

# Image Processing for Color Pattern Recognition

## 1-) Introduction

The aim of this project is to develop a robust software system using MATLAB to automatically detect and interpret color patterns from given images. This task mimics real-world challenges, where users recreate a displayed image from memory and have the results verified by computer vision technology. The ability to automatically read and decode color patterns has numerous applications beyond gaming, including in security, data storage, and advanced imaging systems. Color data matrices, for example, offer enhanced data density over their black and white counterparts, providing a rich field for deploying advanced image processing techniques.

## 2-) Methodology

The methodology of this project is structured around the development of a cohesive image processing application in MATLAB, designed to accurately decode predefined color matrices from images. The application is built through a sequence of interconnected functions, each tailored to handle specific aspects of the image processing task:

### **Function 1:** loadImage(imagePath)

**Purpose:** This function initializes the image processing pipeline by loading an image from disk into MATLAB. It prepares the image for further processing by normalizing its pixel values.

**Process:** The image is loaded using MATLAB's imread, and the pixel data type is converted from an integer-based format (typically uint8) to double precision using im2double. This normalization scales the pixel values to a range of 0 to 1, standardizing them for subsequent operations.

#### **Function 2: findCircles(image)**

**Purpose:** To detect four specific black circles within the image, which are used as reference points for accurately orienting and scaling the image in the processing steps that follow.

**Process:** The image is converted to grayscale to simplify the data using `rgb2gray`, then smoothed with Gaussian blur via `imgaussfilt` to reduce noise. Edge detection is performed using the Canny method to outline potential circles. Morphological operations such as `imclose` and `imfill` are then applied to enhance the visibility of these circles. The function finally isolates these features based on their size and shape using `regionprops`.

#### **Function 3: correctImage(circleLocations, destinationPoints, image)**

**Purpose:** This function aligns the detected reference circles to predefined positions, ensuring the image is normalized for consistent analysis across different samples.

**Process:** Utilizes the coordinates of the detected circles and predefined destination points to compute a projective transformation matrix with `fitgeotrans`. The transformation is applied using `imwarp`, adjusting the image so the circles are correctly positioned. This step ensures that subsequent color analysis is performed on a consistently oriented image.

#### **Function 4: getColors(image)**

**Purpose:** To analyze the corrected image and accurately determine the dominant color in each section of a predefined grid. The output is a structured matrix of color patterns, crucial for applications requiring detailed color interpretation.

**Process:** Converts the image from RGB to Lab color space using `makecform` and `applycform` to enhance color differentiation. The image is then segmented into a grid, where each block is analyzed to compute its average color values. These values are compared against specific thresholds in the Lab space to classify the dominant color in each block.

#### **Function 5: findColors(imagePath)**

**Purpose:** Serves as the main orchestrator of the image processing workflow, ensuring seamless integration and operation of all components from image loading to color pattern output.

**Process:** Begins with loading the image using `loadImage`, followed by using `findCircles` to detect orientation markers necessary for geometric corrections by `correctImage`. Once the image is correctly oriented and scaled, `getColors` is applied to classify and format the color data into a structured matrix. This matrix, representing the color configuration, is the final output of the process.

### **3-) Results and Performance**

The system was tested across a variety of images, each containing a distinct color pattern. The overall success rate of the algorithm in recognizing and classifying the color patterns was 100%, indicating the robustness of the implemented methods under various conditions.

#### **4-) Discussion**

While the current implementation successfully achieved a 100% success rate, continuous improvement is essential for maintaining robustness in new scenarios. Potential enhancements could include:

**Enhanced Color Differentiation:** Implementing more sophisticated algorithms for color recognition could prepare the system for even more complex images.

