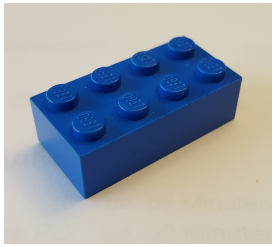
High-Level Computer Vision with MATLAB

1. Introduction

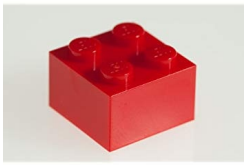
This coursework provides the opportunity to design and implement a computer vision systems that can identify and count specific types of object in an image. You will need to work on this project in your own time. If you require any clarification to the instructions please use the "Questions on Lab 3" forum on KEATS.

2. The Task

At the end of these instructions are 12 images containing various pieces of LEGO. Your code needs to provide a count of the number of Blue 2 by 4 bricks (objectA) and the number of Red 2 by 2 bricks (objectB) in such an image. Close-up images of these two types of LEGO brick are provided here:



ObjectA



ObjectB

A zip file containing the training images is available from the module's KEATS webpage. You can use these images in any way you wish, but they are provided primarily to allow you to evaluate the performance of your code.

3. System Requirements

Your code must include a m-file called "count_lego.m". The main function in this file should have the following definition:

```
function [numA,numB]=count_lego(I)
```

Hence, this function takes one input argument (`I`), which is a MATLAB variable storing a colour image (i.e. `I` will be a 3-dimensional matrix that has been produced using the command: `I=imread('legoX.jpg');`). The two output variables (`numA`,`numB`) should be integers. The first output, `numA`, should provide a count of the number of Blue 2 by 4 bricks that occur in the input image. The second output, `numB`, should provide a count of the number of Red 2 by 2 bricks that occur in the input image.

The `count_lego` function may call other functions. Unless they are official MATLAB functions, these additional functions should be included in the same m-file, or be include in the zip file you submit. Your code can also load data, as long as this data is included in the zip file you submit. Your code should run on a PC and not need access to special hardware, such as a GPU.

4. Submission

You should submit a single m-file, or a single zip file containing your code and any additional files that it requires to execute. If submitting a zip file you must ensure that when this file is unzipped that the count_lego.m file appears in the directory in which the zip file was unzipped, and not in a sub-directory.

Together with your code you should also submit a one paragraph description of the method you have used. This can be in the form of a separate pdf document or a block of comments at the top of your count_lego.m file. Within this description you must clearly acknowledge any code that you have copied or modified from online sources, books, or articles, and include full details of the source of this code by providing a URL or a citation. Re-using open-source software is good practice and demonstrates your research skills, however, not acknowledging other people's work is both intellectual property theft and plagiarism. Any submission that is identified to include other people's code which has not been clearly referenced will be reported for misconduct. You may **not** share code with fellow students, even if this is acknowledged. You can use any built-in MATLAB function or any official MATLAB toolbox without acknowledgement.

You need to submit through KEATS by the deadline specified on the module's KEATS webpage.

5. Terms and Conditions

Distribution or sharing of your submitted code, or any code derived from it, is forbidden. By submitting your

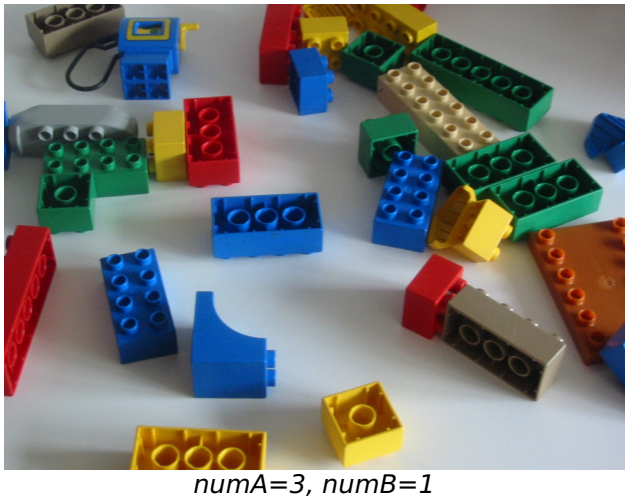
coursework you agree to abide by these terms and conditions.

