Master's Project

Babyfoot

Fine control

Sameer Roger
Supervised by : Christophe Salzmann

July 01, 2022

Contents

1 Abstract

The babyfoot project's first iteration started in Fall 2012, and has evolved constantly since then both on the mechanical and software side. It has gone through a large number of student projects each trying to improve some aspect of the robot, all under the supervision of Christophe Salzmann. This multidisciplinary project regroups mechanical, electronic and programmatic elements to form a very complex but powerful and autonomous structure. The initial goal of the project was, and still is to put in practice the skills learnt at EPFL while showcasing robotics applications in a fun way that can speak to a wide audience. It can be a good entry point into the field for people wanting to discover it. The project is open ended, there are no clear goals to reach, except making it the strongest possible and reproducing human like playing behaviour. The babyfoot always has room for improvements, new strategies and new functions.

Some important accomplishments have already been achieved such as juggling the ball between players of the same rod or capturing the ball in motion. However, the succession of projects which have been more or less successful have left some minor bugs and small tweaks made by different students to correct undesired behaviors. This has led the babyfoot to be less and less performant and advancements made such as juggling became hardly reproducible and very unpredictable. In order to counteract this accumulation of problems, a lot of aspects needed to be reviewed and possibly corrected. This work has started in fall 2021 and is continuing partly in this 2022 spring semester. The version of the babyfoot at the beginning of this project was therefore less performant than previous versions and has less functionalities. One of the goal of this project is therefore to restore the babyfoot at least to where it was before the reset in a cleaner and more robust way.

The initial objective of the babyfoot, was to mimic as best as it could human behavior and make it unbeatable to a human. In the weakened state that the babyfoot is in at the beginning of the project and after a lot of corrections and calibrations, the babyfoot is already almost unbeatable, even though the strategy is extremely basic. As is, the babyfoot can defend extremely well as it can easily place one of its player in front of the ball, making it very hard for the human opponent to score. However, attacking wise a lot of improvements can be made, the machine only shoots straight in front with no perception of the human players or the direction of the opponent's goal.

The babyfoot does play correctly but its play style doesn't resemble in any way the one of a real player, of course this is normal given that it analyzes 500 frames per second which is much faster than what a human can do, giving it an edge especially in defense. The reaction time can't be matched. However, for the rest no humans would play in this way. Most people will try to some extent to aim for the goal and their movement is not exclusively translational or angular but a combination of both. Humans are also able to forecast future actions and play accordingly, such as when dealing with rebounds on the side of the board or stopping an action if the environment changes and the action will for sure fail, for example when striking the ball in front of an opponent player.

All of these are possible paths of improvements for the babyfoot.

1.1 Objectives

This master's project has multiple objectives. The first one follows the issues encountered in 2021, which have led to a backtrack of some functions. The goal is to recreate a cleaner version of the main VI which would be more reliable and less dependant on tweaks and fixes added to make it work as well as creating a more solid foundation giving desired and reproducible behaviour. Before the start of the project a few issues were already solved but a lot of elements remained to be verified, validated and corrected if necessary and when possible. These points include, the ball position measured by camera, the homing sequence of the translational motors, angle measurement by the encoders and that the rods are correctly fixated at the correct initial angles mechanically. Until the remaining issues are corrected it is not possible to improve the control part of the babyfoot.

The second main objective of the project is too have an in depth analysis of the shooting abilities of the machine modelling the different shooting methods and finding the best method to use given a target to hit and the position of the ball. Three methods will be tested : shooting strictly in a rotational motion, shooting strictly in a translational motion and finally a combination of both a rotational and translational motion. A lot of testing is necessary in order to create the models and assess them.

Finally, the last objective is to put in practice the new shooting models in a dynamic setup in order to be able to shoot accurately in any situation while possibly avoiding the opponent's players. Some of the possible leads would be to incorporate the capture of the ball as well as juggling the ball between players of a rod, to manage to both catch the ball when the ball is coming at too high speed and also move the ball around to find an angle that will allow to shoot to the goal without hitting one of opponent's players.

2 Introduction to Babyfoot structure and functioning principle

In order to understand the behaviour and functioning of the babyfoot a quick introduction to its mechanical and control structure is given.

2.1 Mechanical structure

First of all, the dimensions of the board and the positions of the blue rods automatically controlled are given in Fig.1. All positions are given from the reference in the center of the board, with the x axis being in direction of the white's goal and the y axis in the direction of the blue side.

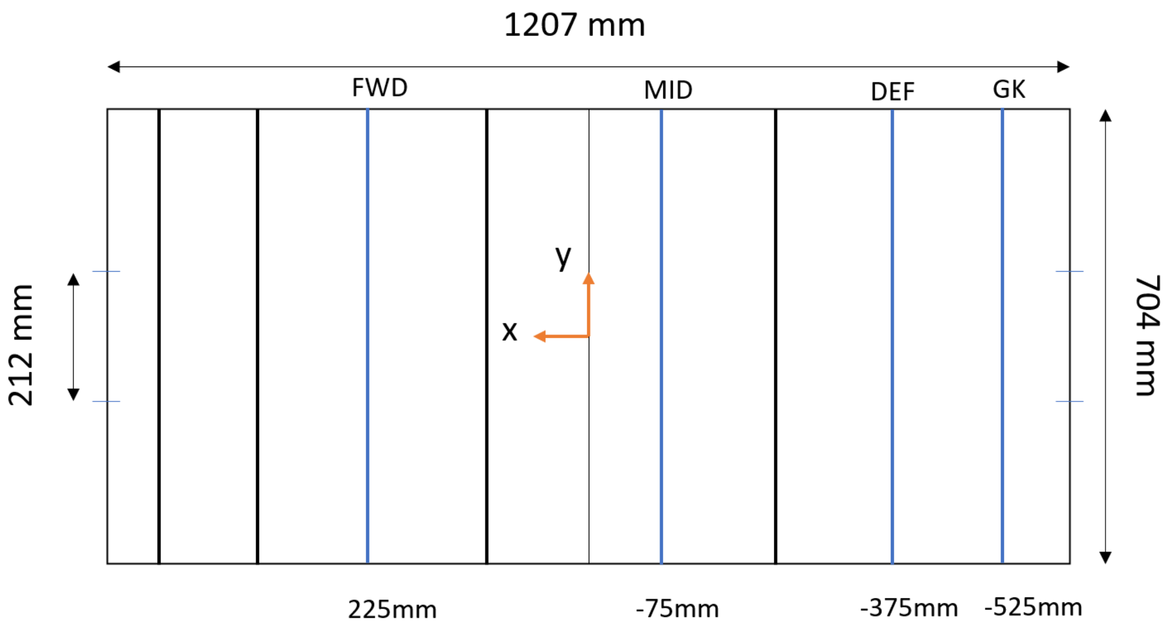


Figure 1: Board dimensions

The mechanical structure is quite straightforward, just like a normal babyfoot, there are 8 rods, 4 on each side which are composed of :

- Goalkeeper rod with 1 player
- Defenders rod with 2 players
- Midfielders rod with 5 players
- Forwards rod with 3 players

Concerning the robot's side of it, the four rods are completely independent from each other. Each rod can be controlled both in translation and in rotation. The two motions are commanded by separate motors. The range of displacement is different for all rods considering they have different number of players and therefore, some rods have to move greater distances from one side to the other.

Different types of sensors are used to control the machine. A camera situated below the bottom board made of transparent Plexiglas is used to track the ball. The angles of the controlled rods are measured with angular encoders mounted on the rods. The lateral displacements are measured directly from the motor's sensors. Finally, the human controlled rods's lateral position and angles are given by a set of two lasers for each rods.

For more information on the mechanical and control characteristics of the babyfoot, a lot of resources can be found on the babyfoot webpage of the automatic control laboratory.

2.2 Control architecture

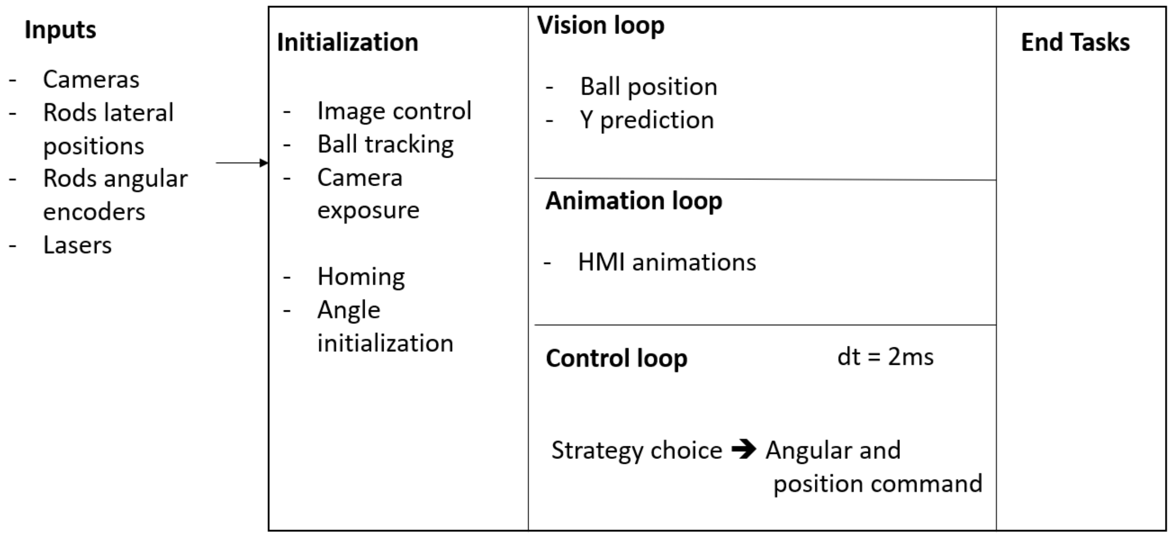


Figure 2: Control Structure

Concerning the control architecture of the babyfoot, it is composed of three main parts as shown on Fig. 2 and done on LabView 2019.

With the inputs coming from the different sensors detailed previously for each rod as well as the camera and lasers the initialisation part can begin. It is essential for the Babyfoot to function correctly. It is needed to calibrate the initial positions through a homing maneuver from the Coppley controllers, to find the bounds and initializing the position of the rods in the middle of the board. For the initialization of the rod's angles, the z index is used. It triggers once every revolution of the encoder and aligns the rod at the given 0 position when the z signal is emitted. The ball tracking is also initialized, the algorithms need an initial picture of the ball that it will then track. It is important for this initial image to be clear and bright enough to be recognized but not too bright or else it could be confused with other lights in the room. The image must also be centered exactly on the center of the ball so that the function that need very precise position of the ball can work properly.

The two other parts of the control architecture are running in parallel, they are the vision and the control loops.

The vision loop, outputs the ball position every millisecond and gives a prediction of the position of the ball.

The control loop, which runs with a 2ms period is the part responsible for the strategy of the babyfoot and is responsible for sending orders to the rods. The strategy can be chosen from the front panel.

Finally when the program is stopped by the user, it is necessary that all the tasks are terminated so that the motors completely stop and remain in this state.

3 Verification and validation

At the start of the project a number of issues remained to be discovered and fixed in multiple parts of the babyfoot, especially in the vision (ball detection), homing maneuvers and angle measurements.

3.1 Vision Issues

It was observed that the ball measurements could be very off on some parts of the board, especially on the side of the blue goalkeeper. The origin of the issue was not clear as it was not a constant offset on all positions but a variant one which were intensified on one half of the board. The issue could come either from the camera itself which was not well calibrated or on the software side.

