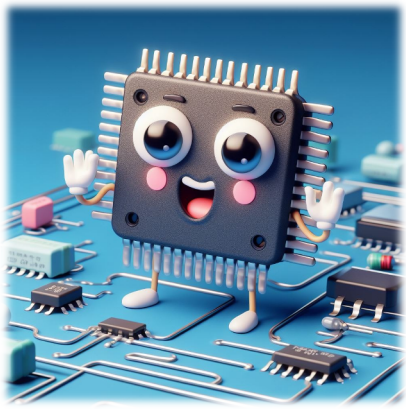
CS413/CS933 Coursework 2023-2024

 [1]

Automatic Inspection of Electronics

The aim of the coursework this year is to use image analysis methods to help inspect electronic components and to automatically identify defects on a printed circuit board (PCB).

There are three main programming tasks which you should attempt, writing python code which reads in the given images and generates the results automatically. You are also required to submit a short report describing your methods and results.

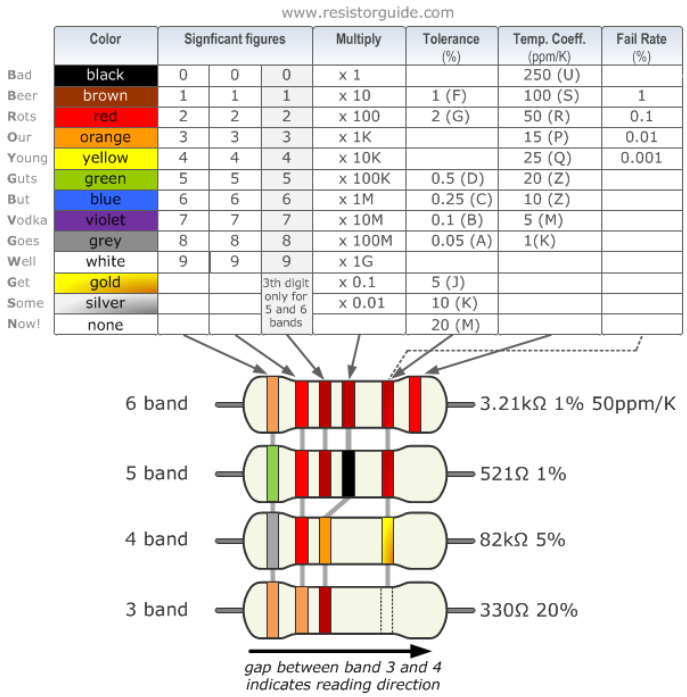
Use only code from standard python libraries: e.g. numpy, PIL, opencv, skimage, sklearn, and any examples you were given in the module's lab tutorials.

The image data files required for this coursework can be downloaded from the module resources page, together with a PDF of this document.

[1] A cheery micro-chip, courtesy of Bing's Image Creator.

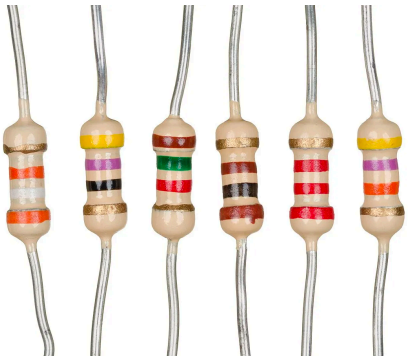
Part 1: Resistance is Futile

Electronic resistors often use colour codes to indicate their value. Such resistors are roughly cylindrical in shape and the code is arranged in several "bands" which can be decoded to determine the value. The figure below shows how to read the values:



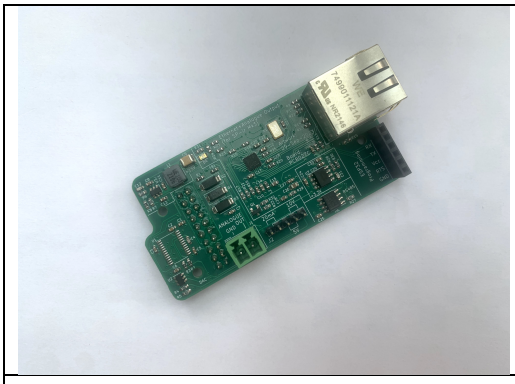
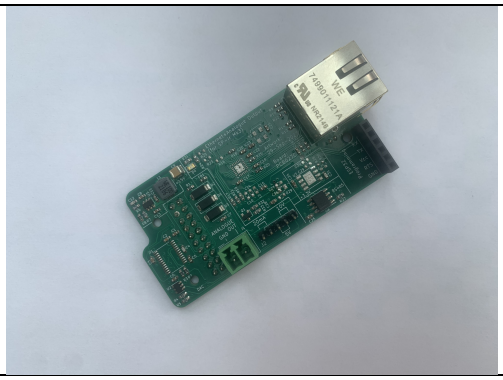
There is a handy calculator here https://www.calculator.net/resistor-calculator.html