6. Assessment

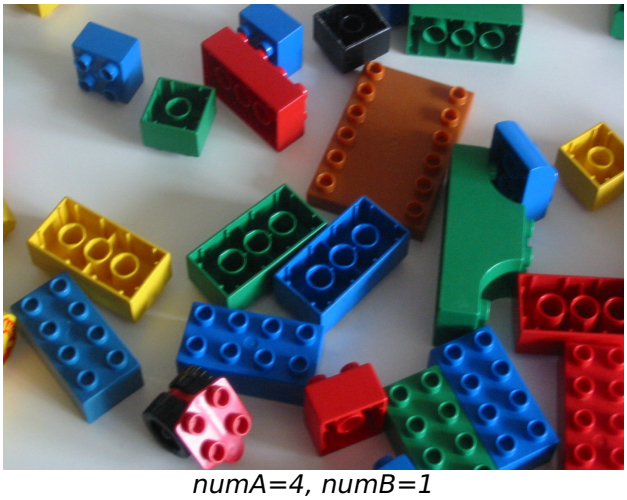
The performance of your code will be assessed by testing the accuracy with which it counts numbers of objectA and objectB in 12 testing images. The testing images are distinct from the training images given below, but were obtained under the same conditions.

Marks will be awarded primarily for how accurately your code counts the objects. In addition, credit may be given for how sensible an approach you have chosen to implement and how efficiently and correctly that approach has been implemented. Marks will be deducted if your code needs to be fixed in any way to make it execute.

