UNIVERSITY OF SUSSEX	**Candidate Number**	237707
	Module	521H3
	Word Count	1182

Image Processing Lab Project

Table of Content

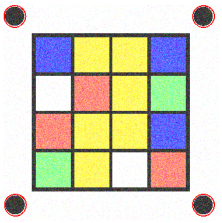
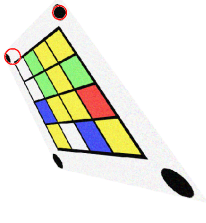
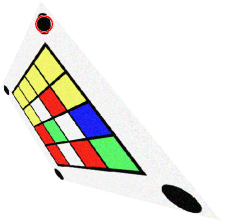
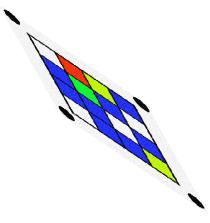
Report

Introduction

The purpose of this project is developing a software that is able to effectively identify colours from 16 blocks of squares (arranged 4x4). Each square is filled with a random choice of the following colours including white, red, green, blue or yellow. The sample experimented upon includes 30 images with 10 undistorted images as well as 20 distorted images. Each image in the sample includes 4 circles located in the corners of the 4x4 blocks. In this respect, the task requires locating the coordinates of the undistorted images as a reference point. Subsequently, using the coordinates that have been located, distorted images must be adjusted/rotated to remove the distortion before processing the images using various techniques to remove noise and extract features. Finally, having processed our images, the task requires the software to give an output of a 4x4 array of strings identifying the colours in the images from the sample.

Detection of Circles

Considering that we are dealing with distorted images, the Hough Transform method which is well regarded for dealing with imperfect images was selected to achieve this task. This method was implemented using the `imfindcircles()` function which returns a two column matrix with the x and y coordinates of the centres of each circle detected in the image. Radius ranging between 18 and 27 were implemented as a key parameter effectively enabling this method to detect circles which came in various sizes within the sample of images provided. The object polarity was set to "dark" effectively enabling the identification of the circles which were all coloured in black. The sensitivity was also increased from default value of 0.85 to 0.93 effective allowing this method to better detect weaker and partially obscure circles due to distortion in some images.

Image				
Results	All 4 circles detected on undistorted images ("noise_1.png").	2 circles detected on distorted images ("proj_1.png").	1 circles detected on image that sees further distortion ("proj_2.png").	No circles detected on highly distorted image ("proj1_3.png").

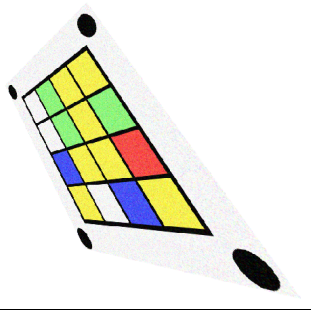
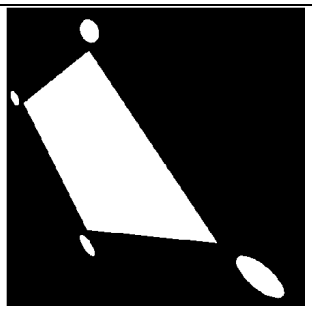
The function effectively detected all circles on images that saw very little or no distortion at all. However, when faced with greater distortion, a noticeable impact can be observed in the shape and size of the circles. In this respect, images with greater distortion in the sample saw fewer circles or no circles detected.

Files	*Noise_X.png*	*Org_X.png*	*Proj_X.png*	*Proj1_X.png*	*Proj2_X*	*Rot_X.png*
Success	5/5 → 100%	5/5 → 100%	0/5 → 0%	3/5 → 60%	4/5 → 80%	5/5 → 100%
Average	100%	100%	25%	60%	80%	100%

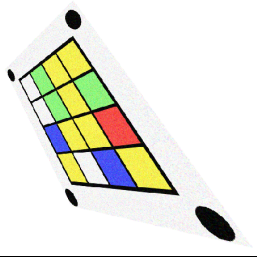
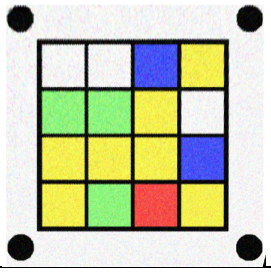
Overall circles were detected in 22 of the 30 images in the sample. The worst results were witnessed in “proj_X.png” where images were highly distorted and an average of 1 circle was detect in each of the 5 images that were processed. Likewise, “proj1_X.png” and “proj2_X.png” also witnessed complication to a lesser extent on images which were highly distorted.

Correction of Images

In order to undistort the distorted images in the sample, “org_1.png” was used as reference point. Firstly, the input image was converted to black and white and elements (16 blocks and 4 circles) were filled in white using **`imfill`**`(img, "holes")` function before being denoised using a gaussian filter (check table below).

