CONCENTRATION
Epson Manipulator Card Sorting System

Kausthub Krishnamurthy
312086040
Email: kkri3182@uni.sydney.edu.au

James Ferris
311220045
Email: jfer6425@uni.ssydney.edu.au

Sachith Gunawardhana
440623630
Email: sgun5213@uni.sydney.edu.au

***Abstract*—An implementation of an Epson robotic arm as a card sorting mechanism aimed towards mimicing human style situational problem solving regarding the game Concentration (A.K.A. Memory) with the intent on addressing methods to simplify decision making processes for the game Set. By first addressing a simpler problem we can assess the suitability of this system overall in performing complex sorting operations in a way that directly mimics the way a human would perform the task.**

I. Introduction

The card game Concentration begins with a set of cards placed face down on a table. The player's objective is to pick up one card at a time and match it with another they have already seen before based on a specified criteria. This form of concentration is a building block to the game Set in that the similarity criteria can be modified and be made more complex. Something as simple as having the same card in every way to matching cards that share only some features.

The driving interest in this project was trying to exlore the similarities and differences between the series of processes and actions a human would take to complete a job and compare them with those that a robotic manipulator would need to undertake in order to match those actions. Since there is an inherently different way that machines work to humans we wanted to look into those differences and consider how best to make a system come as close to the original human playing style as possible.

Previous efforts from other research students around the world, on card sorting systems, seem to involve a robotic system that sorts cards it in a way that is specific to robots but lacks the capacity to PLAY the game.

II. Background

This section outlines the background knowledge that was important to the setup of this project. It has been split into the paradigms and concerns that are pertinent to each of the modules.

A. Program Control Paradigms

There are a few key factors to designing an Artificial Intelligence software to deal with problem solving games, many of which can be seen in good programming style across many disciplines of Mechatronics and Computer Science. These practices are used to simplify the way the computer "perceives" the world around it and affords us certain layers of abstraction that makes it both easier to program and less susceptible to semantic and logic errors.

1) Object Oriented Programming: OO Programming paradigms are a key feature in the way we handle the idea of a "card" in this system. A human will be able to look at a card and immediately associate with it specific traits that make it unique like the colour, shade, shape and number of shapes. In an Object Oriented programming style it's important for the program itself to be able to deal with the concept of a card (and its traits) in the same way. This incorporates the use of data structures that logically organise the information (pertinent to each card) in a way that it makes it easy to access (unlike storing values in arrays and having arbitrary correlations between objects and array indices) irrespective of the number of instances of the cards that are present.

2) Modular Programming: Maintaining code that can be modular and split into individual functions that can be reused on demand by multiple parts of the same program. The advantage of this is that it allows for less convoluted code leading to fewer logical failures and leaves potential for each individual function (already proven to work) to be used in a multithreaded version of the system. It also creates a clear function dependency heirarchy which makes the system easier to test both piecewise and in a cascaded series of running tests.

3) Error Handling: As with any game system there will be situations in which the AI reaches fail cases in which it cannot succeed in its objective. As such it is crucial for the system design to take this into account and to allow it to either recover or gracefully exit from these issues.

B. Vision Detection Principles

The task was to identify the model of the card from the variations on shape, shape count and the filler. Figure shows

the different variables in the used card set.

Fig. 1. *Different types of cards*

Shape	Shape Count	Filler
Triangle	One	Shaded
Rectangle	two	Non-Shaded
Ellipse	three	Block

Fig. 2. *Variables*

In the robot arm we had there is a camera mounted on top of the table which can be used to get images of the objects place on the table. This images can be used to identify the object properties using image processing techniques.

There are plenty of approaches to find features in image processing. Following are some of the processing techniques.

1) Edge detection
2) Binary image processing[2]
3) Corner detection
4) Hough transformation
5) Swift operator
6) Histogram comparisons

In this project Edge detection and Binary image processing methods have been used to identify the cards from the table and shapes in the cards.

1) Edge Detection: An edge in an image is a significant local change in the image intensity, usually associated with a discontinuity in either the image intensity or the first derivative of the image intensity[2]. Matlab toolbox has an inbuilt function for Edge detection. Canny operator[1] have been used in this project.

2) Binary image processing: Using the histogram of the image a threshold can be identified to convert the image to a binary image with areas of interests. This method was feasible to this project as the image of back of the card gives pixel value near to black and a image of the front of the card gives pixel values near to white.

After creating the binary image with interested regions matlab inbuilt functions can be used to identify centroids and orientations of the interested regions. Local positions can be transferred to world coordinates using a reference point in this case are 3 fiducial boxes at known positions.

