École Polytechnique
Fédérale de Lausanne

Foosball project
Master 3

Vision and strategy: Ball capture

Author:
Yann Morize

Professor:
Christophe Salzmann

January 2020

Contents

1 Words for reader **4**

2 Introduction **5**
2.1 Project objectives . . . 5
2.2 Current state of the project . . . 5
2.2.1 Vision precision . . . 6
2.2.2 Vision loop speed . . . 7

3 Vision **8**
3.1 Speed computation . . . 8
3.1.1 Adaptive algorithm . . . 9
3.2 Camera issues . . . 9

4 Strategy **11**
4.1 Current state . . . 11
4.2 Goal of the strategy . . . 11
4.3 Lateral position prediction . . . 11
4.3.1 Speed extrapolation . . . 12
4.3.2 Auto-regression . . . 13
4.3.3 Linear motors control . . . 13
4.4 Rotation motion - Ball capture . . . 14
4.4.1 Rotation strategy . . . 14
4.4.2 Control reference position . . . 16
4.4.3 Control reference speed . . . 17
4.5 Attack . . . 19
4.6 Issues . . . 19

5 System Identification **20**
5.1 System description . . . 20
5.2 Dataset . . . 20
5.2.1 Data set verification . . . 20
5.3 Parametric identification . . . 21
5.4 Conclusion . . . 23
5.5 Further Projects . . . 23

6 Annex **24**
6.0.1 Identification . . . 26

Bibliography **27**

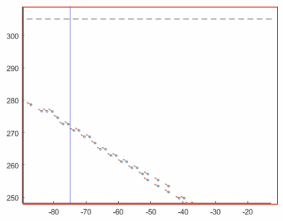
Executive summary

Mission Statement

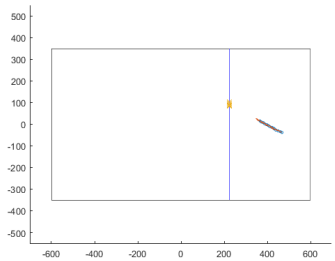
The goal of this project is upgrade the existing strategy of the automatized Foosball by intercepting the ball in order to have static possession. To do so, the speed of the ball had to be computed as well as a prediction of future positions to place the players in advance on the trajectory of the ball.

Highlights

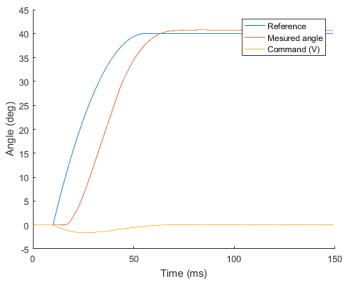
- **Speed computation**
  Accurate smoothed speed has been computed with a adaptive window with moving average method.



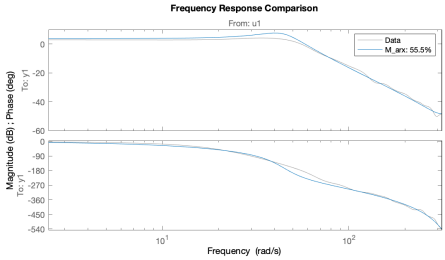
- **Lateral ball position prediction**
  1cm precision is achieved on lateral ball position prediction by applying Auto-regressive linear model.



- **Ball damping**
  A strategy using a specific position reference profile is generated to make the player damp the ball and keep its possession.



- **System identification**
  The report puts in light that the dynamic of the linear motors is too slow for the game. This is a particularly weak point during diagonal shots. The closed loop system has been estimated with an ARX model. The accuracy of this model is about 40% which shows that the vibrations make the system nonlinear.



Laboratoire d'Automatique

EPFL STI IGM LA
ME C2 408 (Bât. ME)
Station 9
CH-1015 Lausanne

Téléphone : + 41 21 693 38 49
E-mail : christophe.salzmann@epfl.ch
Site web : https://la.epfl.ch

Lausanne, September 2019

Semester Project ME
Fall 2019

Candidate: **Yann Morzine**
Mechanical Engineering Section

Subject: **Babyfoot: Ball capture**

Le but de ce projet est l'amélioration du babyfoot existant. La stratégie actuelle bénéficierait fortement de la possibilité de pouvoir capturer la balle et de la positionner à un endroit atteignable pas les joueurs du robot.

Un algorithme d'interception de la balle sera développé et implémenté. Pour ce faire la vitesse devra être estimée pour positionner correctement le joueur en cas de tirs en diagonale et une stratégie d'arrêt devra être imaginée et mise en œuvre.

Le nombre de crédits réservés au plan d'études pour ce projet est de **dix.**

Dr. Christophe Salzmann

1 | Words for reader

Dear Reader,
If you are a student looking for an exciting project to work on, the Foosball is the perfect option. You will spend some time at the beginning to understand the actual system and all the different work that has already been done on this project by previous colleagues. You will learn a lot during this phase, not only about the Foosball but more generally about how to vehicle information within a project. Once you will have understood the soft and hardware, you will be able to implement codes and put them into direct application with some great technological tools (linear motors, encoders, drivers, dissipators,...). Moreover, you will be supported by passionate people, I have myself been supported by Prof. Ch. Salzmann who has been very present during the whole semester. I have learned so much thanks to him.

Acknowledgement

I would like to address a very special thank you to Prof. Ch. Salzmann who has been highly present, supportive and always motivated for this project. He has guided me to understand the current state of the project and its different systems while constantly supporting new ideas. I would also like to thank Dr. Alireza Karimi and Mr. Philippe Schuchert for their help in identifying the linear motors.

2 | Introduction

During this semester, I have decided to work on the Foosball as a semester project during my master 3. Working on a project on which several students have already been working for couple of years and trying to find new ways to upgrade it boosted my motivation. Before implementing any upgrades or changes, it was first necessary to clearly understand the work that has already been done in the past. I have spent the first month going through all the reports and the codes, trying to see which ones are currently included in the actual code and how they work.

2.1 Project objectives

My main goal was to be able to control the ball. I thereby wanted to capture the ball in an elegant way so that the future students working on this project could concentrate more on implementing strategies by having static ball possessions. To reach this goal several sub-objectives where fixed:

- Understand the hardware and software of the Foosball.
- Vision
  - Compute accurately the speed of the ball.
  - Predict future positions of the ball.
- Strategy
  - Capture the ball by damping it with the player.
  - Position the ball in order to attack or do a pass.

2.2 Current state of the project

As the Foosball has been presented during the 50 years of EPFL exposition, some modifications and improvements have been done by Prof. Ch. Salzmann. In this section a brief description of how the program gets the position of the ball is made.

The different steps from acquiring the image to getting the position are described on Figure 2.1.

1. **Camera**: The actual camera is a Ximea MQ003CG CM, with a maximum frame rate of 500 fps.
2. **Pattern selection**: On the Labview front panel one has to mark the ball by putting it in a box pattern. Note that it may be necessary to adjust the luminosity previously in order to see the ball.
3. **Image acquisition**: This VI acquires the most recent frame from the camera and imports it into the program.