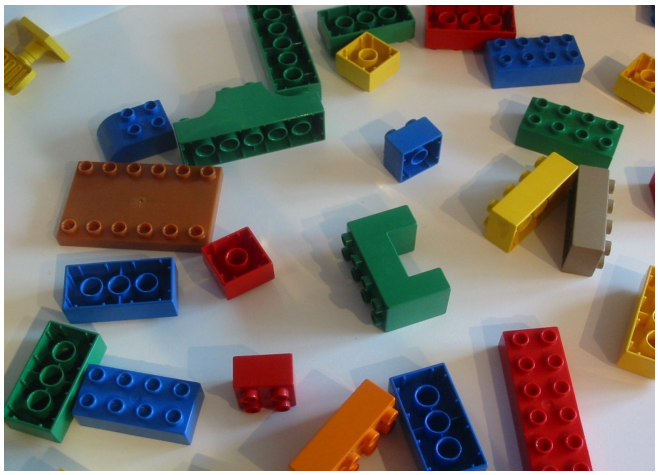
7. Training Images



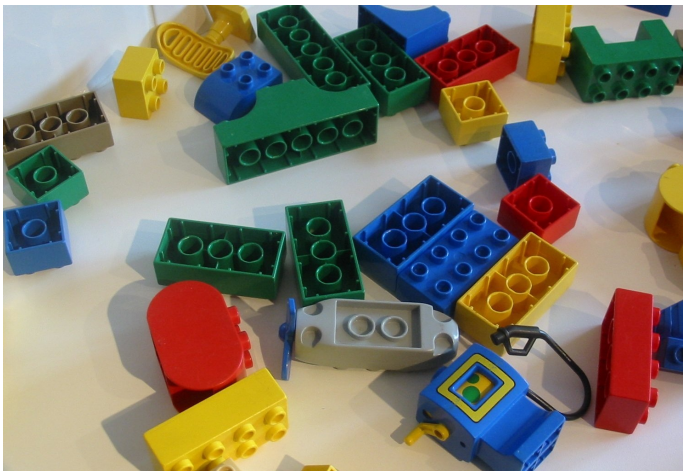
numA=3, numB=1



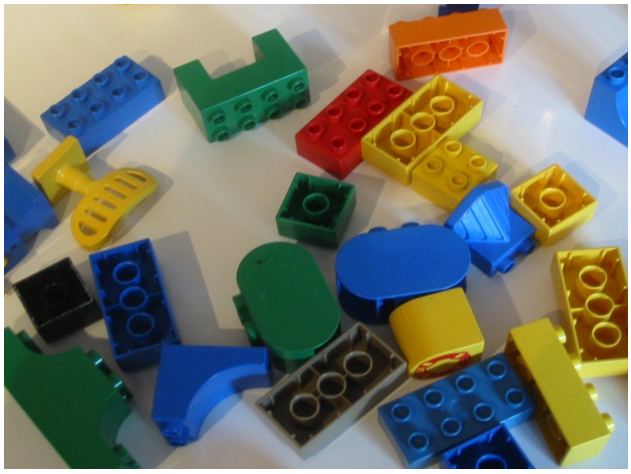
numA=4, numB=1



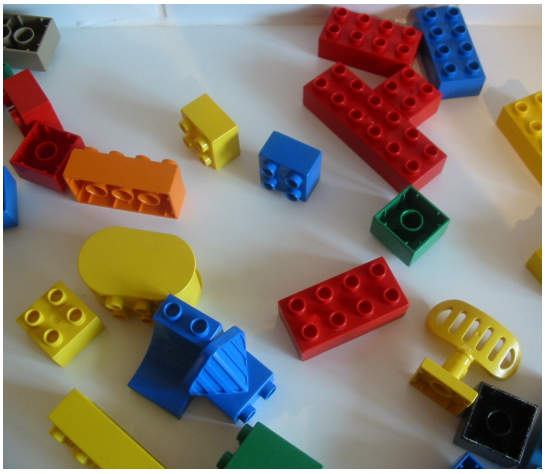
numA=4, numB=2



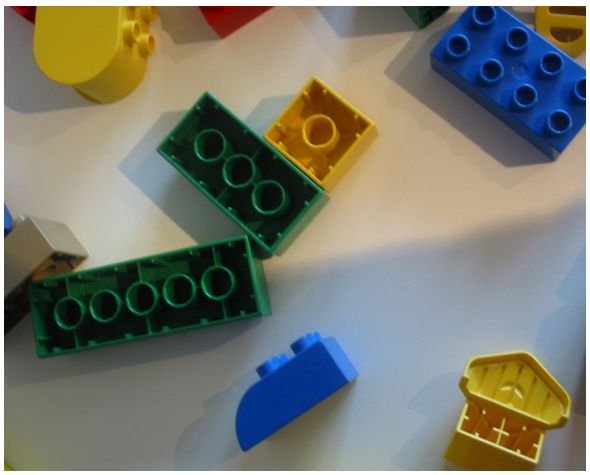
numA=2, numB=1



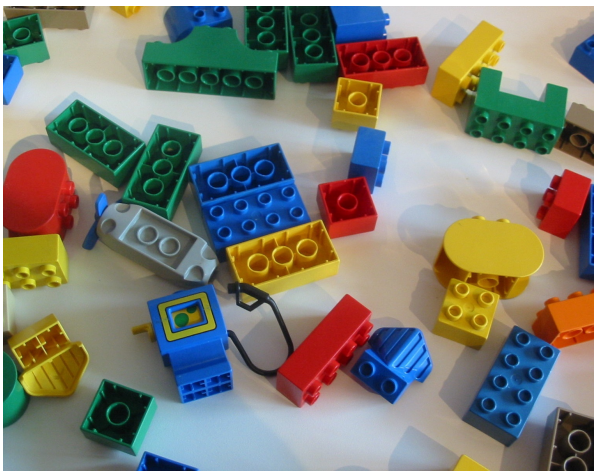