The software side has been checked out. The dimensions of the babyfoot were taken again and both vision correction algorithms accounting for light refraction passing through the Plexiglas and the angle of the ball from the camera, were also verified. Small corrections were applied to these algorithms such as removing constants that were previously added but were responsible of small errors in the y coordinate.

The problem turned out to be from the camera which was not well calibrated. Even though it had been calibrated a few months back and left untouched since then. The babyfoot can sometimes have violent and abrupt movements from the rods that could cause unexpected changes in the calibration of the camera. Therefore, in order to insure proper ball detection, the cameras should be recalibrated often, at least once every 3 months. The camera was recalibrated with the 'VIBabyfoot2021-Calibration-Chs', by realigning the defined calibrations points to the image shown from the camera.

Other issues coming from the vision aspect is the actual initialization of the ball. The two aspect of the initialization which are the manual selection of the ball and the choice of exposure are both very important to have an accurate measurement of the ball position.

For the selection of the ball it is very important that the selection is square is properly centered on the middle of the ball and that the whole is ball is in the square. It is important to keep in mind that the position where the ball will be initialized also plays a role, it is an area where the detection accuracy will be increased. Moreover, it is best to avoid putting the ball where white lines are on the board as it could decrease the detection.

For the selection of the exposure, there is no exact rule as what is the correct number as it is very dependant on the brightness of the room. The darker the room is the better the ball will be detected. A luminous room also usually leads to a difference in brightness of different parts of the board due to shades. As a rule of thumb, the exposure selected during initialization of the ball should never be less than 700 and over 2000.

Even when taking all these steps to ensure having a good initialization, small errors of a few mm are very frequent especially on the sides of the board near the goalkeepers where the ball is very hard to detect. These errors can have important impacts in the accuracy of shots especially in when the ball is in motion.

3.1.1 Ball used

Another point tested regarding the vision is the choice of the ball. The ball that is used is a plastic ball with a very bright orange color to improve ball detection by the camera. However, it is not a ball commonly used for foosball.

To improve the feeling of the human player and make it closer to a normal foosball table, it would be better to use a ball made of cork. It is heavier, rougher and has a more tern color which might be harder to detect as it denotes less from the rest of the points the camera sees. The cork ball due to its surface roughness and elasticity would also allow to capture it easier and improve the control over it.

Testing the cork ball gave very satisfying results the camera could detect as well as the previous ball. No particular issues were detected. However, if the room is to luminous it could cause detection issues as it would be harder to differentiate the ball from the environment, this is already something that can happen with the bright orange ball. One solution would be to simply to paint the ball in brighter color.

3.2 Homing procedure

The homing procedure is causing some issues in the position measurement of the rods. It is entirely controlled by the Coppley controller and with the CME2 software.

Homing is the behavior that enables a robot to return to its home position. In our case the translational home position for the babyfoot is set when all rods are in the middle of the board in the y direction. The homing method used in our case is the positive hardstop method which is one of the methods integrated in the Coppley Xenus controller.

For the Xenus controller the home position is set from the point of contact with the babyfoots's left border (on the human side). This hard stop is reached when the amplifier outputs the current limit continuously for the amount of time specified in the delay time (100 ms).

With this method the rods will be correctly positioned in the middle of the table if no obstacles are blocking the movement. However, the homing procedure will be perturbed if there is an obstacle blocking the way towards the spring on the human side.

This possible obstacle issue causes calibration errors which can be dangerous for the babyfoot as the servo doesn't know where the borders really are and therefore the motor can continue to give orders to move even when the rod has already reached the border.

Moreover, even though at the end of the homing all rods are in fact in the middle of the board in the y direction, the distance it can move on either side is not exactly the same. Since the controllers are not aware of that, they falsely believe that the rod can move the same amount in both direction. However, moving it all the way to -10V which is the negative limit will push the rods too much pressing hard against the border. On the contrary, at 10V which is the other bound, the rods will not go all the way to the border. These differences are more or less marked for the different rods, the distance from each side is different for each rod.

Rod	Dist to left border	Dist to right border
Goalkeeper	91	103
Defenders	175	185
Middfielders	60	73
Forwards	91	91

Table 1: Distance to borders for each rod

This is not a critical issue as the babyfoot still functions with it but it could damage the structure of the babyfoot. The ball can always be reached because of the slopes added near the borders.

A few solutions were explored to solve this issue without success. Since the whole process is done with an innate program of Coppley control it is hard to modify the behaviour. Different methods of homing were tested as well as adding an offset after homing directly on the CME2 software. However, the issue remains. Tries to displace the rods after homing to put them in a position where the distance to the border is equal on both side were also made, however it wasn't possible as the limits of -10V and 10V can't be changed and it is not possible to reinitialize the zero volt position after homing.

It would be necessary in the future to look further into this issue and the CME2 software to understand the source of this error and correct it. The issue is not critical but it is hard to know exactly how much it impacts the translation orders given to the controller.

3.3 Rod angle issues

Some important issues were detected with the angles of the rods, a deviation from the command sent were observed which increased with time as well as translation motion. The more the rod was used the higher the deviation grew. This would render the robot unable to play after a short period of time (1min). This was especially accentuated when moved in translation, even though there should be no relation between the two motors. Because of these points, rods with less players (Goalkeeper and defenders) were especially impacted as they were travelling more distances than other rods. Midfielders angle deviations were much slighter.

This issue was thought to be coming from electromagnetic interferences between the power cords and the angular encoders cables. With the angular encoders not returning the correct rod angles, the rods increasingly deviated from the angle commands sent to the motors.

The first try to fix this issue was to ensure that when the angular encoders return a measure of 0 degrees the rods are perfectly straight. We can either choose the 0 to be set when the player has its head up, or when it is down. Changing from one disposition to the other is done mechanically by unscrewing the fixation between the rod and the motor and putting at the desired angle manually.

With the players facing up, it enabled the activation of the z index, which can be set to reset whenever the player passes through 0 degree angle which can happen very frequently during normal play. Whenever a shot was taken, the angle measure would in theory reset to 0 whenever the feet passed through the 0 angle. Thus, resetting the z index often, should prevent the rods's angle measurement to deviate from the real angles. However, once the angle has deviated by a large angle, it will only reset to 0 when the rod has done a complete 360° rotation. Making the rod completely useless for an important period of time.

However, electromagnetic interferences between the power cord and the angular encoder discussed before also impacted the z channel, falsely giving orders to reset the angle to 0 even though it hasn't passed by the origin. This made the rods rotate by 90 or 180 degrees randomly, making the issue even worst. On Fig. 3, a spike to 0° can be observed on the defender's rod angle measurements by the encoder. When the spiked happened, the encoder was setting the angle of the rod when the spiked happened to be the 0 value, therefore any command after that will be calculated from this initial angle. This caused big angle changes and then the angle was completely off from then.

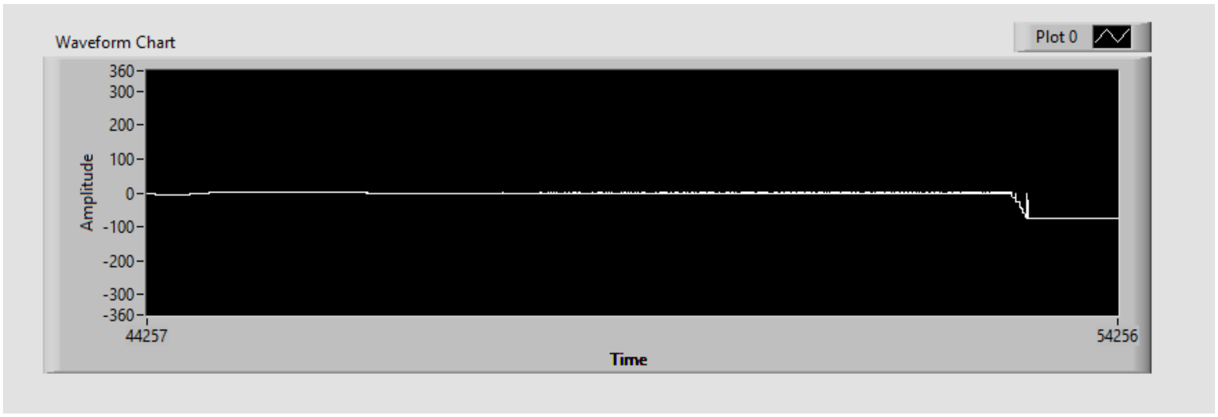


Figure 3: Angular interference of the z index

One possible solution that was tested was filtering out the interferences with a high pass filter by connecting a capacitor of 0.03 pF to the circuit. This has not be successful. New encoders with differentiated entries could also have been a potential fix. However, in the end the solution used has been to replace the cables connected to the encoders by shielded cables blocking the electromagnetic interference. The results are extremely satisfying, the issue is completely fixed, the encoders are now able to measure the rod angles without any problems.

In order to still be able to play and start developing new functions for the babyfoot while the issue was not yet resolved, a function was added to manually reset the z index by pressing a button on the front panel for each rod, resetting the rod angles to the correct one. It was also possible to make the rods reset every few seconds. However, when resetting the z index in these ways, the rods need to do a full rotation which is far from ideal when playing.

4 Shooting Strategies

One of the goals, strategy wise for the babyfoot was to make the offensive actions more constructed and efficient. In the initial state of the babyfoot, the players only shot straight without any consideration of the human players or where the position of the opponent's goal is. Scoring goals therefore has a very random component to it.

These two pieces of information are crucial to score goals, if the babyfoot is able to take them into account and play accordingly, it should be much easier to score goals.

In order to to do so, it is first necessary to be able to shoot to a desired target enabling us to shoot at any desired angle to avoid opponents and shoot at the goal. To do so, three different shooting methods are experimented with: strictly in rotation, strictly in translation and a combination of both.

The goal is to assess the accuracy of these different methods to determine which is the best to use given a specific target angle and ball position. Ideally, every target angle from 0 to 180 degrees could be hit accurately.

Throughout all calculations, some simplifications had to be made concerning the shape of the player feet. They were considered to be perfectly rectangular even tough, it is not the case. The shape is rounded on the bottom part with a slight slope from the top to the bottom in the z direction. These simplifications were made because the exact model of the feet is not known and the simplification is not too far from reality.

However, this could cause quite a bit of uncertainty in the angle that we can achieve after collision. Is is hard to estimate the impact as there are lot of possible sources of uncertainty that also impact the accuracy of the shots. This issue also makes it harder to have an exact 3D model of the collision.

4.1 Rotation

The first method tested is the the shot strictly in rotation. It is the simplest method on paper and the most used by humans. It is efficient for angles between 60° to 120°, which make up for most of the shots that are made in a match.

The basis of this method consists of positioning the player in front of the ball with a calculated offset from the ball's center, in order to hit a desired target point on the field. Even though this method seems pretty straightforward and the calculations could look like a simple trigonometric problem. In reality, hitting the desired target is much more complex and multiple questions need to be answered. First, what is the influence of the ball's position in the vertical direction on the precision of the shoot. secondly, with what part of the player's feet should the shoot be done with (the middle, the border, or a point in between ?). And finally, does the precision decrease with an angle increase ? Is there a threshold where another shooting method should be used instead ?

To answer these questions a lot of testing is necessary. Starting with a very simple model of the shooting and then improving it gradually to increase shooting precision.