Before	After
	

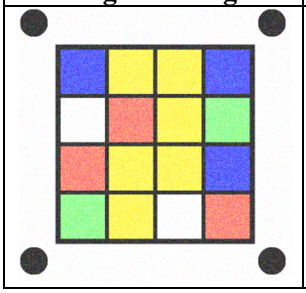
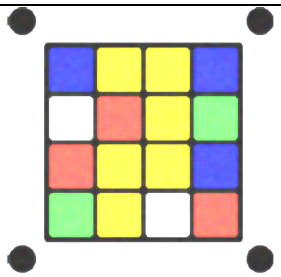
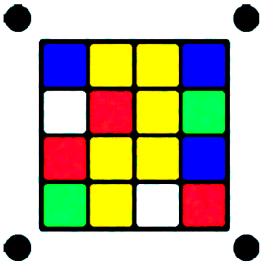
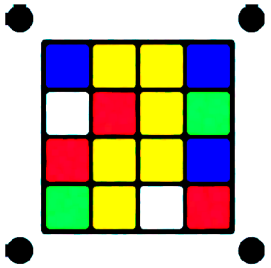
Subsequently, the **`bwconncomp`**`(img)` function locates 5 connected elements in white and the **`regionprops`**`(img, "Area", "Centroid")` identifies centroids as well as the areas of the corresponding components. Thereafter, a 480x480 array of zeros is created and filled with ones in regions where the squares representing 4x4 blocks are located effectively isolating remaining elements and identifying centroids as well as the areas in the isolated image. Using the **`imref2d`**`(size(findCircles(reference_image)))` function, the (x, y) coordinates from “org_1.png” are adjusted and the **`fitgeotrans`**`()` function is used to identify the required transformation for the image. Finally, the input image is undistorted using the **`imwrap`**`()` function which undistorts the input image and returns the output (see example below).

Input	Final Output
	

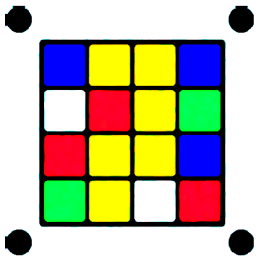
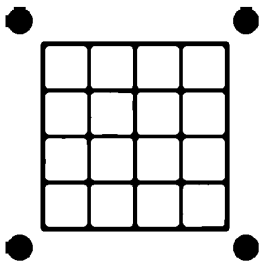
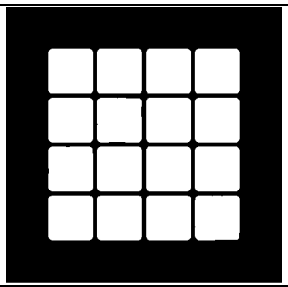
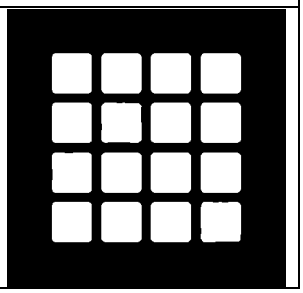
In some cases, the orientation of the output image did not match with input image as shown above. Nevertheless, this process if fully automated and accurately displays an output of undistorted image on all images in the sample with all colours displayed in the undistorted output.

Colour Detection

As a starting point the images in the sample must be denoised to remove unwanted specs and to better recognise the colours in each pixel. A median filter was selected for this task using the **medfilt3**`(img, [11 11 1])` function where each value in the m by n by p neighbourhood around the voxel in the image is padded accordingly. Thereafter, the contrast was increased using the **imadjust**`(img, stretchlim(img, 0.28))` function to better recognize block colours with low contrast. The following output presented an image that is slightly blurred with edges impacted. As a result, erosion and dilation was implemented to further denoise the images and reduce the impact on the edges using **imerode**`(img, strel("square", 9)` and **imdilate**`(img, strel("square", 9)` functions (as shown below).

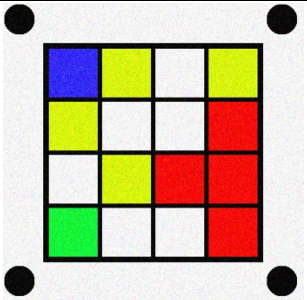
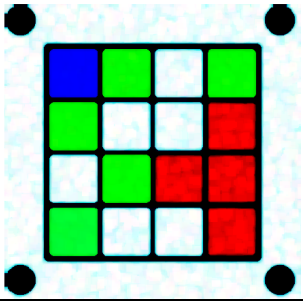
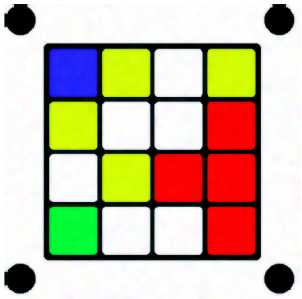
Original Image	Median Filter	Increased Contrast	Erosion/Dilation
			

Having completed the pre-processing on the images, further steps were undertaken converting the images to grayscale as well as thresholding using **rgb2gray**`(denoised_img) > 0.08` script. Thereafter, the outer borders surrounding the blocks are removed using the **imclearborder**`(img)` function in order to focus on the blocks and erosion was once again performed to prevent any effect on the edges of each block. This presents an output of black and white 4x4 blocks (as shown below).

Denoised Image	Grayscaling and thresholding	Clearing borders	Erosion
			

Using the output displayed above, segmentation is performed on the images to identify the regions with the square blocks using the **bwlabel**`(img)` function and a 16x3 array of zeros is created to store the average colour of each square block. Subsequently, the centroids of each block is identified using **regionprops**`(img, "centroid")` function and the index of each centroid colours are identified by finding the closest match in based on predefined colour codes and rearranged based on the correct order. Finally, using indexing consistent with array storing the colour code is used to identify the matching strings and return the final output as an array which is reshaped to form a 4x4 matrix.