numA=3, numB=0



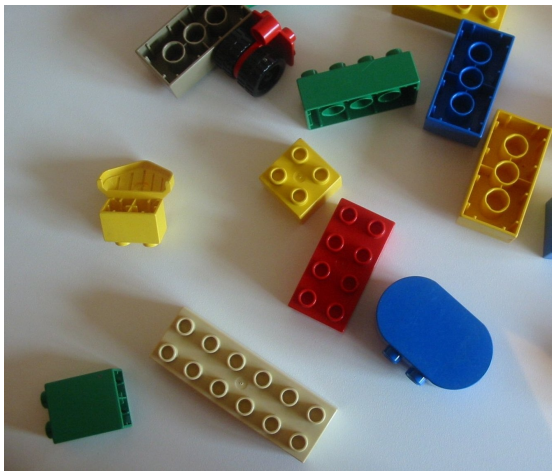
numA=1, numB=3



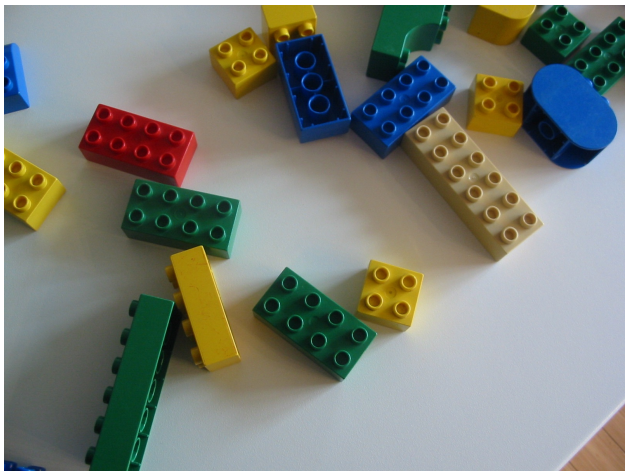
numA=1, numB=0



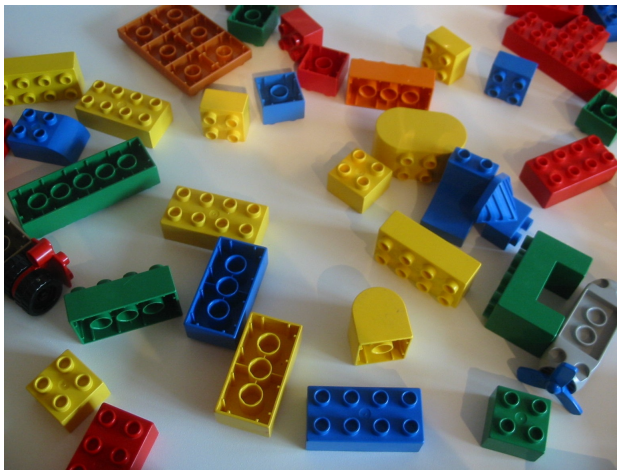
numA=3, numB=2



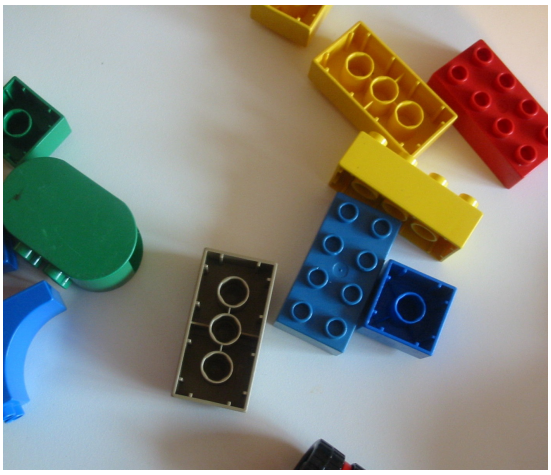
numA=1, numB=0



numA=2, numB=0



numA=2, numB=3



numA=1, numB=0