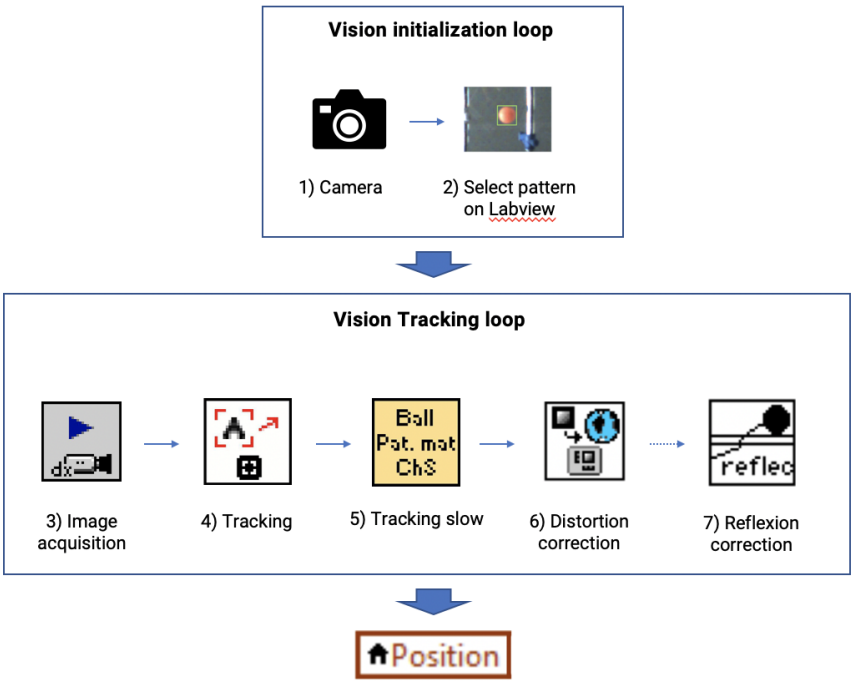


Figure 2.1: Structure of the vision part.

4. **Tracking algorithm**: The fast algorithm scans a limited part of the table and computes the correlation between the pattern we have selected in the initialization loop and the pixels it sees. The sensitivity of this function can be set with the control *"Vision min score"*. Where 1000 would represent a 100% match required between the pattern and the image in order to affirm the ball has been found. The actual value of 600 seems to be plausible.

5. **Tracking slow**: If the fast algorithm hasn't found the ball, the slow algorithm is activated. This algorithm will scan all the pixels of the table until he find a pixel of orange color. He will then apply the fast algorithm on this specific part of the picture to compute the exact position of the ball. This subVI is about 10 times slower than the fast one. It is usually used when the ball cannot be detected (sides of the Foosball, thrown in the air, too fast speed or abrupt change of direction).

6. **Reflection correction**: This subVI corrects the deformation of the angle perceived by the camera due to the refraction of the Plexiglas.

7. **Angle correction**: It has been created last summer by Prof. Salzmann. It takes into account that the algorithm only localizes the center of the ball. Therefore, when the ball is not right over the camera, the center of the ball is not recorded correctly. An angle correction is thus performed.

8. **Ball position**: The precise position of the ball is stored in the variables x and y. The accuracy of the results will be discussed in the next section.

2.2.1 Vision precision

The precision of the measured position is crucial for this project. Several tests have been performed to define the actual precision of our system.

The first step was to verify if the algorithm could return the correct position of the ball. To do so, a physical measurement has been performed and tape has been put on the specific measured spot. Then, the coordinates returned by the algorithm have been compared to this physical measurement. The results were very satisfying as one could observe less than **2mm** of error between the position returned by the algorithm and the real position.

Als, when trying to compute the speed (by finite difference) it was clear that something was wrong. The recorded position was doing jumps over time, therefore the speed could not be computed accurately. After a week of observation, it was concluded that the problem was caused by the camera. It has thus been decided to connect the camera to the USB3 and the strange measurement disappeared. Now that the camera was working properly, measures made more sense. On Figure 2.3 one can observe the recorded position of the ball on the x axis with respect to time. These measurements are very similar to what Yoann Moret had observed in his report [2].
The gaps are due to the pixels size. One cannot be more precise than 2mm as the maximum precision of the camera seems to be reached.

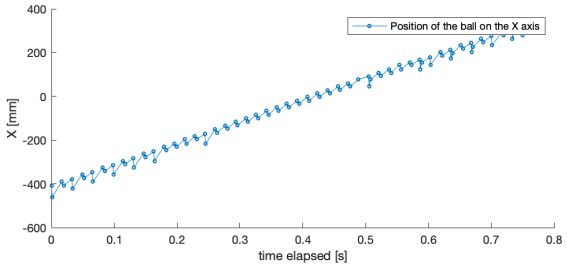


Figure 2.2: Position of the ball regarding the x axis when connected to *USB2* port.

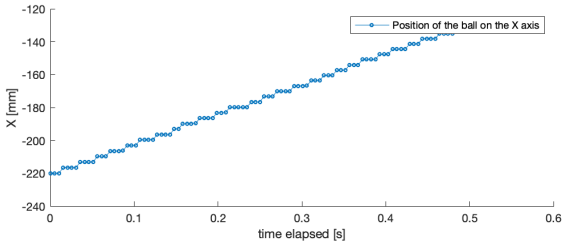


Figure 2.3: Position of the ball regarding the x axis when connected to *USB3* port.

It was now possible to find out a way to compute the speed as accurately as possible. The major challenge being to smooth the gap caused by the pixel precision.

2.2.2 Vision loop speed

Another determinant parameter is the speed at which we acquire the new positions. By running the current algorithms with the camera connected to the USB3 port, we reach a rate ranging between 300 and 500 [Hz] depending on the luminosity parameter we have chosen as the aperture of the camera will change in consequence. This means, we record a new position every 2.5 [ms] in usual conditions. Knowing that a professional player can move the ball up to 15 [m/s][4], this would mean that we would get one measurement every 37mm. Nevertheless, when trying to shoot as fast as possible we only reached speeds of about 3 [m/s], which means we will get measurements every 7.5 [mm]. This is far more reasonable knowing that the players are distanced by 150mm between each row. We could thus theoretically compute the accurate speed of the ball and even predict it's position before it reaches the next row of players.

3 | Vision

3.1 Speed computation

Now that the tracking seems to work precisely, the speed of the ball at each point can be computed. To do so, the positions of the ball recorded in labview have been written in a Text file and processed in Matlab to ease the assessment of different approaches. The main issue being the smoothen the noisy measurements of the ball position due to the pixel precision (see Figure 2.3) as simple finite difference would compute very inaccurate velocities due to this noise (see Figure 6.1 in Annex).
Two methods have been assessed:

- **Step:**
  The idea was simple, for speed computation, remove some measured positions to avoid having position redundancy under the two-millimeter precision of the camera. Therefore, every nth measurement will be kept. This method seems to work properly as the speed computation is smooth (Figure 3.1).

- **Moving average:**
  To smooth the velocity and keep all the data points, an averaging method has been assessed. The concept being to compute the average of the position on a certain range and do a finite-difference on a shifted average. Note that, by doing this method one actually assigns to the point i the speed of the point $i-(n/2)$ with n being the range on which the average has been computed. This creates some kind of inertia because after an abrupt change of direction the speed changes more slowly. This specificity could be useful for the strategy.

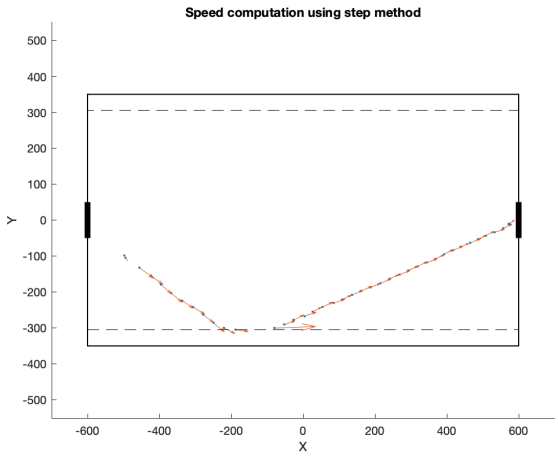


Figure 3.1: Step method with n=8.