Results and Discussions

Org_1.png		
Original Image	**Pixel Saturation = 0.27**	**Pixel Saturation = 0.1**
		
N/A	{'B'} {'G'} {'W'} {'G'} {'G'} {'W'} {'W'} {'R'} {'W'} {'G'} {'R'} {'R'} {'G'} {'W'} {'W'} {'R'}	{'B'} {'Y'} {'W'} {'Y'} {'Y'} {'W'} {'W'} {'R'} {'W'} {'Y'} {'R'} {'R'} {'G'} {'W'} {'W'} {'R'}

Overall 28 out of the 30 images in the sample returned the correct array storing to colours of each square block as a 4x4 array of string. However, the yellow blocks in "rot_3.png" and "org_1.png" were identified as green. As illustrated above, when the pixel saturation parameter used to change the contrast is set to 0.1, we were able to return the correct output. In this respect, the program had a trade-off to maximize the number of images correctly recognized which resulted in the misclassification of block that appear to look borderline green/yellow (even through manual observation of the human eye). Therefore, the program could have exploited further experimentations to determine the optimal parameter for the contrasts to circumvent the problem. Nevertheless, the program should typically be able to identify what would typically be seen with certainty through manual observation as demonstrated with the rest of the 28 images in the sample.

Code

Load Image

```
% Converts image to double precison type
function [img] = loadImage(filename)
img = imread(filename); %read the image file
img = double(img)/255; % Convert to double precision
if isa(img,'uint8') == 1 % Error prevention
    error('Ensure image is type unit 8'); % Error message
end
end
```

Find Colours

```
% Function that returns denoised image
function denoised=denoise(img)
filtered = medfilt3(img,[11 11 1]); % Applying median filter to denoise the image
darkened = imadjust(filtered,stretchlim(filtered, 0.27)); % Reduces brighness of the image
metrics = strel('square', 9); % Metrics used as parameter for erosion and dialation
eroded = imerode(darkened,metrics); % Erosion applied to cancel the blurred image
denoised = imdilate(eroded,metrics); % Dialtion applied for final touches
end
```

```
% Function returns image block in black an white
function transformed=transform(denoised)
grayscaled= rgb2gray(denoised) > 0.08; % converting the image into greyscale
removeBorders = imclearborder(grayscaled); % Clear image borders to focus on the square blocks
transformed = imerode(removeBorders, ones(10)); % Erosion applied to mask edges that may be
altered
%imshow([grayscaled, transformed])
end
```

```
% Function returns 4x4 array of strings representing color blocks
function colourMatrix=findColours(img)
denoised = denoise(img); % Returns denoised image
transformed = transform(denoised); % Returns image block in black an white
colournames = {'W','R','G','B','Y'}; % Specify colour names
colorcodes = [1 1 1; 1 0 0; 0 1 0; 0 0 1; 1 1 0]; % Specify colour code
[IMB, totalBlocks] = bwlabel(transformed); % segmenting the image regions with the square blocks

% getting average color in each image mask region
averageColours = zeros(totalBlocks, 3); % Creating an empty array of three zero for each square
block detected

for blk = 1:totalBlocks % Looping through the detected blocks (16 blocks )
    val = IMB == blk; % Validation using boolean (0-1)
    region = denoised(val(:,:,[1 1 1])); % Finding regions
    averageColours(blk,:) = mean(reshape(region,[],3),1); % Applying average color for each
region
end

% Finding the centre of blocks
centers = regionprops(transformed,'centroid'); % Centre of blocks
centroids = vertcat(centers.Centroid); % Merging the central regions
centroidlimits = [min(centroids,[],1); max(centroids,[],1)]; % Finding limits
centroids = round((centroids-centroidlimits(1, :)) ./ range(centroidlimits, 1) * 3 + 1);
% Rearraging the colours
idx = sub2ind([4 4], centroids(:, 2), centroids(:, 1));
averageColours(idx,:) = averageColours;
% Finding colour distances
distance = averageColours - permute(colorcodes, [3 2 1]);
distance = squeeze(sum(distance.^2, 2));
% Finding the index of the best match
[~,idx] = min(distance, [], 2);
% Returning colours as 4x4 matrix
if totalBlocks ~= 16 % Error prevention
    error('Unable to detect all the square blocks!') % Error message
else
    colourMatrix = reshape(colournames(idx), 4, 4);
end
end
```

Find Circles

```
% Functiomn identifying the circles in the images and marking it in red
function [centpt, R] = findCircles(image)
```

```
%imshow(image)
[centpt, R] =  imfindcircles(image,[15
25],'ObjectPolarity','dark','Sensitivity',0.93,'Method','twostage'); % Returns coordiates of the
circle on the images
%circle = viscircles(centpt, R) % Circle surfaced marked in red
end
```

