
    dist_matrix = build_distance_matrix(tracked_skeletons, pose_center_xy)
    used_poses ← []
    mapped_ids ← [null] * len(poses)
    for tracked_key in dist_matrix do
        min_index ← index of minimum of dist_matrix[tracked_key]
        min_distance ← value of minimum of dist_matrix[tracked_key]
        distance_threshold ← 25% of diagonal length of the pose at min_index
if min_index not in used_poses and min_distance < distance_threshold do
    tracked_skeletons[tracked_key][‘last_seen’] = frame_nr
    tracked_skeletons[tracked_key][‘coordinates’] =
pose_center_xy[min_index]
            append min_index to used_poses
            mapped_ids[min_index] = tracked_skeletons[tracked_key][‘idx’]
end if
    end for
    not_used_poses ← indices of poses not in used_poses
```

References

1. Kolaja, J., & Ehlerova, J. K. (2019, September). Effectivity of sports timing RFID system, field study. In 2019 IEEE International Conference on RFID Technology and Applications (RFID-TA) (pp. 220-223). IEEE.
2. Deconinck, Guillaume ; Grynczel, Wojciech. Open Source RFID race timing system. Ecole polytechnique de Louvain, Université catholique de Louvain, 2017. Prom. : Schaus, Pierre.
3. Sun, C. I., Huang, J. T., Weng, S. C., & Chien, M. F. (2019). City Marathon active timing system using bluetooth low energy technology. Electronics, 8(2), 252.

4. Arbués-Sangüesa, A., Ballester, C., & Haro, G. (2019). Single-camera basketball tracker through pose and semantic feature fusion. arXiv preprint arXiv:1906.02042.
5. Santhosh, P. K., & Kaarthick, B. (2019). An automated player detection and tracking in basketball game. *Computers, Materials & Continua*, *58*(3), 625-639.
6. Chakraborty, B., & Meher, S. (2013, December). A trajectory-based ball detection and tracking system with applications to shooting angle and velocity estimation in basketball videos. In 2013 Annual IEEE India Conference (INDICON) (pp. 1-6). IEEE.
7. Fang, H. S., Xie, S., Tai, Y. W., & Lu, C. (2017). Rmpe: Regional multi-person pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2334-2343).
8. Tammina, Srikanth. "Transfer learning using VGG-16 with deep convolutional neural network for classifying images." International Journal of Scientific and Research Publications 9.10 (2019): 143-150.
9. Li, P., Xu, Y., Wei, Y., & Yang, Y. (2020). Self-Correction for Human Parsing. IEEE Transactions on Pattern Analysis and Machine Intelligence.
10. Malis, E., & Vargas, M. (2007). Deeper understanding of the homography decomposition for vision-based control (Doctoral dissertation, INRIA).