Image analysis report

Muhamad Harith Arash (2291111)

January 2023

1 Missing Puzzles

1.1 Solution Flow

The process begins with an image of a jumbled arrangement of the Jigsaw Pieces. The white background in the image is eliminated and the image pieces are processed to get the outline of the individual pieces. In this first part, all pieces edges will be compared to the empty puzzles edges and a best possible match is chosen. After compatibility is matched, the missing pieces will be placed on the chosen empty pieces.

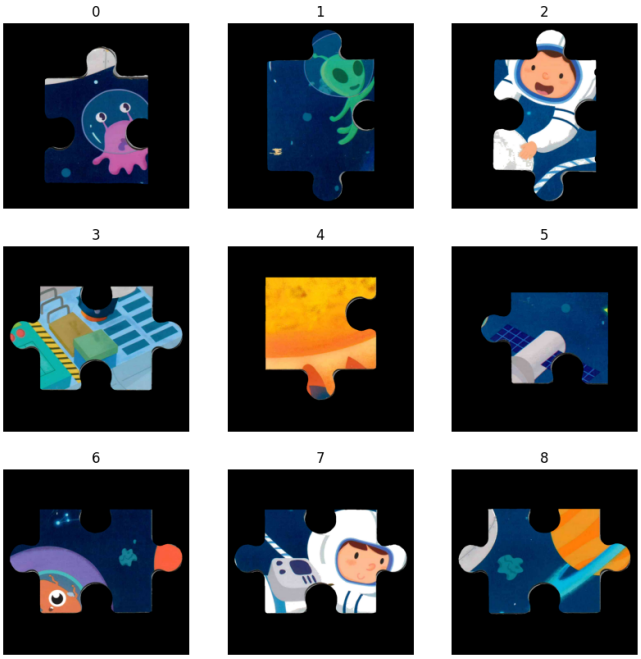
1.2 Explanation process

1.2.1 Image Processing

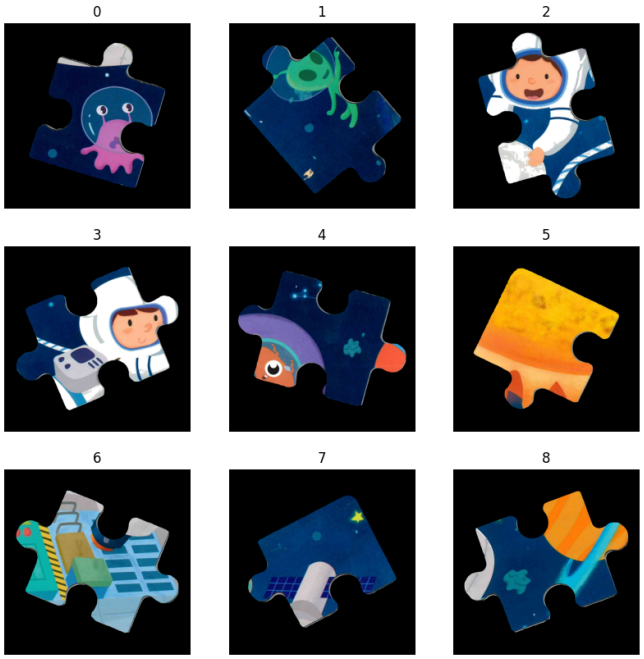
Euclidean distance method : Initially, the image will undergo a process of maximising white intensity via the Euclidean distance method. Using pixel [255,255,255] as reference to increase the score of whiteness. The lower the distance the higher the chance that pixel value is a shadow pixel and be converted to the actual white pixel (255). This is to ensure the background would be totally white background.

Thresholding and blurring : The image will then be thresholded and blurred, which will remove shadows from the edges and convert it to a binary image.