4.1.1 Initial calculations

To begin with, the simplest model possible was created. The problem was simplified to a maximum, by making it a 2D problem. The ball was reduced to a disc and the player's feet to a single point. Making the collision calculations very straightforward as it can all be simplified to simple trigonometric equations. The shots tested were done for three different points of the feet : the middle, the edges and a point in between.

The diagram of the simplified 2D problem is given in Fig. 4

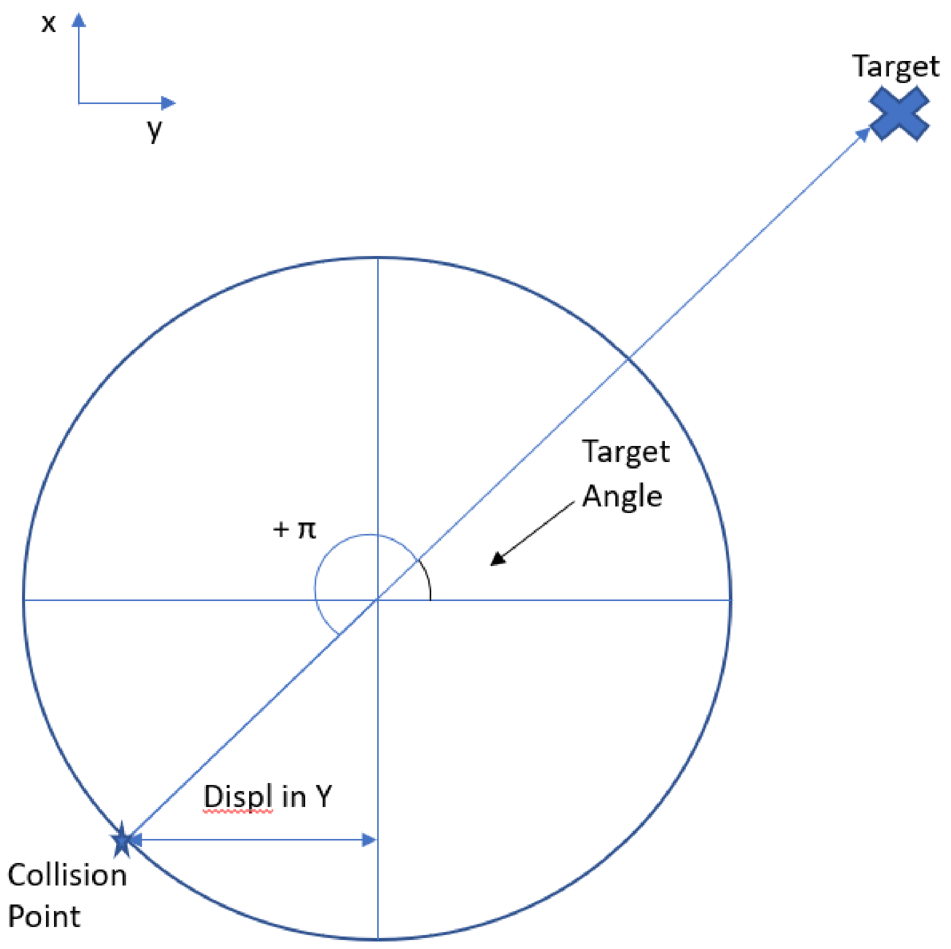


Figure 4: Simplified 2D collision diagram

The target angle represents the angle from the center of the ball to a point fixed by the user, or the middle of the opponents goal when implemented in the final dynamic function. It can be calculated as follows :

$$TargetAngle = atan2(\frac{X_{target} - X_{ball}}{Y_{target} - Y_{ball}}) \tag{1}$$

The function atan2 works similarly to the classic atan except that the signs of the arguments are used to determine the sign of the result. The function always return a result between $-\pi$ and π.

The idea is that the player will hit the ball at a certain point on the disk creating a force in the direction going from the point to the center of the ball. In this 2D model, the position in x and z of the ball are considered to have a negligible impact on the direction of the vector force. Therefore, the only input that is needed is needed to achieve the angle desired is the displacement in the y direction of the player compared to the center of the ball.

This displacement in the y direction from the ball center can then be calculated as follows :

$$InitDisp = cos(TargetAngle + \pi) * r \tag{2}$$

r being the ball radius. π is added to the Target angle as the collision will take place on the other side of the ball.

To this *InitDisp* the distance between the closest player on the rod to the ball needs to be added to insure the player is in front of the ball, before adding the displacement for the angle desired.

The Labview implementation is given in Fig.5. The thresholds are the maximum displacement the defender's rod can move in both directions.

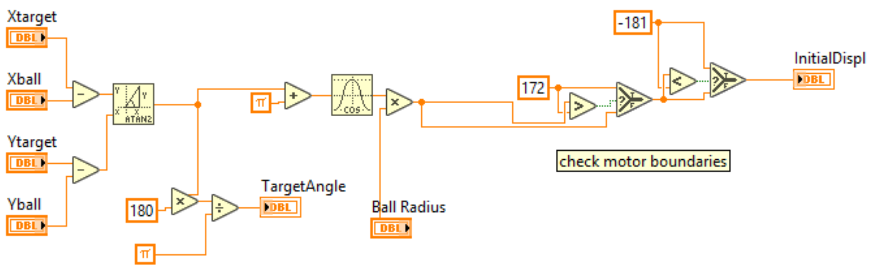


Figure 5: Part of Shoot.VI representing the initial displacement for rotational shooting

With this first hypothesis, the shots where then tried with either the center of the feet (no added displacement), the edge of the feet (added displacement of 10 mm) and a point in between (added displacement of 5.5mm). All the tests were performed on the defender's rod. The displacements added, are always positive for negative target angles, and negative for positive angles.

The results for this first model are as follows. The angles are given from 90°, meaning that a 0° degree angle in the plots is a straight shot of 90°.

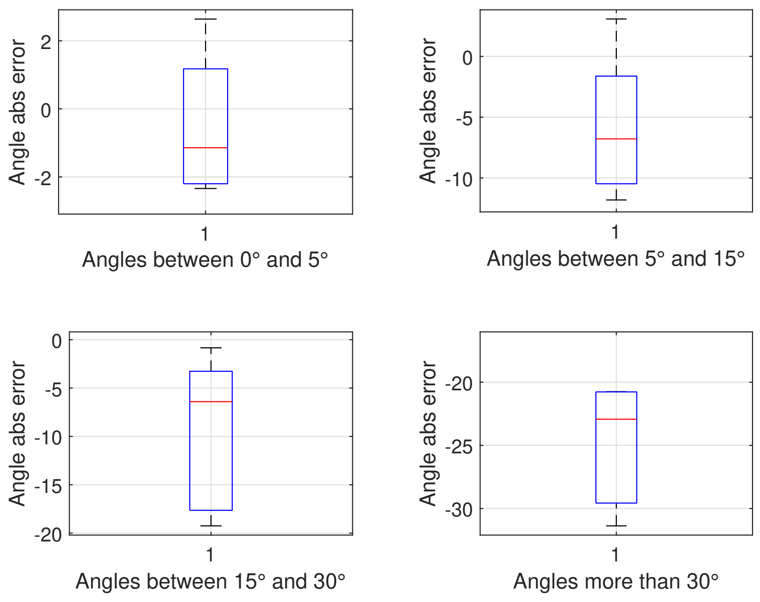


Figure 6: Boxplots representing angle errors for 0 mm added y displacement

The graphs [6] shows that hitting the ball with the middle of the feet gives relatively good results for very small angles (lower than 5°). For angles between 5° and 15° and 15° and 30° the results give an average angle deviation of around 5° which can be considered to be too high as it can lead to an offset of 10 cm in the y direction from the desired target, if the shot is taken from one end of the board to the other. For angles larger than 30°, the average error is 26°, shots are completely inaccurate.

Moreover, the angle error is heavily skewed towards negative values meaning the angles achieved is always smaller than the one desired. It can be concluded that hitting the ball with the middle of the feet is only accurate for very small angles.

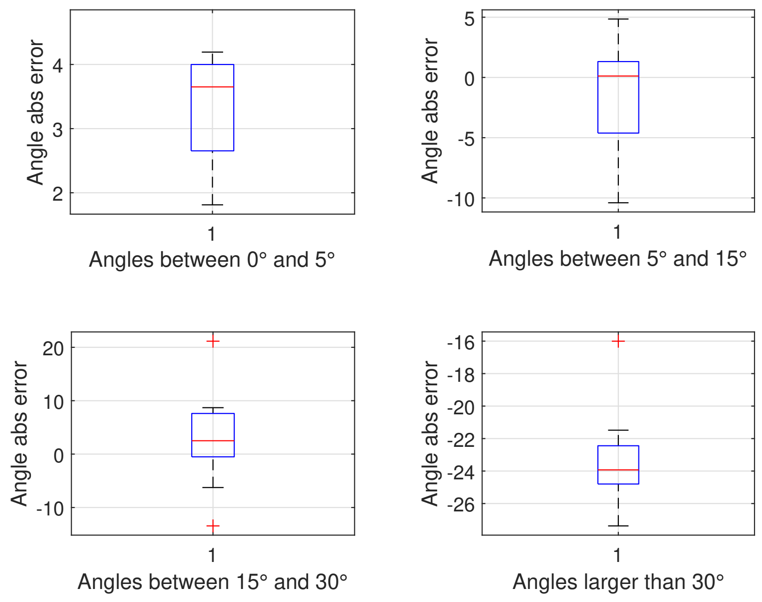


Figure 7: Boxplots representing angle errors for 5.5 mm added y displacement

The graphs [Fig. 7] shows that with an added displacement of 5.5 mm corresponding to a point between the middle and the edge of the feet the accuracy is very high for angles between 5° and 15°. For the 0 to 5 and 15 to 30 degrees range, the error is around 3° which could be fine for shot at small distances but not for larger distances. Moreover, the presence of very big outliers is also an issue. Finally, similarly to the 0mm displacement shots, the accuracy is very poor for angles larger than 30°.

Overall, hitting the ball with a y displacement of 5.5 mm give better result than with 0 mm but the result are still not satisfactory especially for large angles.

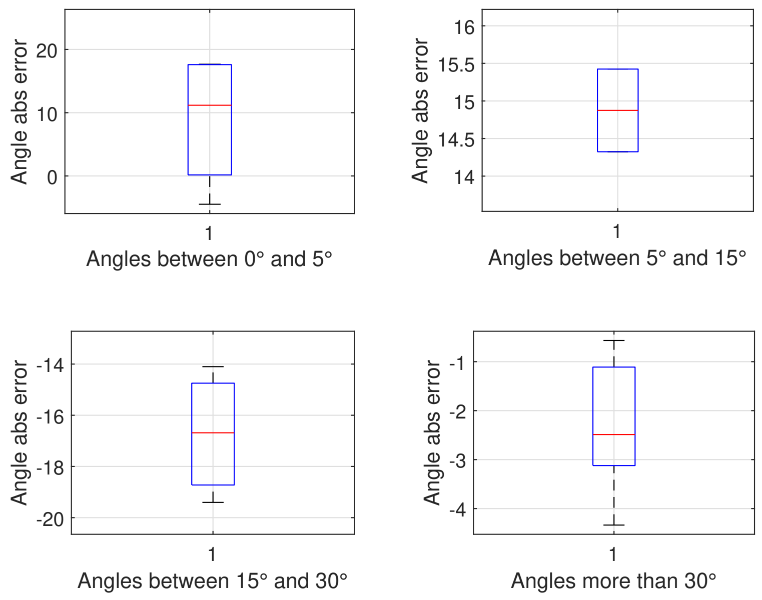


Figure 8: Boxplots representing angle errors for 10mm added y displacement

Finally, for an added displacement of 10 mm that should correspond to the edges of the feet, the graphs [Fig. 8] shows that results are not good at all. Results are only good for angles over 30° and very inaccurate for smaller ones.

