***Stonewall***

A Computer Vision System for Live Analysis of Chess Gameplay and Feedback from a Chess AI

Sam Donald

Mechatronics Engineering

University of Canterbury

Christchurch, New Zealand

Sdonald.uc@gmail.com

Supervisor: Richard Green

Computer Science

University of Canterbury

Christchurch, New Zealand

Richard.green@canterbury.ac.nz

1. **Introduction**

Chess is an exceptionally complex and strategic game played throughout all areas of the world. Advancements in technology only allowed machines to surpass humans during the late 20th century with Deep Blues victory over Garry Kasparov. A major method of gaining knowledge is analysis of previously played games against opponent’s. To analyse a previous game, common practice is to manually record each move on pen and paper, prior to placing the move sequences into chess analysis software.

I aim to implement an automated system able to interpret live video footage provided via a side mounted webcam, either within a laptop or on a similarly heighted non-intrusive mount. This software will be able to determine the exact series of moves played within a game in the form of chess co-ordinates algebra. With this collected data, implementation areas include:

.

* Analysis of previous games after inputting into Chess analysis software.
* Providing move input to a Chess AI software, which itself will output its own move in reply.
* Sending of live data across the internet, allowing for word wide physical chess games.
* Tracking of multiple chess games in a tournament environment.

.

With the usage of the created software, recording moves by hand will become redundant. Analysis of moves for improvement or training will be far more accessible and the ability to coach or play with on another across the world would now be possible with the play on a physical board.

1. **Previous Related Work**

Tracking moves and identification of playboard surfaces is a common subcomponent of not only chess tracking software, but many other board game related programmes. There are an abundance of difference software and hardware approaches to the common task of board and piece identification.

Gambit [1], a chess robot created in the 2010 AAAI Small Scale Manipulation Challenge [2], uses a Prime Sense RGB-D camera, which is technologically identical to an Xbox Kinect. The chess detection software is advanced and divided into four sections: square detection, piece vs background detection, piece color and piece-type. This complexity is such

that the robot can enter the game at any point, without any prior knowledge of the game state, identify occlusion of pieces from a hand or other pieces along with allowances for pawn promotion. It detects 2D corners of the chessboard image, combines this with depth information at those points, and fits a 3D plane on the board using RANSAC [3] (a process of eliminating outliers). The model is matched to a 3D 8×8 template grid to determine the exact position and corners of the board along with location and identification of pieces.

Jonathan Coens [4] programmed a Chiara hexapod robot which competed in the same competition as Gambit. The hexapod is maneuverable around the board, allowing for different images captured by the simple RGB camera. The camera is mounted very low, approximately 15cm vertically from the playing surface and as such occlusion of pieces is an issue. This is rectified by comparisons between images gathered at different locations, mimicking possessing multiple cameras. The method of extracting piece location was unique, it searched for pixels of target color, whose southern neighbor was of a differing color. This produced a U-shaped edge under each piece and subsequently localize the pieces, movement of the camera was used to gather all piece positions. Hough Transforms were used to locate the edges of the playing surface.

Goncalves [5] implements a simple RGB webcam and transformed the corners of the chessboard to a normalized image. This was done so via manual human detection of the corners from a simple GUI. To determine the occupant of each square, the pixels bounded by the square are compared to the predetermined color of each average piece, again done so at the start of a game using a GUI.

Victor Wang [6] used a unique flood fill technique to locate squares using a simple overhead RGB camera setup. On an already populated board, the image is scanned for any one square through an iterative approach, once one square is found, its coordinates can be used to approximate the location of adjacent squares which are then tested. The process is repeated until the entire board is filled. Homography matrixes were used to map the chess board onto a regular predefined square. Pieces are then classified via comparisons of the bounded squares pixels to predetermined RGB color averages of the black and white pieces. These averages are taken at the start of the game before any move is made as the pieces are in a known location.