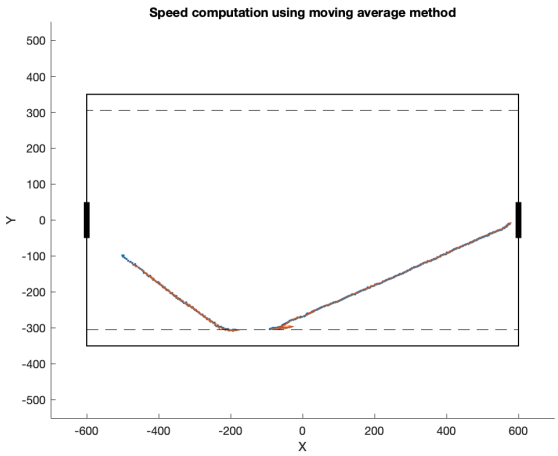


Figure 3.2: Moving average method with n=8.

By comparing theses figures it is obvious that the step method loses important information as it deletes many measured points. Moreover, the "inertia" on the speed caused by moving average can be interesting as the prediction will be smoother and therefore the commands given to the motors won't be too abrupt. It was thus decided to keep the averaging method for the speed computation.

3.1.1 Adaptive algorithm

Having a constant averaging range (n) makes little sense as it's purpose is to remove the noise of small movements (<2mm). Surely when the ball exceeds a speed (vx or vy) of about $2mm/loop_{iteration}$ no averaging would be needed. Nonetheless, in order to have a smoother speed profile, the margin has slightly been increased. Therefore an adaptive algorithm has been implemented. Its specifications are summarized in the following table:

Averaging window (n)	**Distance between measurements [mm]**	**Speed of the ball [m/s]**
3	>7.5	>3
6	4-7.5	1-3
10	<4	<1

Note that the value *Distance between measurements* is compared to $max(\Delta x, \Delta y)$ because the noisy measurement is caused by the pixel precision along the x or y axis. Therefore, with a shot along the x axis, little noise would be perceived. The worst-case being a slow diagonal shot. The value for *Speed* in this table is only indicative. It has been computed with a rate of 400 fps but this value will change depending on the exposure of the camera. This adaptive algorithm will be crucial for the defensive strategy implementation. As the defense will need to know precisely from where the ball comes and at which speed. Adapting the window size is necessary as the speed will have to be computed accurately on very short distances (See Section 4.4). The results look very satisfactory as noise is drastically reduced, Figure 3.5 show this in image:

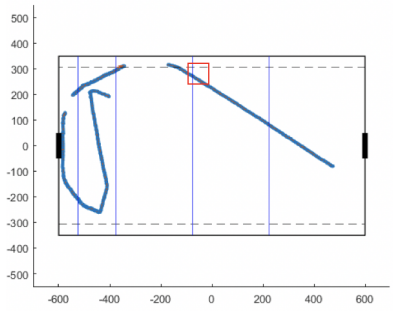


Figure 3.3: Ball position with speed computation

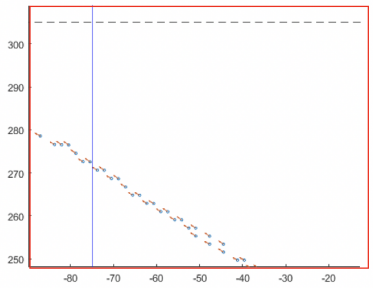


Figure 3.4: Zoom

Figure 3.5: Speed visualisation

3.2 Camera issues

In order to compute the speed of the ball, precise measurements of the ball position are needed. This can only be achieved by connecting the camera to the USB 3 port (see Section 2.2.1). A problem when connecting the camera to this port has already been observed in previous reports: the system was shutting down and showed an error from the camera image acquisition VI. As in the previous version of the Foosball the speed of the ball did not have to be known, the camera stayed connected in the USB2 port. For this project though, the speed computation is necessary

to implement more advanced strategies. It was thus necessary to solve this issue. During this semester, prof. Salzmann and myself have tried many different setups to identify the source of the problem. An exhaustive list of the different test and their observations has been made:

1. **Influence of the motors**
   As the crash occurred when the actuation was active the issue may come from an electromagnetic interaction between the driver and the USB3 card. It was thus attempted to find a pattern of input signals for which the system would fail.
   After trying different random noise input signals with different amplitudes and with periodicity (PRBS) the system failed often for high-frequency random noise (1 input every loop or every two loop iterations) but no clear pattern could be observed. Same observation when looking at the output voltage of the motor. Note that if some more tests are done it is important to reduce the timeout of the image acquisition VI and to take it into account when observing measurements.

2. **Vibration**
   It was also observed that the table was creating non-negligible vibrations. The computer has thus been taken out of the shelf and put on the floor.
   This test did not change anything.

3. **USB3 card**
   Maybe the card was in some way damaged. A new card has then been installed and tried. This newly installed card required an external power supply.
   This test limited the frequency of failures but they still occurred when using the original VI (front only).

4. **Software**
   To ensure that the problem was not caused by Labview it was attempted to read the camera from another software (MAX) while exiting the input of the linear and angular motors with white noise. MAX showed a timeout error message. Thereafter it has been attempted to do the same test by changing cameras and USB drivers. The results were:

   - Camera color:
     - NI (MAX): Timeout Error
     - Ximea: No error
   - Camera B/W:
     - NI (MAX): No error
     - Ximea: No error

   Later on, Prof. Ch. Salzmann removed the vision part from VI that inputted the white noise signal. He then created a new VI with only the vision part. When using the color camera and running both VIs at the same time the simulation lasted for 2 hours without crashing.

The conclusion is not very clear. The last test is the most surprising as it points out that the problem resides in the USB driver from NI. But this is confusing as the system does not crash when the motors are not actuated or when the B/W camera is used.
The only thing one can deduce is that the problem is an interaction between the camera, the motors and the NI USB driver.

4 | Strategy

4.1 Current state

In this version of the Foosball, the lateral position of the payers is set by the y position of the ball. Even though this seems simplistic it works very well thanks to the speed of the motors. The rotation motion is defined by a state machine that is defined in the VI "Strat". The essence of this strategy resides in shooting the ball as soon as it is reachable by a player.

4.2 Goal of the strategy

The goal of this section will be to implement prediction of the future position of the ball in order to place the player in advance and anticipate the position they should have to intercept the ball. Two sub-goals have been set:

1. Place players laterally based on prediction of the position.
2. Implement a defense strategy where the player would intercept the ball and keep the possession.

4.3 Lateral position prediction

Once again, different methods have been assessed. The first one was to project the position of the ball on the axis of the players based on the actual speed of the ball. The second was a projection by least-square autoregression.

Some constraints have been set to avoid unnecessary and wrong predictions. The following conditions have to be respected to use the predicted y_{pred} position. If any of them is not respected the current y_{ball} position of the ball will be set as reference.

1. $250mm/s < \|v\| < 8000mm/s$
   Decent speed rates to avoid noisy and inaccurate predictions.
2. $v_x < 0$
   Negative x-wise speed as we are only testing the defense mode of the forward player. This condition will have to be adapted when implementing the strategy to other players such that passes between rows of players can be performed.
3. $\left|y_{pred}\right| < 350mm$
   Predicted y_{pred} must be within the Foosball dimensions.
4. $d_{ball-player} > 25mm$
   The distance between the actual x position of the ball and the axis of the player ($d_{ball-player}$) must be greater than a threshold. As it will be illustrated on Figure 4.1, the x position where the ball touches the player is not exactly the position of the player's axis.