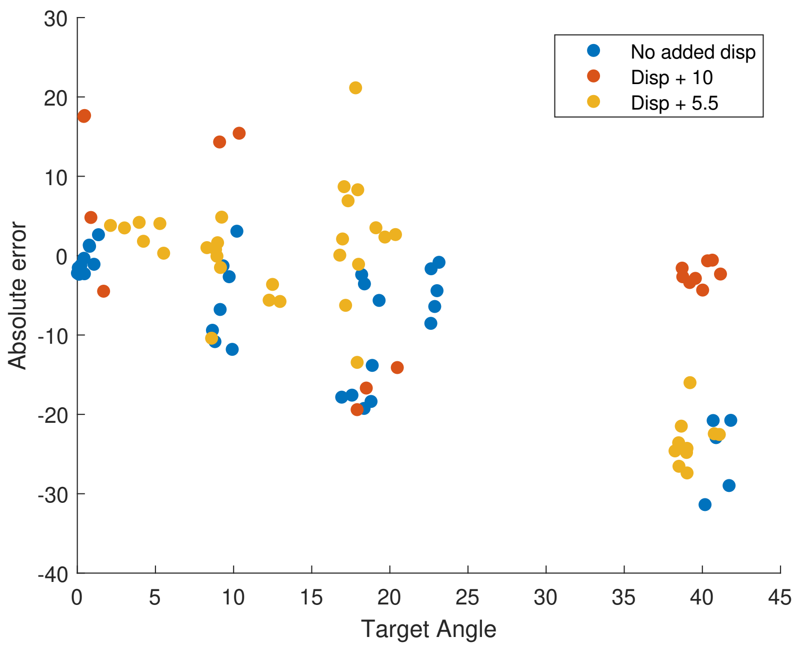


Figure 9: Angle error compared to added displacement

As it can be observed, none of these results are satisfactory for all angles. Each displacement seems to have a range of target angles for which the results are good but for the rest, the accuracy is not great. Fig. 9 highlights this issue. It can be seen that around 0° the best is to shoot with the middle of the feet, whereas between 10° and 20° a displacement of 5.5mm is more accurate, and finally for large angles around 40° the edge of the feet is the best option. It is therefore very clear that the displacement added to the initial calculations and therefore the point of the feet that hits the ball plays a very important role in the angle of the ball post collision. However, Fig.9 shows that it is not the only source of error as the standard deviation of the error for a given target angle and displacement can be very high. Some large outliers are observed

4.1.2 3D collision model

One aspect that was suspected to have an important role in the uncertainty of the angle achieved in the previous model is the position of the ball in the x direction with regards to the position of the rod shooting. For instance, for a defined target angle the previous model would give the same displacement from the center of the ball whether the ball is in front of the rod or behind. Of course in the two cases the point of contact between the ball and the player's feet will be different and can therefore influence the trajectory after collision. The magnitude of how it impacts the angle post collision is however hard to estimate without modelling it.

To check for this potential issue a Matlab model was developed simulating the trajectory of the rotating player to estimate exactly the point of impact and therefore the angle of the ball after collision. In this model the player is represented as a line in the z direction for simplicity. The Matlab function can be found in Appendix A.

The algorithm has as inputs the target angle and the initial displacement calculated in the simplified 2D model and will try different y displacements around the initial one calculated in [2] to find the one that will achieve the target angle.

The goal is to iterate for a range of displacement in y, and then iterate again for a range of rod angles. The algorithm stops when the target angle is attained. It then outputs the displacement in y.

Since the period of a loop for the main VI on LabVIEW is only 2ms, it is important that this displacement searching algorithm is as fast as possible while remaining precise and finding a solution for every inputs. To lower to a maximum the time needed to run the code the range of y displacement and angles tried are kept to a minimum.

The displacement in y is ranged between $InitDisp - 2$ (2) and $max(0, InitDisp + 2)$ for positive target angles and $min(0, InitDisp - 2)$ and $InitDisp + 2$. With an exception for small angles (between -10 and 10 degrees) where the range is between $InitDisp - 2$ and $InitDisp + 2$.

For the angle range, we have to find the highest angle needed to hit the ball which depends on the position of the ball in x relative to the position of the bar. Higher angles are not necessary as it can't result in an impact with the ball.

This maximum is angle is given by :

$$maxangle = asind(\frac{-r - (Xball - PosRodX)}{L}) \tag{3}$$

with r being the ball radius, L the length of a player from the rod and $PosRodX$ the position of the rod in the x direction. $asind$, refers to $asin$ converted from radians to degrees.

Finally all calculations are made for the lowest point of the player's feet up to 10 mm above this point.

For all these loops the position set of equations for the player's feet are given by :

$$x = sind(\alpha(i)) * (L - k) + Xball - PosRodX \tag{4}$$
$$y = j \tag{5}$$
$$z = L - cosd(\alpha(i)) * (L - k) + 6 \tag{6}$$

with i indexing *alpha* angles to test corresponding to the angle of the rod. k iterates to test different points of the feet in the z direction. The addition of 6 in the z corresponds to the distance from the end of the feet to the table board.

To check for the points of collision between the player's feet and the ball, we check the following equation considering the center ball to be in the center of the plan in the x and y direction and at r in the z direction.

$$x^2 + y^2 + (z - r)^2 = r^2 \pm 2 \tag{7}$$

The ± 2 ensures to find a solution while not having to refine the iterations too much.

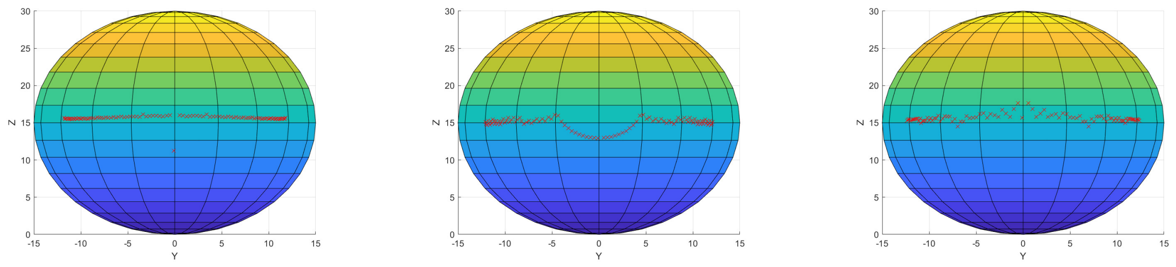
For all points that validates this condition, the angle the ball will take after collision is computed. The algorithm stops if the angle corresponds to the $TargetAngle$ or if all displacements, angles were tested.

The post collision angle is as follows :

$$PCangle = atan2(0 - x, 0 - y) \tag{8}$$

x and y are the x and y coordinates of the collision.

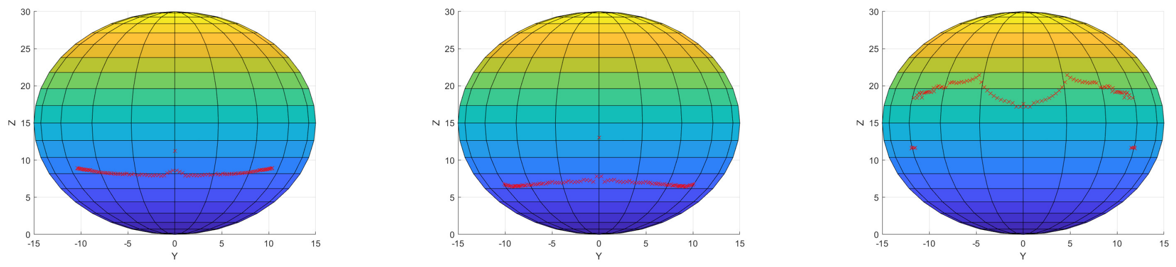
Fig. 10, shows the collision points on the ball respectively when the ball is 30mm behind the rod, when it is right under it and when it is 20 mm in front. The target angles range from 40° to 130°.



(a) Ball 30 mm behind rod (b) Ball under rod (c) Ball 20 mm in front of rod

Figure 10: Collision points between ball and feet for 3 different x positions (1st solution)

It is interesting to note that there are multiple solutions to the equation [7] which give the desired target angle. Another set of solutions corresponding to the same inputs as Fig. 10 are given in Fig. 11. A large difference can be observed between the 2 solutions. In theory, both solutions should lead to the same angle after collision, however in practice it is not an evidence as the displacement in y can be quite different. This could cause some uncertainties and make the model less reliable and results not reproducible all the time.

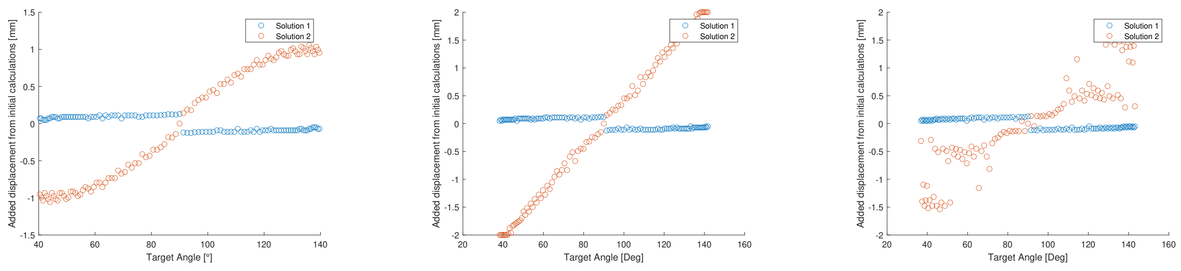


(a) Ball 30 mm behind rod (b) Ball under rod (c) Ball 20 mm in front of rod

Figure 11: Collision points between ball and feet for 3 different x positions (2nd solution)

Fig.12, shows the difference in added y displacement to the initial calculations [2] between the two solutions for the three different cases. For the first solution, the added displacement is fairly negligible if we consider that the position of the ball in x with respect to the rod only modifies the point of collision in the y direction. However, it doesn't take into account that it could also influence the rotation of the ball on its own axis which can affect the trajectory of the ball.

The second solution however, shows the added displacement in y from the initial displacement increases with the target angle, up to around 2 mm for a 130° target angle. This difference is very important, a 2mm variation can in theory change the angle post collision by 7 degrees. In practice, the error could be even greater.



(a) Ball 30 mm behind rod (b) Ball under rod (c) Ball 20 mm in front of rod

Figure 12: Difference in added y displacement compared to initial calculations for the 2 solutions

After testing this new algorithm with an added displacement of 10mm [Fig.13], an important improvement compared to Fig.8 can be observed for angles smaller than 30 degrees but the shots are still highly inaccurate and imprecise. For the same target angle, the angles post collision could have a difference of 10 degrees, which is a big issue as it would make future functions such as avoiding opponents impossible to work. For large angles, the accuracy has also decreased.