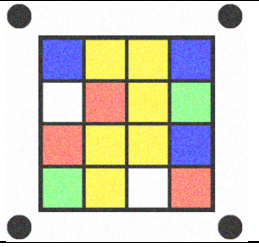
Correct Image

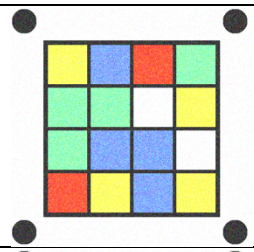
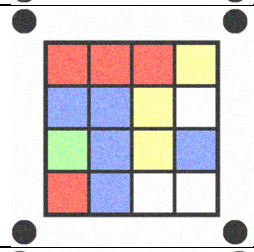
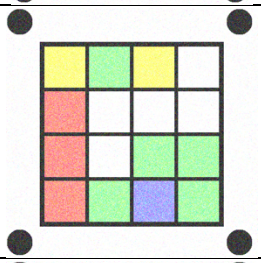
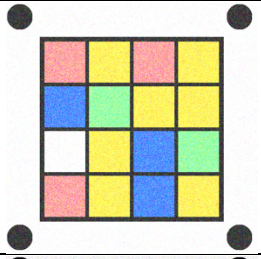
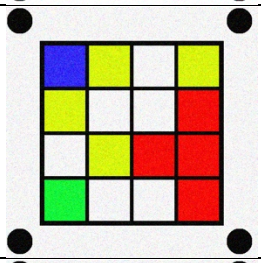
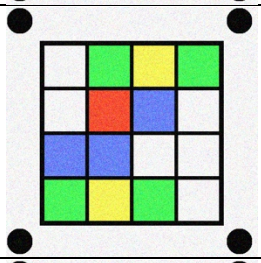
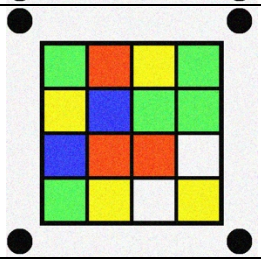
```
% function that adjusts distorted images and rotates based on "org_1.png"
function undistorted= correctImage(img)
refferenceImage = loadImage('org_1.png'); % Using the org_1 image as refference for rotation
bwImg =~ im2bw(img, 0.5); % Converting the image to black and white
imgFilled = imfill(bwImg,'holes'); % Filling elements detected in white
gaussFilt = fspecial('Gaussian',[3 3],2); % Creating a Gaussian type filter.
filtered = imfilter(imgFilled, gaussFilt); % Denoising the image using the Gaussian filter.
elements = bwconncomp(filtered); % Locating connected elements in the image
imgData = regionprops(filtered,'Area','Centroid'); % Finding the area and centroids in the
filtered image.
Blocks = zeros(elements.ImageSize); % Creating an array of zeros with the shape of the square
blocks using size of the image in number of pixels

for B = 1:elements.NumObjects % Filling in area with the square blocks with ones
    if imgData(B).Area<max([imgData.Area])
        Blocks(elements.PixelIdxList{B}) = 1;
    end
end

centroids = [regionprops(bwlabel(Blocks, 8),'Area','Centroid').Centroid]; % Locating centroids
for connected elements
refPoints = transpose(reshape(centroids,2,[])); % Reshaping matrix array
staticPoints = flip(findCircles(refferenceImage)); % Setting the fixed points as centroids of
org_1.png
sizeRef = imref2d(size(refferenceImage)); % Referencing the size of org_1.png
rotate = fitgeotrans(refPoints,staticPoints,'Projective'); % applying the transformation
% Output - Identifying pixel values using reference image and distorted image
undistorted = imwarp(img,rotate,'OutputView',sizeRef);
%imshow(undistorted) % Displaying distorted image and corrected image side by side
end
```

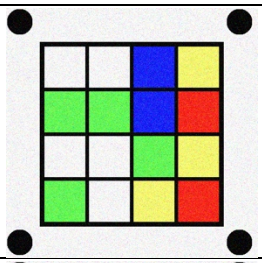
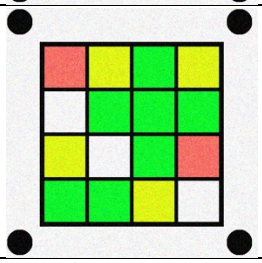
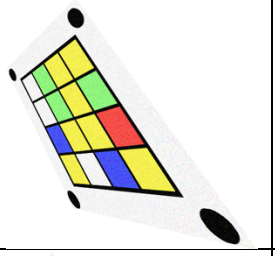
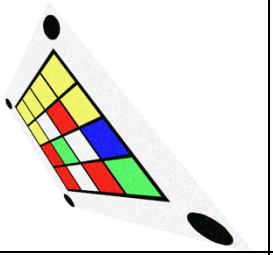
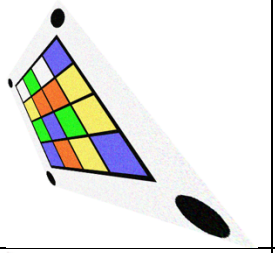
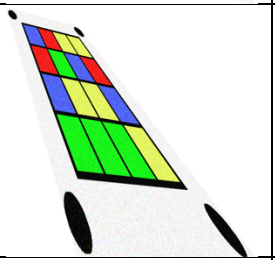
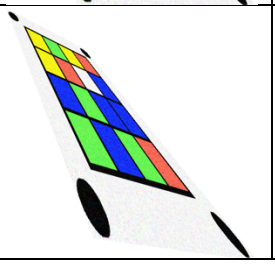
Colour Matrix