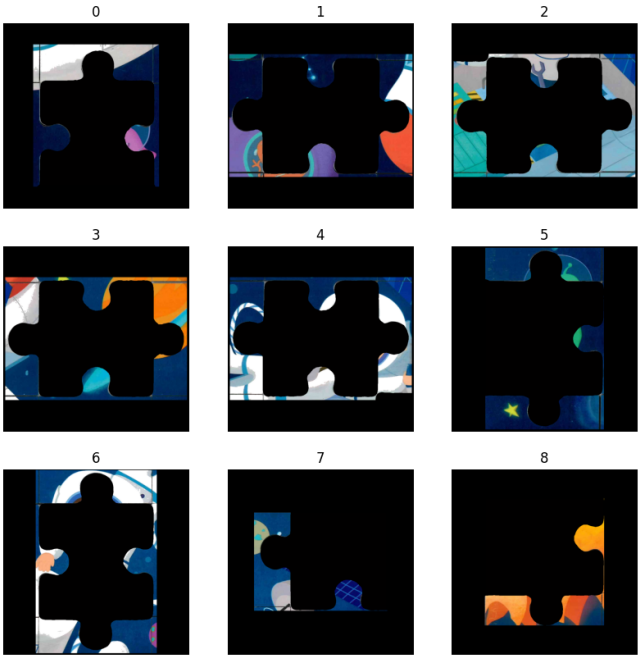
Cropping Out Puzzle Pieces : Now that the image is binary, each puzzle can be extracted via the contouring process, and the image will be multiplied on the original image to extract coloured tiles one by one. Function boundingRect() provides box points for each puzzle. The separated puzzles from missing pieces 1, 2, and missing pieces can be seen in the figure below:



Separated-puzzles-part-1-a



Separated-puzzles-part-1-b



Separated-empty-puzzle

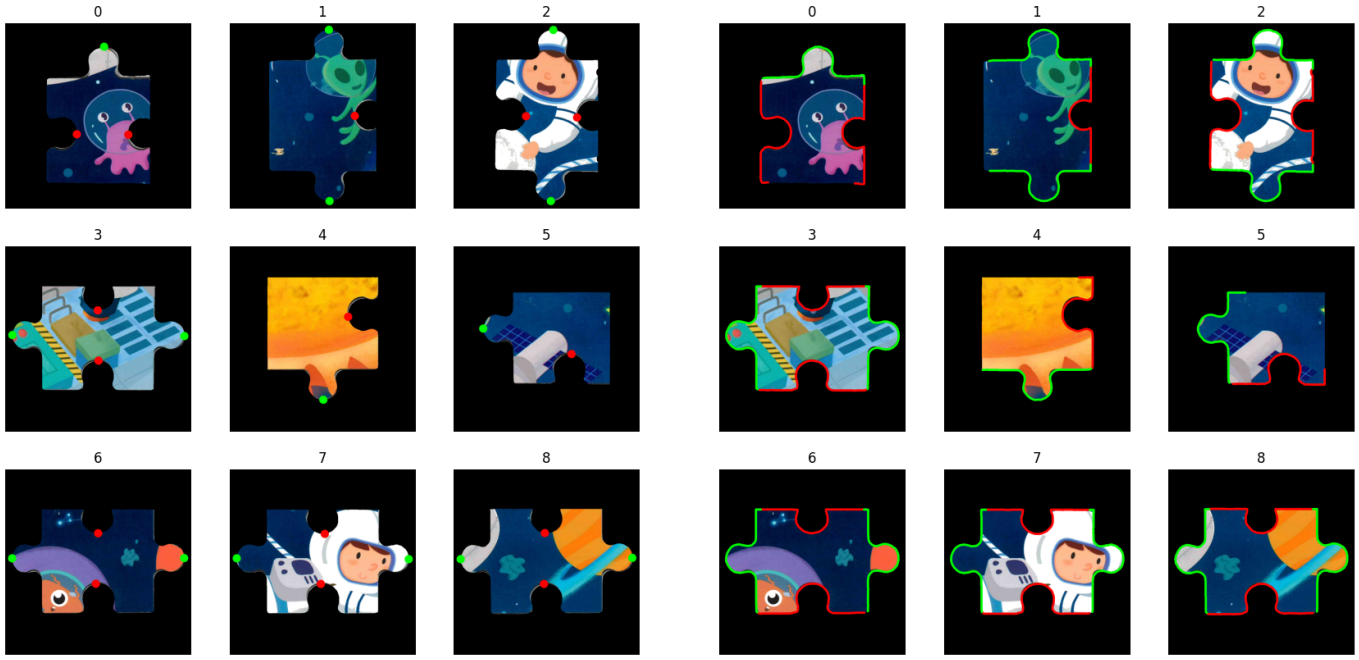
Figure 1: Cropped Pieces

1.2.2 Data Extraction

Aligning Puzzles in the Upright Position : In Part 1a, all missing pieces are already correctly aligned with the empty pieces, so there is no need to rotate. While puzzles in Part 1b are still chaotically rotated, by using the angle of the bounding box, all chaotic puzzles will be rotated to the nearest axis.