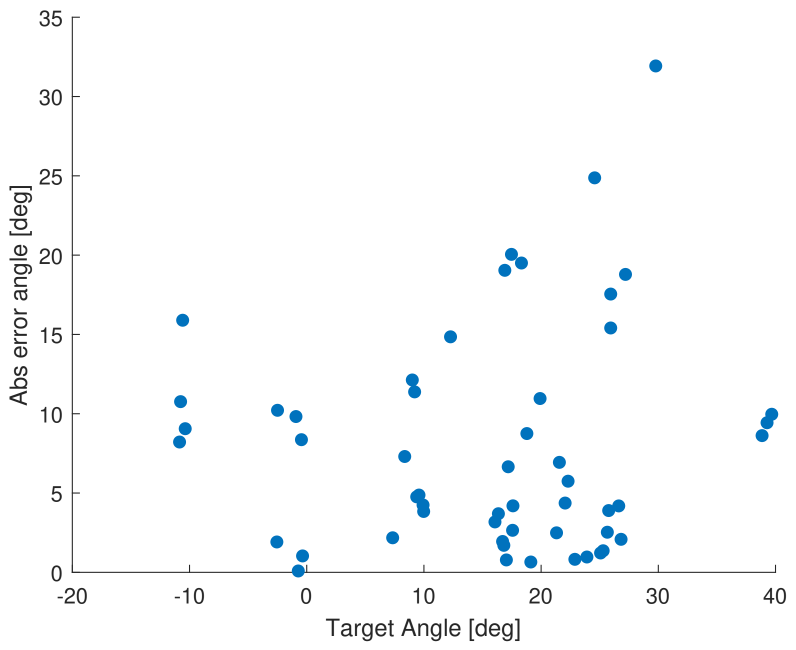


Figure 13: Angle error compared to added displacement

The algorithm seemed to have improved results for a 10mm displacement but the results are still not satisfactory, the shots are not precise enough to use them in any functions.

In theory the addition of this algorithm should improve the accuracy, however it is hard to know exactly if in practice the collision point computed is actually the point hit or not. Moreover, in order to make it really function the model precision needs to be improved, meaning more iterations for the angles, y displacement and Z position of the feet and a smaller acceptation range for eq. 7. However it is not possible to increase the precision without impacting the computation time too much. As the whole LabView SubVI should be run in 2ms the computation of this part shouldn't go above 1ms to stay safe. It is however very difficult to keep this limit. In order not to accumulate uncertainties, to keep computation time low and considering that the accuracy improvement is not sufficient, it was decided not to use the algorithm in the shooting model. However, it is an interesting basis for possible future works on the matter. Maybe the solution would be to run a lot of simulations on the model for a large number of ball positions and target angles and save the results so that a LabView function would only need to fetch the result from these already computed solution given the ball position and target angle.

4.1.3 Final correction

With the accuracy still not being satisfactory, a more data centered approach to the problem was taken.

Up to this point the player was always considered as either a point or a line in the z direction. And a constant displacement was added to see if the ball should be hit with the middle of the foot, the border or a point in the middle.

However, after a lot of testing, it was clear that depending on the target angle, the point with which the player hits the ball should be different. It should hit it with the middle for very small angles (less than 5 degrees) and up to 10 for angles larger than 20 degrees.

A lot of testing was needed in order to find how much added y displacement was needed to hit a particular target angle. For each target angle, the y displacement was added manually to the initial calculations [2] until the correct one was found. This method is very time consuming as for each angle a lot of displacements must be tested to ensure that the right one is found. With all this data, a function can be fitted which is able to output the necessary displacement given a target angle.

The function constructed from this dataset is given in Fig.14. The model is a polynomial function of degree 7 given with the following coefficients in ascending order:

0.541	3.013	-0.52	0.051	-0.0028	9.20e-5	-1.56e-6	1.07e-8

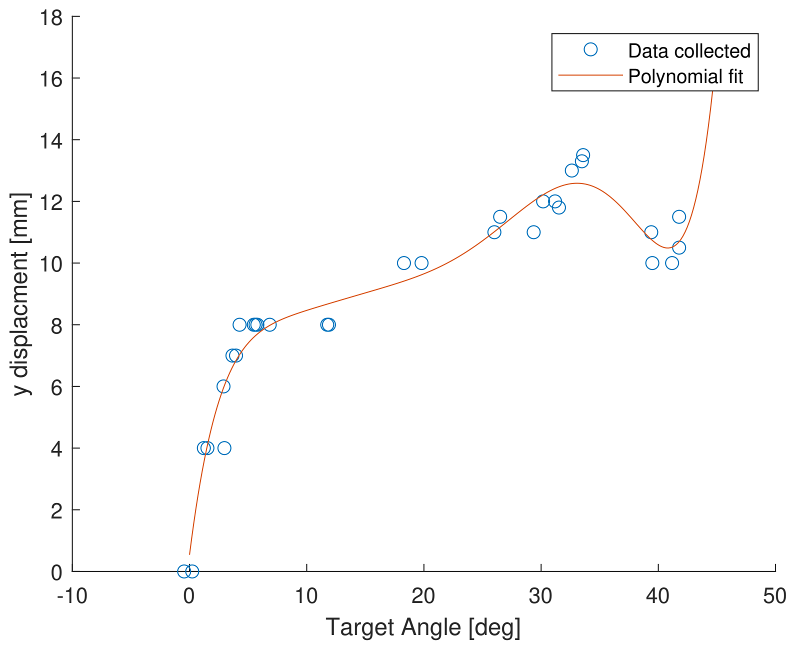


Figure 14: Function fit for defenders rod

The model was made on top of the initial displacement calculated in [2]. Using the polyfit function in Matlab.

The LabView code of the integration of this function in the subVI 'Shoot.VI' is given in Fig.15, a threshold was added to the function as if the added displacement goes higher than 13.5 mm the rod will not be able to hit the ball correctly.

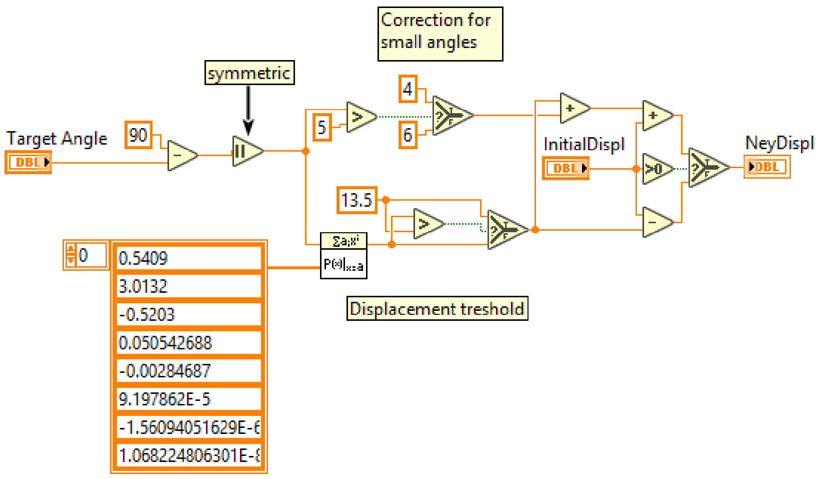


Figure 15: Part of 'Shoot.VI'

It is important to note that the model works for positive angles only, for negative angles a constant of 4mm needs to be added in order to obtain the correct target angle for angles larger than 5° and 6mm for angles smaller than 5°. The cause of this asymmetry is not clear, it could be due to the homing issues or it could be mechanical. The model is also different for each rod. Only the defenders and forwards were modelled for now as there are the two rods who needs to shoot the most accurately as the midfielders are normally not able to shoot following the true rules of foosball.

Moreover, the added displacement isn't capped at 10mm corresponding to the edge of the feet, it can go up to 13.5mm. This shows that this new model takes into account the point of the feet that hits the ball but also corrects the initial calculations which do not work in practice as seen in the first subsection. It also corrects multiple factors impacting the trajectory of the ball which have not yet been identified.

With this model the results obtained shown in Fig.16 are quite satisfying for all angle ranges tested. The average error doesn't go above 2 degrees, which stays very precise. A part of this error could also be due to measurement error as the camera is not always very accurate to give the exact position of the center of the ball, especially on the edge of the board and when the speed of the ball is high. There is still some uncertainty and outliers, however the accuracy is good enough to be able to shoot in the direction of the goal or to avoid opponents player in most cases. The two functions created can of course be improved by testing even more points for different target angles.

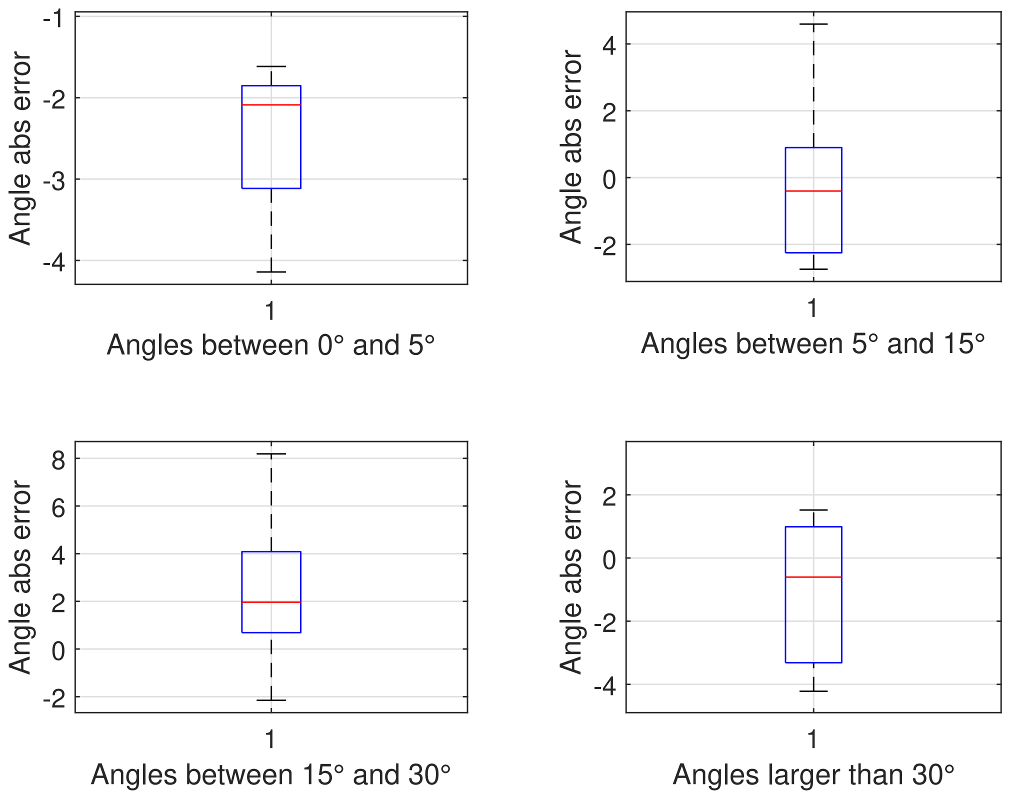


Figure 16: Boxplots of angle error with polynomial function

For the forward rod, during the tests made to construct the function in the same way as for the defenders in Fig. 14, it was observed that the function for shooting with a positive and negative target angle were completely different. For positive angles the model corresponded exactly to the one in Fig.14, however for negative angles the function corresponded to a 4 degrees polynomial function as shown in Fig.17.

The coefficients of this function are given in ascending order by :

10.807 -4.357 0.539 -0.0211 2.604e-4

The forward rod was more complicated to model because for some ball positions around -100mm and on the edges the players have a hard time reaching the ball and even less with the desired y displacement given by the model.