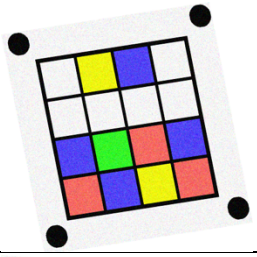
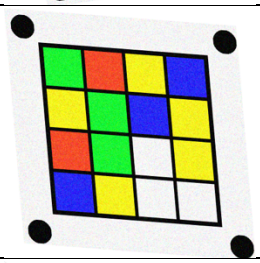
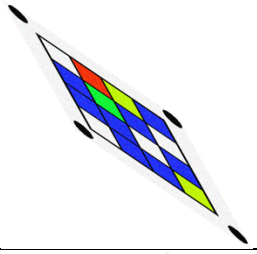
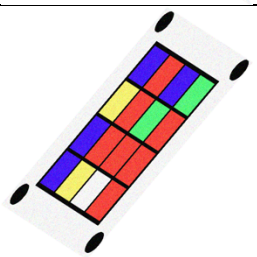
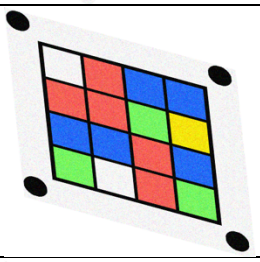
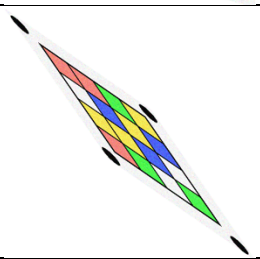
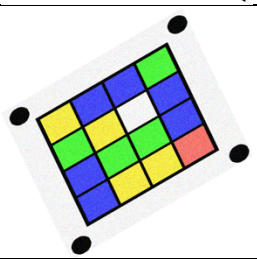
```
% Function undistorts double precision images and return array of string for each colour
function colourArray = colourMatrix(filename)
img = loadImage(filename);
undistortedImage = correctImage(img); % Undistorting the images
imshow(undistortedImage)
colourArray = findColours(undistortedImage); % Output of identified colour as 4x4 array of
strings
end
```

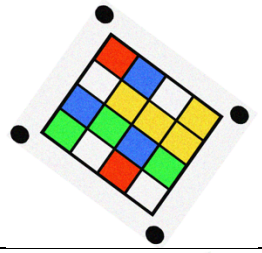
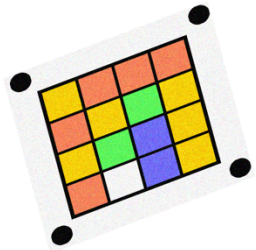
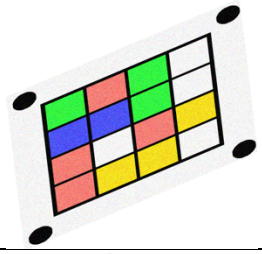
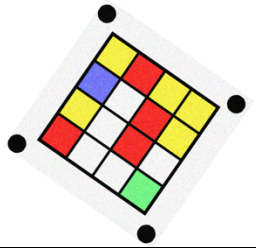
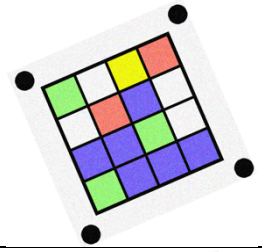
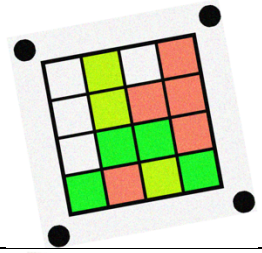
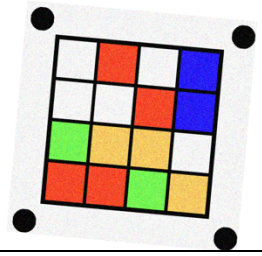
Colour Detection Results

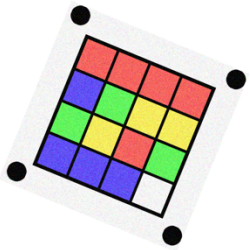
File Name	Image	Output	Success	Notes
noise_1.png		{'B'} {'Y'} {'Y'} {'B'} {'W'} {'R'} {'Y'} {'G'} {'R'} {'Y'} {'Y'} {'B'} {'G'} {'Y'} {'W'} {'R'}	Yes	None