Detecting Puzzles Locks : After rotating process for each puzzle, the process of detecting inner locks and outer locks begins. The algorithm have been developed to locate the interlocking parts of puzzles – inner and outer locks.

First, the program calculates the convex closure of the puzzle and its convexity defects—the deepest points on the contour. If the distance from the convex hull to the defect is insubstantial, that is, smaller than the error margin, e.g., 2000 px, that point is not considered to be a defect. The resulting points will be used to calculate the area between one point and the adjacent point; if the area is less than 25000, then the middle of the point is considered an outer lock, whereas the other is an inner lock. In general, all inner and outer locks can be pinpointed. From the locks' coordinates, the edges can be located as well. Figures below show the locations of all the locks and edges in each puzzle.



Locks identification Edges identification

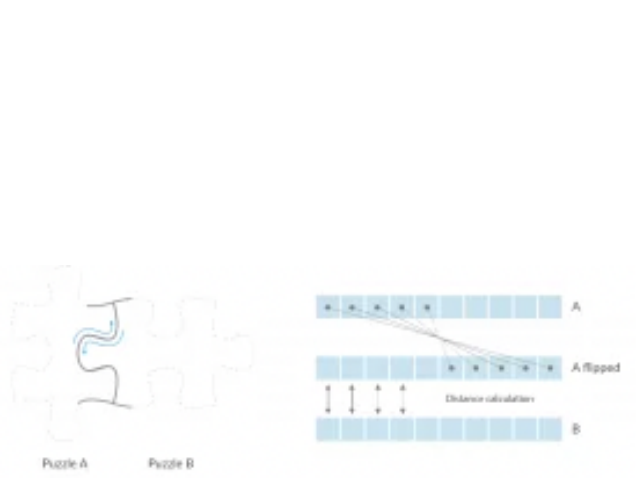
Figure 2: Locks and Edges identification

Data packaging on edges: After Identification of locks, all puzzle information such as coordinate, contour edges, position and locks type will be stored into a list for matching process.

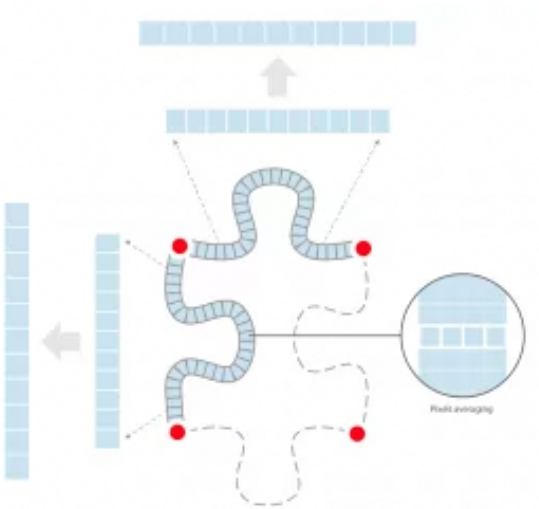
1.2.3 Matching Compatibility

Shape Matching : In this process, for Part 1a, each puzzle is not required to be rotated, so each missing piece can be compared directly to the empty pieces. Only pieces with the same shape will be considered suitable candidates to reduce the time required to compare the pieces. However, in Part 1b, all missing pieces need to be rotated by four quadrilateral angles. So, each piece has four different versions, and only pieces with the same orientation will be considered suitable candidates. All candidates that passed this condition will proceed to the next colour matching.

Colour Matching : Here, all puzzle edges will be compared to the empty puzzle edges. The RGB edges will be converted into HSV values for the best comparison output. The distance between two colour lists with one of them flipped is calculated using the Euclidean distance method. The average of the distances for each edge for each piece will be stored in another list. According to this method, the lower the value of distance, the more likely it is that missing puzzles and pieces will be fitted together. Finally, all missing pieces will be paired with empty pieces.



Flipped On Edge



Average On Edge

Figure 3: Color Matching

1.2.4 Assembly

Superimposed and replace : After finding all the best candidates for missing puzzles, both images will be added together and replaced in the original image of the empty puzzle. The final image after completion can be seen below:

Completed-part-1-a

Completed-part-1-b

Figure 4: Assemble Puzzles

2 Neighbouring pieces

2.0.1 Image processing :

Thresholding and Blurring : In Part 2, each image undergoes a maximum white background process, thresholding and blurring to smooth, removing shadows on edges, and converting an RGB image into a binary image.