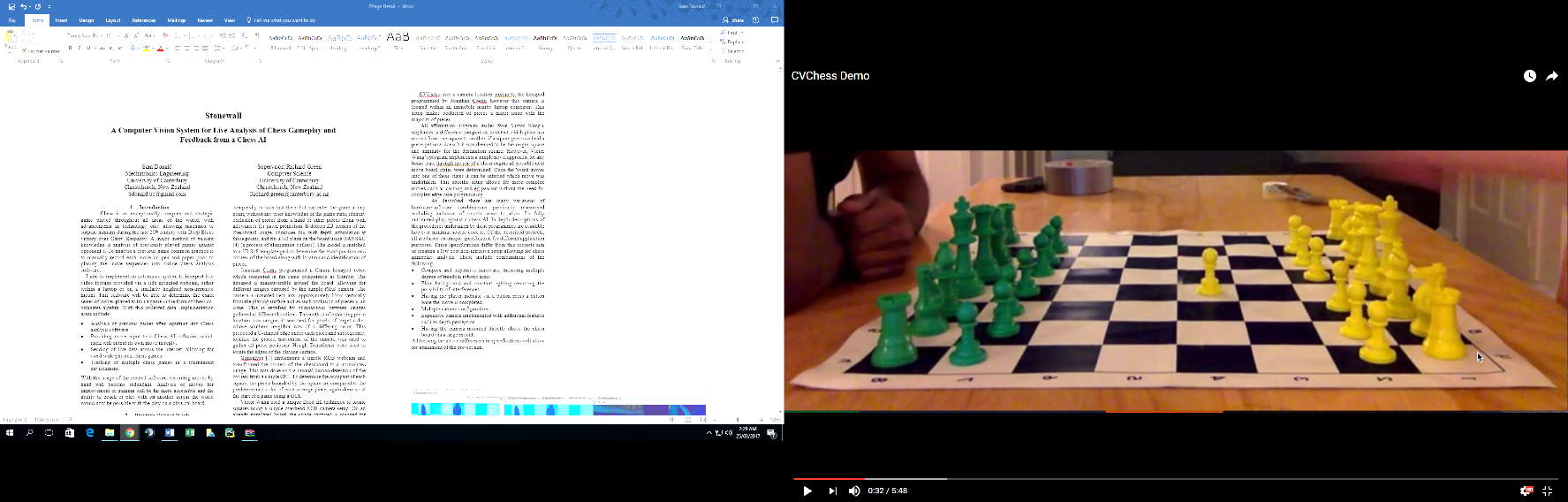
CVChess [7] uses a camera location like the hexapod programmed by Jonathan Coens, however this camera is located within an immobile nearby laptop computer. This setup makes occlusion of pieces a major issue with many pieces being obstructed from view as displayed in figure 1.

Figure 1: Occlusion of chess pieces from side mounted camera [7]

Board and piece detection is completed through corner detection, Hough transform and pixel color comparison and is similar to the methods implemented in Jonathan Coens hexapod [4] and Victor Wang’s [6] software. An algorithm was created to account for the occlusions within the image. This involved comparing each squares pixel composition to a reference color, however unlike previous designs, multiple squares experience a color change when the camera is mounted at this angle as displayed in figure 1 with the pawn accommodating squares above it as well. As such the algorithm placed more weighting on squares closer to the bottom of the board. This method worked in simple situations however the program failed to track over half of the sample games due to the occlusion present within the frames.

All affirmation programs asides from Victor Wang’s implement a difference comparison to detect which piece has moved from one square to another, if a square previous held a piece yet now doesn’t it was deemed to be the origin square and similarly for the destination square. However, Victor Wang’s program implements a novel approach, for any board state using an integrated chess engine, all possible next move board states were determined. Once the board moves into one of these states it can be inferred which move was undertaken [6]. This specific setup allows for more complex moves such as castling and en passant without the need for complex edge case programming.