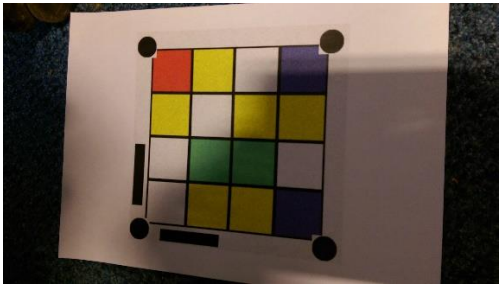
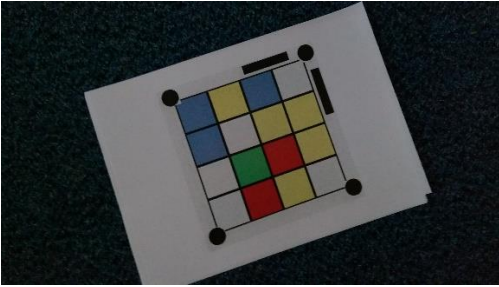
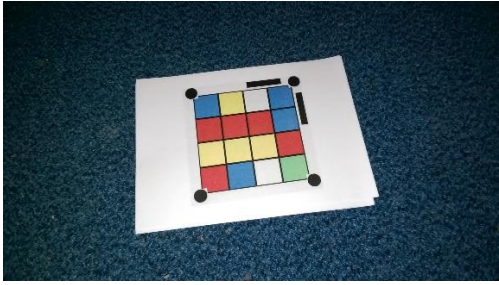
**Noise Reduction:** Further preprocessing to reduce image noise could enhance edge and color detection accuracy.

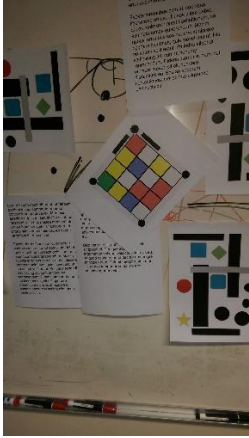
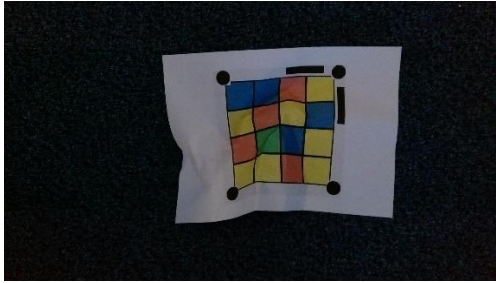
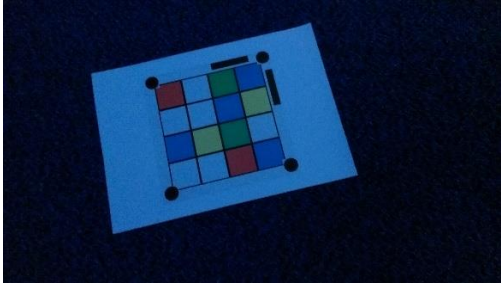
**Adaptive Thresholding:** Utilizing adaptive methods for edge detection could cater to varying lighting conditions across different images.


#### **5-) Comments on Real Photographs**

As part of the image processing project, a set of real photographs was provided for analysis. These images were not processed by the algorithm; instead, they were reviewed to understand the potential challenges they could present for automated image processing systems. Below are the comments on each photograph:

File Name	Image	Comment
IMAG0032.jpg		The image exhibits a minor skew which might impact the detection of reference circles. Good lighting is noted, but reflections on some blocks could affect the color recognition accuracy.
IMAG0033.jpg		Significant skew and perspective distortion are observed, which would necessitate correction before further processing. Additionally, shadows in the image may pose challenges in segmenting the colors properly.
IMAG0034.jpg		This photograph is taken from an angle, introducing perspective distortion and uneven lighting. Both factors would need to be addressed to ensure accurate color segmentation.

IMAG0035.jpg		<p>Despite the uniform lighting, the image's rotation and perspective distortion could hinder standard processing techniques and require corrective preprocessing.</p>
IMAG0036.jpg		<p>Uniform lighting is a positive aspect of this image; however, low contrast and shadows might interfere with color classification.</p>
IMAG0037.jpg		<p>Overexposure and reflections, along with image rotation, are the main concerns for this photograph, which could compromise the color accuracy.</p>

