Foosball Robot Object Detection and Angle Estimation

Joseph Lundy

Abstract—This project looks at implementing a visual system for a foosball robot. This system must track positions of the ball and foosmen, and also estimate the rotation of foosmen. Our approach was to train a YoloV7 model to detect the balls and foosmen, and use the bounding boxes of the foosmen to estimate the rotation angle. To generate the training data, a single video was used, and passed into the SAM2 segmenter to obtain masks which were used to derive bounding boxes. The Yolov7 model was trained on 100 iterations, and generalized well in other videos. To obtain data for angle estimation analysis, ArUco markers were placed on the rod ends, and two videos were captured - an overhead video to record the bounding boxes of the foosmen, and a side-view video to record the ArUco markers. The ArUco markers were used to compute the 'ground-truth' angle, and this was compared against the angle inferred by the bounding boxes. We found that the arcsine method can give a decent approximation of the rod angle, and that if the foosman quadrant is known, the foosman angle can be estimated across the full 360 degrees of motion. As well, the trend of the bounding box center can indicate the sign of the footman's rotation angle. However, this simple approximation requires calibration and results in some nonlinearities. In conclusion, we found that the Yolov7 model was ideally suited as an real-time object detector, and for future work, the Yolov7 model should be retrained to classify the foosman quadrant as well.

Index Terms—ArUco, YoloV7, SAM2, Object Detection, Object Tracking

1 Introduction

As part of a previous project, we constructed a simple robotic system for a miniature foosball table [1]. In order for the robot to play against a human player, at minimum, it needs to be able to estimate the position of the ball. Since the robot is actuating two of the rods, we can assume that its actuators also have corresponding sensors to measure the angle of the rods it is attached to. However, due to manufacturing imperfections, there is a lot of variance in the actual mounting angle of the foosmen, and it would be ideal if another measurement of the foosmen orientation can be obtained. As well, it would be beneficial for the robot AI to receive information of the pose of the opponent's foosmen, to allow effective passing and aiming. An overhead camera system is an ideal sensor for this measurement problem, as it requires minimal changes to the board layout and can easily be mounted to different foosball tables. Previously, we implemented a rudimentary ball tracking system using color detection [2]. However, this is not a robust tracking method across different lighting conditions, and is unsuitable for tracking the position and orientation of the foosball rods. In Figure 1a, we see the annotations output by this color tracker, and 1b shows the computed color masks. The masks are quite imperfect. While adequate for ball tracking, they require tuning for lighting conditions and adjustments to the algorithm settings for different colors. It also would not generalize well for a patterned ball. For tracking the foosmen, the color masks are highly unsuitable. A more robust approach for foosmen and ball detection is required, as well as an algorithm that can be used to measure the foosman rotation angle from the video feed. We discuss various approaches for these two problems in the next section.

Tracker	FPS	Performance
CSRT	44	Very good - Tracks ball until very last frames
Boosting	62	Mediocre – loses ball after occlusion; tracks foosman poorly
KCF	69	Very poor - Loses ball and foosman very quickly
MIL	26	Mediocre – loses ball immediately; can track foosman
TLD	29	Good – can track ball throughout, very noisy detection; performs worse on foosman
Median Flow	80	Poor – loses ball immediately; can track foosman but detection box grows
MOSSE	80	Very poor – loses ball immediately; loses foosman after rotation

TABLE 1: Performance of correlation-based trackers on single foosball

2 Related Work

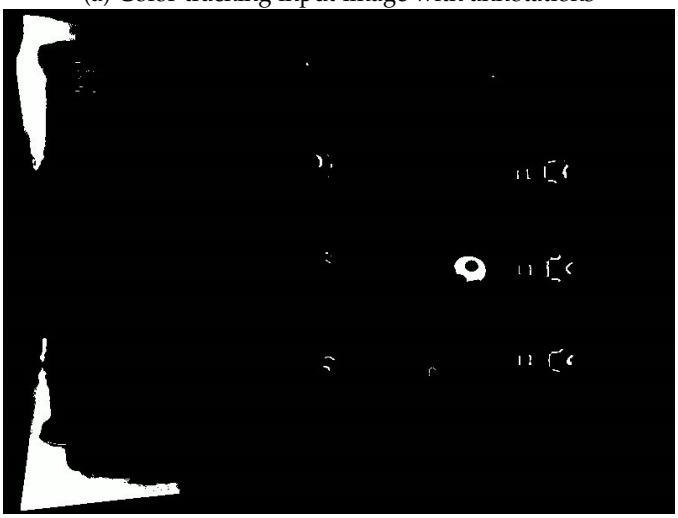
The first problem our system must solve is the tracking problem. In [3], the authors present an overview of tracking algorithms, dividing them into 2 categories:

