Autonomous Air Hockey Opponent

System Requirements

SFWR ENG 4G06 / MECHTRON 4TB6

GROUP 3

Nima Akhbari

Daniel Chaput

Nicholas Cianflone

Michael Rowinski

Joshua Segeren

Evan Skeete

Contents

Table of Revisions

Version	Date	Authors	Description of Revision
0	2/11/2014	Nima Akhbari Daniel Chaput Nicholas Cianflone Michael Rowinski Joshua Segeren Evan Skeete	Initial revision of the *System Requirements* document
1	3/3/2015	Evan Skeete	R1 update; notably, safety & HMI changes
1.1	4/3/2015	Joshua Segeren Michael Rowinski	Minor revision; removed/modified non-critical features from scope Added student numbers to title page

Table 1 – Table of Revisions

1 Purpose

The project will involve the design and development of an autonomous robot capable of playing air hockey against a human opponent. This will entail delivering an effective solution to object detection, tracking, and response problems. The team will investigate, determine and implement subsystems based on best practices across machine vision, motion planning, predictive analytics, control systems, mechanical design, and hardware-software interfacing.

There is also significant opportunity to explore principles and design strategies for human-machine interaction, and consider user-centered features for improving the entertainment value of the game. More broadly, the project aims to make a positive contribution toward the broader goal of integrating robotics into society safely and beneficially.

2 Scope Discussion

As a demonstration of generalized user-human interaction, physical event detection and handling, the project will specifically be based around the arcade game of air hockey. The system will be able to play a standard game against a human at a skill level comparable to that of an "average" adult player (validated against each member of the project design team). Game rules will be based on those for USAA (United States Air-Table-Hockey Association) competitive (tournament) play [1].

The system will be designed as a retrofit/extension of a standard air hockey table. While the design of the table itself—including relevant peripherals and their operations (such as the timer and scoreboard)—is not specifically in scope, such considerations may need to be included over time to achieve the primary objectives.

In-scope items of functionality for the system include the following:

- Awareness of game start event (based on user input/action)
- Detection and awareness of goal scored events for each side
- Defensive play (blocking) against high-speed pucks
- Offensive play (puck handling and shooting) where possible
- Edge case detection and handling (discussed in Undesired Scenario)
- Pausing or stopping the game based on appropriate user input/action
- HMI, including information display and strategy mode selection

Additionally, the following items are deemed to be out of scope:

- Resetting puck after goal scored
- Resetting puck shot or deflected off table
- Resetting puck stuck/jammed/stopped on user side of table

3 Normal Operation (User Side)

3.1 Description

In general for normal gameplay, the system must be able to detect/predict the location and trajectory of a puck, strategize appropriately (either defensively or offensively), then accurately and precisely deliver the puck to the desired location (typically with the intention of scoring) by striking it with the mallet accordingly.

The collective and sequential behaviors of the subsystems must ultimately model and perform this behavior.

3.2 Normal Use Cases/Scenarios

3.2.1 Game is started/requested

The game is to be manually started by the player (i.e. system is enabled/turned on), but it will subsequently wait for the puck to drop before initiating normal game playing behavior.

3.2.2 Goal is scored (either side)

If a goal is scored, the system is not designed to reset the puck position (i.e. execute the "face-off" puck drop); the user must therefore retrieve the puck and perform this step manually. The system is then designed to handle the game resume scenario appropriately (without additional intervention or input).

3.2.3 Game is won/lost/finished

The system will detect and acknowledge completion of the game, provide appropriate ("sportsmanlike") gesturing to the player, assume default position, and terminate normal playing operation.

3.2.4 Difficulty detection and setting

The system will be able to alter its difficulty level based on the score difference between itself and the user (potentially alongside other factors indicating skill level, such as detected puck velocities). The system will attempt to play at a degree of skill similar to that of the user (determined by speed, for example). If the system is heavily trailing the user in score it will adopt a more difficult/aggressive level of play; conversely, if it has a strong lead with respect to the user, it will reduce its level of effort with less aggressive play. This is intended to ensure the game remains competitive and entertaining for users. A secondary requirement/feature involves the ability to adjust the difficulty level based on explicit user configuration (e.g. preconfigured settings for "easy", "normal", and "hard" or "expert"). These settings will affect the dynamic change in difficulty.