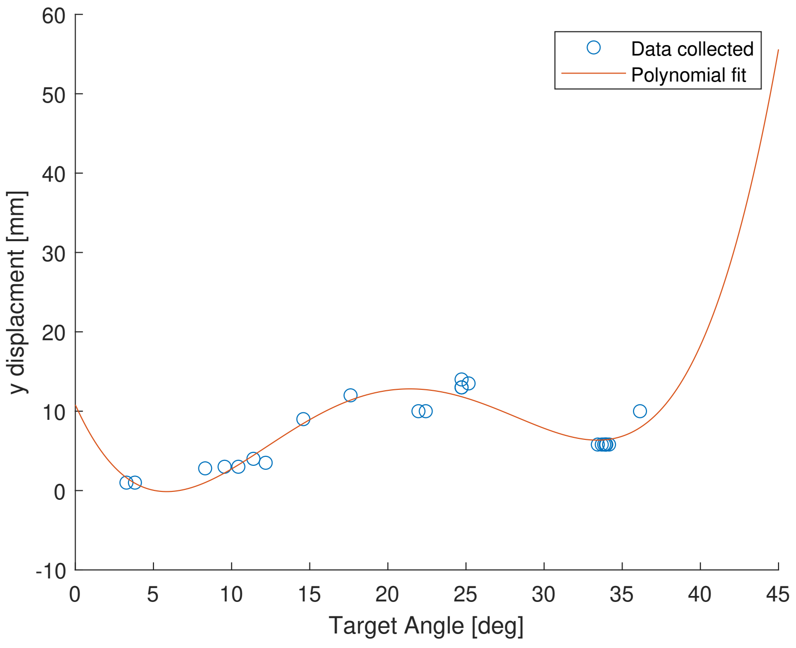


Figure 17: Function fit for forward rod

4.2 Translation

The second method tested is the shot strictly in translation. This method is more complex to implement than the rotation one as some targets can't be hit depending on the position of the ball and it requires a more complex shooting pattern to shoot correctly. This method is mostly efficient for angles around 0° and 180°.

The basis of this method consists of looking at the target angle sign, if the angle is positive, the shot is only possible if one of the player on the rod can be on its left in order to shoot with a motion from left to right. Inversely for negative angles. The next step is to position the player at a 40mm distance from the ball so that the player hits the ball with enough momentum. Then, the command is given of a fixed rod angle computed to hit the ball with an x displacement to obtain the desired post collision angle, and the translational command corresponding to the position of the ball in y with and added 40mm. Finally the rod is quickly rotated to a 90 degrees angle in order to avoid another player on the rod blocking the shot after collision.

The calculated offset from the ball's center, in order to hit the desired target point on the field is calculated in a similar way than for the rotation shot, only the displacement is in the x direction and the movement of the rod when taking the shot is in y not x.

With this method the oval shape of the player's feet on the bottom has a bigger impact than in rotation as the ball will be hit the bottom part of the feet which is rounded. Therefore the uncertainty of the shots is much higher.

4.2.1 Initial calculations

Similarly to the shots in rotation, the problem was also simplified to a 2D model. Considering the ball as a disk and the feet as a point. The problem is very similar except that the ball is hit from its side and not behind.

The target angle [1] is calculated in the same way. However the displacement from the ball center needs to be adapted as it will be in the x direction. And the player will need to be at a specified angle to hit the desired point on the ball in the x direction.

The initial displacement in the x direction can then be calculated as :

$$InitDispX = r * sin(TargetAngle + \pi) \tag{9}$$

π is added to as the collision happens on the other side of the ball.

The angle of the rod needed to hit the point in x is then :

$$RodAngle = asind(\frac{InitDispX - (PosBarX - Xball)}{L}) \tag{10}$$

$PosBarX$ is the position of the rod in the X direction, L is the length of the player from the bar to the bottom of the feet.

After some testing, it was noticed the angle of the shots were systematically lower than the target angles. It was then found that adding a constant of 15 degrees to the $RodAngle$ corrected this issue.

The Labview code of the translation shooting method can be found on Fig.18, the top part of the code finds the closest player that can shoot in the desired direction and then creates a path for the shot, with a list of position and angle orders. Fig.18 shows the case for a positive target angle.

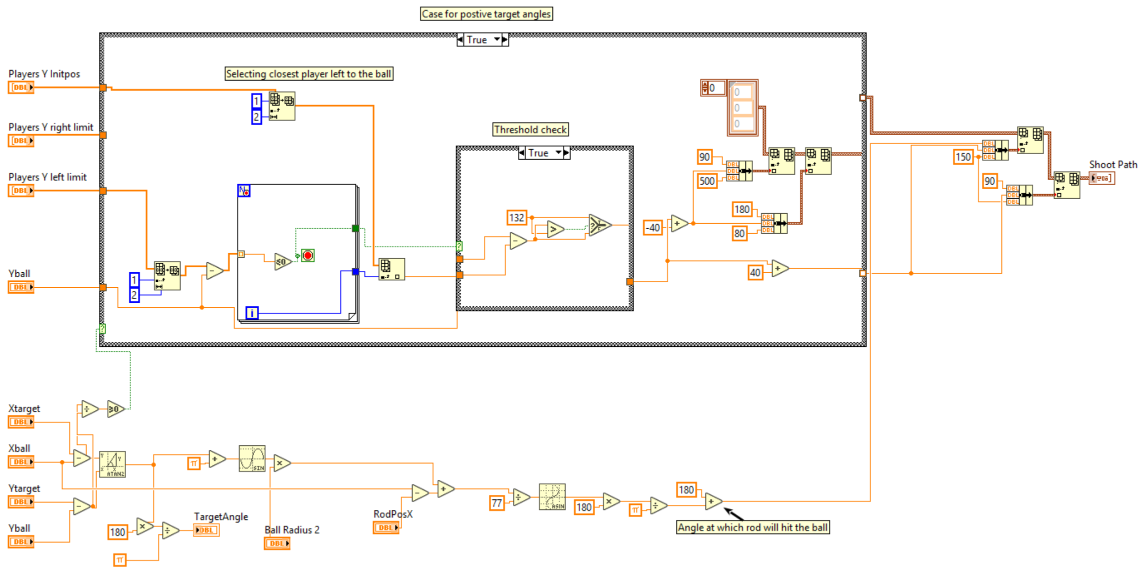


Figure 18: Labview code of the translation shooting method

4.2.2 Results

The results after the correction are given in Fig.19. The results are not extremely precise as the camera had a hard time giving the exact position of the ball on the edge of the table, therefore some of the angle post collision had to be estimated visually.

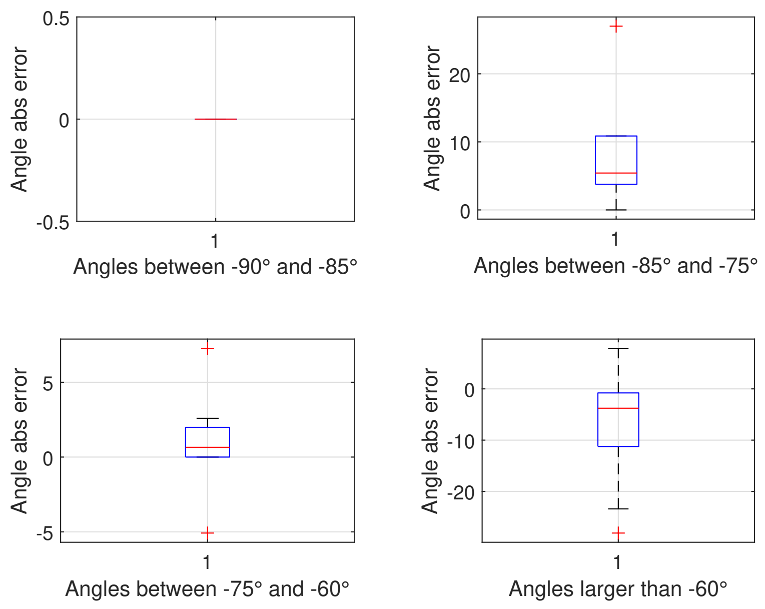


Figure 19: Boxplots of angle error for translational shooting

The accuracy is pretty high, the median value of the error doesn't go above 5 degrees, especially for very small angles. However, there are some big outlier points, the greater the target angle gets. This could be explained by the shape of the feet which is curved and this has a much higher impact in translation than for rotation. For angles larger than -60° the dispersion might be too big to ensure the shot hits the target consistently.

The precision is not perfect but should be enough if the use of this shooting method is to use for target angles from respectively 90° and -90° to 60° and -60°. The model could probably be improved by fitting a function in the same way as the rotational method.

4.2.3 3D model

A 3D model of the collision was also constructed on Matlab in a similar manner than for the rotation model. The algorithm has as input the desired target angle and outputs the needed added displacement in the x direction. The Maltab function can be found in Appendix B.

The algorithm iterates for a range of rod angles then displacement of the rod in y and finally different points of the feet in the z direction.

The equations of motion are given by :

$$x = sind(\alpha(i)) * (L - k) \tag{11}$$

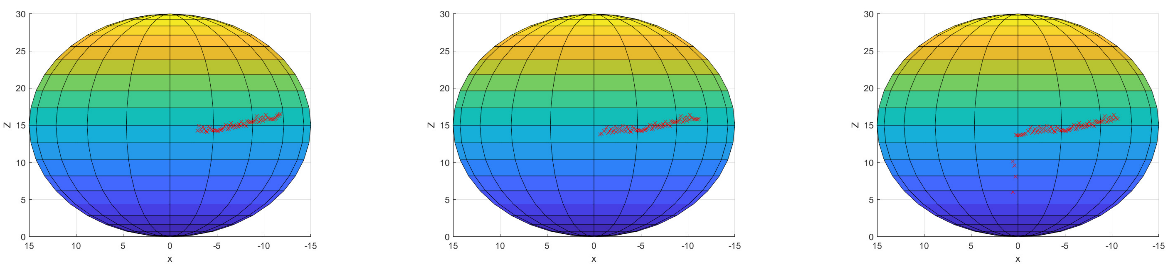
$$y = j \tag{12}$$

$$z = L - cosd(\alpha(i)) * (L - k) + 6 \tag{13}$$

with i indexing rod angles to test, j is the lateral displacement of the rod, for angles smaller than 90° the movement is in the positive direction if it's greater the movement is in the negative direction. k iterates the different points of the feet in the z direction.

The points of collision are also given by the equation 7.

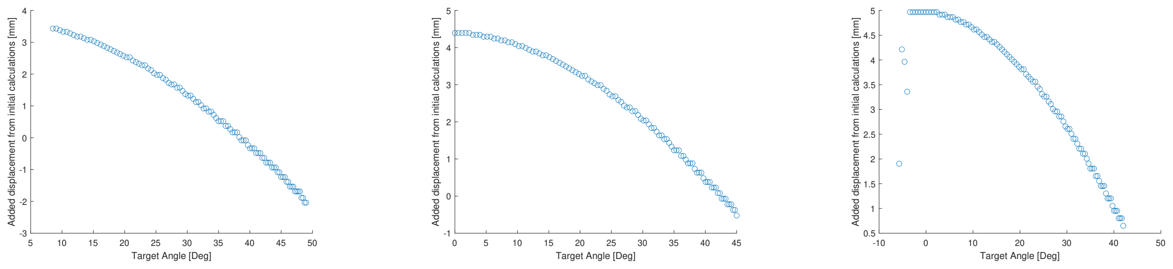
The points of collision on the ball for angles between 0 and 45 degrees are given in Fig.20



(a) Ball 30 mm behind rod (b) Ball under rod (c) Ball 20 mm in front of rod

Figure 20: Collision points between ball and feet for 3 different x positions

The difference in x displacement with the initial calculations is quite high especially for small angles, it can be up to 5mm which once again can cause a very important difference in angle.