1) Correlation-filter based tracking
2) Deep learning based tracking

While some correlation-filter trackers can be quite computationally efficient, many of these methods have poor performance handling occlusions. We performed an experiment on the following algorithms to see how well they tracked the foosball in a sample video. Table 1 summarizes the performance and typical FPS obtained for each method. These tests were performed on a single GPU machine. From this experiment, we see that none of the correlation-filter approaches are suitable for this application.

The SORT [4] algorithm is a real-time tracking method, which uses a CNN object detector to obtain the bounding

(a) Color tracking input image with annotations



(b) Color tracking detection mask

Fig. 1: Outputs of color tracking algorithm

boxes of objects, then simply uses the Kalman Filter and Hungarian algorithm for tracking. This algorithm is highly suitable for our application, but requires a suitable object detector model. One very fast object detector model is YoloV7 [5]. While it would need to be retrained, we found that it is a highly suitable candidate for transfer learning, as it was able to weakly detect the foosball and foosmen with two of its pre-existing classes: 'person' and 'sports ball'. This can be seen in Figure 2.

The second problem we need to solve is the angle estimation problem. ArUco markers [6] are a highly effective method for visual pose estimation. One simply needs to place a marker on the target object and calibrate their camera so that the camera matrix and distortion coefficients are known. When an image of the target object is captured, the marker is detected in the image and from the black and white pattern its 3D orientation can be estimated. If the size of the marker is known beforehand, its position to the camera can also be estimated. This would be a highly suitable method for measuring the angle of the foosmen, but there are two problems with this approach. The first is

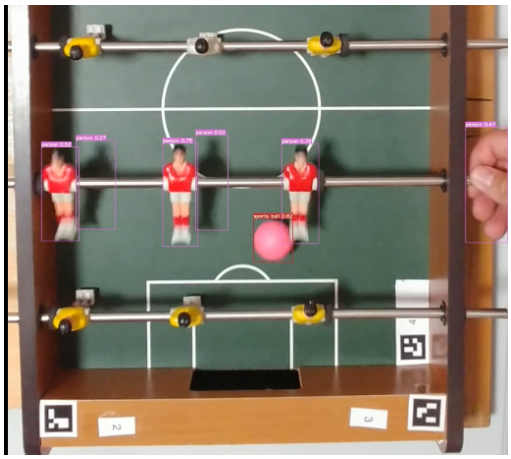


Fig. 2: Detected objects in foosball table image from pre-trained YoloV7 model

that it requires additional modification to the foosball table, which is undesired. The second is that since the foosmen can make 360 degree rotations, a single marker would not be sufficient to fully capture the foosman pose. Even if a second marker were placed on the rear of the foosman, there would be a 'blind spot' when the foosmen are in the upright position. Having multiple markers on each foosman also makes the angle estimation more complicated, and requires extra calibration of the orientation of each marker on the foosman. Nonetheless, as we will discuss below, we made use of ArUco markers to obtain a 'ground-truth' measurement of the rod angles for our angle estimation analysis.

3 Proposed Method

3.1 Object detection and tracking

From our conclusions above, we propose using the SORT algorithm with YOLOv7 as the object detector. We will retrain the model on a previously captured video, to detect two classes:

- 0 - ball
- 1 - foosman

Due to time constraints for this project, we will focus only on the detection performance of our model, and leave aside tracker implementation for future work.