4.3.1 Speed extrapolation

This method is the most straight forward: compute future positions of the ball by projecting the actual speed of the ball on a certain axis.

The projection has been calculated from the following relations:

$$t_{\text{until impact}} = \frac{x_i - (x_{player} + d_{ball-player})}{v_{x,i}} \tag{4.1}$$

$$y_{projected} = y_i + v_{y,i} \cdot t_{\text{until impact}} \tag{4.2}$$

Where x_i, y_i and $v_{x,i}$,$v_{y,i}$ are respectively the current position and speed of the ball. x_{player} is the x coordinate of the row of players and $d_{ball-player}$ is the distance between the centre of the ball and the x coordinate of the row of players. $d_{ball-player}$ is necessary to include in the projection as the player will have to be placed at the point where he will touch the ball. This notion is very important as it is crucial to understand that the point and line drawn in Figure 4.2 are in reality physical objects. Figure 4.1 clearly shows how $d_{ball-player}$ is measured.

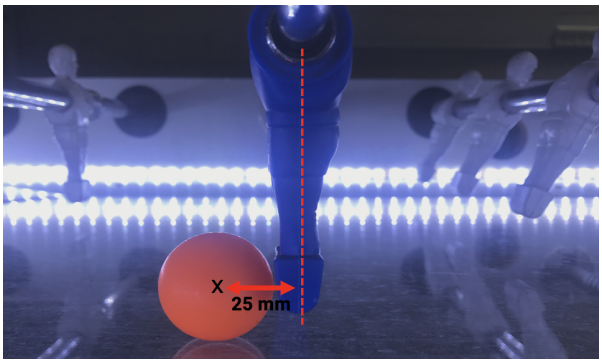


Figure 4.1: Measure of $x_{ball-player}$

As one can observe on Figure 4.2, the position prediction is not as accurate as desired. Even if the noise is reduced by the moving average speed computation method, there is still some left. When projecting future speed, this small noise has big effects on the prediction. On Figure 4.2 one can observe an error of about 100mm for the prediction. This results in fast oscillations of the defender on his axis. This method has poor performances when it comes to capturing the ball. Moreover it is not pleasant to play against a constantly oscillating robot.

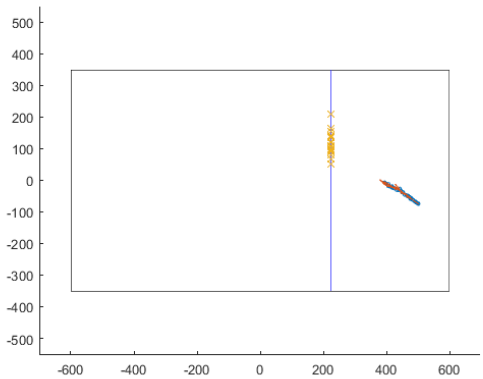


Figure 4.2: Lateral prediction of defense with speed projection method.

4.3.2 Auto-regression

The main assumption of this method is that the ball does linear trajectories. Therefore, instead of computing the speed at each point and extrapolating it, a linear regression is performed on the last 10 measured points. Then the coefficients of the linear regression are used to evaluate the future position of the ball on the axis of the defenders. This method turned out to be highly performant as it is extremely precise. Figure 4.3 shows the estimation of a diagonal shot at about 2m/s. The prediction has a precision of 10mm which is about the width of the player.

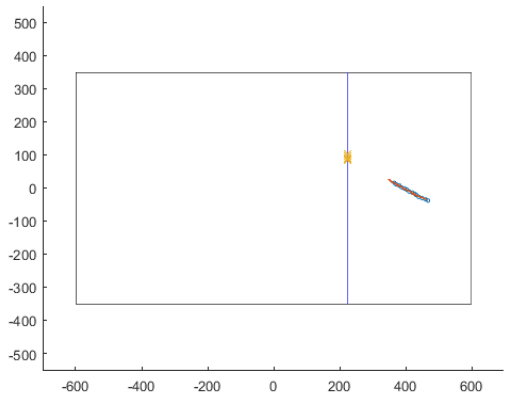


Figure 4.3: Lateral prediction of defense with auto regression method

An other advantage of this method is the smoothness of the prediction. If the ball changes direction, the prediction will change progressively. The regression window on 10 measurements is kept constant (unlike the speed computation) in order to avoid noisy predictions and oscillations. Moreover, this strategy will be used on all the rows of players simultaneously. Therefore, if the first row of players misses the ball, the other players will already be in position to intercept it.

4.3.3 Linear motors control

It has been observed that even with the precision of the auto regressive method, if one performs a fast diagonal shot ($\sim 3m/s$) the robot cannot intercept the ball. After several investigations the issue was localized: the linear motors cannot follow the reference input fast enough. This can be observed on Figure 4.4.

The response of a step input of 100mm takes approximately 110ms to reach the reference point. A simple calculation can show the relation between the raise time of the system and the time for the ball to reach the row of players:

- Assuming a diagonal shot from the goalkeeper at $3m/s$ with an angle of $30degrees$ and trying to stop it with the robots forwarders.

  $$v_x = 3 \cdot cos(30deg) \approx 2.6m/s$$

  The time it will take to the ball to go from the shooter (adverse goalkeeper) to the defender (robot forwarder) will be:

  $$t_{shoot-touch} = \frac{d}{v_x} = \frac{200mm}{2.6} \approx 75ms$$

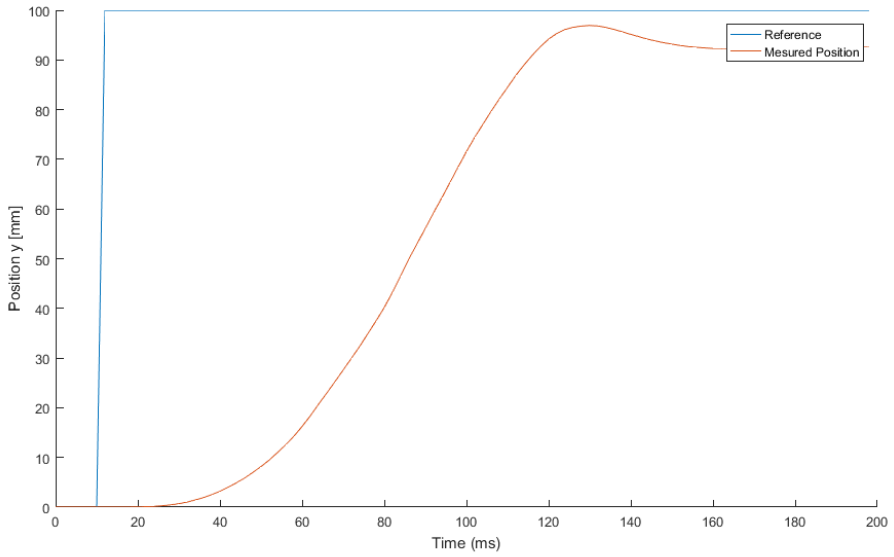


Figure 4.4: Step response of the actual linear motors controller.

Note that the distance d was slightly underestimated in order to make a simple comparison and obtain a predicted position 100mm from the initial reference point (to compare with the step response). Thereby, one clearly deduce that the motor control is not fast enough ($110mm > 75ms$). The slow dynamics on the linear motors are mostly due to the weight of the motor and the rod.

Keeping in mind that even if the first row of defenders does not capture the ball, the rows of players behind it will have more time to reach their predicted positions. This strategy is thus applicable and might be way more efficient than the actual one.