4 System Diagram

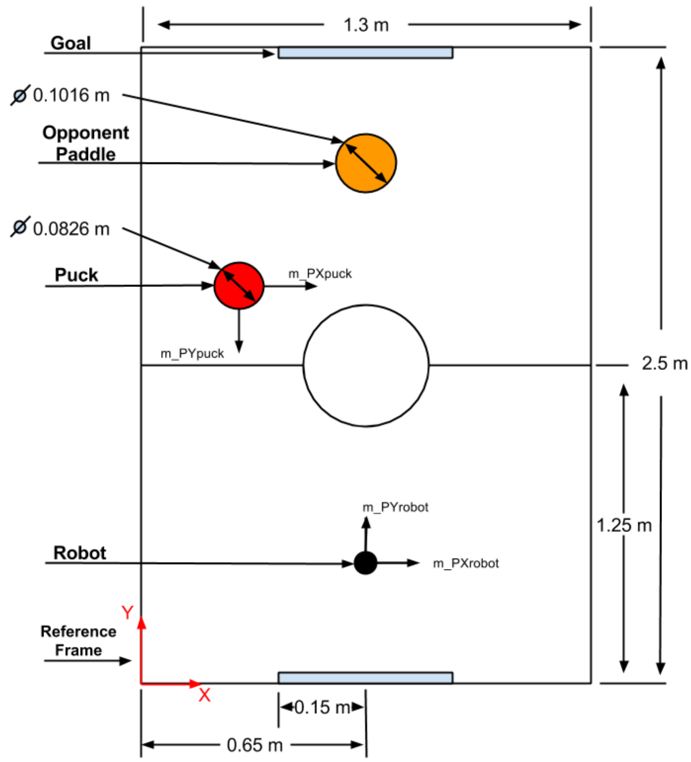


Figure 2 – System diagram with standard table dimensions

5 Context Diagram

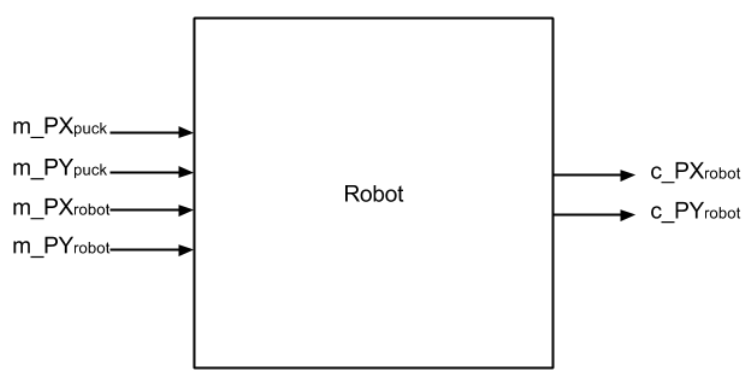


Figure 3 – System context diagram

6 System Level Variables

System level variables are labeled on Figure 2. The variable names, descriptions, and units (or values where applicable) can be found in the tables below.

6.1 Constants

Constant	Unit	Value
Table Length	m	2.5
Table Width	m	1.3
Goal Width	m	0.3
Standard Puck Diameter	mm	82.6
Standard Mallet Diameter	mm	101.6

Table 2 – Constants

6.2 Monitored Variables

Variable	Unit	Description
m_PXpuck	m	Position of the puck in the x-axis
m_PYpuck	m	Position of the puck in the y-axis
m_PXrobot	m	Position of the robot mallet in the x-axis
m_PYrobot	m	Position of the robot mallet in the y-axis

Table 3 – Monitored Variables

6.3 Controlled Variables

Variable	Unit	Description
c_PXrobot	m	Position of the robot mallet in the x-axis
c_PYrobot	m	Position of the robot mallet in the y-axis

Table 4 – Controlled Variables

7 Requirements

The collective behaviors of the subsystems must ultimately model and perform the desired system-level behavior; however, the actual specifications (requirements and constraints, including those for timing) for each individual subsystem can exist in any number of combinations. Thus, only system-level requirements have been provided, as any additional detail requires further design analysis.

7.1 Functional Requirements

For each specification, a metric (description) is provided, along with a relevance rating from one to five (higher values corresponding to higher importance), marginal value (indicating the minimum degree of acceptability), and ideal value (the design target). For qualitative specifications, a subjective rating following a scale from one to five (higher values being better) is the adopted convention. Finally, the volatility rating indicates the likeliness of a particular specification (either its precise metric description, or provided marginal/ideal values) to change in subsequent design phases or iterations. Again, a proportional convention is used (i.e. higher values indicating greater volatility).

7.1.1 Required Variables and Tolerances

7.1.1.1 System-controlled mallet response time

Description: Response time is the amount of time the system takes to respond to the user striking the puck. Ideal and marginal response times are listed below.