3.2 Angle estimation of foosmen

After examining the layout of the robotic system, we determined that it is possible to directly infer the angle of the foosman with the width of the detection bounding box, using the arcsine method. Figure 3 provides an illustration of the trigonometry. There is a major assumption here which is that the camera perspective is relatively level, and that its incline can be ignored. This can actually be accounted for if we measure the relative position of the camera to the table beforehand or automatically with foosball markers, but for our analysis we will see what accuracy can be obtained with this imperfection. The second assumption is that the camera has relatively little distortion. Distortion can be corrected if the parameters of the camera model are known. For our analysis, we assume the camera has relatively low distortion.

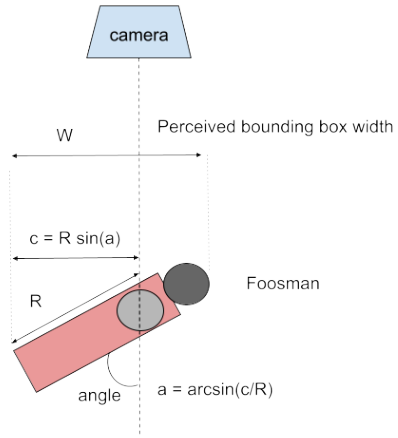


Fig. 3: Heuristic for estimating foosman angle

The proposed angle detection algorithm has two stages:

1) Online Calibration stage
2) Detection and estimation stage

The online calibration stage works as follows:

1) Rotate each rod 360 degrees
2) Record the bounding boxes
3) Record the maximum and minimum box extremities for each foosman
4) Record the maximum and minim box width
5) Compute the rod center using the recorded maxima and minima

The estimation algorithm works as follows:

1) Obtain the foosman bounding box
2) See if the center-x of the bounding box is above or below the rod center - this will tell you the whether the rotation angle is positive or negative with respect to the vertical z-axis
3) Compute angle from

$$c = W - box_{min}$$

$$c = R = box_{max} - box_{min}$$

$$\theta = \arcsin(\frac{c}{R})$$

Taking another look at the estimation algorithm, one will realize that it will compute the same angle if the foosman is rotated by (θ) or ($\theta+90$), the bounding box width will be the same at these corresponding angles. We propose training a detector to directly classify the foosman rotation quadrant. From visual inspection of the photos, this classification task seems plausible as the foosmen are distinctive from the front and back. The positive and negative information can be inferred either using the bounding box center, or from a model detecting a foosman facing 'up' or 'down'. To determine whether the angle is greater or less than 90 degrees, a model could detect whether the bottom of the foosman's feet are visible or not. Given the quadrants the angle is computed as follows:

if q == 0 **then**
    theta = theta
else if q == 1 **then**
    theta = 180 - theta
    theta = -180 + theta
else if q == 3 **then**
    theta = -theta
end if

Due to time constraints, we did not train a model for this particular classification problem. In our later analysis, we look at the usefulness of the center-x method, and for our estimation of the 'full-rotation' accuracy we assume the quadrant of the foosman is known.

4 Datasets

4.1 Object detection datasets

4.1.1 Training data

To generate the data for retraining the YoloV7 model, a sample video was chosen. We used the SAM2 [7] segmentation model to detect masks for the foosmen and foosball. Due to memory constraints from using a single GPU machine, the labeling task was split up into 3 groups of 3-4 objects. The algorithm for labeling worked as follows:
For each object group:

1) Take the first frame of the video
2) Mark 2 coordinates to initialize the mask of each object
3) Pass in the coordinates and all image frames to the SAM2 video, which returns masks of the objects in each frame
4) For each frame:
    - Reject outliers coordinates in the mask which deviate by more than 5 standard deviations from the mask center
    - Bounding box coordinates are the maximum and minimum coordinates of the pixels in the mask

This method was used to generate approximately 501 image labels from the selected training video, which were split into 400 training images, and 50 validation and 51 test images.

4.1.2 Test data

Several videos were captured in order to test the ability of the re-trained YoloV7 model to generalize. In total, 7 additional videos were used to generate separate test data sets. The first labeled video was the cellphone video captured in for the bounding box and angle estimation analysis. The second video was captured with a webcam that has significant distortion and reduced resolution. The other 6 videos were captured using various ball colors - pink (the same color used in the training videos), orange, light orange, green, blue and white (with a soccer ball pattern). In order to reduce labeling time, a sequential subset of images were selected from each video. The videos were converted into JPEG image frames using the ffmpeg tool. For the 1st video, every 10th frame was selected in order to sample the full rotation motion of each rod. For the webcam video, every 50 frames were sampled. For the ball color videos, 10 evenly spaced frames were selected from each video. Initially we

attempted to use the SAM2 model to automatically generate object masks for the test videos. However, the SAM2 model can be quite sensitive to initial prompts, resulting in unsatisfactory bounding box computations. For this reason, we opted to use the Roboflow platform to manually label the test images. The performance of the model on each test set is discussed in Section 5.