4.4 Rotation motion - Ball capture

The objective of this part was to capture the ball in movement in order to have it in static possession. This strategy could indeed boost the performance of the Foosball as advanced control could thereafter be implemented to either pass the ball between players or do very precise attacks.

Many previous reports have tried to stop the ball in different ways. Most usually they tried to put the player at a certain angle such that the ball would be blocked. Unfortunately, this strategy did not perform as well as expected. It this report another approach will be tested: damping the ball with the player. To do so the objective is to input a ramp speed command starting at the speed of the ball and finishing at zero.

It has been discussed whether to control the player in position or in speed. Previous reports having essentially commanded the rotation motion with a PID on the angle position of the player, this controller was already implemented.

4.4.1 Rotation strategy

The VI *'YANN DEFENSE'* executes 2 steps one after each other:

1. Record speed of the ball
2. Apply damping command by activating position profile generation.

This sequence has been performed because the VI 'Generate speed' takes a constant initial speed as input. During the first tests, this method did not perform well: the ball just bounced against the player before he even started moving. Obviously the system had some delay. By having a closer look at the response of the system to a step input one could observe a delay of about 3ms (see Figure 6.3 in annex). For the parabolic position input reference, the delay before reaching the desired speed was about 20 ms (see Figure 4.7). In order to compensate this delay two adjustments where made:

- **Sending the command earlier**
  The speed of the ball is stocked in a variable when its position is between $D_{\text{start rec speed}}$ and $D_{\text{send command}}$. Once it passed $D_{\text{send command}}$, the input speed is kept constant in a buffer during the damping of the ball. When the ball passes $D_{\text{send command}}$ the speed profile is generated and the command is activated.

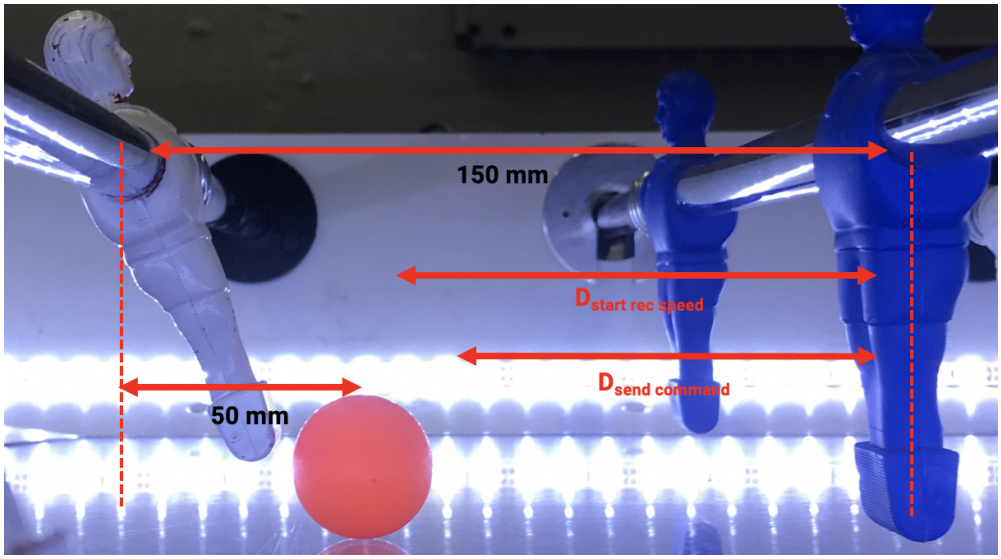


Figure 4.5: Distances explanations.

Note that the speed of the ball can be computed accurately for $D_{\text{send command}} < 76mm$ because:

$$D_{\text{send command,max}} = 150mm - 50mm - (2ms * 4mm/ms * 3frames) = 76mm \quad (4.3)$$

Recalling that for a fast shot (4m/s in this case) the speed computation would use an averaging window width of 3 and a vision iteration loop speed of 2ms. It can be observed that $D_{\text{send command,max}}$ is actually very small. If $D_{\text{send command}} > D_{\text{send command,max}}$ the speed of the ball will be underestimated. The rotational speed of the defender will be too slow and thus the ball will bounce against the player. This is not a big issue because even tough the ball will not be damped, it will at least be stopped from going closer to our goals. A larger $D_{\text{send command,max}}$ was chosen in order to be able to capture the ball with the other rows of players. This because Equation (4.3) is only valid when the shot comes from the rows of

players right in front of the defender. It is important though that $D_{\text{send command,max}}$ does not exceed 100mm because for this values the next row of players would still be in possession of the ball while the command will be sent.

Finally, the distance at which the command is sent will be the following:

$$\begin{cases} D_{\text{send command}} = d_{ball-player} + v_x \cdot delay & D_{\text{send command}} \leq 100mm \\ D_{\text{send command}} = 100 & D_{\text{send command}} > 100mm \end{cases} \tag{4.4}$$

With $delay = 15ms$, $d_{ball-player} = 25$.

- **Increasing the speed of the rotation**
  A factor x1.2 has been added to the actual speed of the ball in order to reach the desired speed of the ball while slowing down the angular speed of the player.

To summarize, this method performs very well for:

- A low speed shot if it comes from a close range (<2m/s). Typically, the next row of player.
- At medium speed range (<3m/s) for a further shot.

The main advantage being that if the ball is intercepted, the player has a static possession and he can therefore implement an attack or a pass strategy.

4.4.2 Control reference position

As for this strategy, a ramp speed profile will be implemented to damp the ball, the speed profile had to be converted to position profile. The following equations have been used to compute such a profile.

$$\begin{cases} \phi_{k+1} = \phi_{final} & \omega_k \leq 0 \\ \omega_{k+1} = 0 & \omega_k \leq 0 \\ \phi_{k+1} = \phi_k + \omega_k \cdot dt + \frac{1}{2}\alpha \cdot dt^2 & 0 \leq \omega_k \\ \omega_{k+1} = \omega_k + a \cdot dt & 0 \leq \omega_k \end{cases} \tag{4.5}$$

With ω_{k+1} the angular speed of the player at the new loop iteration, dt the time of one loop iteration and ϕ_{final} the final angle position (set to 40°).
The angular acceleration α is computed by setting the constrains and solving the kinematic equation. The initial angular speed is based on the speed of the ball and the initial position of the player is vertical. The units for theses variables are respectively $[°], [°/s], [°/s^2]$.

$$\begin{cases} \phi_0 = 0 \\ \omega_0 = \dfrac{\omega_{ball}}{d_{rotation-ball}} \cdot \dfrac{360}{2 \cdot \pi} \\ \alpha = \dfrac{\omega_0}{2 \cdot (\phi_{final} - \phi_0)} \end{cases} \tag{4.6}$$

Where $d_{rotation-ball}$ is the distance between the axis of rotation of the player and the point where the player touches the ball. This distance has been measured as being $d_{rotation-ball} = 65mm$.

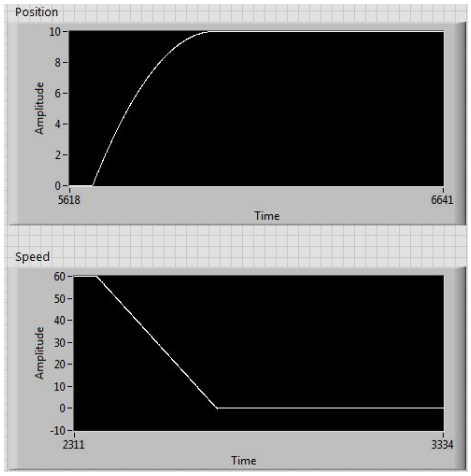


Figure 4.6: Speed and position profile for rotation control.