IMAG0038.jpg		<p>The image presents noticeable shadows and perspective warp that would require advanced correction methods to prepare for analysis.</p>
IMAG0041.jpg		<p>The image is mostly well-oriented but would benefit from adjustments for slight perspective distortions. Variations in the background could be mistaken for part of the color pattern.</p>
IMAG0042.jpg		<p>Clear perspective distortion is visible and must be addressed before processing. The high variance in lighting conditions presents additional challenges for color recognition algorithms.</p>

IMAG0044.jpg		This photograph has optimal lighting and minimal shadows. However, the paper's curvature introduces a challenge that would need to be addressed with a flattening transformation pre-processing step.
--------------	---	---

## 6-) Conclusion

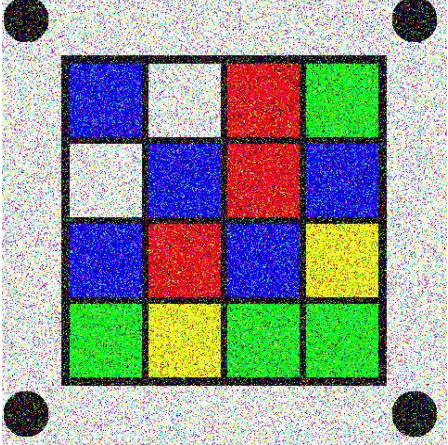
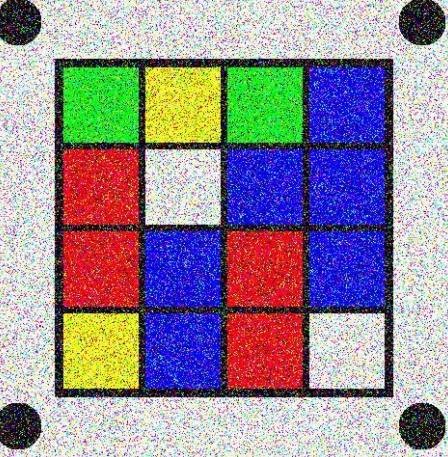
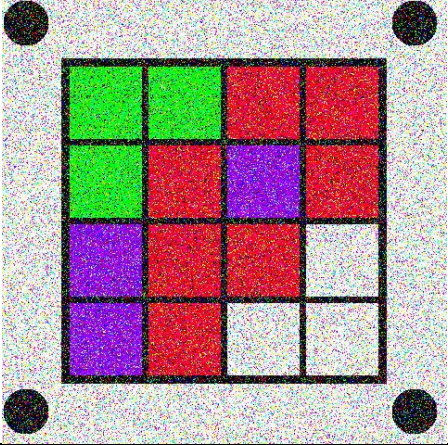
The project successfully demonstrates the potential of using automated image processing techniques for recognizing and interpreting color patterns in images. The implementation showcased not only high accuracy in color detection but also highlighted the scalability and adaptability of the developed MATLAB functions to different image conditions.

## 7-) Appendix

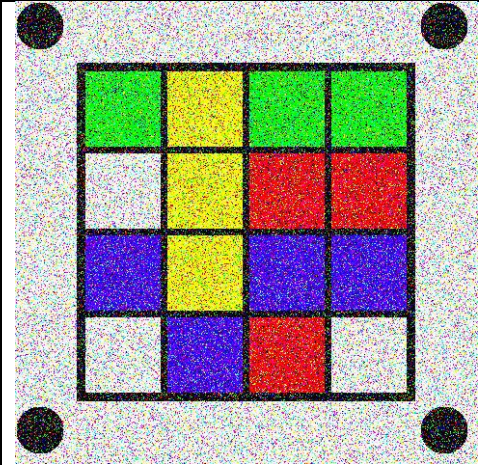
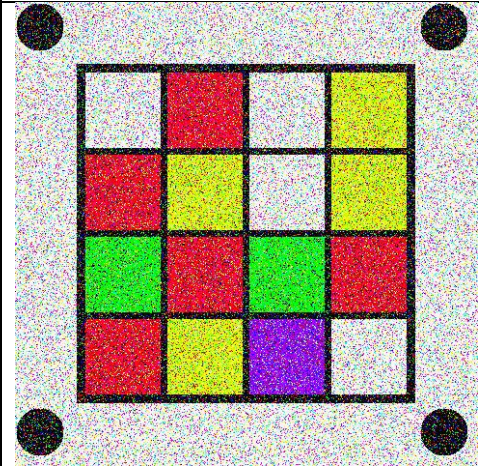
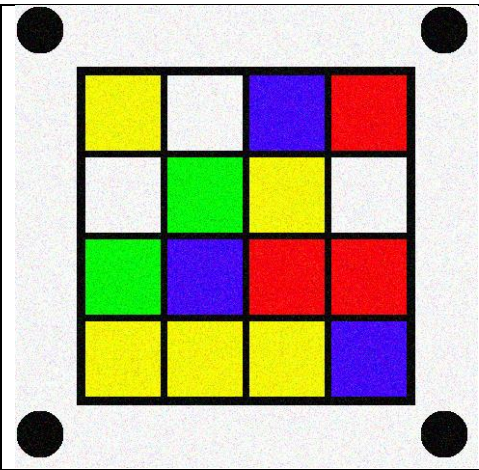
### 7.1-) Results Tables