As described, there are many variations of hardware/software combinations previously researched including inclusion of robotic arms to allow for fully automated play against a chess AI. In-depth descriptions of the procedures undertaken by these programs are available however minimal source code is. Of the described projects, all are based on unique specification for different application purposes. These specifications differ from this projects aim of creating a low cost, non-intrusive setup allowing for chess gameplay analysis. These include combinations of the following:

* Complex and expensive hardware, including multiple degree of freedom robotic arms.
* Plain background and constant lighting removing the possibility of interferences.
* Having the player indicate via a button press a button once the move is completed.
* Multiple camera configurations.
* Expensive camera implemented with additional features such as depth perception.
* Having the camera mounted directly above the chess board via a large mount.

Addressing the above differences in specifications will allow for attainment of the project aim. To do so the design will incorporate a single low cost readily available webcam that will be mounted at a non-intrusive, low angle, on the side of the board. It will also be able to detect itself once a move is completed by the player removing the need to indicate via a button press

1. **Methodology**

Testing of the Stonewall Chess analysis program was done on a home desktop computer running Windows 10 with an Intel i5-7500 CPU @ 3.40GHz and 16GB of RAM. The camera used for testing was a Logitech HD C525 webcam available for $49.99 NZ.

A. Chessboard and square detection

Previous works done on chess vision software have disregarded the inbuilt OpenCV *findChessboardCorners* function. This is due to the apparent poor performance at low angles and inconsistent results. However, after testing the function myself in an array of environments including and low angles, far away, close to the board and with many other objects in the background, it performed exceptionally well. OpenCV has since gone through multiple updates since the creation of the previously discussed programmes and perhaps some aspects of the *findChessboardCorners* underwent improvements. Using this function requires the board to be empty of all pieces and returns a list of coordinates corresponding to the inner corners of the board displayed in figure 2. This list must be sorted via an algorithm such that are in a known order, as the function returns the corners in differing orientations.