Rationale: This is based on an estimated shot speed of an average adult player; this is the time within which the system must be able to react—for both recognition and movement—to stop a goal. Currently this is an estimate (details provided in Appendix A), and the value is likely to be refined via experimentation.

Importance: 4

Unit: milliseconds (ms)

Marginal Val: 190

Ideal Val: 35

Volatility: 4

7.1.1.2 Maximum puck shooting speed

Description: The maximum speed the system must be able to launch the puck. Ideal and marginal speeds are listed below.

Rationale: This is based on an aggregate (estimated) measure of the response time and accuracy of an average adult player; this is the speed at which the puck must be shot/launched in order for a reasonable chance of overcoming the reaction of the player and scoring. Currently this is an estimate (details provided in Appendix A), and the value is likely to be refined via experimentation.

Importance: 5

Unit: meters/second (m/s)

Marginal Val: 4

Ideal Val: 35.8

Volatility: 5

7.1.1.3 Combined Mallet Position and Puck Position Detection Accuracy

Description: The accuracy for detecting the position of the puck summed with the accuracy of manipulating the mallet.

Rationale: This is essential for making accurate shots. In order to make a shot from any position on the board, the system must be able to determine the point of collision between the puck and the mallet to, in turn, be able to control the angle of the collision. Calculation of the tolerances are provided in Appendix B.

Importance: 5

Unit: millimeters (mm)

Marginal Val: 2.0

Ideal Val: 0.5

Volatility: 4

7.1.1.4 Range of Motion

Description: Percentage of system side of table within reachable workspace

Rationale: This is important for being able to make a move regardless of puck position, preventing delays of the game, and enabling a sufficient number of strategic alternatives.

Importance: 5

Unit: % surface area

Marginal Val: 95

Ideal Val: 100

Volatility: 4

7.1.1.5 Total System Cost

Description: The cost of all parts and technologies necessary to build the system/robot. This does not include the cost of tables or additional infrastructure as the marketable product only consists of the robot and control systems.

Rationale: Given analysis of comparable systems built for academic research ($2000+), the only budget constraint is that imposed by the MECHTRON 4TB6/SFWR ENG 4G06 course. The system must be built within the maximum allowable budget, although striving to minimize the cost (while still achieving performance and functional requirements) will allow greater room for error and potentially make aspects of the system more marketable for future endeavors.

Importance: 5

Unit: $ (CAD)

Marginal Val: ≤ 750

Ideal Val: ≤ 750

Volatility: 4

7.1.2 Required Functions

7.1.2.1 Path prediction

The system will be able to leverage current and previous state information to determine the path and speed with which the puck will be moving. The system will be able to determine the future position of the puck up to a specified number of wall collisions.

7.1.2.2 Strategy

Strategy defines the robot mallet response given the puck's position and velocity. The overall strategy is to be composed of a collection of sub-strategies which are selected based on specified intervals/ranges for puck position and velocity. Sub-strategies are to be ranked on a continuum from naive to advanced. Naive strategies have a quick response time and are less computationally expensive, to be used when the puck is close to the robot goal and/or is moving quickly. Advanced strategies are preferable in terms of placing the robot at an advantage but are more computationally expensive and therefore used when the puck is at a further distance, or moving slowly enough. Strategies are further broken down into offensive and defensive strategies.

> *Defensive*: The robot will take a defensive action if an offensive action is not feasible (i.e. an offensive action can't be produced in the time/space available to the robot). A defensive action is considered successful if a goal is not scored against the robot. In general, this action will include blocking, or in the worst-case, altering the trajectory of the puck in any way such that a goal is not scored against the robot. Ideally, a defensive action will ultimately result in the robot gaining "possession" and control of the puck, such that a subsequent offensive action can be made.
>
> *Offensive*: The robot will respond offensively whenever possible; that is, if the action is determined to be feasible. An action is considered offensive if it involves an attempt to score a goal against the opponent; this could include (but is not limited to) 1) a stationary deflection which redirects the puck toward the opponent's goal, 2) striking the puck directly toward the opponent's goal, and 3) striking the puck such that the sequence of wall bounces results in an ultimate puck trajectory toward the opponent's goal. An offensive action is considered successful if it results in a goal scored against the opponent. An offensive action should never result in the puck scoring on the robot itself.

The overall strategy is deemed acceptable if the robot is able to win over 50% of games played against it. That is, given a statistically significant sample size, the robot wins more games than it loses.

7.1.2.3 Goal detection

The initial value for both the player's and robot's goal score is 0. If the puck is located within the bounds of the player's goal, the system's goal score must be incremented by one. If the puck is located within the bounds of the system's goal, the player's goal score must be incremented by one. Any other location of the puck will not result in a goal being registered. The current goal information will also be made accessible to the user.