(a) Ball 30 mm behind rod (b) Ball under rod (c) Ball 20 mm in front of rod

Figure 21: Difference in added y displacement compared to initial calculations

This model isn't used either for the same reasons as for the rotation method but, it could also be used in future works.

4.3 Translation + Rotation

The third and last method tested is the the shot in both translation and rotation. In the previous methods, one of the two motor commands were kept constant during the shot, the lateral displacement for the rotation shot and the rod angle for the translation shot. Actuating both the rotational and translational motors could be closer to the reality of a human shot, however it is hard to say before testing if this method will perform better than the two others.

For this method, the main complexity comes from hitting correctly the point of contact with the given y displacement, which ideally should be the same in both the rotation method and this one while not keeping the displacement in y constant.

This method is fairly similar to the rotation method as said before the angle post collision should in theory be mostly given by the displacement in y as given in the calculation for the rotation. This is because most shots in a game are usually around 90° up to ±45, therefore as the rotation method works better than the translation method in this range, it should have a more important role than the translation component of the shot.

4.3.1 Calculations

In order to ensure that the point of contact in y is the displacement given by 2, the path of the shot was divided in two. In the first part the command given for the rod angle is ranged between 90 degrees and the angle at which the rod will hit the ball given by :

$$RodAngleCollision = asind(\frac{(Xball - PosRodX) - r}{L}) \tag{14}$$

r being the ball radius, and L the players vertical length from the rod to the end of the foot.

The angle 14 should only be attained when the correct displacement is attained as well, otherwise the shot would be mostly translational. Therefore the lateral displacement of the rod is ranged between $dispY - 10mm$ and $dispY$ if the target angle is positive and between $dispY + 10mm$ and $dispY$ if the target angle is negative. $dispY$ being given by 2.

The second part of the shooting path is then the trajectory after the first contact of the ball. For that part the angle command ranges from 14 to 300°. The translational command ranges between $dispY$ to $dispY + 40mm$.

The LabView code for this method can be found on Fig.22. The two for loops are here to create the range of angle and position command respectively before the impact and after the impact. The cutoff is at the impact point to ensure that the orders given are exactly what was computed.

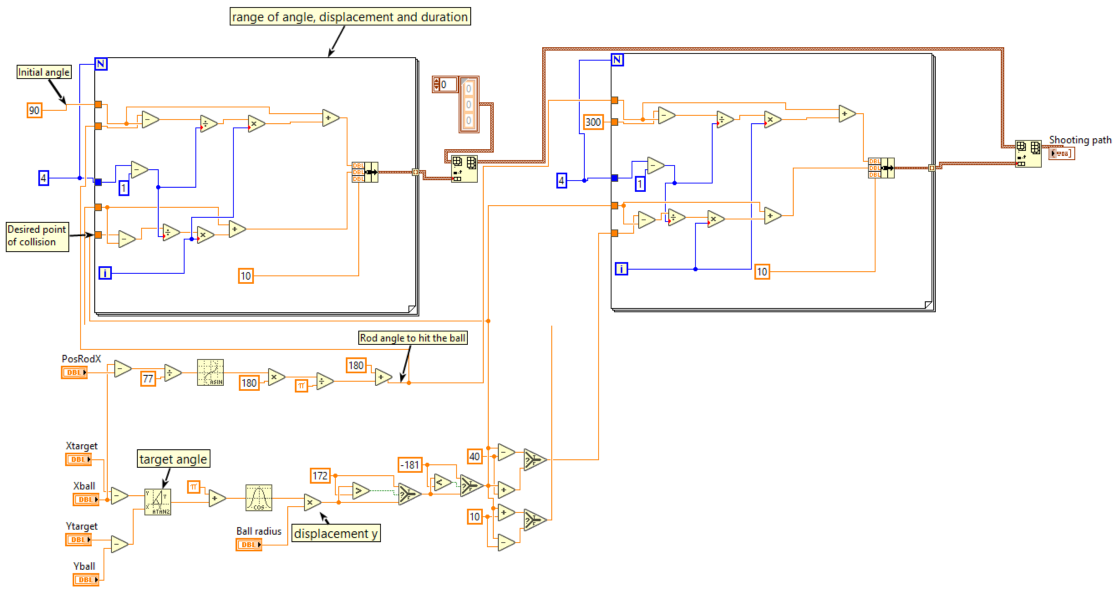


Figure 22: Labview code of the rotation + translation method

4.3.2 Results

With this method, the accuracy of shots for angles larger than 15° is fairly similar to the rotational method. The displacement added should of course be adapted, a lot less added y displacement is necessary as it is compensated by the translational component of the shot. Because of this component, it is almost impossible to shoot straight and for small angles as the translational motion can't be compensated enough.

The results for angles between 15 and 30 degrees and larger angles are given in Fig.23.

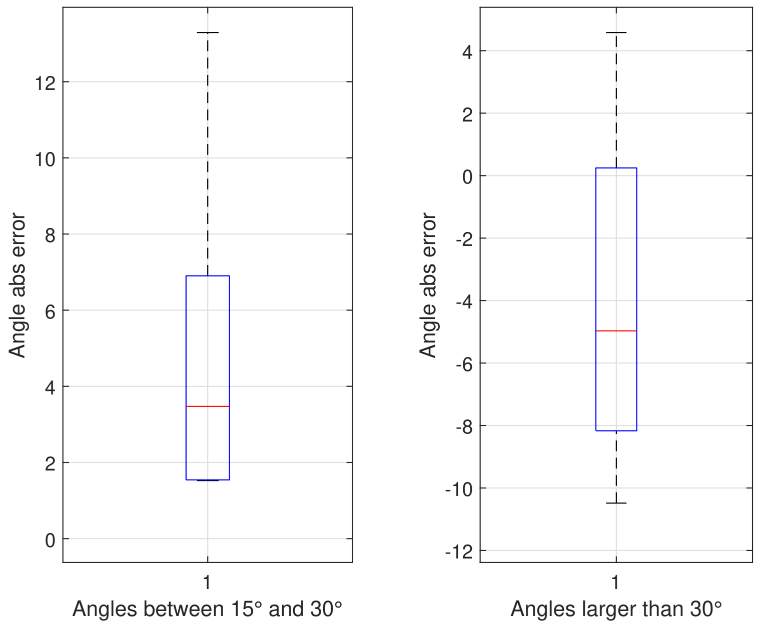


Figure 23: Boxplots of angle error for rotational + translational shooting

This method could be interesting to explore, however, for now there are more drawbacks to this method than advantages. The fact that it can't handle small angles is an issue.

4.4 Shooting conclusion

After a lot of testing of the three different method it is clear that the best way to shoot accurately from 0° to 180° is to use a combination of the strictly rotation and strictly translation methods. The rotation method has a larger range of angles that it can hit accurately. The translation being less accurate for angles larger than 30° and smaller than 150°, the final model is as follows: the rotational shot should be used for angles between 30° to 150° and the translational shot should be used for angles from 0° to 30° and 150° to 180°. There are however angle ranges which will be less accurate as both methods are not very efficient for them. This is the case for the range 30° to 45° and 135° to 150°. However these angles are rarely used so it shouldn't be a critical issue.

5 Dynamic

The models for the shots were developed in static mode, meaning the ball is not moving when taking the shots. The next logical step is then to test the shots dynamically in a real match-like setup. Since the best performing model for angles between 30° and 150°, is the rotational model, it will be the one used in general for the dynamical settings. The translation method and rotation + translation could be of use in future functions however right now they are not useful as the target will be set as the opponent's goal. In this section the tests will be focused on the forward rod which is the most likely to score.

5.1 Shoot

First, the shots were tested dynamically in a simple model. When the ball is further than 40mm away from the rod in the x direction the algorithm simply puts the closest player in front of the ball in order to block a possible shot. When the ball comes near the rod (less than 40mm) than the rod in question is switched to shooting mode, which corresponds to the last model of the rotation method with as a target the point (550,0) corresponding to the middle of the opponent's goal.

The shooting method is then chosen by looking at the position of the ball and the angle to the opponent's goal. In this model, if the angle is smaller than 10° or larger than 170° and there is a player which is able to shoot the ball in translation than the method chosen is the translation one otherwise the rotation method will be chosen. A player is considered being able to shoot if it there is a player close enough on its left is the target angle is smaller than 90° and on its right if the angle larger than 90°. The range for translation is purposely very small to focus on testing the rotation method in a dynamic setting. The LabVIEW implementation is given in Fig.24.

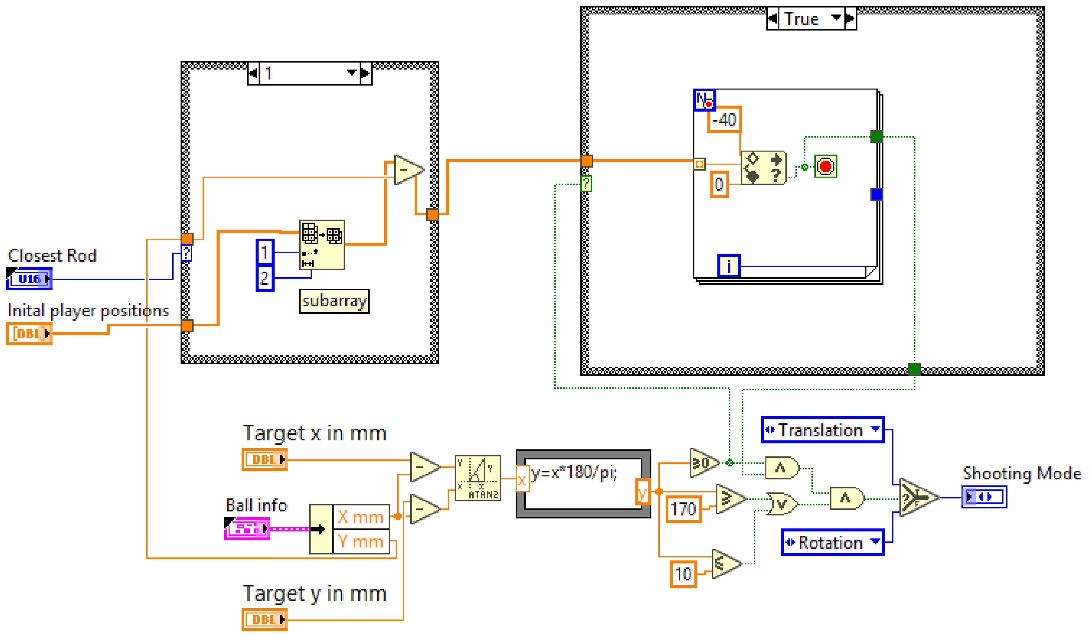


Figure 24: LabVIEW code of shooting method selection

The operation of the shots is the same as in static mode, the only difference is the ball's speed. It can impact the precision of the shots, depending on the speed of the ball. The faster the ball and the larger the angle at which it comes to the rod, the lower the precision will be. If the speed is too high the ball might pass under the player when charging the shot. It is therefore important to estimate the cutoff speed for which the shot's precision becomes not sufficient anymore. In which case another solution must be found. The two possibilities being to either compensate the incident angle by adapting the target angle that is given as input in LabVIEW, or the second solution would be to capture the ball and then shoot statically. The second solution, which is capturing the ball is of course much easier to implement and less random as when the ball is too fast it is hard to move the rod fast enough to adapt and the rebound can be complicated to anticipate

After some tests, it was found that the cutoff speed is around 500mm/s, this is an estimation, it is hard to have an exact value for it. For lower speeds, the velocity of the rod when shooting is much higher than that of the ball and therefore compensates any unwanted horizontal forces after collision.