Extracting pieces : The first 16 large contours in each image will be used as a mask and multiplied with the actual image to extract the actual image. However, there are still shadows on the edges. Using the scaling contour method, the contour size will be reduced by 1%. Each contour will be used to generate the mask, which is then multiplied with the actual image to extract the actual puzzle, which has a black background.

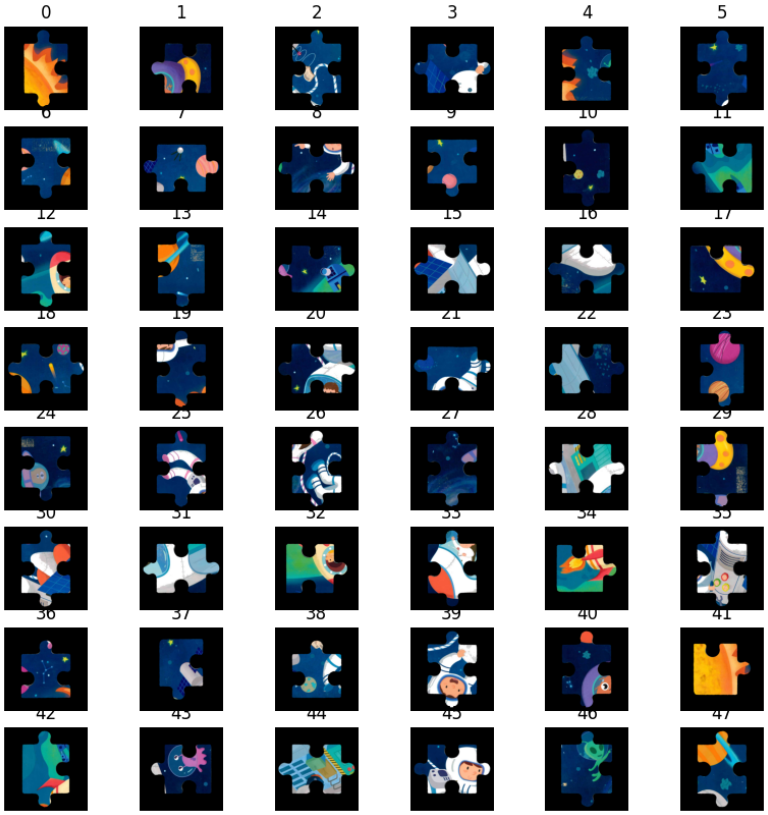
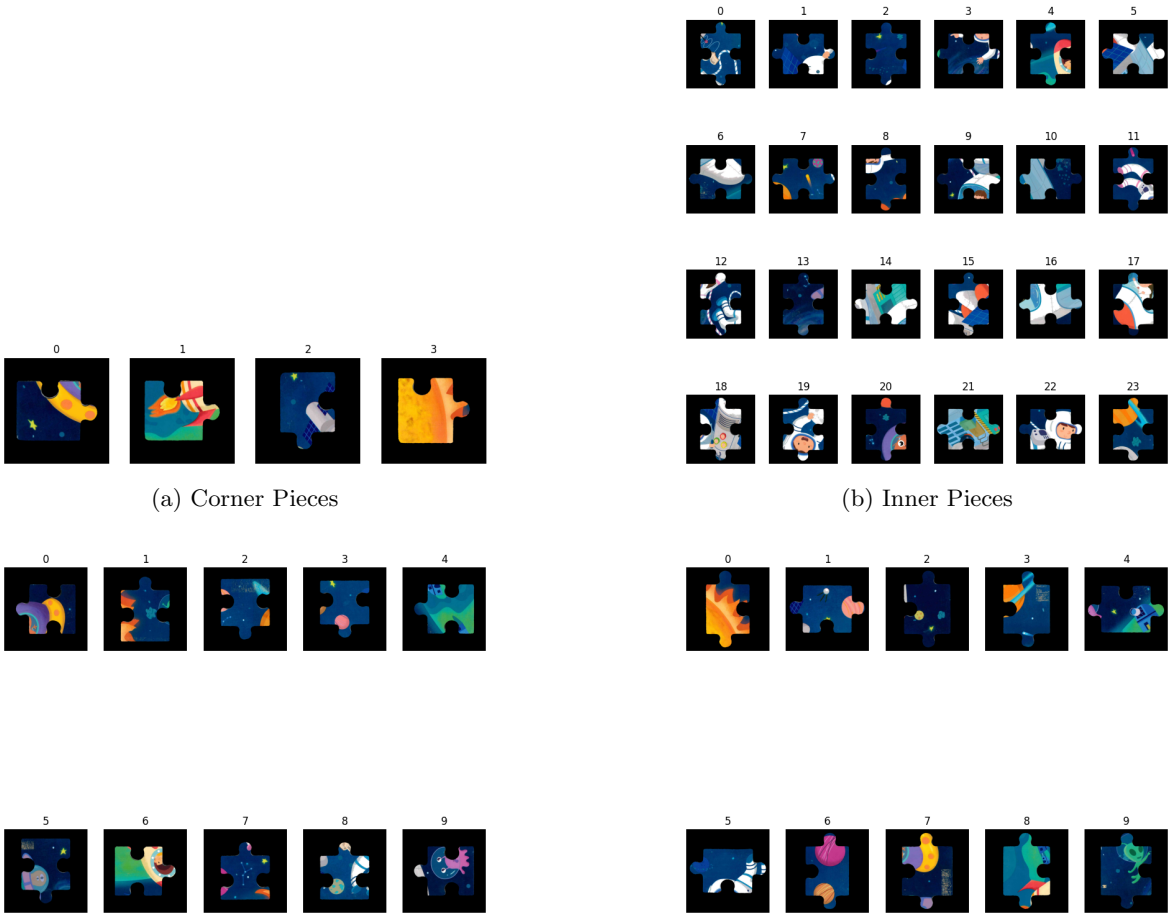