noise_2.png		{'Y'} {'B'} {'R'} {'G'} {'G'} {'G'} {'W'} {'Y'} {'G'} {'B'} {'B'} {'W'} {'R'} {'Y'} {'B'} {'Y'}	Yes	None
noise_3.png		{'R'} {'R'} {'R'} {'Y'} {'B'} {'B'} {'Y'} {'W'} {'G'} {'B'} {'Y'} {'B'} {'R'} {'B'} {'W'} {'W'}	Yes	None
noise_4.png		{'Y'} {'G'} {'Y'} {'W'} {'R'} {'W'} {'W'} {'W'} {'R'} {'W'} {'G'} {'G'} {'R'} {'G'} {'B'} {'G'}	Yes	None
noise_5.png		{'R'} {'Y'} {'R'} {'Y'} {'B'} {'G'} {'Y'} {'Y'} {'W'} {'Y'} {'B'} {'G'} {'R'} {'Y'} {'B'} {'Y'}	Yes	None
org_1.png		{'B'} {'G'} {'W'} {'G'} {'G'} {'W'} {'W'} {'R'} {'W'} {'G'} {'R'} {'R'} {'G'} {'W'} {'W'} {'R'}	No	The yellow blocks appear to look green. So the program was unable to correctly pick its colour.
org_2.png		{'W'} {'G'} {'Y'} {'G'} {'W'} {'R'} {'B'} {'W'} {'B'} {'B'} {'W'} {'W'} {'G'} {'Y'} {'G'} {'W'}	Yes	None
org_3.png		{'G'} {'R'} {'Y'} {'G'} {'Y'} {'B'} {'G'} {'G'} {'B'} {'R'} {'R'} {'W'} {'G'} {'Y'} {'W'} {'Y'}	Yes	None

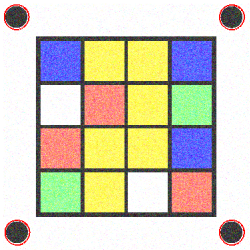
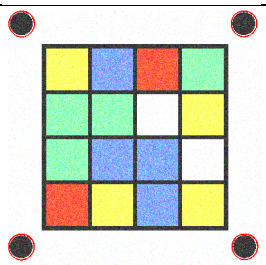
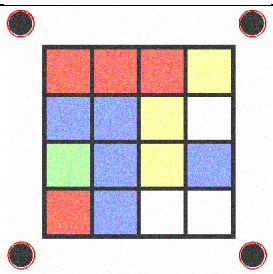
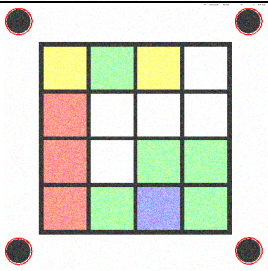
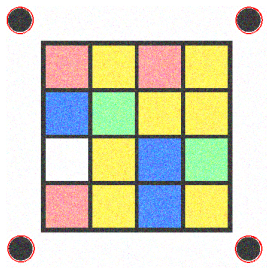
org_4.png		{'W'} {'W'} {'B'} {'Y'} {'G'} {'G'} {'B'} {'R'} {'W'} {'W'} {'G'} {'Y'} {'G'} {'W'} {'Y'} {'R'}	Yes	None
org_5.png		{'R'} {'Y'} {'G'} {'Y'} {'W'} {'G'} {'G'} {'G'} {'Y'} {'W'} {'G'} {'R'} {'G'} {'G'} {'Y'} {'W'}	Yes	None
proj_1.png		{'W'} {'W'} {'B'} {'Y'} {'G'} {'G'} {'Y'} {'W'} {'Y'} {'Y'} {'Y'} {'B'} {'Y'} {'G'} {'R'} {'Y'}	Yes	None
proj_2.png		{'Y'} {'Y'} {'R'} {'R'} {'Y'} {'W'} {'G'} {'W'} {'Y'} {'R'} {'W'} {'R'} {'Y'} {'Y'} {'B'} {'G'}	Yes	None
proj_3.png		{'W'} {'G'} {'W'} {'B'} {'Y'} {'R'} {'R'} {'Y'} {'G'} {'B'} {'G'} {'Y'} {'B'} {'R'} {'Y'} {'B'}	Yes	None
proj_4.png		{'B'} {'R'} {'Y'} {'Y'} {'R'} {'G'} {'B'} {'R'} {'B'} {'Y'} {'Y'} {'B'} {'G'} {'G'} {'G'} {'Y'}	Yes	None
proj_5.png		{'Y'} {'G'} {'Y'} {'R'} {'Y'} {'R'} {'W'} {'B'} {'B'} {'G'} {'B'} {'B'} {'G'} {'B'} {'G'} {'R'}	Yes	None

