Assignment Report

*https://github.com/francesco-vaccari/CV-Project

Francesco Vaccari
francesco.vaccari@studenti.unitn.it
239927

Leonardo Giaretta
leonardo.giaretta@studenti.unitn.it
248719

Michele Minniti
michele.minniti@studenti.unitn.it
247168

***Abstract*—This report presents an analysis of detection and tracking algorithms used on a top view court video, stitched from multiple different views, of a volleyball match. The detection of players or of the ball can become problematic, especially considering the frames in which the objects of interest are not visible. As we will see, the problem consists of five different parts: calibrating and adjusting the video, performing and evaluating motion detection on players, performing and evaluating motion tracking, creating an algorithm that can distinguish between players using a color-based technique, and then performing tracking on the volleyball. We will describe our methodology and how it's influenced by the state of the art. We will then explain the results obtained and how our work can be improved.**

I. Problem description

Sports analysis is a very popular task in computer vision, since it can be used to analyze the behaviour of each single player, or the strategy of the whole team compared to the progress of the game. Our project consists in creating a top view stitch of a volleyball match recorded in the Sanbàpolis court, and then performing detection and tracking evaluating the performance. We were provided three starting videos showing different views of the court: top view, center view and bottom view. Each video was recorded with multiple cameras and then stitched together into a single clip. All cameras and videos are synced temporally, although at different frame rates. We stitched and adjusted the videos in order to have a full court view, and then we run our algorithms for the tasks of detection and tracking. Once we managed to achieve that, we implemented a color-based method for team identification of players, and an algorithm for ball tracking. In Figure 1 we can see the starting point from one of the section of the three videos we had at our disposal.

In the next sections we are going to analyze the state of the art, our approach, the results, and some conclusions regarding our work.

II. State of the Art

In this section we are going to analyze briefly some of the techniques we are going to use in our project.

- **Motion Detection:** Refers to the process of identifying a change in position of an object relative to its surroundings or detecting an object moving through a field of view. It is widely used in various applications, such as security systems, video surveillance, automated lighting systems, and many more. Once implemented, a detector can highlight motion by building a bounding box around the object moving, or by drawing motion vectors to indicate movement of specific keypoints in the image [3]. A particular technique of detection called background subtraction takes it the other way around by removing the static background and detecting the moving objects as foreground. This technique works at its best only in environments with few illumination changes and with a static background, for example an indoor facility.
- **Motion Tracking:** It's the process of following the movement of objects over time within a specific environment. Unlike motion detection, which simply detects where movement occurs, motion tracking continuously follows the trajectory and behavior of moving objects. Since it's different from performing detection in every frame of the video, additional data such as speed and direction are collected for analysis or visualization purposes. In techniques such as Optical Flow we perform tracking by analyzing the changes in pixel intensity, while in techniques like the Kalman Filter we make an estimate of the position of an object, and then we correct that prediction using a measurement taken directly from the environment [3].
- **Performance measure:** The performance measures used

Fig. 1. Image from a section of the distorted video

are the following

1) **Precision**: The ratio of true positive detections (correctly identified objects) to the total number of detected objects. Precision indicates the accuracy of the detection algorithm in identifying only the correct objects without producing false positives.
2) **Recall**: The ratio of true positive detections to the total number of ground truth (GT) objects. Recall measures the algorithm's ability to detect all desired objects within the frame.
3) **IoU** (Intersection over Union): A metric used to evaluate the overlap between the predicted bounding box and the GT box. IoU thresholds are varied to assess the sensitivity of the algorithm to different levels of overlap.
4) **mAP** (mean Average Precision): A comprehensive metric that averages the precision over multiple IoU thresholds, providing a single score that reflects the algorithm's performance across different overlap requirements.
5) **F1 Score**: The harmonic mean of Precision and Recall that provides a single metric that balances both accuracy and completeness.
6) **MOTA** (Multiple Object Tracking Accuracy): A comprehensive metric that takes into account false positives, false negatives, and identity switches. It ranges from $-\infty$ to 1, where higher values indicate better performance.
7) **MOTP** (Multiple Object Tracking Precision): Measures the average alignment between the predicted and GT bounding boxes. It ranges from 0 to 1, with higher values indicating better spatial precision.

III. Our approach

In this section we are going to explain how we managed to obtain the results described later.