Figure 5: All 48 pieces

2.0.2 Classification

Classification : After identifying the convex hull and defect points (explained in Part 1), all puzzles will be divided into four groups (corner, inner, border1, and border2). This is to avoid a long process of matching.



(a) Corner Pieces

(b) Inner Pieces

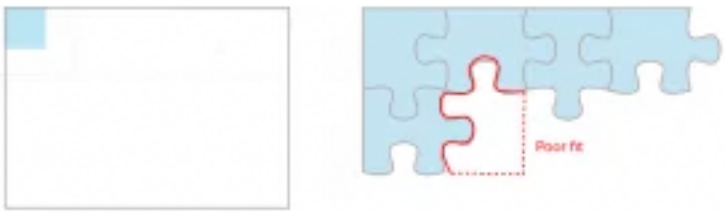
(c) Border Pieces

(d) Border Pieces

Figure 6: Puzzle Classification

2.0.3 Matching Compatibility:

Sorting and Color Matching : All puzzle pieces are in a 6 x 8 arrangement. The puzzle will be sorted starting from the first top-left corner to the last bottom-right corner. So the correct template layout was created for each iteration to reduce complexity. In the first row, the puzzle will be compared from the right edge of the current puzzle to the left edge of the candidate puzzle. For the first column pieces, each candidate puzzle top edge will be matched up to the previous row puzzle bottom edge. The rest of the pieces will be compared to both the top and left pieces. The Euclidean distance method is used to compare all pieces' colour edges. The best candidate puzzle will be the one with the shortest average distance.



Orientation: Similar to Part 1b, since each puzzle is not in the correct rotation, it will be rotated into four quadrilateral angles. Only the candidate puzzles that have the correct orientation and type can proceed to the next colour matching process. In general, only one of the best candidate puzzles will be selected in each iteration. All neighbouring puzzles can be seen in the figure below:

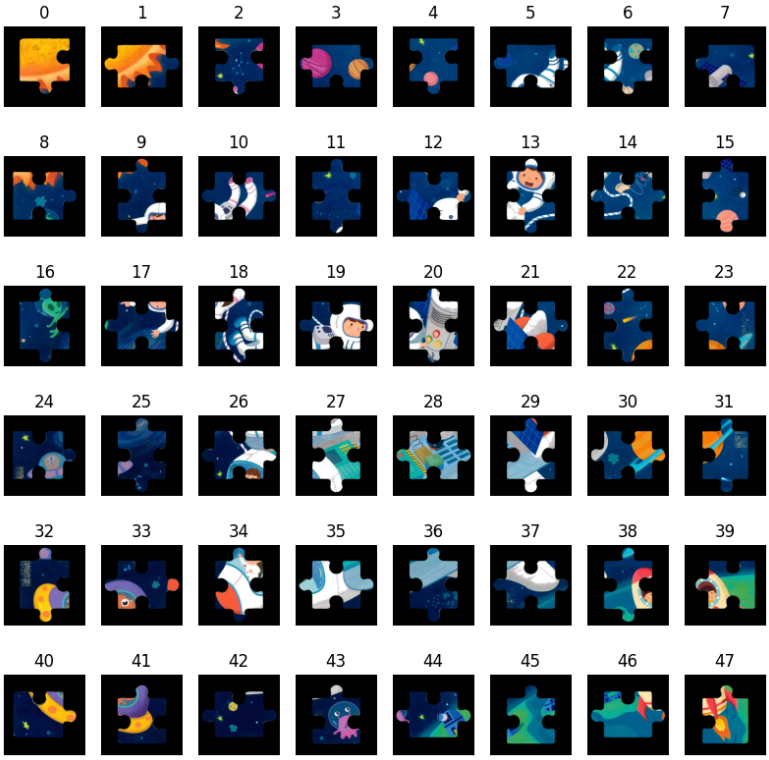


Figure 7: Neighbours Puzzles

3 Completed Scattered Jigsaw Puzzle

Assembly of scattered puzzles : Continuing from Part 2, each puzzle will be shifted through the translation matrix method to be placed next to the previous puzzle in each iteration, using the first top-left corner piece as a reference. Since all pieces have inner and outer lock coordinates, they will be used to calculate the shifted values of horizontal and vertical. The completed jigsaw puzzle can be seen in the figure below.

Figure 8: Completed Jigsaw Puzzle

4 Rotated Camera Jigsaw Puzzle

4.1 Pre-processing

In this Part 4, the image will undergo adaptive thresholding by OpenCV. With Gaussian adaptive thresholding, each region will have a different threshold value. When masking the pieces, the filter median blurring is used to remove excess edges.

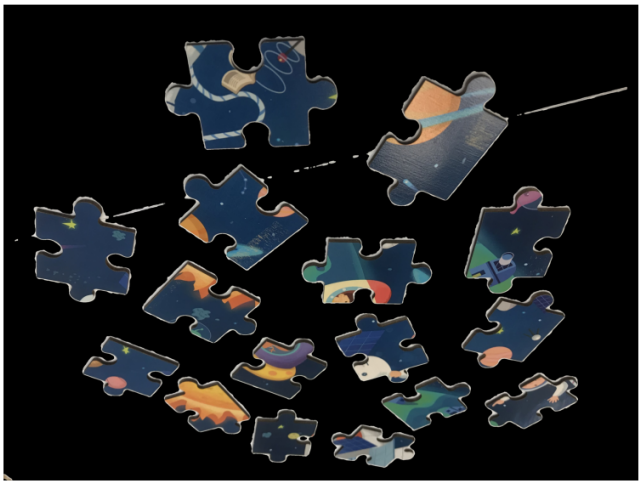


Image 4

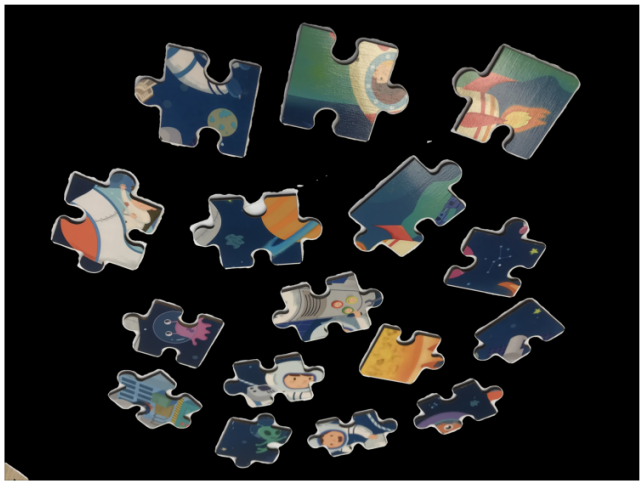


Image 5

Image 6

Figure 9: Removed Background Color

4.1.1 Image Enhancement

The image needs to be enhanced, such as with brightness and contrast tuning, so when using SIFT feature detection, it can be identified with all the completed pieces from Part 3. Hence, a function have been created to enhance for each image.