4.2 Obtaining bounding boxes and ground truth angles for angle estimation

To obtain angle data for the angle estimation analysis, ArUco markers were placed on the end of each of the four rods, as well as a fifth ArUco marker on the table wall. In order to perform pose estimation of the ArUco markers, a calibration routine was performed with a camera, using a checkerboard pattern. A set of calibration images were taken, with the foosmen aligned to be perpendicular with respect to the table surface. From the relative positions of the rod markers and the 'reference' marker, a calibration rotation was computed. The rotation of the foosman relative to the table is given by:

$$R_{calibration} = R^{-1}_{reference} R_{aligned}$$

$$R_{rod} = R_{ref} R_{marker}$$

$$R_{foosman} = R_{calibration} R_{rod}$$

An experiment was performed capturing 2 simultaneous videos. The first camera was positioned to capture the rotation of the ArUco markers on the rods. The second camera was placed overhead to capture images of the foosmen. A simple frame alignment between videos the videos was performed. Ground truth rod angles were computed for each frame from the ArUco markers using the equations described above. Using a similar method as described in 3.1, ground truth bounding boxes were computed for each foosman. These bounding boxes were used to compute the 'best possible' angle estimation using the methods described in 3.2. Due to time constraints, we did not use YoloV7 generated bounding boxes for the angle estimation analysis.

5 Experimental Results

5.1 Object detection

5.1.1 Training

The YoloV7 model was trained on 400 training images over 100 iterations, using the base yolov7-tiny parameters. All metrics were evaluated based on a confidence threshold of 0.001 and an intersection over union of 0.64. Tables 2 and 3 shows the effect of one training setting, which enables rotation of input images when performing training. Two models were trained, the first without this setting, and the second with this setting. These are compared on two test videos. The first test video has a horizontal view of the foosball table, and captures the full 360 degree rotation of each rod. The second video was captured with a webcam with high distortion and lower resolution. As can be seen, enabling this setting resulted in a large improvement in all performance metrics, especially recall and mean average precision. The rest of the results use the performance of the second model (Model 2), which rotated input images during training and had much better performance detecting the foosmen. The performance of Model 2 on the initial dataset is summarized in Table 4.

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Model 1	0.886	0.795	0.812	0.396
Model 2	0.973	0.858	0.943	0.483

TABLE 2: Detection performance - all classes - for test video 1 (rod rotation)

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Model 1	0.841	0.517	0.685	0.321
Model 2	1	0.92	0.993	0.65

TABLE 3: Detection performance - all classes - for test video 2 (warped webcam)

Dataset	Precision - all classes	Recall - all classes	mAP@0.5	mAP@0.5:0.95
Train	0.996	0.991	0.996	0.802
Test	0.999	0.989	0.997	0.83
Val	0.988	0.968	0.984	0.802

TABLE 4: Performance on training data

5.1.2 Generalization on different videos

Table 5 summarizes the performance obtained on the first test video data set, which captured the full 360 rotations of each rod. Here we see the precision, recall and mAP@0.5 scores remain high, but the mAP@0.95 suffers.

Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95
all	0.976	0.929	0.969	0.457
ball	0.979	1	0.995	0.432
foosman	0.973	0.858	0.943	0.483