The speed and angle position generated by Equation (4.5) can be visualized in Figure 4.6.

The PID parameters have been tested and tuned to avoid any oscillation. The parameters have been selected as: $k_p = 0.07$, $k_i = 0.03$ and $k_d = 1e-6$.

The response of the system for such an input can be observed on Figure 4.7. The delay to reach the reference speed (derivative of position) is about 20ms.

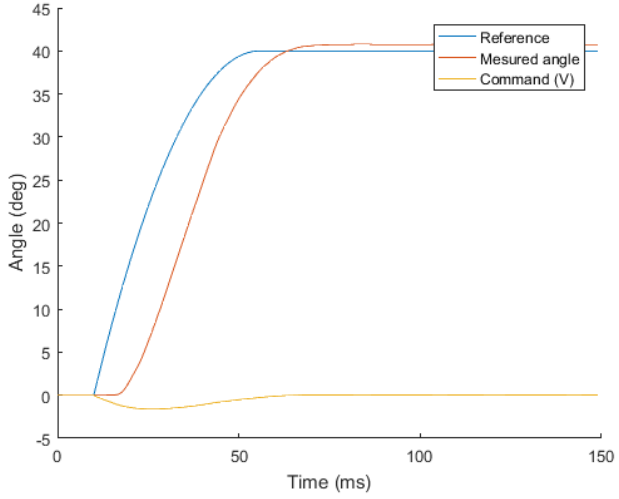


Figure 4.7: System response to parabolic position input

4.4.3 Control reference speed

The command written on the motors is in voltage level. It would thus make more sense to directly control them by speed reference as there is a linear relation between the speed and the current. First the relation between speed and current has been obtained by measuring the angular speed for different (constant) voltage references. The linear relation obtained is (see Figure 6.2 in annex)

$$w[deg/s] \approx 886 * Voltage \tag{4.7}$$

To simulate the damping of a ball with $v = 2m/s$ the speed reference should start at $w_0 = \frac{2000}{65} = 30.8rad/s = 1750deg/s$ (recalling the distance from the center of rotation to the ball $d_{axis-ball} = 65$).

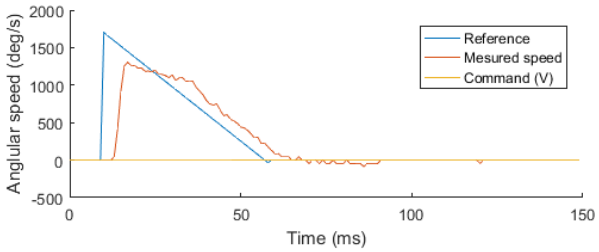


Figure 4.8: Response to an inverted ramp input with open loop speed control.

Figure 4.8 shows the result of such an input. The result is not satisfactory as the player does not reach the desired speed and a delay of about 8ms is still observed.

As one could observe difficulties for the system to reach the reference speed, an attempt was made to include this delay in the input (in other words let some time to the system to respond).

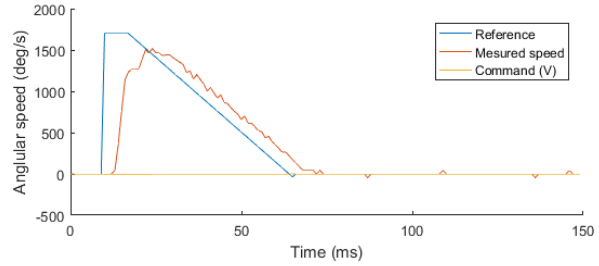


Figure 4.9: Response to an inverted ramp input with open loop speed control. Delay of 8ms included in the input.

This method improved the results only slightly, the time for the system to respond is still way too high.

Control decision

Controlling the rotation motors in position is advantageous for two reasons:

- Less complex: Controlling in velocity is possible but an outer control loop in position will have to be implemented to ensure constrains.
- No delay improvement: Controlling in velocity does not improve the delay of the system, moreover the performances seem to be less interesting as (due to the delay) the player will never reach the speed of the ball (see Figure 4.8).

It was thus decided to keep the control in position

Notes for tests

When observing the delay of the system response it is important to apply an input not directly from the first iteration. In this case a larger delay has been observed. In this project, when testing the systems response, the input was applied at the 10th loop iteration.

4.5 Attack

Due to lack of time, a very basic attack strategy was finally implemented: If the speed of the ball is below 20mm/s and it is within the players reaching zone, the attack strategy is activated. The player hovers left right with an amplitude of 20mm, then after 2+*rand* seconds, the player shoots. *Rand* being a randomly generated number between 0 and 1 seconds.

4.6 Issues

It has been observed that after performing several shots and ball captures, the front players get an angle shift. This issue may be due to the encoder.

5 | System Identification

After noticing in Section 4.3.3 that the linear motors had relatively slow dynamics in regards to the speed of the game, it was decided to identify the system.

5.1 System description

The linear motors already have their own built-in controllers. Therefore, a black box open loop identification will be conducted to better understand the relation between the reference input and the actual output of the system. On the following picture, the identified system is shown as G_{tot}. The identification will be done on the front row of players.

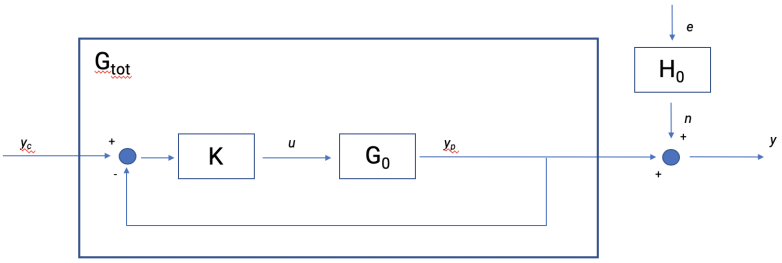


Figure 5.1: Black box identification

5.2 Dataset

The system was excited with a periodic ($p = 700$) white noise signal. The amplitude of the excitation was chosen at 6mm to avoid exciting the non-linearities of the system while applying a *close to game situation* excitation (see Figure 6.5 in Annex).

The sampling time was chosen by observing the raising time T_r of Figure 4.4 and by applying the relation [1]:

$$\frac{T_r}{10} < T_s < \frac{T_r}{5} \tag{5.1}$$

A sampling time of 10ms was thus chosen. For the parametric identification a train test ratio of 80% was chosen.

5.2.1 Data set verification

To verify the accuracy of the obtained data, the FIR (First Impulse Response) was compared to the real impulse response of the system.

From the FIR (Figure 5.2), a delay of about 10ms can be observed. The settling time seems the be around 0.25 seconds. By observing the real impulse response of the system (Figure 5.3), a settling time of 0,25s seems to be slightly underestimated but plausible, the real delay of the system is between 3 and 5 ms. Thereby, the data set seems to be a fairly good representation of the real system and it will be used for parametric identification.

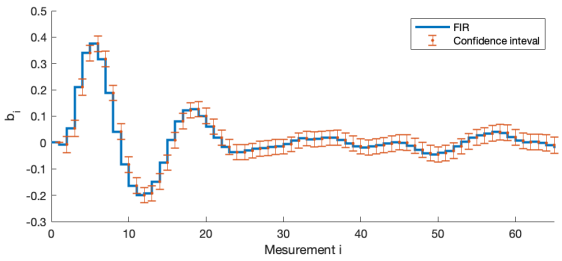


Figure 5.2: FIR with order 65.