A. Top view stitching

Creating a single view of the court has proven to be more challenging than we originally anticipated. At our disposal we had three different videos of the Sanbàpolis facility, each one taken by multiple cameras. The goal was to create a single view by stitching those three videos. Some areas of the court were recorded by more than one camera, and since there wasn't a precise calibration we had many object duplicates in our videos. But in the end we managed to obtain a decent result with the following work:

1) We cut the videos to take only a much shorter clip to reduce the computational burden of following steps, especially considering that most of the video provided didn't actually show players playing volleyball;
2) We then extracted from each video the four horizontally stitched views, keeping only the two central ones. Doing so, we ended up with a total of two cuts (left cut and right cut) of the same court for each video;

Fig. 2. Example of usage of the software [1]

3) We adjusted the frame rate of the central video to match the frame rate of the others, thus preparing for the final stitched version;
4) We combined the left and right splits after a horizontal translation to solve partially the object duplication problem;
5) We dewarped our combined splits with [1] to apply the transformation desired: this allows to remove duplicated objects and to fix non-converging lines by warping the frame using Delaunay triangulations [2]. An example of this can be viewed in the fig. 2;
6) We applied the transformations obtained in the previous step and we finally stitched the top, central and bottom videos together.

B. Detection

After having obtained the final top view court video, the next step was to implement motion detection and analyze the performance. We decided to manually annotate the position of each player on the field, creating an evaluation dataset that will be used to judge the quality our work. The annotations folder contains a file for each player on the field (and also for the ball), and each annotation references the frame of the video, and the positions $x1, x2, y1, y2$ of the bounding box surrounding the player. If the player is not present on the frame analyzed, the four points of the bounding box were set to $0, 0, 0, 0$. As preprocessing, we applied erosion and dilation to apply the opening operator on the mask [3]. We considered many techniques such as frame differencing, and both standard and adaptive background subtractors (FD, BGSUB and ABGSUB) [3]. We also searched some methods present in the open-cv library [4], such as the background subtractors based on K-Nearest Neighbours [6] or on a Mixture of Gaussians model [7]; or then again some supplementary methods based on other algorithms [5]. For example, we used

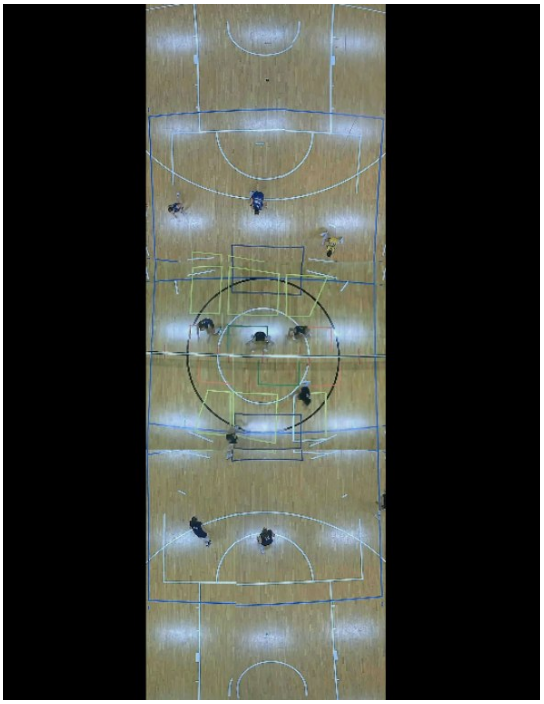

Fig. 3. Image of the final result obtained by the top view stitching, after starting from three separated videos as shown in fig. 1

the algorithm based on counting by Sagi Zeevi [8], or the GMG which combines different methods such as background subtraction, Bayesian segmentation and Kalman Filters [9], the GSOC algorithm, not originated from any paper [10], or a background subtraction method that uses local singular value decomposition (SVD) binary pattern [11].

In the end we chose to analyze the performance of:

- **MOG2**: An advanced background subtraction method using a Mixture of Gaussians to model the background and detect moving objects. This method is robust against changes in lighting and offers shadow detection.
- **KNN** (K-Nearest Neighbors): A background subtraction algorithm that uses a K-Nearest Neighbors approach to classify pixels as foreground or background.
- **GSOC**: A robust method that improves upon traditional background subtraction by incorporating statistical modeling and temporal consistency.
- **CNT, GMG, LSBP**: Additional background subtraction methods that vary in their approach to modeling the background, temporal dynamics, and noise handling.