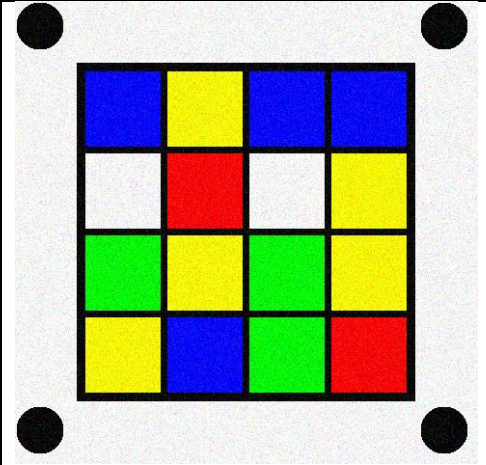
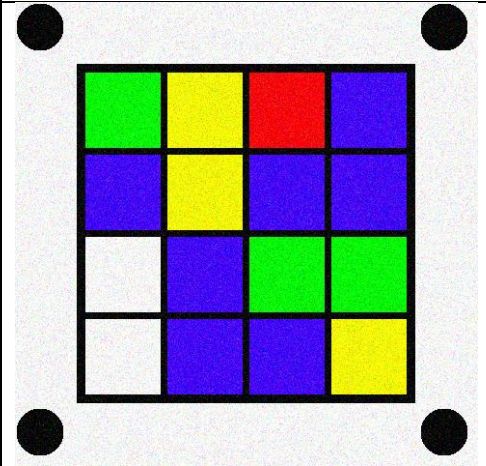
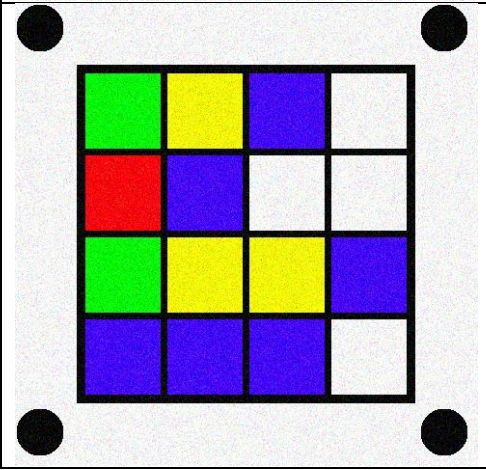
The results table below summarizes the outcomes of the image processing functions when applied to a series of test images. Each entry includes the filename, the corresponding output matrix indicating the recognized color patterns, and any additional notes relevant to that particular test case.



