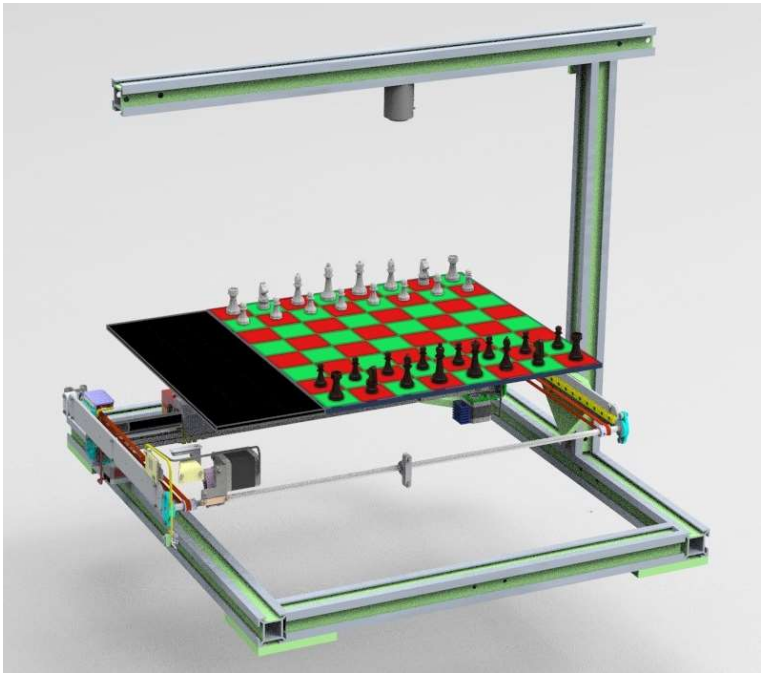
RPL - Computer Engineering. UAB (Barcelona) 2019-2020



BetaGone

An autonomous vision-based chess robot

PROJECT SPRINT #5.
DATE: May 26th 2020

Marc Espinosa Gil
Esteve Jimenez Vidal
Miquel Tena Morales
Pau Torras Coloma

Table of Contents

Sprint #5
Date: 26 May 2020

BetaGone

An autonomous vision-based chess robot

Project description

The idea behind BetaGone is to craft a robot capable of playing chess on a real board against a human. It will identify the position through a camera placed right above the board, which will take pictures periodically in order to assert whether a move has been made. Then, a custom-made Monte Carlo Tree Search based chess engine will explore the playing possibilities and decide on one, leaving a Movement System to calculate the path to perform the chosen move.

The uniqueness in our approach lies in the fact that, aside from the camera, the playing area will remain clean of any contraptions to the eye of the player: instead of having a robot arm perform the moves, BetaGone will have an electromagnet move the pieces from below the board. This magnet will move in the bidimensional plane through two perpendicular axes moved by two stepper engines.

The human interaction with the robot will be performed through the vision system, similar to the interaction system seen in professional broadcasting chess boards: placing pieces in certain squares will trigger the program to start running and the it will remain on standby should the game end.

All of this will be running on a Raspberry Pi model 4B and will be written primarily in Python, although if testing evidences such need it, we shall implement the chess engine in C++.