C. Motion

Degeneracy and Dexterity

- positions with multiple inverse kinematic solutions - can't select one, so stalls

III. Experimental Design

The following section details the design rationale and the process followed for each part of the system.

A. AI Setup

Program Flow: The program is set up in a way that adheres to the following flow structure:

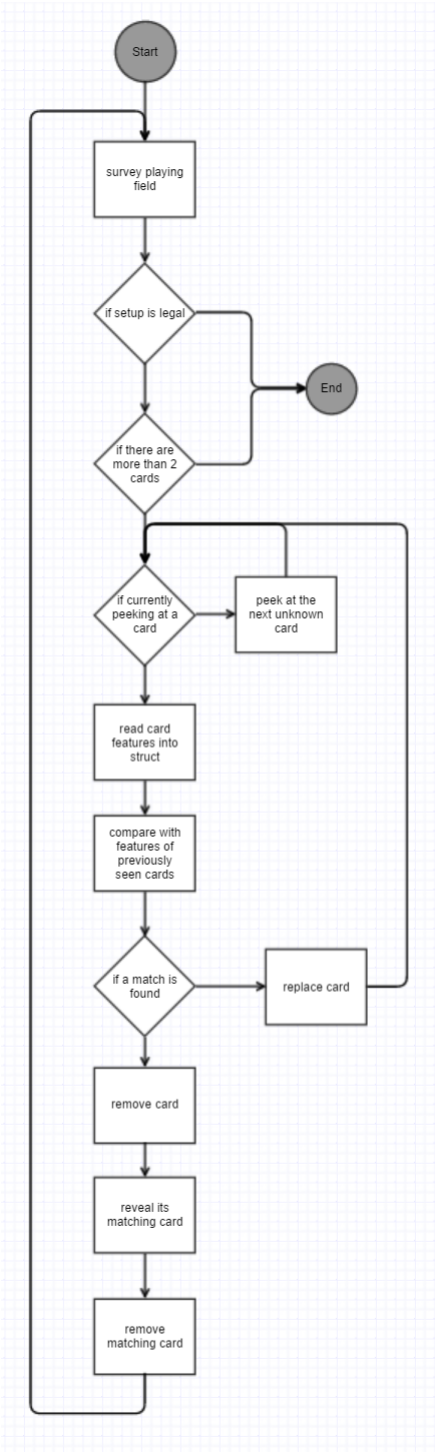

Fig. 3. *Program Flow Diagram*

This structure in Fig.3 focuses on claiming a matching pair as soon as it is found.

Game Modes: Consider the logic table in Fig.4 which shows us 16 different numbers that can be expressed as a single character where the the 4 right hand bits represent a flag value for what should be compared in each run of the game based on the game mode that is selected. The reason for this comparison structure is that if individual flags are set high we would have to poll all the flags but by using bits in a character as a flag system we can simply compare with a final value that we expect to see which reduces the number of flag checking to a single poll irrespective of the number of factors that need to be compared. This maximises the scalability of this program to series of cards with many more differences.

It's important to note that this compare function is designed to be modified for the game Set. In the game Concentration we use a Game Mode that's specified in code that masks the bits in a way that we only get identical pairs. Whereas to develop this for set we would have to see assign a "score" for each set that was found and at the end of examining EVERY card on the field figure out the combinations that will maximize the score that can be made from the series of cards at hand. In this setup we had Game Mode 11 as we couldn't acquire colour images from the camera. This game mode will accept sets of cards that have the same Shape, Filler, and Count.

Game Mode (dec)	Game Mode (binary)	Shape	Colour	Filler	Count
0	0000 0000	0	0	0	0
1	0000 0001	0	0	0	1
2	0000 0010	0	0	1	0
3	0000 0011	0	0	1	1
4	0000 0100	0	1	0	0
5	0000 0101	0	1	0	1
6	0000 0110	0	1	1	0
7	0000 0111	0	1	1	1
8	0000 1000	1	0	0	0
9	0000 1001	1	0	0	1
10	0000 1010	1	0	1	0
11	0000 1011	1	0	1	1
12	0000 1100	1	1	0	0
13	0000 1101	1	1	0	1
14	0000 1110	1	1	1	0
15	0000 1111	1	1	1	1

Fig. 4. *Game Modes*

Card Data Structure One of the more important parts of the setup is that the program is able to see what it considers a set of cards where each card has:

- X Coordinates
- Y Coordinates
- Orientation
- Shape
- Colour
- Filler
- Count

This makes it very easy to address individual traits on any set of cards. This needs to also have a trait that links it to other cards that it could form a set with and the score potential

B. Vision Detection Principles

In order to achieve the vision detection task we had broken down the task into 4 sub tasks as following

1) Identify the Cards on the table.
2) Coordinates and pose of the cards.
3) Match the local coordinates to the world coordinates.
4) Identify the features of the cards.