TABLE 5: Performance on rotation test video

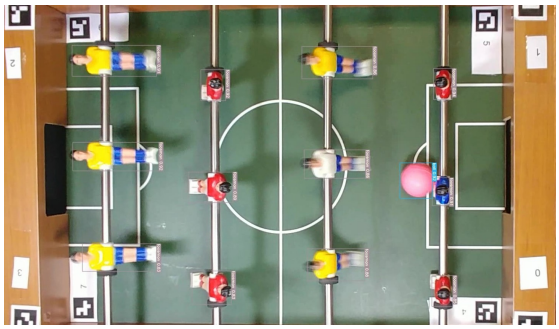
Table 6 summarizes the performance was measured on a video captured using the webcam . Here we see that the precision for the precision and recall for the foosman class remains high, but the precision of the ball detection is impacted. Curiously, the model performs better on this video on the mAP@0.5:0.95 metric. After examining the footage, it was found that for one of the rods in the rotation video, the bounding boxes of the foosmen at large angles only captured half of the actual foosman. Curiously, the model worked well capturing the bounding boxes for the foosmen on the other three rods. The output of the retrained model on a cellphone image and webcam image are shown in Figure 4

Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95
all	0.827	0.938	0.936	0.627
ball	0.655	0.957	0.879	0.604
foosman	1	0.92	0.993	0.65

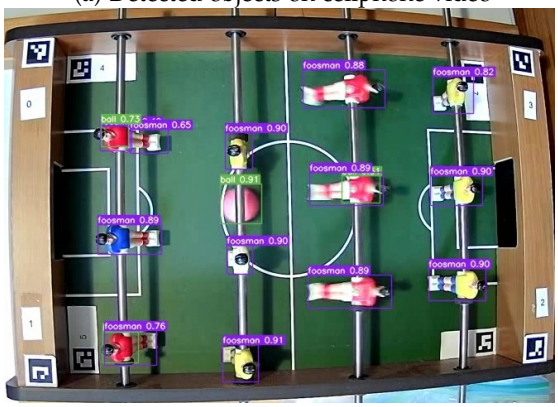
TABLE 6: Performance on high-distortion webcam video

5.1.3 Generalization on different ball colors

Six additional test videos were captured with different ball colors. Table 7 summarizes the model performance for each



(a) Detected objects on cellphone video



(b) Detected objects on warped webcam video

Fig. 4: Detection frames from retrained model

ball color. From this we see that the ball detection generalizes poorly on certain colors, worst of all being green, white and light orange. Nonetheless, given that it has moderate performance on other colors like orange and blue, the model seems to have learned to detect the ball based on more than just the color pink. After inspecting the detection video for the white ball, it appeared that the ball was classified in almost every frame as a foosman. From this we can infer that the model has learnt to associate the white color with a foosman, which is not unsurprising given that one of the foosmen has a white torso, and all the foosmen have white shoes. This suggests the model would benefit from further training on images of balls with various colors, or perhaps training on grayscale images.

Color	P	R	mAP@.5	mAP@.5:.95
Green	0.998	0.5	0.632	0.359
Pink	0.991	1	0.995	0.667
Orange	1	1	0.995	0.845
Light orange	0.666	0.4	0.5	0.306
Blue	0.778	0.7	0.757	0.323
White	0	0	0.0371	0.0217
All	0.692	0.6	0.623	0.394

TABLE 7: Ball detection performance on different colors

5.1.4 Detection speed

When the model is run as a detector on an input video, the typical FPS was 270 on average. The slowest measured performance was 172 FPS. All tests were performed on a PC with a NVIDIA GeForce RTX 3060 GPU. Table 8 shows the inference, and non-maximum suppression times reported by the model. From these results, assuming that an added tracking algorithm is not too computationally expensive, we expect that a tracker based on this yolov7 detection model will massively outperform the 'classical' algorithms listed in Table 1.

Video	Inference (ms)	NMS (ms)	Total FPS
Average	3	0.7	270
Worst-case	3.3	2.5	172

TABLE 8: Typical detection speed

5.2 Angle estimation

Figure 5 below shows the relative position of the bounding box centers relative to the computed center. As can be seen, there appears to be a strong correlation between the center offset and the sign of the angle, however with this computation there is a noticeable negative bias. We can conclude that using the center offset has promise as a means to compute the sign of the angle. For the next part of our analysis, we assume the quadrant has been inferred and look at the average accuracy over a full 360 degree rotation.