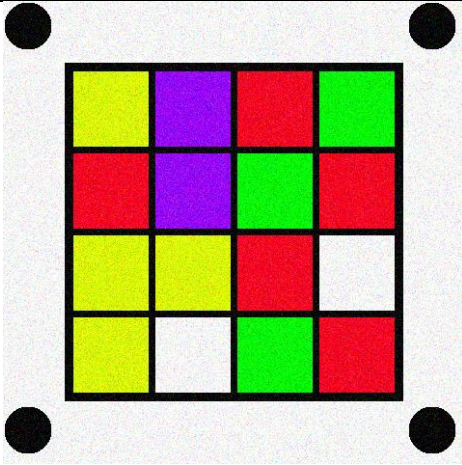
File Name	Image	Output	Success Rate
noise_1.png		<div><div>"blue"</div><div>"white"</div><div>"red"</div><div>"green"</div></div> <div><div>"white"</div><div>"blue"</div><div>"red"</div><div>"blue"</div></div> <div><div>"blue"</div><div>"red"</div><div>"blue"</div><div>"yellow"</div></div> <div><div>"green"</div><div>"yellow"</div><div>"green"</div><div>"green"</div></div>	100%
noise_2.png		<div><div>"green"</div><div>"yellow"</div><div>"green"</div><div>"blue"</div></div> <div><div>"red"</div><div>"white"</div><div>"blue"</div><div>"blue"</div></div> <div><div>"red"</div><div>"blue"</div><div>"red"</div><div>"blue"</div></div> <div><div>"yellow"</div><div>"blue"</div><div>"red"</div><div>"white"</div></div>	100%
noise_3.png		<div><div>"green"</div><div>"green"</div><div>"red"</div><div>"red"</div></div> <div><div>"green"</div><div>"red"</div><div>"purple"</div><div>"red"</div></div> <div><div>"purple"</div><div>"red"</div><div>"red"</div><div>"white"</div></div> <div><div>"purple"</div><div>"red"</div><div>"white"</div><div>"white"</div></div>	100%



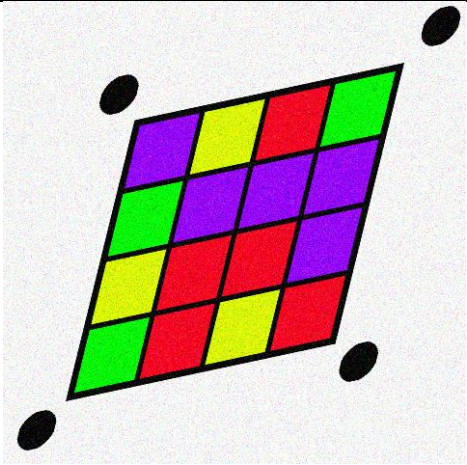
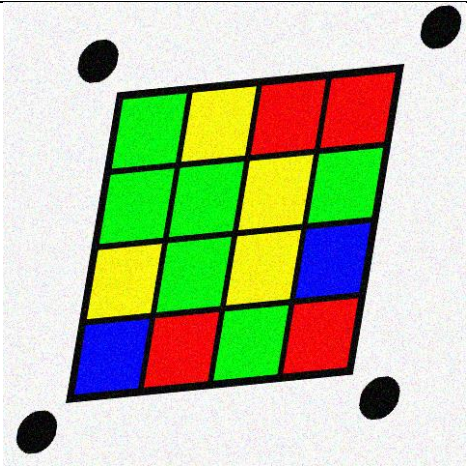
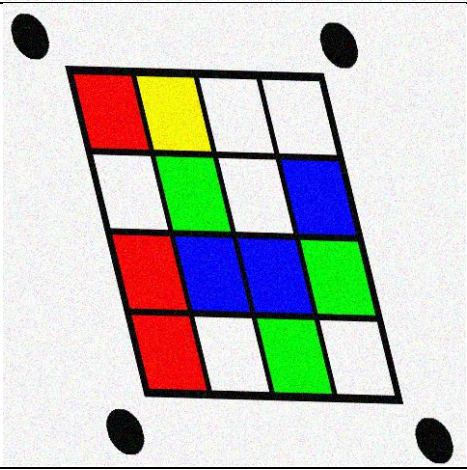
noise_4.png		<div><div>"green"</div><div>"white"</div><div>"blue"</div><div>"white"</div></div> <div><div>"yellow"</div><div>"yellow"</div><div>"yellow"</div><div>"blue"</div></div> <div><div>"green"</div><div>"red"</div><div>"blue"</div><div>"red"</div></div> <div><div>"green"</div><div>"red"</div><div>"blue"</div><div>"white"</div></div>	100%
noise_5.png		<div><div>"white"</div><div>"red"</div><div>"white"</div><div>"yellow"</div></div> <div><div>"red"</div><div>"yellow"</div><div>"white"</div><div>"yellow"</div></div> <div><div>"green"</div><div>"red"</div><div>"green"</div><div>"red"</div></div> <div><div>"red"</div><div>"yellow"</div><div>"purple"</div><div>"white"</div></div>	100%
org_1.png		<div><div>"yellow"</div><div>"white"</div><div>"blue"</div><div>"red"</div></div> <div><div>"white"</div><div>"green"</div><div>"yellow"</div><div>"white"</div></div> <div><div>"green"</div><div>"blue"</div><div>"red"</div><div>"red"</div></div> <div><div>"yellow"</div><div>"yellow"</div><div>"yellow"</div><div>"blue"</div></div>	100%