The video was processed frame by frame, applying the selected motion detection algorithm to generate a binary mask highlighting moving objects. The mask was then processed to improve the detection quality, after which the bounding boxes were extracted to indicate detected objects. The extracted bounding boxes were compared against ground truth annotations using the various performance metrics explained above.

C. Tracking

For motion tracking we took into account some of the main algorithms studied in the lectures [3], such as Optical Flow and the Kalman Filter. Furthermore, open-cv provides some already implemented trackers, such as Discriminative Correlation Filter with Channel and Spatial Reliability tracker (CSRT) [12], Multiple Instance Learning (MIL) tracker [13], Kernelized Correlation Filter tracker (KCF) [14], Distractor-aware Siamese Networks for Visual Object Tracking (DaSiamRPN) tracker [15], Generic Object Tracking Using Regression Networks (GOTURN) [16], and two lightweight DNN-based general object trackers called Nano [17] and Vit [18]. We should mention that we were not interested in implementing real-time trackers because we wanted to explore and experiment with different approaches. We initially intended to use SIFT and similar feature extraction methods to enhance tracking quality, but this approach proved ineffective due to the highly textured floor, which introduced too many edges and corners, complicating the identification of distinctive keypoints.

In the end we implemented three kinds trackers: two based on Optical Flow and one based on the Kalman Filter.

- Optical Flow: OpenCV provides three implementations of optical flow, two based on Dense Optical Flow, and one based on Pyramidal Lucas-Kanade optical flow. We decided to use all three in the same fundamental way. The trackers receive in input the initial locations of the players we want to track, and for each of them we sample a number of points that fall inside the bounding box provided. We adopted a gaussian distribution for sampling random points because it is more likely for the player to be close to the center of the bounding box. Then, each time the tracker receives the next frame, it tries to compute the optical flow for all the points sampled for each player. If the tracker fails to compute the optical flow for a specific point, that point is marked as not valid and will be discarded. For all other points, we apply the optical flow found by translating them with the magnitude and direction calculated. After having done so, we know roughly where the player is estimated to be, meaning that we can construct the minimal bounding box that contains all the points for that player. But this is not enough, in fact, we also use a motion detector to look for movement in the estimated position of the player and we combine the bounding box coming from the motion detector with the box previously constructed to produce the final output of the tracker. After that, we resample all the points that are now outside the new final bounding box. If the motion detector indicates that there is no movement in the estimated position of the player, then it means that the tracking has probably been lost. In this case, we try to reinitialize the tracker by looking for a match among the boxes returned by the motion detector. To find a match we use an history of box sizes, box positions, and color histograms saved in the previous 36 frames. By comparing this information with boxes that do

not correspond with the position of other tracked objects, we try and recover from players leaving the frame or getting occluded by other elements in the scene.
- Kalman Filter: Just like before, the tracker receives the initial position of each player to track, but this time we take only the center of the box to use it for predictions. The prediction step of Kalman Filter is straight-forward: given the initialized position of the point, we receive the new estimated position of the player. The difficult part is the correction step since we have no way of knowing the actual position of the tracked player. For this purpose, we decided to use Yolo to identify the position of the player closest to the estimate computed, and use this information to perform the correction step. If the player recognized by Yolo is too far or if there is no player detected at all, we skip the correction step for a maximum of 36 consecutive frames, after which we try to recover tracking by matching with other bounding boxes given by a motion detector, like in the optical flow trackers. The Yolo used was provided by Niccolò Bisagno since it needs to be finetuned to detect correctly volleyball players from an overhead view like ours.

As dataset we were still using the same preprocessed video used for detection, alongside the same ground truth annotations. Each annotation provides the bounding box coordinates for the objects in the first frame, which serve as the initialization for the trackers.

The tracking results were captured for each frame, and the metrics explained above were computed to quantify the performance.

D. Color based identification

For this task our approach was based on the use of color histograms, background subtraction and the k-means clustering algorithm provided by OpenCV. The main problems we have encountered and the solutions we have found while tackling this task were:

- The color histograms of the players were too similar to each other, mainly because of the inclusion of the floor. Our solution to this problem has been the application of a mask obtained through background subtraction to the histogram extraction and the removal of purely black pixels from histograms.
- The labels assigned to each team by the clustering algorithm are completely random, making the solution thus far only a clustering method and not a mean for identification. Our solution has been to calculate the mode label of each team, allowing us both to track the correctness of each team member assignment and to evaluate how often the two teams were assigned the same label.