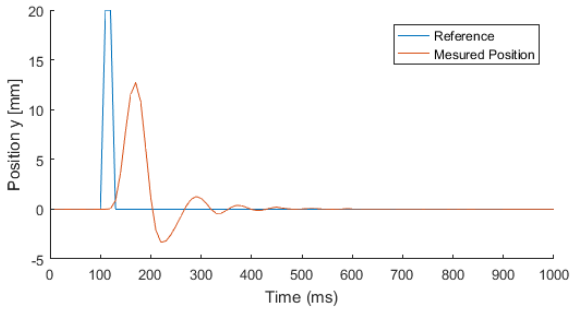


Figure 5.3: Impulse response of the system.

5.3 Parametric identification

Now that the recovered data seemed to well represent the system, parametric identification was conducted to fit a model and obtain it's transfer function. First, the loss function of the ARX (AutoRegressive with eXternal intput) was computed for different orders.

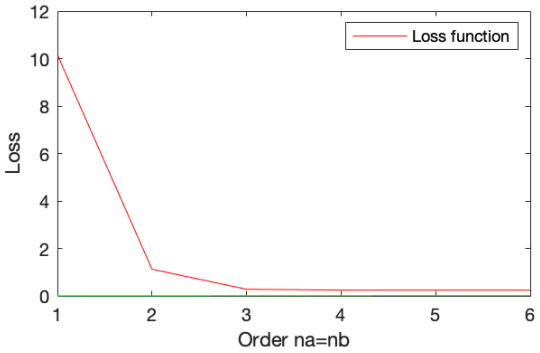


Figure 5.4: Loss function for $n_a = n_b$

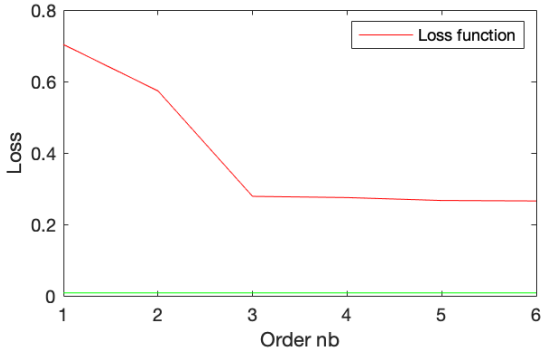


Figure 5.5: Loss function for n_b with $n_a = 3$

By observing Figure 5.4, the order of the system can be identified as being $n_a = n_b = 2$ or 3, both orders were compared on the testing sample and finally order 3 was chosen as it presented significantly better results: order 3 ARX model fits the testing data at 94.03% (prediction focus). Recalling the ARX model being:

$$y(k) = \frac{q^{-d}B_0(q^{-1})}{A_0(q^{-1})}u(k) + \frac{1}{A_0(q^{-1})}e(k) \tag{5.2}$$

One obtains the discrete time transfer function of the ARX model:

$$G_{tot} = \frac{0.00308z^{-1} + 0.03925z^{-2} + 0.05772z^{-3}}{1 - 2.258z^{-1} + 1.829z^{-2} - 0.5062z^{-3}} \tag{5.3}$$

Fitting this model to the testing data Figure 5.7, one obtains a decent but not excellent fit:

Other more complex models have also been assessed (See Figure 6.6 in Annex) but theses models did not show any drastic improvement.

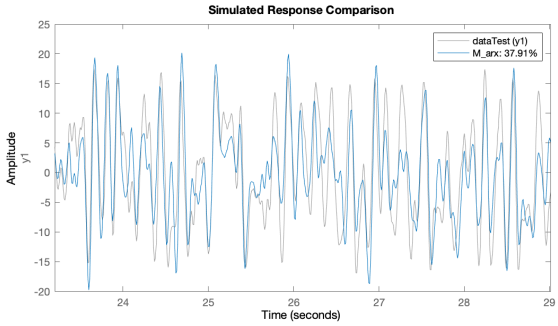


Figure 5.6: State-space comparison ARX

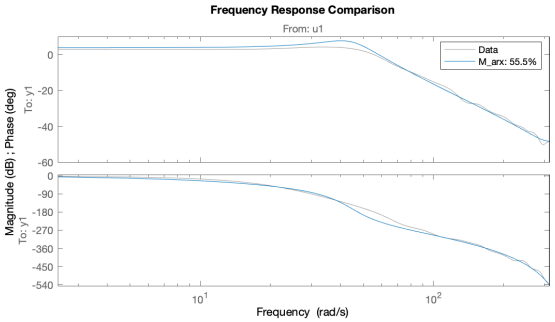


Figure 5.7: Frequency domain comparison ARX

From the analysis of this dataset we can deduce that the system seems to be highly non linear as an optimal fit of only 38.3% is achieved. This may be due to the vibrations as when applying the white noise input signal, the whole table was moving consequently. Theses vibrations make the system non-linear which explains the relatively poor performances of the assessed models.

5.4 Conclusion

As the camera issues have not been fixed yet, most of the dynamic test could not be conducted in real game conditions with all the rods activated. Nevertheless, I really believe that the speed computation, the position prediction and the ball interception will play a major role in the upgrade of the Foosball towards the ultimate goal: *build an unbeatable robot.*

5.5 Further Projects

There are still many aspects to develop on the Foosball in order to reach the ultimate goal. Here are some thoughts about what future projects could be focused on.

- **Strategy 1**
  It will be necessary the rethink the state machine of the strategy by implementing the damping of the ball only for low speed shots, improving the linear defensive placement of the players to avoid letting the ball pass and include the speed in the state decisions. Moreover, creating an adaptive PID controller for the rotation motion could be very useful as the ball-damping strategy requires some coefficients while a shot may be more efficient with other coefficients.

- **Strategy 2**
  Now that the player has static possession of the ball, more advanced attacking and passing strategies can be implemented. Once the camera will be fixed the defense strategy can also be upgraded by using the prediction and optimizing the position of all the rows players among one another simultaneously.

- **Vision**
  The crucial weak point of this Foosball still is it's edges as they are not see-through. The camera loses track of the ball for split seconds and we are thereby considerably vulnerable. An ingenious way of avoiding this blind point without losing vision precision (by distortion) could make the system more robust to very good players.

- **Electrical/mechanical**
  During this semester Marxon motors contacted us to design some tailored motors for the Foosball. This could be very useful as it has been observed that the dynamics of the motors are too slow regarding the speed of the ball when playing against very good players (shooting the ball fast and with an angle). Maybe the controller of the linear motors could as well be better tuned to have a faster response.

  The mechanical aspect of the Foosball could also be improved as it was observed that we have significant noise due to vibrations when the linear motors are moving rapidly.

- **Data analysis**
  Once all the components of the Foosball will have been accurately identified and modeled, a virtual model of the game could be built [3]. Then, a Neural network could be trained [5] in order to place the players to avoid conceding a goal. Obviously, identifying all the components will be a consequent challenge.

6 | Annex

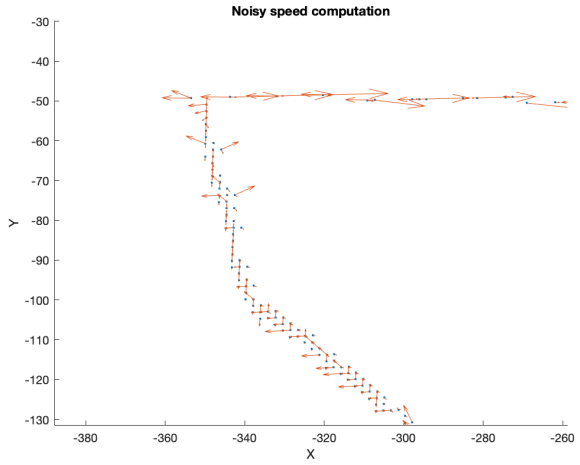


Figure 6.1: Simple finite difference speed computations: very inaccurate noisy measurements.