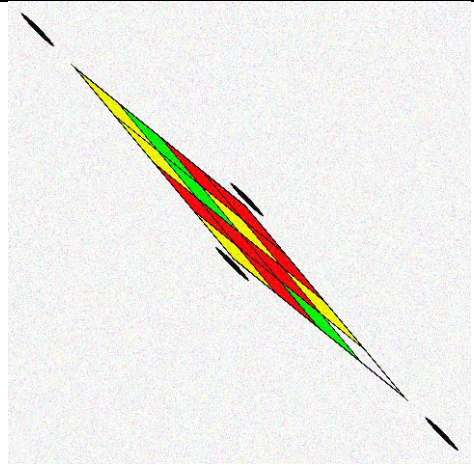
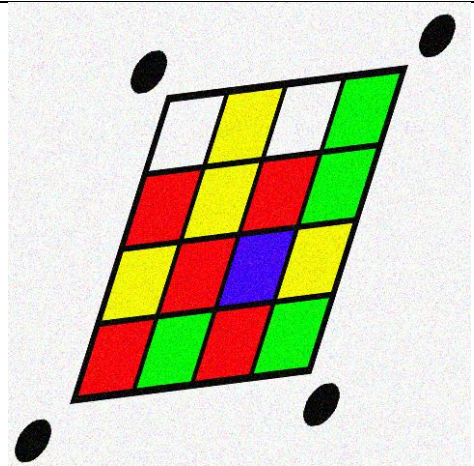
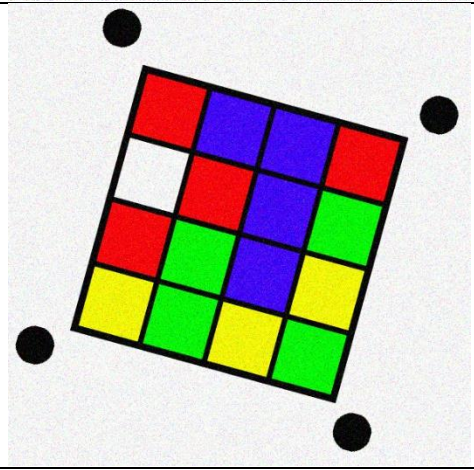
org_2.png		<div><div>"blue"</div><div>"white"</div><div>"green"</div><div>"yellow"</div></div> <div><div>"yellow"</div><div>"red"</div><div>"yellow"</div><div>"blue"</div></div> <div><div>"blue"</div><div>"white"</div><div>"green"</div><div>"green"</div></div> <div><div>"blue"</div><div>"yellow"</div><div>"yellow"</div><div>"red"</div></div>	100%
org_3.png		<div><div>"green"</div><div>"blue"</div><div>"white"</div><div>"white"</div></div> <div><div>"yellow"</div><div>"yellow"</div><div>"blue"</div><div>"blue"</div></div> <div><div>"red"</div><div>"blue"</div><div>"green"</div><div>"blue"</div></div> <div><div>"blue"</div><div>"blue"</div><div>"green"</div><div>"yellow"</div></div>	100%
org_4.png		<div><div>"green"</div><div>"red"</div><div>"green"</div><div>"blue"</div></div> <div><div>"yellow"</div><div>"blue"</div><div>"yellow"</div><div>"blue"</div></div> <div><div>"blue"</div><div>"white"</div><div>"yellow"</div><div>"blue"</div></div> <div><div>"white"</div><div>"white"</div><div>"blue"</div><div>"white"</div></div>	100%