The boxes used during the evaluation of this task have been extracted from our annotations in order to avoid the accumulation of inaccuracy between the various tasks. The ordered nature of the annotations has allowed us to use indexes to differentiate between players, making the evaluation process particularly easy.

E. Volleyball tracking

This task has revealed to be very difficult to tackle for the following reasons:

- The ball is very small in size when compared to the size of the court.
- The ball is often blurred with unclear edges and its shape is elongated due the constant movement.
- The color of the ball can be hardly distinguished from the background, especially when it is moving for the reason explained above.
- The ball is very often not in frame. In fact, the evaluation dataset tells us that about 75% of the time the ball is not visible.

For all these reasons, performing tracking is really not an easy task. Techniques that analyze color histograms, edges and gradients are not a viable option. In the end we decided to simply use a pretrained Deep Learning model to solve the task.

The very first simplification is due to how the problem is constructed: since we know that there can be only one ball in frame at any given point, we just need to find its position. For this purpose, we use 2 versions of Yolo: the finetuned one provided by Niccolò Bisagno, which is able to recognize the class *Ball*, and YOLOv8, which is able to detect the class *Sports Ball* (class n.32).

We used these models giving as input images in their full size, and since this is very computationally expensive, we tried to speed up the process by cropping the frame and giving only the cropped portion to the model to look for the ball. If the model is able to find a match, the crop size is reduced and centered around the detected ball, otherwise the crop size is increased. Naturally, the first crop is simply the first frame of the video.

IV. Experimental results

In this section we analyze the experimental results obtained in each part of the project.

A. Detection

Considering the large quantity of algorithms chosen and possible input parameters, we evaluated using precision, recall, and mAP across different IoU thresholds to ensure robust and consistent performance analysis. The results can be viewed in table I and table II.

In all cases BGSUB performs better, with KNN and CNT having similar results, while MOG2 performs always the worst. However, as the IoU threshold increases, we can see a consistent and constant decrease in the performance for all algorithms. We conclude that BGSUB is the best algorithm analyzed: the cause of the superior performance of such a simple algorithm is likely the unchanging illumination and the background in the context of an in-door sport.

B. Tracking

To evaluate our trackers we used as performance measures average Precision, Recall, and F1 Score that indicates the effectiveness in maintaining object identities across frames. We also used MOTA and MOTP to provide additional insights into the overall tracking accuracy and spatial precision, highlighting areas for potential improvement. The results can be viewed in table III.

As can we see, CSRT has the highest Recall measure, indicating its ability in detecting true object, however, other measures such as the high rate of false positives, suggest that it often misidentifies objects, individuating something as object probably too often.

DaSiamRPN has lower precision, recall and f1 score with respect to CSRT, meaning that it struggles more on misses and false positives.

MIL is overall slightly better but it can't reach the performances of CSRT.

NANO has the highest value of ID switches showing that it struggles at maintaining the same label for players across the field. However, it has the highest MOTP value indicating that the tracking of multiple objects is spatially precise.

VIT is overall balanced despite not showing a really good performance across all IoU values.

The DIS Optical Flow tracker has very low values for precision, recall and f1 score, but it demonstrates a good MOTA value.

The Farneback Optical Flow tracker has the highest MOTA and a very good MOTP value suggesting high spatial precision in individuating multiple objects in movement at once.

The Pyramidal Lucas-Kanade Optical Flow tracker has an equal value of MOTP, but a slightly lower value of MOTA. However, it has the highest precision value, and pretty good values also for recall and f1 score, being probably the better trade off between each algorithm.

The Kalman Filter tracker has the lowest number of ID switches and a high MOTA, however, the low F1 score and MOTP suggest that it identifies correctly the objects but has difficulty in maintaining the tracking on previously correctly identified objects.

C. Color Based Identification

For Color Based Identification we decided to measure the following metrics: number of frames where the mode of the labels was identical for the two teams, number of frames where the mode of the label was different between the two teams, and number of correctly (True) and incorrectly (False) assigned labels for each of these two cases. We also computed these values over the whole set of frames.

The most important metric could be considered the combination of number of frames where the teams have different labels and the respective True/False values, as they give a clear indication of how often the algorithm correctly identifies the players' team. The results can be visualized in table IV. We analyze the K-Means clustering algorithm for different combinations with background subtractions at different thresholds and considering in different ways the color histogram, each measure reports also the total number of frames (# frames).