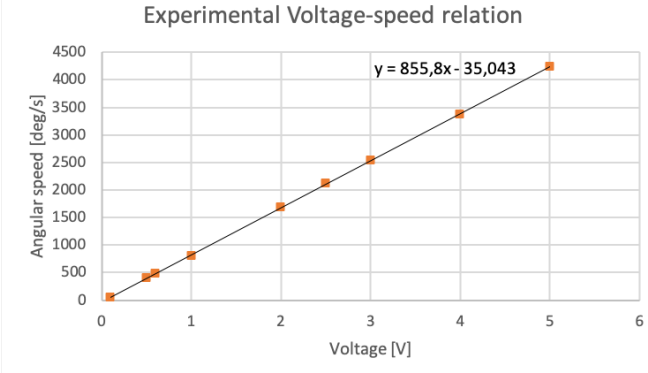


Figure 6.2: Voltage-speed relation: measurements and regression

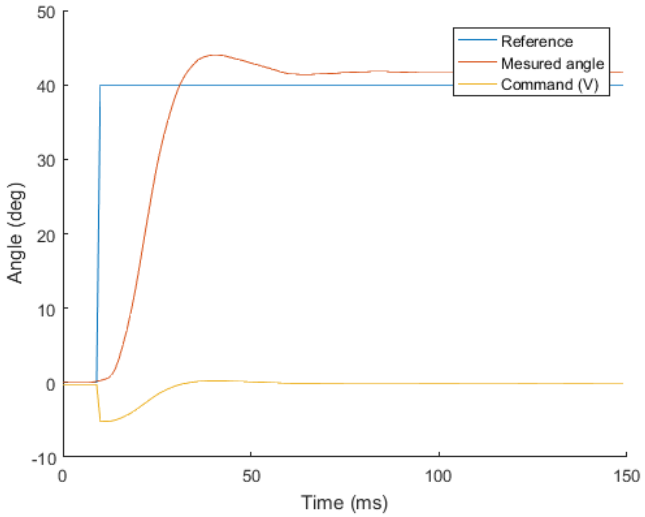


Figure 6.3: Rotation motors: Response to a step input of 40 degrees.

Steady state around 50ms. Reference passed after less than 30ms. (PID parameters kp=0.1, kd=0.01, Kd=0). Note that theses have been changed to avoid over shoot.

	Cinématic						Dynamic	
	Plage de déplacement		Acceleration		Précision			
Axis	Rotation [rad]	Translation [mm]	Rotation [rad/s^2]	Translation [m/s^2]	Rotation [rad]	Translation [mm]	Mass of the bar [kg]	Inertia [kgm^2]
Front	2π	180	107'000	50		2	1,5	6,20E-04
Mid	2π	133	107'000	50		2	1,5	7,80E-04
Def	2π	193	107'000	50		2	1,5	7,00E-04
Goal	2π	203	107'000	50		2	1,5	5,40E-04
Actual system	2π	650		242	See tab2	See Tab3		

Notes	Full rotation	Need to add 60mm because of the springs. (Not taken into account it the above numbers)	Get to a fast speed (15m/s) in one loop iteration (2ms)	Reach 10 cm at each loop iteration (2ms)	Maximum in order to apply precise control.	Precision of the camera	Hard to compute. Apporximate values. Range of validity between [1,2 and 1,8]	Very hard to calculate precisely. Range of validity between [2 and 10]E-04

Figure 6.4: Specifications given to Marxon motors

6.0.1 Identification

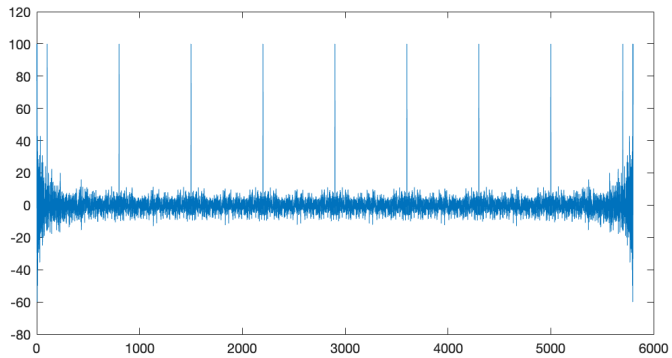


Figure 6.5: autocorrelation of the input signal: xcorr(u,u,'unbiased')

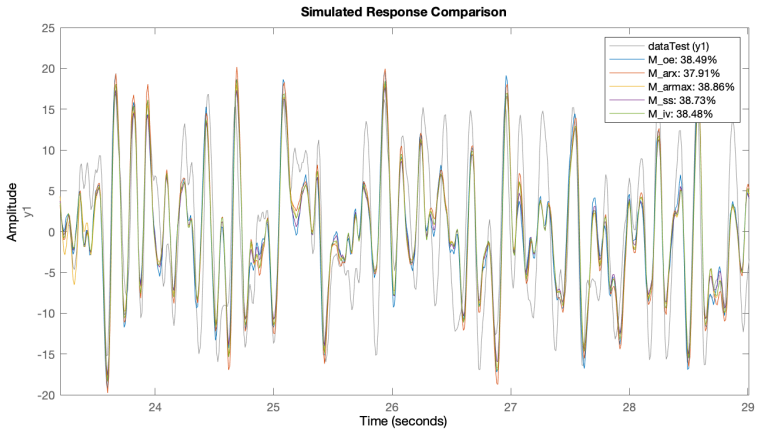


Figure 6.6: Comparison of different models on testing data set

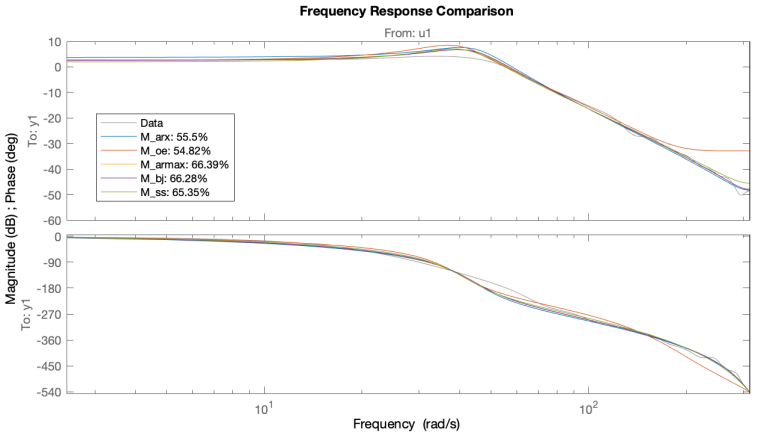


Figure 6.7: Comparison of different models on testing data set

Bibliography

[1] Dr. Alireza Karimi. *System Identification*. 2019.

[2] Yoann Moret. *Babyfoot Vision et Stratégie*. 2017.

[3] *Source code for inverted pendulum, cartpole.py*. 2019. URL: https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py.

[4] *Table football*. 2019. URL: https://en.m.wikipedia.org/wiki/Table_football.

[5] *Using a neural network to solve OpenAI's CartPole balancing environment*. 2019. URL: https://pythonprogramming.net/openai-cartpole-neural-network-example-machine-learning-tutorial/.