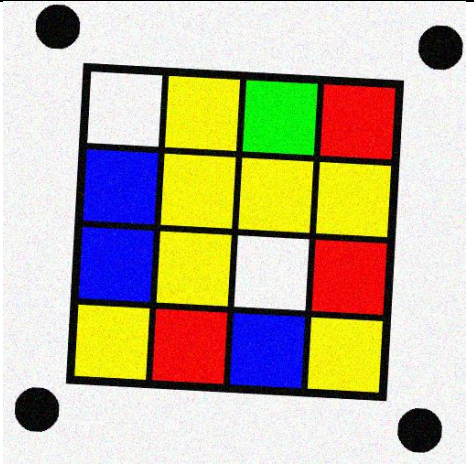
org_5.png		<div><div>"yellow"</div><div>"red"</div><div>"yellow"</div><div>"yellow"</div></div> <div><div>"purple"</div><div>"purple"</div><div>"yellow"</div><div>"white"</div></div> <div><div>"red"</div><div>"green"</div><div>"red"</div><div>"green"</div></div> <div><div>"green"</div><div>"red"</div><div>"white"</div><div>"red"</div></div>
-----------	---	---

100%

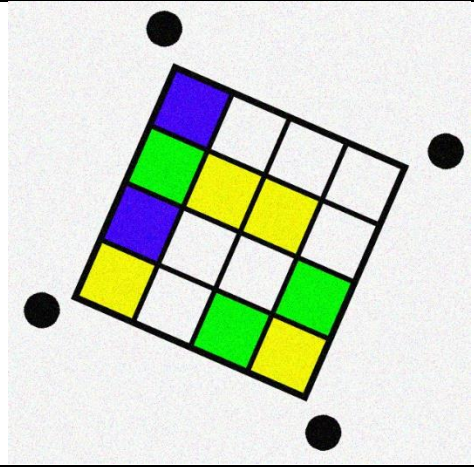
proj_3.png		<div><div>"purple"</div><div>"green"</div><div>"yellow"</div><div>"green"</div></div> <div><div>"yellow"</div><div>"purple"</div><div>"red"</div><div>"red"</div></div> <div><div>"red"</div><div>"purple"</div><div>"red"</div><div>"yellow"</div></div> <div><div>"green"</div><div>"purple"</div><div>"purple"</div><div>"red"</div></div>	100%
proj_4.png		<div><div>"green"</div><div>"green"</div><div>"yellow"</div><div>"blue"</div></div> <div><div>"yellow"</div><div>"green"</div><div>"green"</div><div>"red"</div></div> <div><div>"red"</div><div>"yellow"</div><div>"yellow"</div><div>"green"</div></div> <div><div>"red"</div><div>"green"</div><div>"blue"</div><div>"red"</div></div>	100%
proj_5.png		<div><div>"red"</div><div>"white"</div><div>"red"</div><div>"red"</div></div> <div><div>"yellow"</div><div>"green"</div><div>"blue"</div><div>"white"</div></div> <div><div>"white"</div><div>"white"</div><div>"blue"</div><div>"green"</div></div> <div><div>"white"</div><div>"blue"</div><div>"green"</div><div>"white"</div></div>	100%