7.1.2.4 Game start/finish

The game will be considered finished if either the parameterizable score limit or time limit is reached. The system will expose a mechanism to allow the user to manually end the game (clearing the current scores and time), and manually start the game (resetting the scores and time, as well as position of the system-controlled mallet).

7.1.2.5 Emergency stop

The system will expose a mechanism to immediately terminate all mechanical and electrical operations in case of an emergency. This will be easily accessible to the human player throughout gameplay.

7.1.2.6 Limit stop

The system will expose a mechanism to cut power to the motors in the case of a software or electrical malfunction to prevent the robot from moving out of its defined working boundary.

7.1.3 Non-Functional Requirements

7.1.3.1 Performance

The system should be able to deliver 90% of offensive "shots"/actions with respect to desired target and within relevant tolerances.

The system should be able to defend against 90% of shots made by the opponent (i.e. prevent a goal from being scored).

7.1.3.2 Security

The system must be reasonably resistant against unauthorized "tampering"; that is, the system should not be vulnerable to external and unintended manipulation—either hardware or software-based—to its normal behavior and/or performance.

8 Functional Diagram

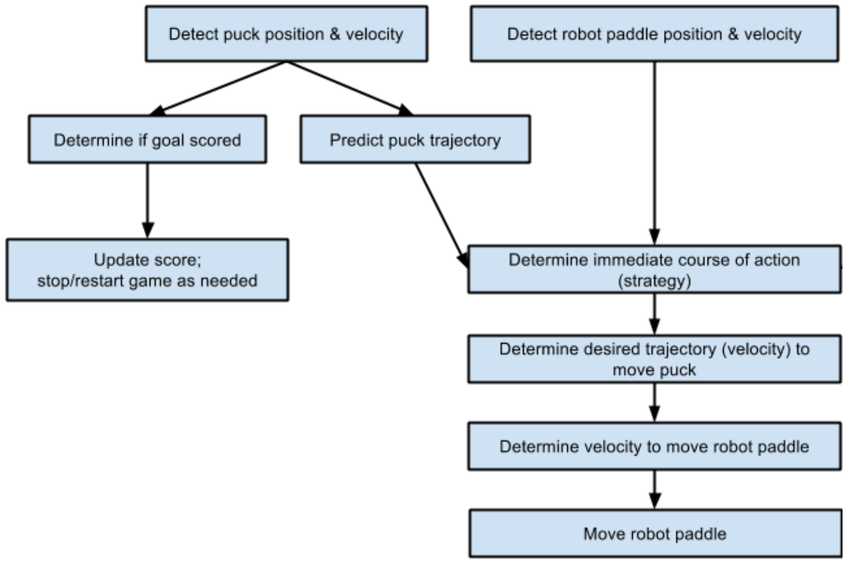


Figure 4 – Functional diagram

9 Undesired Scenario Handling

9.1.1 Puck removed from, or otherwise not detected on table

- Puck knocked off table during play
- Puck intentionally moved off table (i.e. user intervention)
- Vision subsystem failure

If the puck is removed from or otherwise not one the table, the system will recognize this condition. The robot will return to a designated default/home position and stop all movement.

9.1.2 Puck not moving (robot side of table)

- Puck stuck/jammed/unreachable
- Robot stuck/jammed
- Strategy subsystem failure

If the puck is jammed or otherwise not moving on the side of table controlled by the robot, the system (specifically the vision subsystem) will recognize this. The robot will return to the designated default/home position and stop all movement, and the system will shut down in anticipation of required human intervention; safety measures for fully locking out the machine will also be provided. A signal will notify the user that a fault has occurred, with all available and relevant reason/logging information provided. The system will wait for operator approval/confirmation before attempting to resume normal function.

9.1.3 Puck not moving (user side of table)

- Puck stuck/jammed/unreachable
- Puck intentionally stopped (i.e. user intervention/cheating)

If the puck is jammed or not moving on the user's side of the table, the vision system will recognize this. The robot will return to the designated default/home position and stop all movement. If the puck does not move after a specified amount of time, the system will shut down. If the puck starts moving again prior to this timeout limit, the system will resume normal function.

9.1.4 Puck tipped/flipped/moving vertical

- Strong shot from player
- Strong shot from robot
- Deflection off of table boundaries or paddles

If the puck is determined to not be lying flat for a specified duration, the system will automatically treat this as a pause condition and follow the previously defined pause behavior in order to ensure safety of the human player and spectators.