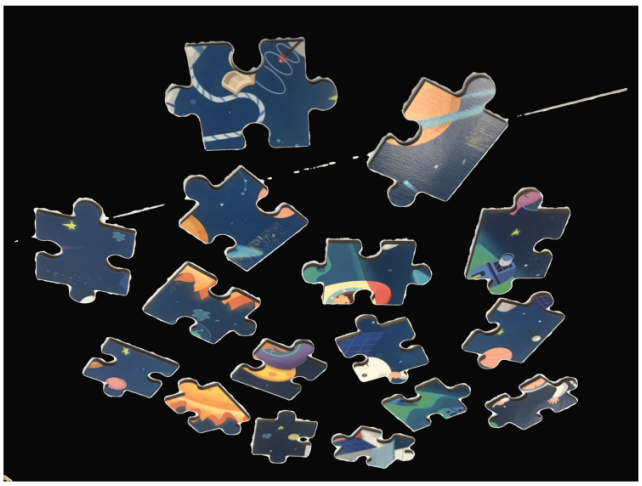


Image 4

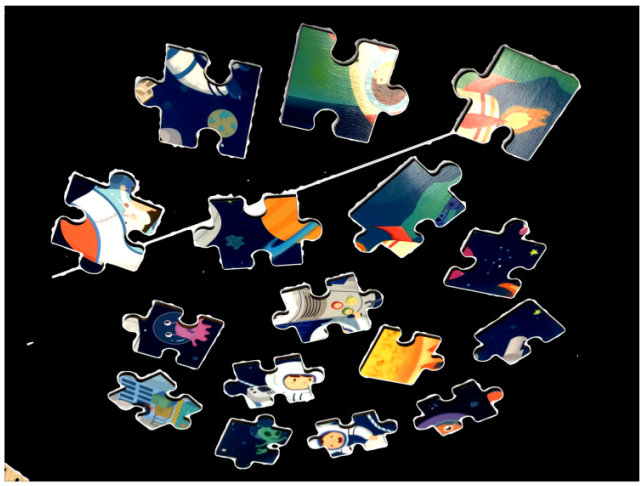


Image 5

Image 6

Figure 10: Image enhancement

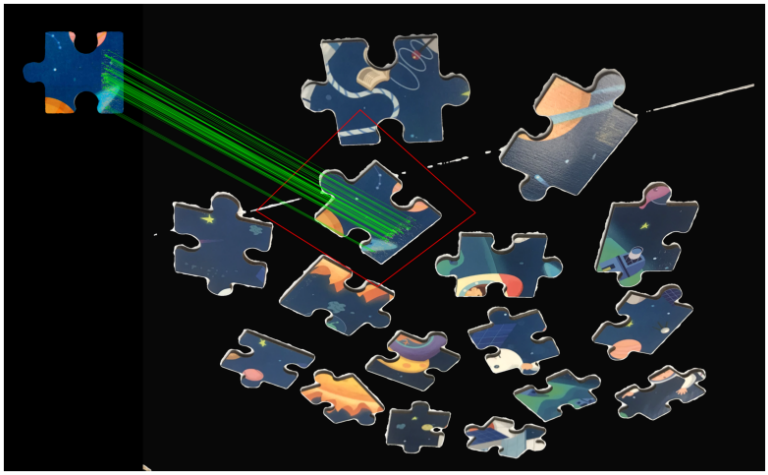
4.2 Features Detection, Detection Warping

All the processes can be seen in Figure 11.

Sift detection : Using all the sorted pieces from Part 3, the sift function will find the top 50 matches on the images after matching keypoints and descriptors between them. By using the ratio method to find the best match, every puzzle has a matching piece from the image.

Planar Homograyphy Construction : The matching points from feature detection will be used for homography, which maps the points in one image to the corresponding point in another image. As a result, a matrix is generated to be used for navigation.

Warping : Using the given matrix as an input for the perspective transformation process, the image can be rotated and scaled to be in an equivalent shape to the completed pieces in Part 3.



Example Sift detection

Example Puzzle Extraction

Figure 11: Puzzle Identification and Extraction

4.3 Assemble

After all the identification processes, the extracted pieces are sorted back according to the completed pieces in Part 3. So far, only 12 pieces can be detected with this method, as shown in the figure below.