5.2 Avoiding opponent players

With the shots being quite accurate, it is now possible to avoid the opponent's defenders and goalkeeper if they are not positioned in a way that block the goal completely, which is rare.

The initial target as said previously is the center of the goals which is at (550,0), however if there is an opponent player in the trajectory from the ball to this point then the target needs to be adapted to shoot at another point of the goal.

The trajectory of the ball is given by the following equation :

$$y(x) = \frac{x - Xball}{tan(TargetAngle)} + Yball \tag{15}$$

If for $x = WhiteRodPosX$ we have $y(x) = WhiteRodPosY \pm 40mm$, than one of the defenders or the goalkeeper is in the trajectory to the goals and will block the shot. The players are 20 mm large and the ball radius is 15mm therefore with a security margin, it is considered that 40mm around the opponent's player position the ball will collide with the player.

When the trajectory is blocked than the target point is modified, it iterates from -90mm to 90mm corresponding to the two borders of the goal with an error margin. When a trajectory is found than the shot happens. However if all trajectories are blocked it will still try shooting on the right border of the goal.

The LabView implementation of this is given in Fig. 25.

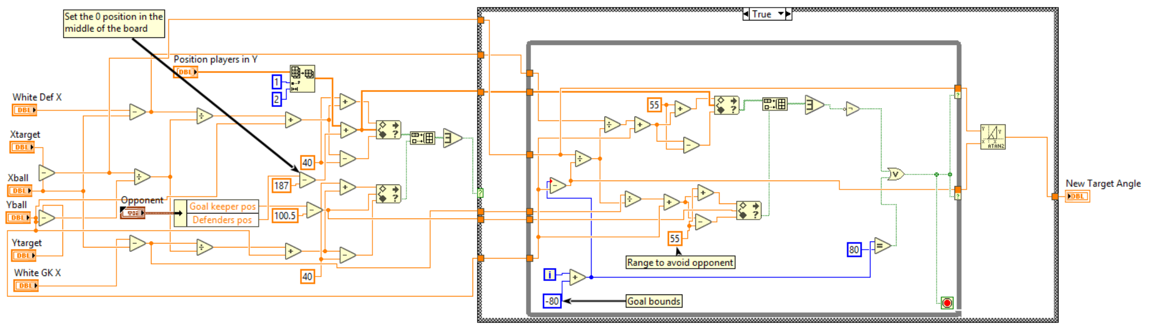


Figure 25: Labview code of avoiding opponents defenders and goalkeeper

This algorithm is quite effective when the opponent rods are static, but it could be improved to take into account the motion of the rods.

5.3 Capture+shoot

As said before, when the speed of the ball is too important it can be hard to shoot the ball with the desired target angle, therefore one solution would be to first capture it and then shoot statically.

A previous project from Yann Morize [1] worked on the capture of the ball, which is working to some extent. This work was done on a previous version of the main VI's before being reworked and also with different angle measurement equipment. Therefore, it had to be readapted to this new version and tuned for the new angular encoders. When simply adapting the inputs and outputs to what is done in the current version, the algorithm wasn't able to capture the ball at all. A few modifications were therefore needed.

The principle of the algorithm is that the player catching the ball will act as a damping mechanism, the rotation speed being ramped from the ball speed to 0. This method works well for ball speeds below 2m/s, but has more difficulty for higher speeds as the rod doesn't have time to send the orders fast enough.

Once the ball is static, it is then possible to shoot in the desired direction as in for static settings. This method is more efficient than trying to shoot at high speed, however it is easier for the opponent to prepare and block the shot.

A possible solution to counteract for this issue would be to first capture the ball and then juggle between players on the rod and shoot when an opening is found.

5.4 Capture + Juggle+ shoot

The next chain of action was then to add a juggling aspect into the picture, to make the shot less predictable while moving the ball around to find a path that is not blocked by the opponent's defenders or goalkeeper.

The juggling function was developed by another student project [2], the principle consists of varying the rod angle according to the position of the ball. The algorithm gives relatively good results, however it works best when the ball is very close to the position of the rod in the x direction. This can be an issue as the ball captured is usually quite behind the bar when stopped. This can cause the ball to be shot out of the rod range by accident when transitioning from capture to juggling.

The algorithm works as follows, the ball is first captured for all ball speeds. Then the trajectory from the static ball to the opponent's goal is checked to see if any players are in path. If the path is clear than the shot is taken aiming for the goal. If the path is blocked than the algorithm goes into juggle mode, in which the players will juggle the ball while checking at every iteration to see if the trajectory from the ball to the goal is clear. Whenever, a good trajectory is found than the shot will be taken.

This chain of action doesn't work very well yet for a few reasons. First, due to the issues with juggling as explained above , but also because, as is, the juggling is too fast, the rod doesn't have time to position itself well enough to hit the correct point in y on the ball to shoot in the desired angle. A deviation of a few mm from displacement in y can cause a big change in angle.

One possible solution to fix the issues would be to modify the juggling function to slow it done and try to improve ball control for larger distances in x around the rod axis. If these issues are fixed and this function is able to work as intended than it would be almost guaranteed to score when the ball is in front of the forward rod.

6 Conclusion

The first and foremost objective set for this project was to identify and fix the issues that were holding back the development of new functions and even made the machine less reliable than it was previously. A lot of the issues were identified and fixed such as the ball detection, and the electromagnetic interferences impacting the angular encoders rendering the babyfoot almost unplayable. Some issues coming from the homing maneuver were also identified but unfortunately weren't able to be fixed. It would be interesting for future projects to look deeper into this topic and the CME2 software responsible of it. The babyfoot is now working well very, small improvements could still be made but overall there are no critical issues impairing the functioning.

The second objective was to create a model that is able to shoot accurately to a desired target between 0 and 180 degrees, comparing 3 different shooting methods : strictly in rotation, strictly in translation and a combination of both. After a lot of testing with the different models constructed for each method it was determined that the rotation method is the very accurate for angles between 30° and 150°. The final model for this method is a combination of simple trigonometric calculations and a polynomial regression returning a displacement in y for a given target angle. For angles between 0° to 30° and 150° to 180°, the best method is the translation method, this model relies entirely on trigonometric calculations. Finally the last method tested, the combination of the first two, is as accurate as the rotation method, except for angles between 75° and 105°. Straight shots are hardly possible due to the transitional component of the motion. 3D collision models were also created on Matlab, they are not usable as is in a real time scenario but they could be an interesting starting point for a simulation of the babyfoot. A more precise model of the player's feet would be necessary for them be really useful, the approximation to a rectangular shape is too far from the reality for the model especially in translation to be really accurate.

Finally, the shots were tested in a dynamic setting, it was determined that for a ball speed of under 500 mm/s the shots are still very accurate, however for higher speeds, the accuracy goes down, if the speed is very high it can even miss the ball completely. With this new ability to shoot at a designated point on the board a lot of functions can be constructed with different chain of actions to improve the offensive abilities of the babyfoot. One very useful function that can use this ability is the possibility to select a ball trajectory to the goals that is not blocked by any of the opponent's defenders or goalkeeper. Moreover, to counteract the issue of shooting the ball at high speed, a function capturing the ball before shooting statically was developed delivering good results. This method is pretty efficient but the static shot could be predictable for the opponent. Therefore, juggling was also added into the chain of action. This last function isn't working perfectly yet but with changes in the juggling function, this function could almost guarantee to score when the ball is in front of the forwards.

The results of this project were really satisfying in the end, all the objectives fixed were attained and this ability to have a better control of the ball's trajectories will enable to create a lot of new functionalities for future projects.

Appendix

A

Listing 1: Collision Model Rotation

```
function  [xcol,ycol,zcol,anglecol]=TestCollision(Xball,r,L,posbarx,
    TargetAngle,initdisp)
minangle = round(asind((-r- Xball+posbarx)/L)) -1 ;
alpha = linspace(minangle,30, 200-minangle) ; % angles of the player to be
    tested (angle of 30 corresponds to lowest feet position in Z of 16.3mm)
AngleFound =0; %flag when angle is found
xcol= 999;
ycol= 999;
zcol =999;
anglecol= 999;
if TargetAngle<=100 && TargetAngle>=80
     low = initdisp -1;
     high = initdisp +1;
if TargetAngle<=90
     high = initdisp -2;
     low = min(0,initdisp+2);
else
     low = max(0,initdisp -2);
     high = initdisp+2;
end
for j = linspace(low,high,100) % disp
     for i = 1:length(alpha)
          for k = linspace(0,10,20) % can touch from z(i) or above
               x = sin(deg2rad(alpha(i)))*(L-k) + Xball - posbarx;
               z = L-cos(deg2rad(alpha(i)))*(L-k) + 6 ;
               if (x)^2 + (j)^2 + (z-r)^2 <= r^2 +2 && (x)^2 + (j-0)^2 + (z-r)
                   ^2 >= r^2 -2
                         angle = rad2deg(atan2(-x,-j));
                    if ((TargetAngle-1) <= angle(end) && angle(end) <=(
                        TargetAngle + 1))
                         AngleFound = 1;
                         xcol= x;
                         ycol= j;
                         zcol =z;
                         anglecol= alpha(i);
                         break
                    end
                    break
               end
          end
          if AngleFound==1
               break
          end
     end
          if AngleFound==1
               break
          end
end
```

B

Listing 2: Collision Model Translation

```
function [xcol,ycol,zcol,anglecol]=TestCollision(Xball,r,L,posbarx,
    TargetAngle,initdisp)
minangle = round(asind((-r- Xball+posbarx)/L)) -1 ;
alpha = linspace(minangle,30, 200-minangle) ; % angles of the player to be
    tested (angle of 30 corresponds to lowest feet position in Z of 16.3mm)
AngleFound =0; %flag when angle is found
xcol= 999;
ycol= 999;
zcol =999;
anglecol= 999;
if TargetAngle<=100 && TargetAngle>=80
     low = initdisp -1;
     high = initdisp +1;
if TargetAngle<=90
     high = initdisp -2;
     low = min(0,initdisp+2);
else
     low = max(0,initdisp -2);
     high = initdisp+2;
end
for j = linspace(low,high,100) % disp
     for i = 1:length(alpha)
          for k = linspace(0,10,20) % can touch from z(i) or above
               x = sin(deg2rad(alpha(i)))*(L-k) + Xball - posbarx;
               z = L-cos(deg2rad(alpha(i)))*(L-k) + 6 ;
               if (x)^2 + (j)^2 + (z-r)^2 <= r^2 +2 && (x)^2 + (j-0)^2 + (z-r)
                   ^2 >= r^2 -2
                         angle = rad2deg(atan2(-x,-j));
                    if ((TargetAngle-1) <= angle(end) && angle(end) <=(
                        TargetAngle + 1))
                         AngleFound = 1;
                         xcol= x;
                         ycol= j;
                         zcol =z;
                         anglecol= alpha(i);
                         break
                    end
                    break
               end
          end
          if AngleFound==1
               break
          end
     end
          if AngleFound==1
               break
          end
end
```

References

[1] Morize Yann, 2019, *Vision and strategy: Ball capture*

[2] Lacroix Antoine, Montorfani Samuel , 2022, *Vision and strategy: Juggling and shooting*