Matlab image processing toolbox has been used for image processing tasks of this project.

1. Identify the cards on the table.

- In the initial sweep after hiding the robot arm the camera will capture an image of the table. Histogram equalisation has been used to equalise the intensity over the image to make the image processing task easier.
- The back of the card gives pixel values near to black colour in gray image. Using this data a logical image has been created by using a threshold of pixel values less than 0.09. After removing pixels areas less than 1000 using function; bwareopen a structured element has been used to morphologically close the founded blobs.
- The same logic has been used to find face up cards as well. In that threshold of 0.91 has been used as face ups are near to white colour.

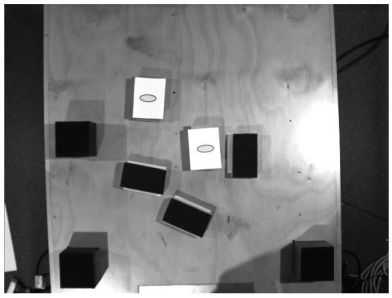

Fig. 5. *Initial captured image*

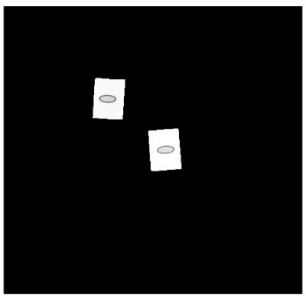

Fig. 6. *Isolated Cards*

2. Coordinates and Pose of the cards

- Logical image created using above mentioned method has been used to find the local centroids and the orientations of the cards. The function; regionprops has been used to find the centroids and orientation. After finding both, centroids and orientations of the fiducials have been removed. In orientations a redundancy has been added to make sure that the robot arm will always work in the safety limits by checking the value of the angle.

3. Match the local coordinates to the world coordinates

- 3 fiducials have been used to get the relative position of the cards. The method used was the perpendicular distance from two fiducials to the centroid of the card. Following figure will illustrate the method.

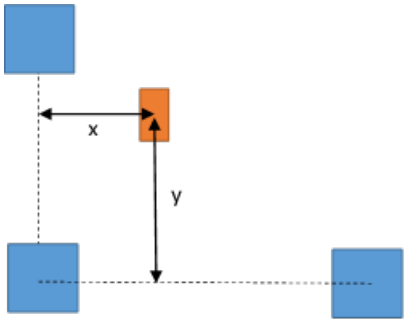

Fig. 7. *Method used to match with world coordinates*

4. Identify the features of the cards

- The main target was to identify the card itself before trying to extract features. Used the same method as above and identify the card using pixel values as thresholds. Used the found logical image as a mask to isolate only the card face.

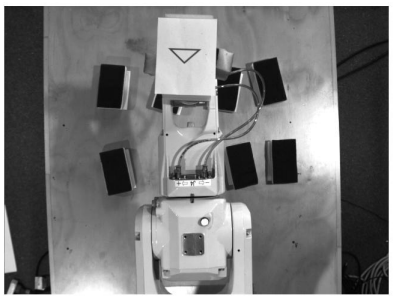

Fig. 8. *Initial image of the faceup card*

- From this image using a bounding box cropped out the card to process to find the features of shape, shape count and filler.
- Edge detection has been used to find the shape. After detecting the edges of the shape unnecessary edges has been removed using clearing edges connected to border and removing connected regions less than a threshold pixel area.

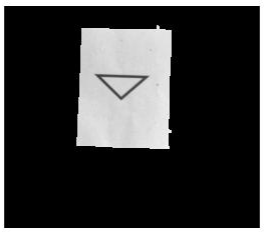

Fig. 9. *Isolated Card*

Fig. 10. *Identified Edges*

- Perimeter of the founded blob has been used to identify the shape as triangle, rectangle and ellipse had different perimeter values. No of connected components in this stage has been used to identify the shape count. If the shape count is greater than 1, the mean perimeter will be checked for identify the shape. The final logical image with the shape has been used with the original image to identify the filler status. Mean intensity of the blob area of the logical image in original image will have a value greater than 0.72 if the shape is not shaded as pixel value for white is 1. For block status the mean intensity will be lowest and for shaded status the intensity value will be between 0.72 and 0.5.