As we can see from the results, overall the k-means clustering works best when most of the background noise, namely the floor, can be eliminated through the use of background subtraction. The fine-tuning process of the threshold value has also played a fundamental part in reaching optimal results, meanwhile simple manipulation over the histograms didn't seem to be able to distance the feature points or improve the identification.

D. Volleyball Tracking

For the volleyball tracking task we decided to use as performance measures the count of True Positives (TP), False Positives (FP) and False Negative (FN), then applied to calculate precision, recall and F1 score; the results can be viewed in table V. The Finetuned models refer to the weights provided to us by Niccolò Bisagno.

Regarding results obtained, we recognize that all the models evaluated clearly struggle with ball detection, probably due to the reason explained in section III-E. Surprisingly, the best performing model is YOLOv8n, however, it only reaches precision, recall and f1 scores values that do not go over 14%. There are some outliers such as a recall value of 19%, but is clear that our solution struggles in reaching good performances when compared to the ground truth.

V. Conclusion

In conclusion, our work in detection and tracking, with and without color identification, shows room for improvement.

In the context of motion detection, the algorithm based on Background Subtraction performed significantly better than all the others, while in motion tracking, methods based on Optical Flow generally outperformed the rest. For color tracking, we observed that k-means clustering achieved better performance when combined with Background Subtraction.

However, we struggled with ball tracking, which did not produce the desired results. With better visual quality or improved calibration our results would probably improve. Overall, each part of our algorithm performed as expected.

Our work could be considerably improved with higher-quality video or by developing a model specifically designed to track the ball in our particular visual scenario. The models we used are not optimized for a top-down view, and this, coupled with the imperfections in our stitched video, resulted in motion tracking that was not entirely satisfactory.

References

[1] "Camera Dewarping." Mattia Franzil, Tommaso Canova. https://github.com/cannox227/camera-dewarping

[2] "Delanuay triangulation" Wikipedia article. https://en.wikipedia.org/wiki/Delaunay_triangulation

[3] "Lectures of CV course" Website. https://didatticaonline.unitn.it/dol/course/view.php?id=38741

[4] "Background subtractures from cv2" Website. https://docs.opencv.org/3.4/de/de1/group__video__motion.html

[5] "Improved Background subtractures from cv2" Website. https://docs.opencv.org/4.x/d2/d55/group__bgsegm.html

[6] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern recognition letters, 27(7):773–780, 2006.
[7] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 2, pages 28–31. IEEE, 2004.
[8] Sagi Zeevi. "Background subractor CNT". https://github.com/sagi-z/BackgroundSubtractorCNT
[9] Andrew B Godbehere, Akihiro Matsukawa, and Ken Goldberg. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In American Control Conference (ACC), 2012, pages 4305–4312. IEEE, 2012.
[10] https://docs.opencv.org/4.x/d4/dd5/classcv_1_1bgsegm_1_1BackgroundSubtractorGSOC.html#details
[11] L. Guo, D. Xu, and Z. Qiang. Background subtraction using local svd binary pattern. In 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 1159–1167, June 2016.
[12] Alan Lukezic, Tom'as Voj'ir, Luka Cehovin Zajc, Jir'i Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. International Journal of Computer Vision, 2018.
[13] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 983–990. IEEE, 2009.
[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In proceedings of the European Conference on Computer Vision, 2012.
[15] "Distractor-aware Siamese Networks for Visual Object Tracking", "Zheng Zhu and Qiang Wang and Bo Li and Wei Wu and Junjie Yan and Weiming Hu", 2018.
[16] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In European Conference Computer Vision (ECCV), 2016.
[17] https://docs.opencv.org/4.x/d8/d69/classcv_1_1TrackerNano.html#details
[18] https://docs.opencv.org/4.x/d9/d26/classcv_1_1TrackerVit.html#details
[19] https://datascientest.com/en/you-only-look-once-yolo-what-is-it

Algorithm	IoU > 0.1		IoU > 0.2		IoU > 0.3		IoU > 0.4		IoU > 0.5	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
MOG2	0.47	0.47	0.42	0.43	0.37	0.38	0.32	0.33	0.26	0.27
KNN	0.52	0.49	0.48	0.47	0.44	0.43	0.40	0.40	0.36	0.36
CNT	0.49	0.50	0.46	0.49	0.44	0.47	0.41	0.43	0.37	0.40
BGSUB	**0.54**	**0.52**	**0.52**	**0.51**	**0.51**	**0.49**	**0.46**	**0.45**	**0.43**	**0.42**

TABLE I

TABLE WITH PRECISION AND RECALL RESULTS FROM THE MOTION DETECTION ALGORITHMS FOR EACH PARAMETER FOR THRESHOLDS UNTIL 0.5.