proj_6.png		<table><tr><td>"yellow"</td><td>"green"</td><td>"red"</td><td>"red"</td></tr><tr><td>"yellow"</td><td>"green"</td><td>"yellow"</td><td>"red"</td></tr><tr><td>"red"</td><td>"red"</td><td>"red"</td><td>"yellow"</td></tr><tr><td>"yellow"</td><td>"red"</td><td>"green"</td><td>"white"</td></tr></table>	"yellow"	"green"	"red"	"red"	"yellow"	"green"	"yellow"	"red"	"red"	"red"	"red"	"yellow"	"yellow"	"red"	"green"	"white"	100%
"yellow"	"green"	"red"	"red"																
"yellow"	"green"	"yellow"	"red"																
"red"	"red"	"red"	"yellow"																
"yellow"	"red"	"green"	"white"																
proj_7.png		<table><tr><td>"white"</td><td>"yellow"</td><td>"white"</td><td>"green"</td></tr><tr><td>"red"</td><td>"yellow"</td><td>"red"</td><td>"green"</td></tr><tr><td>"yellow"</td><td>"red"</td><td>"blue"</td><td>"yellow"</td></tr><tr><td>"red"</td><td>"green"</td><td>"red"</td><td>"green"</td></tr></table>	"white"	"yellow"	"white"	"green"	"red"	"yellow"	"red"	"green"	"yellow"	"red"	"blue"	"yellow"	"red"	"green"	"red"	"green"	100%
"white"	"yellow"	"white"	"green"																
"red"	"yellow"	"red"	"green"																
"yellow"	"red"	"blue"	"yellow"																
"red"	"green"	"red"	"green"																
rot_1.png		<table><tr><td>"red"</td><td>"blue"</td><td>"blue"</td><td>"red"</td></tr><tr><td>"white"</td><td>"red"</td><td>"blue"</td><td>"green"</td></tr><tr><td>"red"</td><td>"green"</td><td>"blue"</td><td>"yellow"</td></tr><tr><td>"yellow"</td><td>"green"</td><td>"yellow"</td><td>"green"</td></tr></table>	"red"	"blue"	"blue"	"red"	"white"	"red"	"blue"	"green"	"red"	"green"	"blue"	"yellow"	"yellow"	"green"	"yellow"	"green"	100%
"red"	"blue"	"blue"	"red"																
"white"	"red"	"blue"	"green"																
"red"	"green"	"blue"	"yellow"																
"yellow"	"green"	"yellow"	"green"																



rot_2.png		<div><div>"white"</div><div>"blue"</div><div>"blue"</div><div>"yellow"</div></div> <div><div>"yellow"</div><div>"yellow"</div><div>"yellow"</div><div>"red"</div></div> <div><div>"green"</div><div>"yellow"</div><div>"white"</div><div>"blue"</div></div> <div><div>"red"</div><div>"yellow"</div><div>"red"</div><div>"yellow"</div></div>
-----------	---	---



rot_5.png		<table><tr><td>"blue"</td><td>"white"</td><td>"white"</td><td>"white"</td></tr><tr><td>"green"</td><td>"yellow"</td><td>"yellow"</td><td>"white"</td></tr><tr><td>"blue"</td><td>"white"</td><td>"white"</td><td>"green"</td></tr><tr><td>"yellow"</td><td>"white"</td><td>"green"</td><td>"yellow"</td></tr></table>	"blue"	"white"	"white"	"white"	"green"	"yellow"	"yellow"	"white"	"blue"	"white"	"white"	"green"	"yellow"	"white"	"green"	"yellow"	100%
"blue"	"white"	"white"	"white"																
"green"	"yellow"	"yellow"	"white"																
"blue"	"white"	"white"	"green"																
"yellow"	"white"	"green"	"yellow"																

7.2-) Code Listings

The code for this project is divided into several functions, each contained within its own MATLAB script file (.m file). The code listings below provide an overview of each function, along with comments explaining the purpose and process carried out by the code.

Function 1:

```
function image = loadImage(imagePath)

    % Loads an image from the specified file path and normalizes its pixel values.

    % [Process details...]

end
```

## **Function 2:**

```
function [circleLocations, destinationPoints] = findCircles(image)

    % Identifies the positions of four black circles in the image for orientation.

    % [Process details...]

end
```

## **Function 3:**

```
function correctedImage = correctImage(circleLocations, destinationPoints, image)

    % Applies a projective transformation to the image based on the circle locations.

    % [Process details...]

end
```

## **Function 4:**

```
function colors = getColors(image)

    % Analyzes the corrected image to extract the color pattern.

    % [Process details...]

end
```

#### **Function 5:**

```
function colors = findColors(imagePath)

    % Orchestrates the process from image loading to outputting the color pattern.

    % [Process details...]

end
```

**Note:** For brevity, only the function headers and brief comments are included here. The complete, detailed commented code is provided in the accompanying .m files.