C. Motion

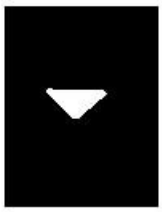

Fig. 11. *Identified Shape*

IV. Results

A. Vision

The image processing algorithm did able to match cards considering all variables, shape, shape count and filler. Results of the image processing can be summarised as follows.

Task	Completed
Identify & Differentiate Face Ups & Face Downs	100%
Identify the pose of cards & match with world coordinates	100%
Identify different Shapes	100%
Identify no of Shapes	100%
Identify the filler of the shape	100%
Identify the colour of the shape	0%

Fig. 12. *Results Table*

Note: We only did able to get gray images from the camera as there was a problem with white balancing with the camera driver which we didn't able to fix. If we had rgb images for check the color of the shape we should have check the 3 channels separately and found the dominant channel.

Results were as following,

1. Shaded one Rectangle

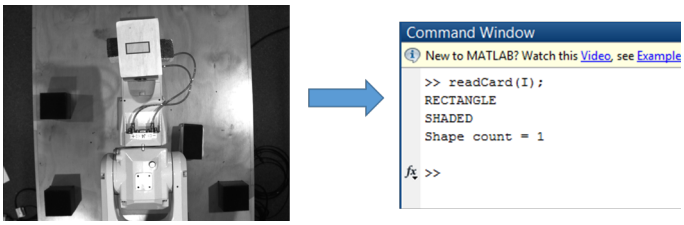

Fig. 13. *Results for a shaded rectangle*

2. Block three Rectangles

3. Shaded one Elipse

B. Motion

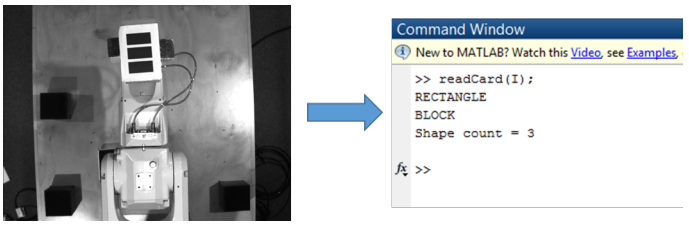

Fig. 14. *Results for non shaded Triangle*

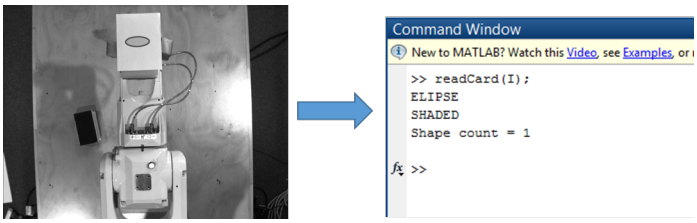

Fig. 15. *Results for Shaded Elipse*

V. Discussion

A. AI

B. Vision

As shown in the results section the image processing algorithm developed was able to find matching cards irrespective of changes in card orientations. There were a number of pitfalls to the vision detection algorithm.

Shadows: When the lighting condition changes we had to manually change the thresholds we used to make the algorithm work. Still when darker shadows exist the method we used won't work. If we had rgb images we would have convert the images into the HSV space and subtract the shadow from the image for a certain extent and use that image to further processing.

Color: As noted under results section as we didn't able to get RGB images this section was removed from our scope which is achievable easily.

C. Motion

VI. Conclusions and Future Work

Considering the successes and failures discussed above the logical future steps would include acquiring a white-balanced colour input in order to be able to deal with colour differences and shadow removal. Generalising this code to work with any set of cards would be a positive step for the game "Concentration" but for the game of Set it would be counter productive. Instead, making the changes to the comparison logic and the card data structures to allow score analysis would be the next step in completing the game such that the robot would play both of these games as a human would, bringing to it the natural advantage that is its robust use of memory (which is the biggest downfall of the human capacity to play these games). Future development may also consider redesigning the gripper arm with something like a suction tip or similar grasping method that allows for the setup to eliminate the need for sponges as the robot will then be able to pick up the cards from the table by itself.

References

[1] Canny, John: *Pattern Analysis and Machine Intelligence* ,6th ed IEEE, 1986/11.

[2] R.Jain,R Kasturi,B.G.Schucnk: *Machine Vision*, 1st ed McGraw-Hill,Inc, 1995.

[3] Register, Andy H.: *A Guide to MATLAB Object-Oriented Programming*, SciTech Publishing Inc., 2007, Georgia Tech Research Institute

[4] Shell, Michael: *IEEEtran*, IEEE formatted LaTeX template http://www.michaelshell.org/tex/ieeetran/