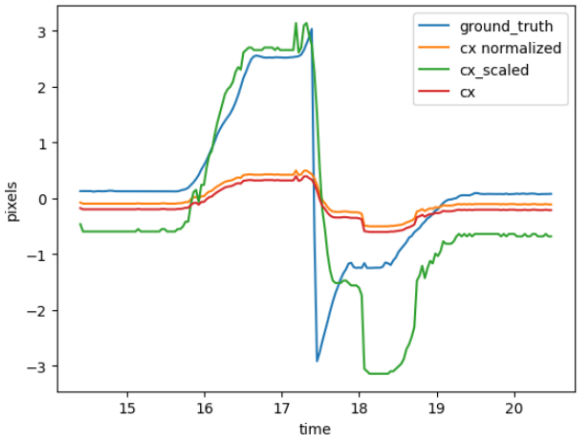


Fig. 5: Bounding box center over 360 degree rotation

For this analysis, we examine the computed angles of a single foosman. In Figure 6 we see the estimated and ground truth angles for a single foosman undergoing a full 360 degree rotation. The first plot shows the ground truth angle in blue. The 'raw' estimated angle is shown in orange, and was simply computed using the equation described in 3.2. The 'full' angle is was computed using the known quadrant information. As can be seen, the estimated angle roughly follows the same trend as the ground truth angle. A simple linear regression was also computed between the estimated and ground truth angles with the following parameters shown in Table 9.

slope (rad)	1.0727
intercept (rad)	0.0661
R-squared	0.975158

TABLE 9: Simple linear regression between ground truth angle and estimated angle using quadrant

Figure 7 shows the angle estimation using the simple linear regression method. In the figure we see that some of the bias

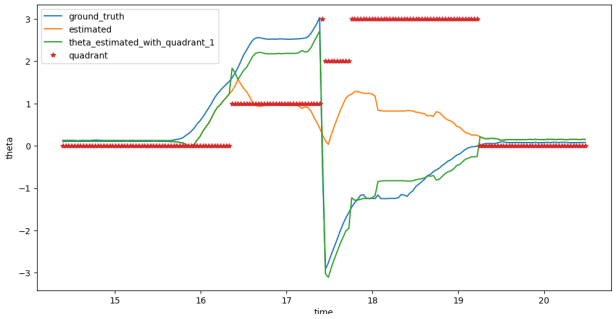


Fig. 6: Estimated and ground truth angle for one 360 degree rotation of a foosman

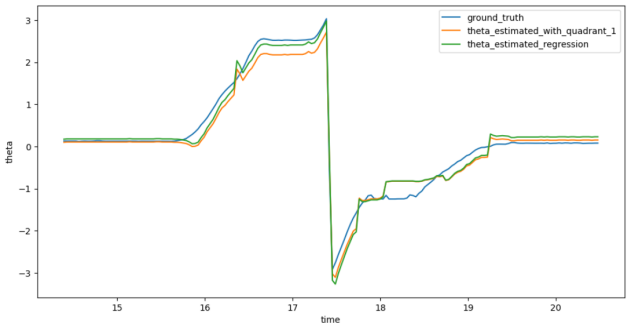


Fig. 7: Angle estimation with simple linear regression

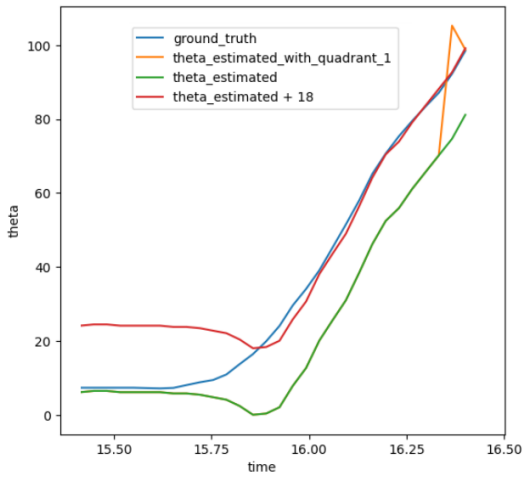
in the positive angels is corrected, however there is a slight worsening of performance in the second half of the graph. The average and maximum error for both methods are shown in Table 10. The simple linear regression results in a slight improvement in average error but a significantly worse maximum error.