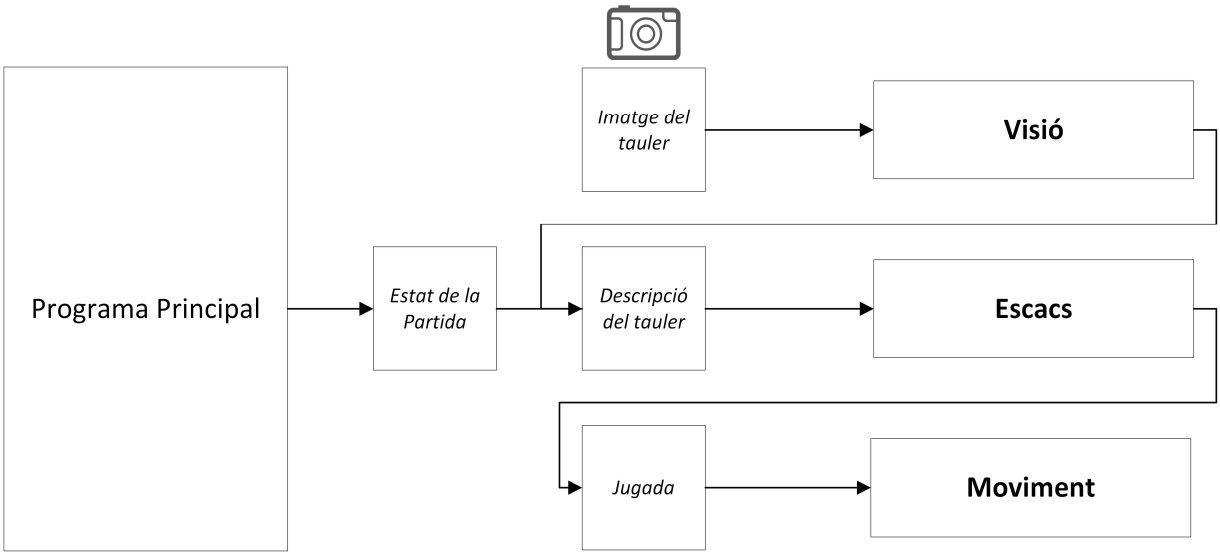
Electronic components

This is the list of the proposed components:

- *2x Motor Paso a Paso 2.4 Kg/cm, Nema 17*
- *Correa dentada GT2 - 6mm (5 metros)*
- *Raspberry Pi 4 Modelo B*
- *Retenedor, Electroimán, Acabado en Zinc Brillante, 25mm*

- *RPI 8MP CAMERA BOARD*
- *2x DRIVER MOTORES PASO A PASO A4988 POLOLU*

Software Architecture



The modules will do the following:

- **Vision Module:** It will generate a board description in FEN (Forsyth-Edwards Notation) format from an image of the board. FEN is a text-based chess board description that allows to completely reconstruct the state of a game without knowledge of any prior movements. In order to account for the lack of information about the previous state of the board, the output FEN from the vision module will be compared and completed with a game state full representation, which will allow to detect potential vision mistakes or illegal moves by the player.
- **Chess Module:** There will be a persistent Chess object that will calculate the best possible move from an input position. It will do so through a **Monte-Carlo Tree Search** (MCTS) algorithm, which will carry out several playouts until the end of the game. The move that wins the most games for the playing side will be chosen and outputted as a 16 bit integer. The lower 6 bits represent the origin square, the following 6 bits represent the destination square and the last 4 bits are dedicated to castling flags and promotion flags.

- **Movement Module:** This will perform the move on the real board. From the input move it will generate the path the robot will have to follow to get the pieces where they belong. It will move the pieces through square borders, so the nodes that will be used to build the path will be corners between 4 squares.

Amazing contributions

Our proposal for an amazing contribution **is an innovative way of playing chess through the use of a Monte Carlo algorithm**. We want to invest a lot of time in making a chess engine from the ground up, inspiring ourselves on the very best chess engines out there (Stockfish, Alpha Zero and others), but also making it unique and personal through a completely different way of playing. Should we succeed in making a strong player, we believe the purpose of this project shall be met completely.

Aside from the chess playing aspect, we have also tried to improve on our inspiration material. We have identified what we consider to be rather an overly complex way of moving pieces in the Raspbery Turk, so we have also designed a state-of-the-art magnet-based system that thoroughly simplifies movement and also clears the way for the overhead camera.

This is of course a very ambitious course of action, but we believe that success in its implementation may very well be deserving of the highest score. Undoubtedly, many pitfalls await in our path, but we have enough faith in our design choices to feel strengthened to overcome them.

Extra components and 3D pieces

*Due to the COVID-19 crisis, no physical implementation shall be carried out. We present here the description of those components we had designed before the confinement situation, but **we do not plan to develop them any further.***

- **Custom piece set and board:** We have designed a chess board that suits our needs by having custom coloring and an ideal size and composition for the camera. We have also designed a custom set of pieces that should fortunately be slightly easier to detect.
- **Camera structural element:** A vertical pillar that will hold the camera above the board.