Algorithm	IoU > 0.6		IoU > 0.7		IoU > 0.8		IoU > 0.9	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
MOG2	0.20	0.21	0.13	0.13	0.02	0.03	0.0001	0.0001
KNN	0.31	0.31	0.23	0.23	0.06	0.07	0.0001	0.0001
CNT	0.32	0.35	0.26	0.28	0.12	0.13	0.003	0.004
BGSUB	**0.38**	**0.38**	**0.32**	**0.32**	**0.16**	**0.16**	**0.007**	**0.007**

TABLE II

TABLE WITH PRECISION AND RECALL RESULTS FROM THE MOTION DETECTION ALGORITHMS FOR EACH PARAMETER FOR THRESHOLDS FROM 0.6.

Tracker	# FP	# FN	Precision	Recall	F1 Score	# ID Switch	MOTA	MOTP
CSRT	9886	3013	40.4%	**66.7%**	**48.8%**	61	0.40	0.39
DaSiamRPN	13866	3607	21.1%	45.3%	26.9%	169	0.18	0
MIL	9557	3271	22.8%	53.9%	28.5%	110	0.40	0.47
NANO	7832	5480	37.0%	47.9%	40.1%	205	0.38	**0.59**
VIT	5431	7289	26.8%	22.0%	23.0%	166	0.41	0.44
Dense Inverse Search (DIS) Optical Flow	2402	972	6.2%	8.1%	6.5%	66	0.84	0.23
Farneback Optical Flow	**1475**	**702**	12.9%	15.5%	12.9%	53	**0.89**	0.57
Pyramidal Lucas-Kanade Optical Flow	1711	1914	**47.3%**	39.5%	37.2%	63	0.83	0.57
Kalman Filter	1568	858	8.7%	12.4%	9.5%	**39**	0.88	0.39

TABLE III

PERFORMANCE MEASURES FOR TRACKING ALGORITHMS.

Algorithm	Total frames			Same mode labels			Different mode labels		
	# frames	True	False	# frames	True	False	# frames	True	False
k-means clustering	1810	10.41	1.59	1228	10.78	1.22	582	9.64	2.36
k-means + bg subtractor thresh 50	1810	9.39	2.61	819	9.83	2.17	991	9.02	2.97
k-means + bg subtractor thresh 60	1810	9.63	2.37	**599**	**9.55**	**2.45**	**1211**	**9.67**	**2.33**
k-means + bg subtractor thresh 70	1810	9.49	2.51	847	10.29	1.71	963	8.79	3.21
k-means + bg subtractor thresh 50 + only lower half of histogram	1810	9.46	2.54	834	10.13	1.87	976	8.88	3.12

TABLE IV

PERFORMANCE MEASURES FOR COLOR TRACKING ALGORITHMS WITH AVERAGE TRUE/FALSE COUNTS.

Version of YOLO	Inference Image Size	# TP	# FP	# FN	Precision	Recall	F1 Score
YOLOv8x	3872	31	323	329	8.76%	8.61%	8.68%
YOLOv8m	3872	28	485	291	5.46%	8.78%	6.73%
YOLOv8n	3872	**46**	302	306	**13.22%**	13.07%	**13.14%**
Finetuned	3872	4	774	253	0.51%	1.56%	0.77%
Finetuned	3200	19	1233	104	1.52%	1.54%	2.76%
Finetuned	2560	23	1261	98	1.79%	**19.00%**	3.27%
Finetuned	2240	22	1254	103	1.72%	1.76%	3.14%
Finetuned	1920	2	**760**	264	0.25%	0.76%	0.38%
Finetuned	1600	23	1256	104	1.76%	18.11%	3.21%
Finetuned	1280	23	1283	104	1.76%	18.11%	3.27%
Finetuned	960	18	1312	**86**	1.35%	17.30%	2.51%
Finetuned	640	19	1268	109	1.48%	14.84%	2.68%

TABLE V

PERFORMANCE MEASURES FOR BALL TRACKING ALGORITHMS.