	Estimated	Linear regression
Average Error (deg)	10.2	9.2
Maximum Error (deg)	24	29

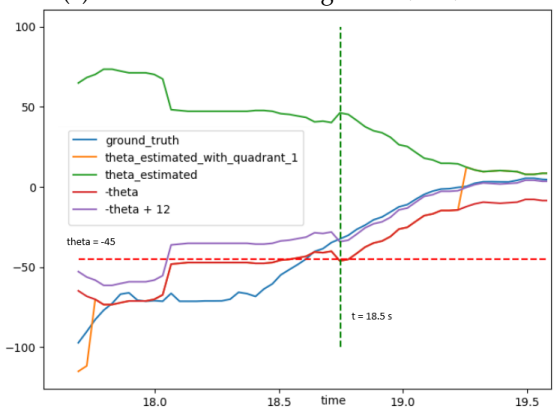
TABLE 10: Angle estimation error

The two graphs in Figure 8 highlight some of the non-linearities in the angle estimation. For the first graph, we see a bias of 18 degrees in for computed angles above 20 degrees, and a non-linearity corresponding to the region of 5 and 20 degrees of the ground truth signal, where the computed angle has an inverse trend. In the second graph, we see a bias of roughly 12 degrees for negative angles above -45, but between -45 and -120 the trend of the estimated angle very weakly correlates to the ground truth angle. From these observations, we can conclude the following:

1) If it is acceptable to only be able to compute angles from -45 to +90, our method, inferring sign from the bounding box center, may be adequate if the accepted error is +/- 18 degrees
2) If a more robust angle estimation is required over the full 360 degrees of rotation, a non-linear regression is likely required.



(a) Non-linearities for region $0 < \theta < 100$



(b) Non-linearities for region $-100 < \theta < 0$

Fig. 8: Angle estimation non-linearities

6 Conclusion

In summary, we have shown that the yoloV7 model is a suitable detection model for this application, and that good detection performance can be obtained from just a single training video. We have also shown that the arcsine heuristic can be used to approximate the true angle of a given foosman. To implement a solution to the tracking problem, additional work needs to be done to implement the SORT algorithm so that the individual foosmen are tracked from frame to frame. In terms of the angle estimation, much more work needs to be done to obtain a robust angle estimation. Firstly, a model should be trained to classify the foosmen based on which rotation quadrant they are in. Secondly, the yoloV7 model should be refined to improve the mean average precision score for an intersection over union threshold of 0.95. This is so that the bounding boxes are as accurate as possible for angle estimation. Third, the pose of the camera should be estimated either by using ArUco markers or by triangulation of the table edges, which can be measured beforehand, so that its perspective can be accounted for. Fourth, we recommend implementing a non-linear regression model to address non-linearities in the angle estimation. Finally, it is likely there will still be noise in the estimated angle, and so our final recommendation is to implement a Kalman filter for the angle estimation.

7 Code, Datasets and Videos

7.1 Code

Project repo:
https://github.com/jclundy/csc2529-project
YoloV7 fork:
https://github.com/jclundy/yolov7-copy
Sam2 fork:
https://github.com/jclundy/sam2-copy

7.2 Datasets

Test data sets can be found here:
https://universe.roboflow.com/foosballproject/foosball-table-qfkml

7.3 Videos

Test videos can be viewed in this youtube playlist:
https://youtube.com/playlist?list=PLge4HMdctiLV0xNT80ETUOM5fFKkDF73V&si=uxMS4V7iaegubKe1

References

[1] J. Lundy, "Design of a robotic foosball goalie system." [Online]. Available: https://joelundy.wordpress.com/wp-content/uploads/2022/01/wkrpt-400-university-of-waterloo.pdf

[2] ——, "Using aruco markers to define region of interest for foosball ball tracking," 2022. [Online]. Available: https://joelundy.wordpress.com/2022/04/21/using-aruco-markers-to-define-region-of-interest-for-foosball-ball-tracking/

[3] K. Du and A. Bobkov, "An overview of object detection and tracking algorithms," *Engineering Proceedings*, vol. 33, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2673-4591/33/1/22

[4] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *CoRR*, vol. abs/1602.00763, 2016. [Online]. Available: http://arxiv.org/abs/1602.00763

[5] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022. [Online]. Available: https://arxiv.org/abs/2207.02696

[6] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314000235

[7] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: https://arxiv.org/abs/2408.00714