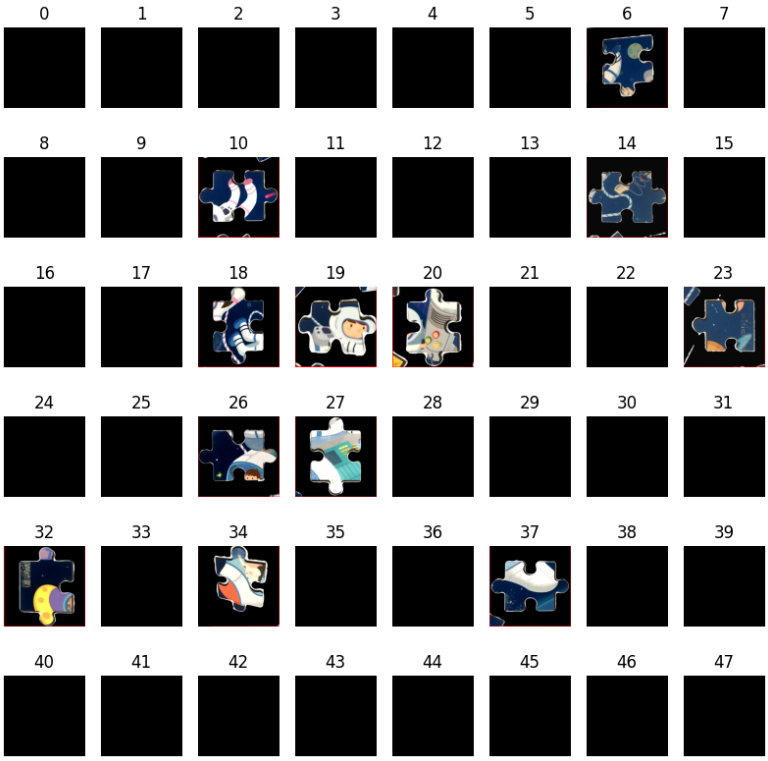


Figure 12: Puzzle Assembly

4.3.1 Discussion

From the figure above, only 12 pieces can be detected and mapped into the actual puzzle. In the future, images can be further enhanced by removing glare or rotating the images first by using one of the pieces as reference (as seen in the figure below).

Image 4

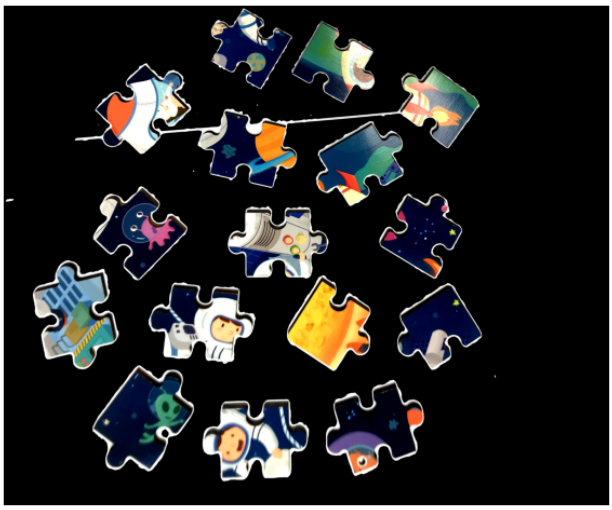


Image 5

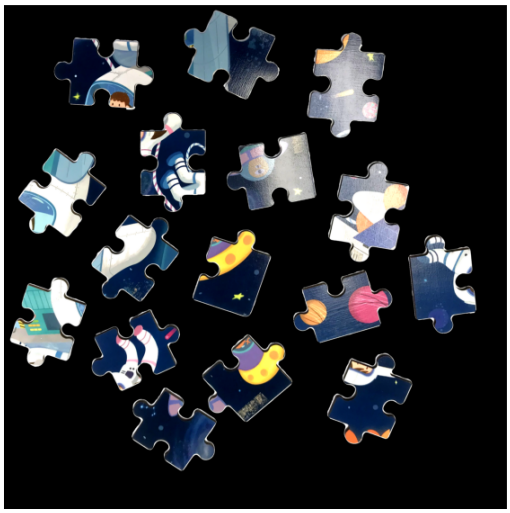


Image 6

Figure 13: Rotated Image

All neighbouring pieces can be detected using feature matching or edges methods.

1420 words