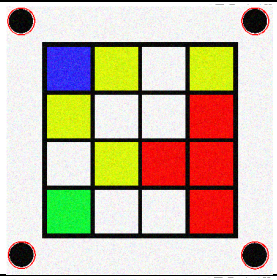
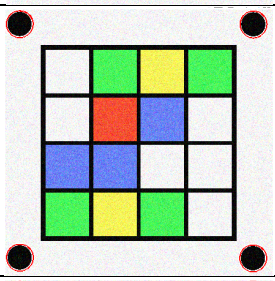
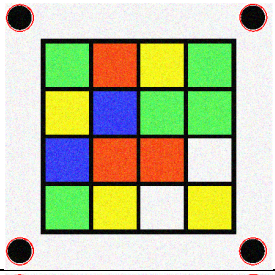
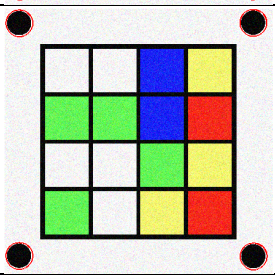
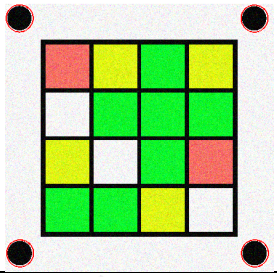
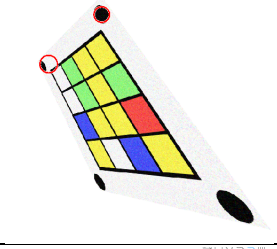
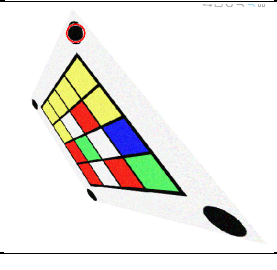
proj1_1.png		{'W'} {'Y'} {'B'} {'W'} {'W'} {'W'} {'W'} {'W'} {'B'} {'G'} {'R'} {'B'} {'R'} {'B'} {'Y'} {'R'}	Yes	None
proj1_2.png		{'G'} {'R'} {'Y'} {'B'} {'Y'} {'G'} {'B'} {'Y'} {'R'} {'G'} {'W'} {'Y'} {'B'} {'Y'} {'W'} {'W'}	Yes	None
proj1_3.png		{'W'} {'R'} {'Y'} {'B'} {'B'} {'G'} {'B'} {'W'} {'B'} {'B'} {'W'} {'B'} {'W'} {'B'} {'B'} {'Y'}	Yes	None
proj1_4.png		{'B'} {'B'} {'Y'} {'B'} {'Y'} {'R'} {'R'} {'R'} {'W'} {'R'} {'G'} {'B'} {'R'} {'R'} {'R'} {'G'}	Yes	None
proj1_5.png		{'W'} {'R'} {'B'} {'B'} {'R'} {'R'} {'G'} {'Y'} {'B'} {'B'} {'R'} {'B'} {'G'} {'W'} {'R'} {'G'}	Yes	None
proj2_1.png		{'R'} {'R'} {'G'} {'Y'} {'W'} {'B'} {'Y'} {'B'} {'Y'} {'Y'} {'B'} {'W'} {'R'} {'G'} {'W'} {'G'}	Yes	None
proj2_2.png		{'Y'} {'B'} {'B'} {'G'} {'G'} {'Y'} {'W'} {'B'} {'B'} {'G'} {'G'} {'B'} {'B'} {'Y'} {'Y'} {'R'}	Yes	None

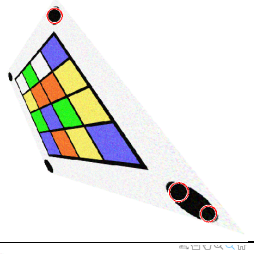
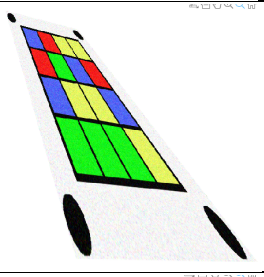
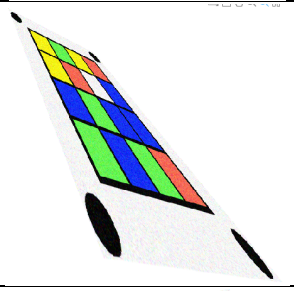
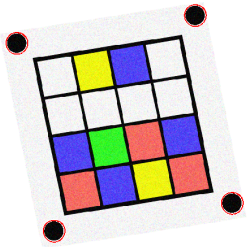
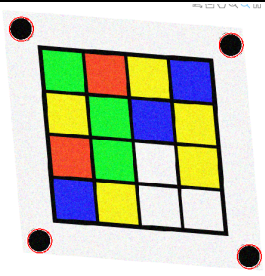
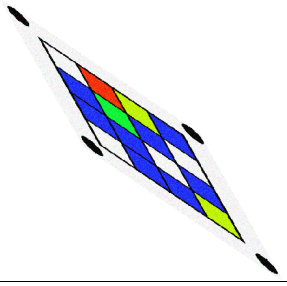
proj2_3.png		{'G'} {'W'} {'R'} {'W'} {'B'} {'G'} {'B'} {'G'} {'W'} {'Y'} {'Y'} {'Y'} {'R'} {'B'} {'W'} {'Y'}	Yes	None
proj2_4.png		{'Y'} {'R'} {'R'} {'R'} {'R'} {'Y'} {'G'} {'Y'} {'Y'} {'G'} {'B'} {'Y'} {'R'} {'W'} {'B'} {'Y'}	Yes	None
proj2_5.png		{'G'} {'R'} {'G'} {'W'} {'B'} {'B'} {'G'} {'W'} {'R'} {'W'} {'R'} {'Y'} {'R'} {'Y'} {'Y'} {'W'}	Yes	None
rot_1.png		{'R'} {'W'} {'W'} {'G'} {'Y'} {'W'} {'R'} {'W'} {'B'} {'W'} {'R'} {'Y'} {'Y'} {'R'} {'Y'} {'Y'}	Yes	None
rot_2.png		{'G'} {'W'} {'Y'} {'R'} {'W'} {'R'} {'B'} {'W'} {'B'} {'B'} {'G'} {'W'} {'G'} {'B'} {'B'} {'B'}	Yes	None
rot_3.png		{'W'} {'G'} {'W'} {'R'} {'W'} {'G'} {'R'} {'R'} {'W'} {'G'} {'G'} {'R'} {'G'} {'R'} {'G'} {'G'}	No	The yellow blocks appear to look green. So the program was unable to correctly pick its colour.
Rot_4.png		{'R'} {'R'} {'G'} {'Y'} {'G'} {'Y'} {'Y'} {'W'} {'W'} {'W'} {'R'} {'B'} {'W'} {'R'} {'W'} {'B'}	Yes	None