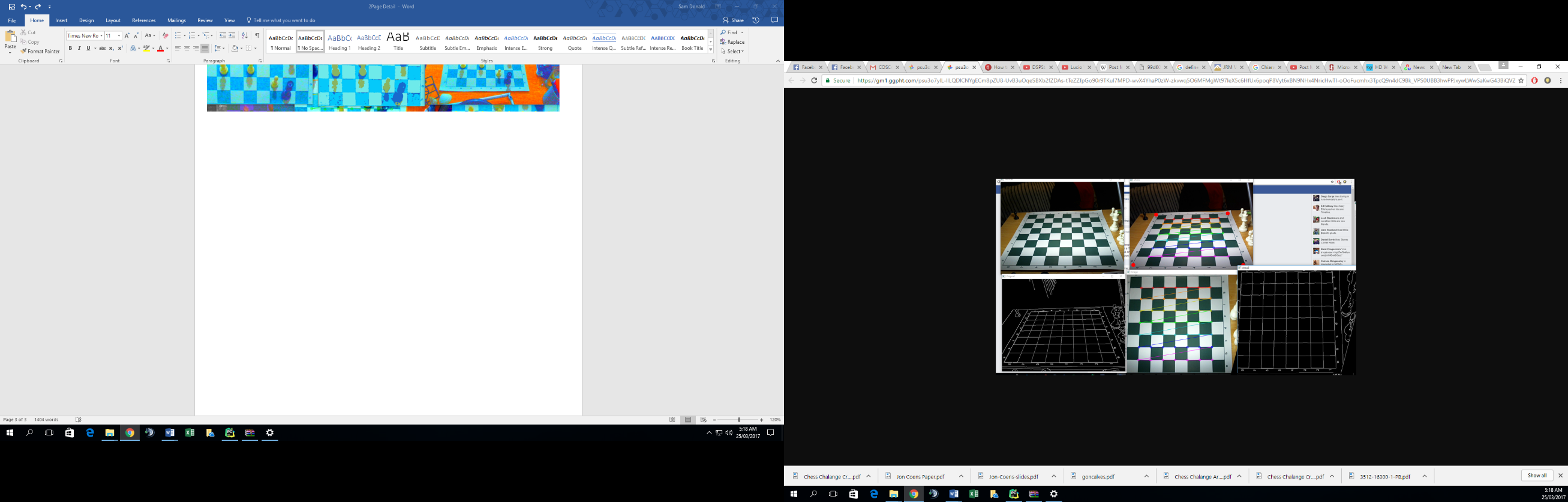


Figure 2: findChessboardCorners function

Due to the nature of the function, it is only capable of locating inner corners. However, with these 49 corner coordinates it is possible using their relationships to approximate the locations of the corners of the board (displayed as red dots on figure 2). These are used as reference points to apply homography to the image, moving the image into a square image of specified dimensions. The *findChessboardCorners* function is run once more to locate the coordinates of each square on this transformed image. To calculate the location of the outermost edges, the relationship between the known coordinates is again used. After this process is completed all squares coordinates are known and allocated to their specific squares variable. Any area exceeds the area bounded by the newly defined outer corners will be cropped to remove the possibility of interferences.

B. Determining square occupancy

The process used to determine the occupancy of each square at any given time was to be done so by comparison of the bounded pixels to a reference value which is predetermined at the initial setup before the game has commenced. This follows a similar process to that used within the Gonclaves program [5].

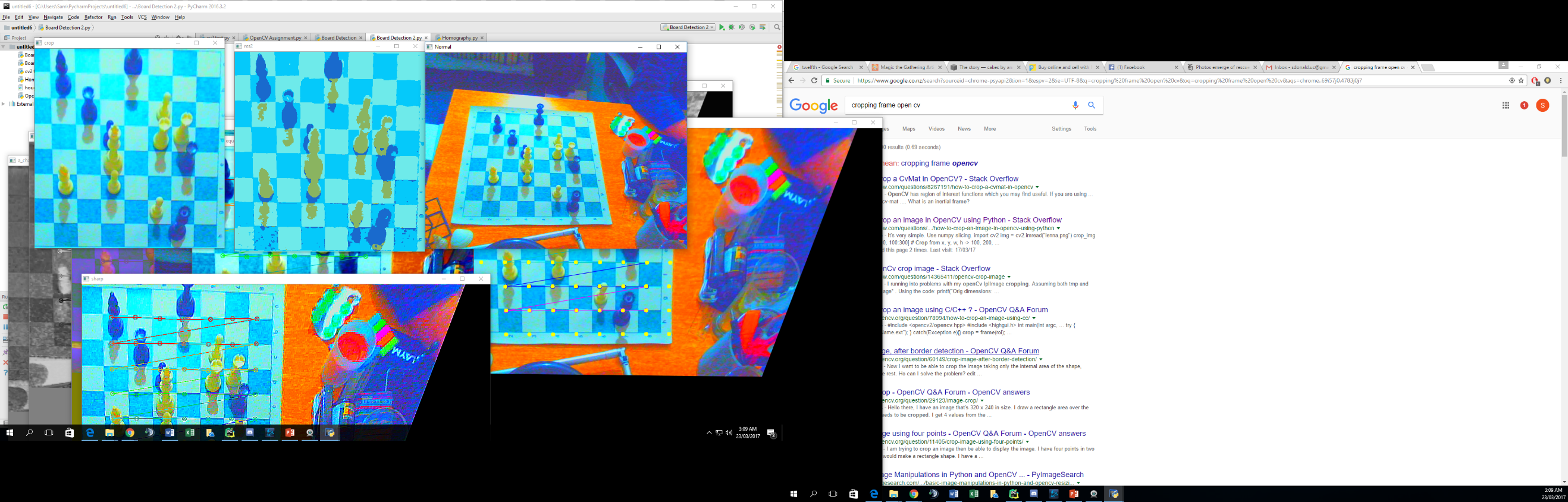
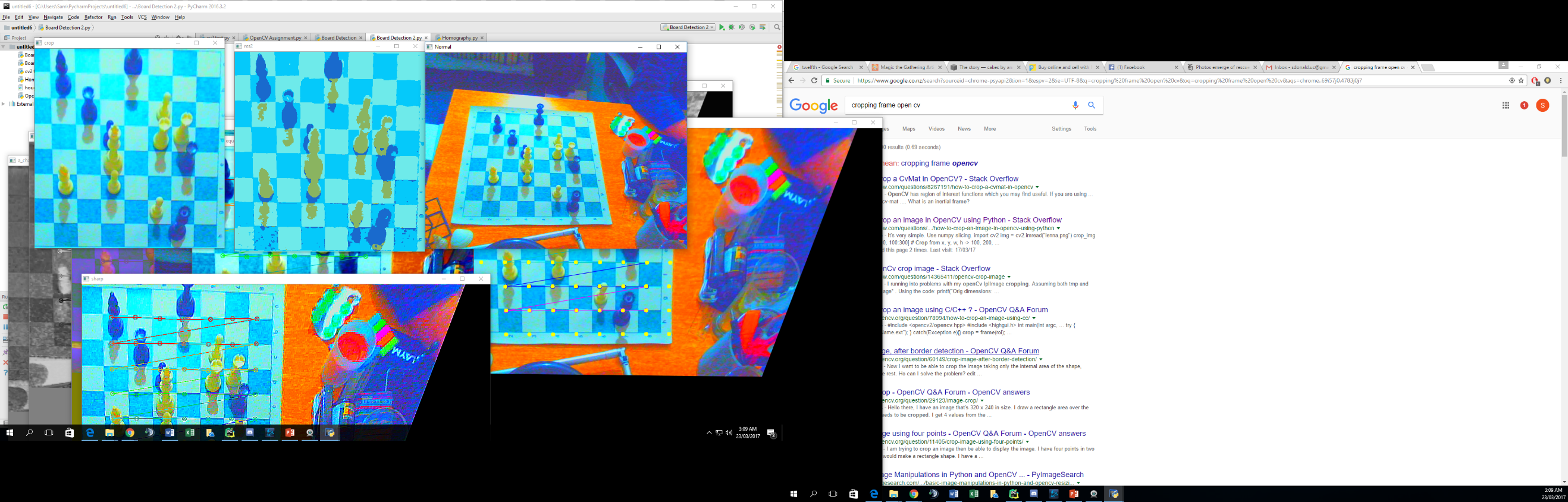
At the beginning of the game after the pieces have been placed onto the board, the four most common colours are calculated via a k-means clustering algorithm. These colours will correspond to the black and white squares along with the black and white pieces. When implementing this methodology on an unmodified image, it struggled to identify the four different classifications of colours due to their similarity. To account of this the input settings on the camera are adjusted such to make the differences in colours more pronounced. This is done so by changing the brightness, aperture and contrast within the Logitech camera settings. The output of this k-means algorithm is displayed in figure 3.