9.1.5 System failure (general)

- Collision with table boundaries
- Electrical malfunction
- Invalid command from upstream (control subsystem failure)
- Failure to detect game conditions

If the system itself fails for any reason--mechanical, electrical, or software--the internal safety checks will acknowledge this state and a fail-stop protocol will be followed (to be specified in later design phases). A signal will notify the user that a fault has occurred, with all available and relevant reason/logging information provided. The system will wait for operator approval/confirmation before attempting to resume normal function.

References

[1] United States Air-Table-Hockey Association (USAA), "Air Hockey Rules," 2014. [Online]. Available: http://www.airhockeyworld.com/usaarules.asp. [Accessed 20 October 2014].

[2] "Retroland," [Online]. Available: http://www.retroland.com/air-hockey/ . [Accessed 20 11 2014].

[3] "Mental Chronometry," [Online]. Available: http://en.wikipedia.org/wiki/Mental_chronometry. [Accessed 20 October 2014].

Appendix A – Paddle Response Time

Paddle response time is based on the speed of an average adult air hockey opponent. The following information is an estimate based on maximum speed ratings for modern air hockey tables. The ideal value will be represented by the maximum achievable puck speed. Marginal times will be reflected by the average adult human response time. This information will be refined further with experimentation during system testing in later development.

- Maximum speed for modern air hockey table design is approximately 80 mph (35.8 m/s) [2].
- Length of a modern air hockey table is approximately 2.5 m.

We are calculating the response time necessary for a shot taken from the halfway mark of the table ((2.5 m)/2 = 1.25 m). This assumption corresponds with the fact that there will be (worst case) approximately half a table length between the player's paddle and the robot's paddle when shots are taken. The response time is calculated as follows:

$$\begin{aligned} v &= \frac{d}{t} \\ t &= \frac{d}{v} \\ t &= \frac{1.25}{35.8} \\ &= 0.035\text{ s} \\ &= 35\text{ms} \end{aligned}$$

Where *t* is the response time, *v* is the maximum puck speed, and *d* is the distance travelled between players. We see the ideal response time necessary is 35 milliseconds. The marginal response time is the average adult response time (defined as reaction time plus movement time), which is approximately 190 milliseconds [3].

Appendix B – Accuracy Tolerances

In order to determine the mallet accuracy, it is first necessary to determine the tolerances within the shot angle. Aiming directly for the center of the goal, there is a margin of error on the angle of the trajectory where the puck will still enter the goal (brushing near the edges of the goal). We first calculate the angle directly towards the center of the goal, then the angles to each edge. For a bouncing shot (a shot which first collides with either of the side walls any number of times before entering the goal), we can "unwrap" the zigzag pattern to get a straight, vector trajectory with the same relevant properties. For a "1 bounce" shot which collides with one wall, we "turn around" the path after the first bounce, such that the slope of the trajectory doesn't become negative. This maintains the absolute value of the slope as well as the length of the line before goal entry.

We calculate the vector of this shot as follows

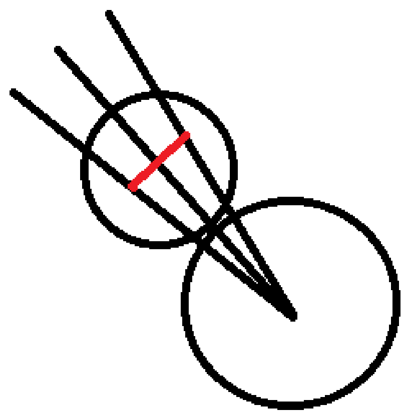
$$x = n \cdot tableWidth + x_{puck}$$
$$y = tableLength - y_{puck}$$

Where *x*, *y* is vector the puck must travel, n is the number of bounces we wish to travel and x_{puck} and y_{puck} are the coordinates of the puck (the origin being the center of the robot's goal).

We then find the vectors towards the left and right edges of the goal, by adding or subtracting the difference between the radius of the goal and the radius of the puck to the x dimension of the trajectory (such that the puck brushes the edge of the goal).

Taking the arctangent of these two numbers (atan2(x, y)) gives us the angles from the horizontal to each of these points and subtracting the center angle from each of the side angles gives us the angle deviation.

We can use these angle deviations to find our linear error, represented by the red line in the following diagram, where the circles represent the puck (smaller) and the mallet (larger) and the black lines represent the center vector and the two outlying tolerances. It is simply a matter of trigonometry to achieve the length of this line.



Repeating this calculation for a large variety of locations and up to 4 bounces, we find we can make the majority of our desired shots with a combined tolerance of up to 1 mm.