

1 Lab Project

1.1 Lab Submission

Lego's Life of George is a game where an image is displayed on the screen for a number of seconds before being blanked. The player of the game then has to construct the shape from Lego blocks from memory. The player photographs the finished blocks and their computer automatically compares the brick pattern to the original image. Similar problems are found in many image processing applications, for example the colour segmentation of real world objects; colour data matrix reading. A colour data matrix has the advantage over conventional blank and white versions in that more information can be stored for a given pattern. An example is shown in figure 1.



Figure 1: An example colour data matrix

For your project you will write software that automatically reads a colour pattern image from hard-disc and returns an array representing the colour pattern. An example image is shown in figure 2. Note the location of the black circles. These are for finding the edge of the image should you need to use them. **There is no correct orientation - so your results may be flipped.**

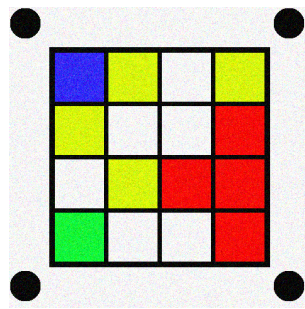


Figure 2: The Project colour matrix example

The images we become progressively more difficult and you may not be able to solve them all. Ideally, the process should be fully automatic (for full marks) but this is not easy so some manual intervention is allowed if you need to. Some images maybe impossible, if you cannot solve them please explain why.

1.2 The image files

There are two sets of images on Canvas

- `images1.zip` and
- `Photos.zip` are a set of real photographs. Do not attempt to solve these, but comment on each one in your report.

1.3 What to do

- Write a function called `findColours(filename)` that given the string `filename`, loads an image and returns an array representing the colours found in the matrix. I would recommend breaking this down in to several steps:
 - `findColours` could call the following functions:
 - A function `image=loadImage(filename)` that given the string `filename`, loads an image and returns the image as type double.
 - A function, `circleCoordinates = findCircles(image)` the locates and returns the coordinates of the four black circles.¹

¹If you don't use the circles you need to adapt this yourself. For example, it could return the intersects of the outside lines that are discovered using a Hough transform.

- A function that correctly un-distorts the images: `correctImage(circleCoordinates, image)`. Note, images may be flipped since there is no correct orientation.
- A function `colours=getColours(image)` that takes the double image array, `image`, and returns an array with the result of the colours. For now this will only use the undistorted images (`org_X.png` and `noise_X.png`) The colours used are red, green, yellow, blue and white.

$$\text{result} = \begin{bmatrix} \textit{green} & \textit{yellow} & \textit{yellow} & \textit{white} \\ \textit{blue} & \textit{white} & \textit{green} & \textit{green} \\ \textit{white} & \textit{green} & \textit{yellow} & \textit{red} \\ \textit{red} & \textit{green} & \textit{white} & \textit{red} \end{bmatrix} \quad (1)$$

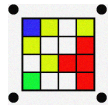
In matlab this is a 4×4 cell array that is holding **char** strings²

Include in your main text what percentage of the images are correctly solved.

- Write in your report the percentage that are correct.
- You may of course write other helper functions to achieve this task, and a main script that calls the `findColours` function.
- In your report, discuss the real images. You do not need to run your program on them, but discuss any problems that might occur.
- In your report you should include a table of results for all of the images. It should have the form:

²See <https://uk.mathworks.com/help/matlab/ref/cell.html>

Table 1: Results Table

Filename	Image	Output	Success	Notes
noise_1.png		$\begin{bmatrix} b & y & w & y \\ y & w & w & r \\ w & y & r & r \\ g & w & w & r \end{bmatrix}$	Yes	None

1.4 Report

- Each of the above functions should have a separate section, describing how they work and discuss and design decisions that you made. The report should include an assessment of the performance of the functions, and suggest improvements that could be made.
- The length of the report should not exceed 1000 words, plus an appendix containing the results tables and a full code with listings of your code. Your code (i.e., the `.m` files or python files) should be commented and also included so that I can run it. The report and the code should be submitted in a single **ZIP** file.
- The report format should be a **PDF** file only. Please **do not** submit Word files. I'm marking this on a Linux machine with no Word installed so submitting Word documents might will get corrupted and will certainly make me grumpy.
- This is to be submitted on-line. See your timetable for the exact date and time.
- You should not submit copies of Matlab's own functions or toolbox functions, though your code can call such functions freely.
- Programs that find accurate matches are, of course, likely to do well. However, the quality of the code and report will be important factors regardless of performance, and originality and ingenuity will be taken into account if you can't solve all of the images.

You can use any approach you wish – though if you want to try something that seems very different to the techniques described in the course it would be sensible to talk to me first. You

are free to use any Matlab function or to download any addition Matlab library as long as it is fully referenced.

1.5 Marks

The break down of the marks:

TODO - I will update this section soon.