Figure 3: Image with pre-processing applied (Left), Image displaying 4 most prominent colours (Right)

The colour differences are now much more pronounced and identifiable from the squares that they are placed on. The shadows cast by the pieces does pose a problem however this test was done so under extreme lighting at an angle, therefore ambient light would solve this problem.

**TO CONTINUE**

* Determining moves
* Input to chess AI
* Output display
* Testing
* Conclusion
* Applications
* Improvements

**References**

1. C. Matuszek et al. “Gambit: A Robust Chess-Playing Robotic System,” in *ICRA*, 2011
2. D. Touretzky. “AAAI Small-Scale Manipulation Challenge: Robot Chess.” Internet: <http://chiara-robot.org/Challenge/>, Mar. 25, 2011 [Jun. 23, 2013]
3. M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Comm. of the ACM*, 1981, pp. 381-395.
4. J Coens. “Taking Tekkotsu Out Of The Plane”, Master’s Thesis, Carnegie Mellon University, 2010
5. J. Gonçalves et al. “Chess robot system: A multi-disciplinary experience in automation," in *9th Spanish Portuguese Congress On Electrical Engineering*, 2005.
6. V. Wang. “Chess Move Tracking using Overhead RGB Webcam”, Honors Report, University of Canterbury, 2013
7. J. Hack and P. Ramakrishnan. “CVChess: Computer Vision Chess Analytics”, Stanford University, 2014