Strategy for validation, testing and simulation

This project has the problem that there are some aspects of it that are tough to assess objectively, those being mostly related to the chess module. Our testing strategy for it will therefore be somewhat manual: we intend to have a relatively strong human chess player test it out and assess the game results, analyze its strengths and weaknesses with a conceptual approach and hopefully help us improve its playing style from his experience.

As of the writing of this document, a chess playing interface has been developed and connected to what is currently developed of the engine (Move Generation).

We also considered setting up a chess webpage bot (for instance in Lichess) being run by our chess engine, so that we can gather a lot more information (anyone can play against it). We have left this idea aside primarily due to time constraints, though we have not discarded it completely.

The vision system is much simpler to test, in the sense that only one possible solution is correct. Our testing strategy consists of setting up a number of FEN position descriptions and rendering them out automatically in a 3D design software (we have deemed best to use Blender for its ease of use and default Python support). These renders will then be fed to the Vision system and its output shall be compared directly to the original FEN description, therefore assessing its correctness directly.

The movement system will be tested in two ways: its path generation aspect and the reverse kinematics required to make it work. Generating paths in a visual way and calculating whether the path is optimal may be carried out in a simple console text-based representation. The optimality test can be done comparing our module's output to that of an exhaustive node search implemented with a Branch & Bound or Backtracking strategy (considering the number of nodes will be around 100 and the branching factor is 4 we do not think this should be too slow).

The reverse kinematics aspect remains undefined for the moment, since the best solution was a Blender simulation but that is not possible now because game engine capabilities have been erased in new releases of the software. We are considering other options such as Unity/Godot.

Due consideration should be given to the fact that only chess and vision will be implemented.

Final State

Currently, for this last sprint we have managed to make great advances in our designs, we have been able to successfully carry out a large part of the parts, only using two contingency plans, numbers 8 and 9.

The first has been used for not getting good power in the chess engine, having to simplify it to a simple alpha / beta search engine.

As far as point 9 is concerned, there was not enough time to develop the chess engine, so Stockfish endorsement has been used and we have focused on developing the vision aspect.

Foreseen risks and contingency plan

Risk #	**Description**	**Probability** (High /Medium /Low)	**Impact** (High /Medium /Low**)**	**Contingency plan**
1	Absolute piece position vision detection not working	Medium	Medium	Fallback to relative position of pieces algorithm (detecting changes of piece placements)
2	Piece set detection suboptimal	Medium	Low	Use a different piece set that has stronger and better defined traits.
3	Persisting piece set detection suboptimality	Low	Medium	Use subtle markings on the pieces to help identification
4	Persisting piece set detection suboptimality	Low	Medium	Use 2D printouts of pieces and detect them through convolution or feature extraction and correlation.
5	Lighting problems (e.g. shadows) lowering performance of detection	High	Low	Addition of a fixed-point light alongside the camera above the board.
6	Relative piece position detection not working	Low	High	Simplify the system and incorporate set rules when capturing the board (width/height of the board, square division…)
7	Slow Python implementation of the chess engine	Medium	Medium	Fallback to C++
8	Chess engine not strong enough	Medium	Low	It is not our intent to make the ultimate chess engine, but if it just does not work we will implement a simple alpha/beta search based engine.
9	Not enough time to develop chess engine	Low	High	Fallback to Stockfish and focus on the vision aspect
10	No way to simulate robot movement	Medium	Low	Console text-based step simulation.

References

This project has been inspired by the following Internet projects:

http://www.raspberryturk.com/ *A very interesting autonomous board system that implements a deep learning vision system and Stockfish for chess move calculation.*

https://stockfishchess.org/ *The best GOFAI chess program in the world, which so happens to be open source*

https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go *A revolutionary approach in chess AI that has inspired a new dawn of engines and a new trend in the way the game is played. The MCTS concept mostly comes from here.*

https://squareoffnow.com/ *A magnet-based autonomous chess board*

https://www.digitalgametechnology.com/index.php/support1/frequently-asked-questions/361-boards/general/193-video-how-does-a-dgt-e-board-work?mavikthumbnails_display_ratio=1.25 *The most widely used broadcasting board system*