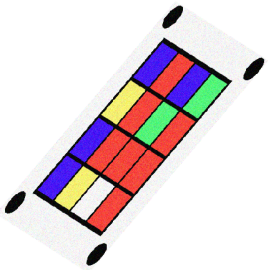
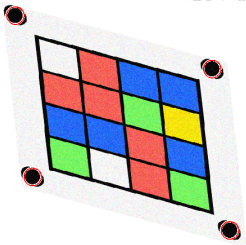
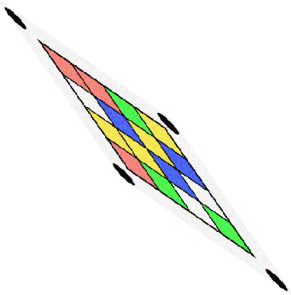
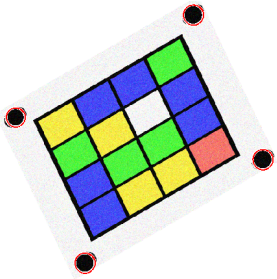
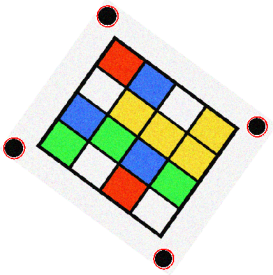
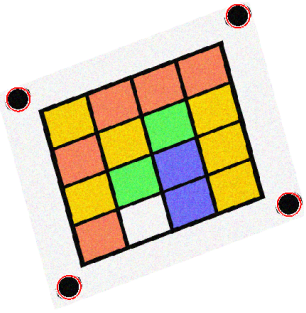
rot_5.png		{'B'} {'B'} {'B'} {'W'} {'G'} {'Y'} {'R'} {'G'} {'B'} {'G'} {'Y'} {'Y'} {'R'} {'R'} {'R'} {'R'}	Yes	None

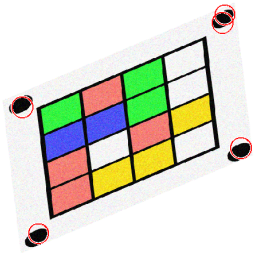
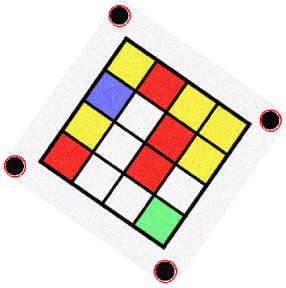
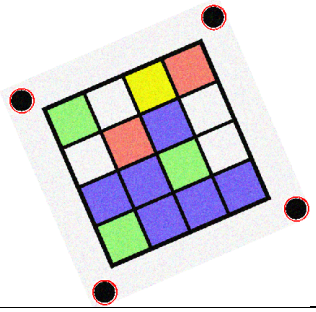
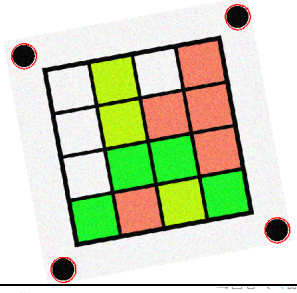
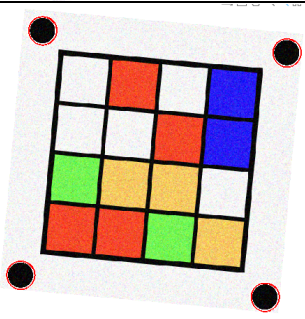
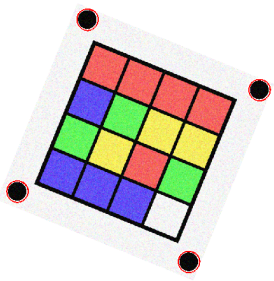
Circle Detection Results

File Name	Circle Detection	Notes
noise_1.png		Success
noise_2.png		Success
noise_3.png		Success
noise_4.png		Success
noise_5.png		Success

org_1.png		Success
org_2.png		Success
org_3.png		Success
org_4.png		Success
org_5.png		Success
proj_1.png		50% detected
proj_2.png		25% detected

proj_3.png		50% detected and picked up one of the circles twice.
proj_4.png		Failed
proj_5.png		Failed
proj1_1.png		Success
proj1_2.png		Success
proj1_3.png		Failed

proj1_4.png		Failed
proj1_5.png		Success
proj2_1.png		Failed
proj2_2.png		Success
proj2_3.png		Success
proj2_4.png		Success

proj2_5.png		Success
rot_1.png		Success
rot_2.png		Success
rot_3.png		Success
rot_4.png		Success
rot_5.png		Success