For example the RESISTORS-1 image has a set of 4-band resistors with values: 39KΩ, 47Ω, 1500Ω, 100Ω, 2200Ω, 4700Ω (Ω = Ohms) (note the 4th-band is the tolerance and not of interest in this task):



The task is to **automatically identify the resistance value** of each resistor in the RESISTORS images.

Part 2: Spot-the-difference

	
Board 3	Board 2

In this part, given images of a three PCBs at various stages of manufacture, the task is to accurately identify places where the PCBs are different, i.e. have missing or misplaced components.

BOARD1 does not have any components on it (unpopulated) and you might wish to use this a reference.

BOARD3 is (almost) fully populated (it is missing one chip), but BOARD2 was defective and some of the components were removed.

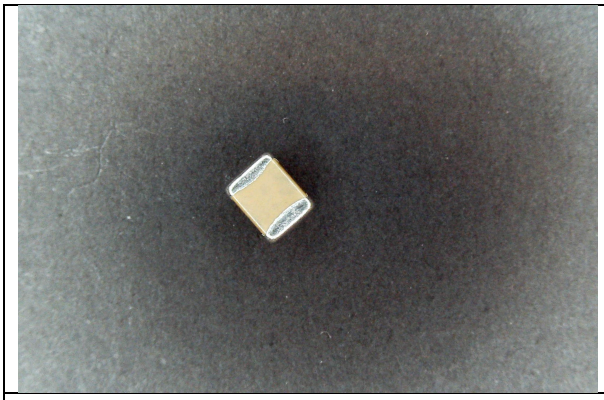
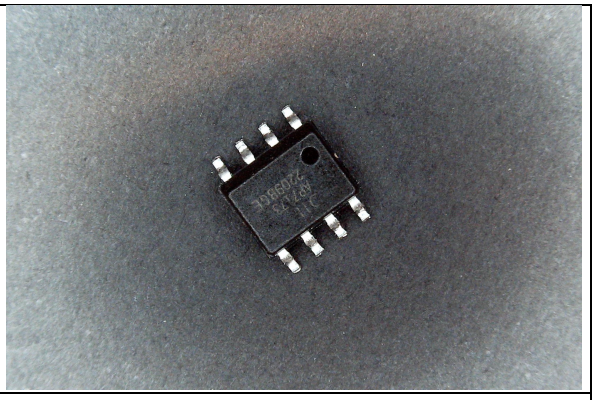
For this part, (1) align BOARD3 and BOARD2, so you can accurately **blend** between the two images of the respective boards; (2) **Highlight** the places where BOARD3 is significantly different from BOARD2.

Note for this task, there are 3 images of each board, which may help you with the solution.

Part 3: Pick-and-Place

The final task is to locate the positions of several components given an image of BOARD3. In the COMPONENTS folder, there are separate images of certain electronic components (chips, resistors, diodes etc) from the board design

Here are two examples:

	
Capacitor	Regulator Chip

There are 3 images of each component. All the component images were taken at the same scale with a high-resolution device, but the resolution is different from BOARD images.

Note also that the DAC is not actually on the board, but you may be able to discover where it might go. Also, there is no place for the DSP on the visible side of the PCB!

Annotations I have provided bounding box annotations for image BOARD3-1 which has all the component sites labelled, plus some other areas. There are also annotations for the individual component images that give the bounding box of the object.

The annotations are stored in XML format and an example notebook called `load-and-display-annotations` shows how to interpret them (note this notebook uses the python package xmltodict.)

Identify the exact position or positions where particular components have been soldered on the board. You might show this as a bounding-box and/or a text label or a colour overlay.

Note that you should **only** use the annotations to support the solving of this task, for example using it as a ground-truth, or to perhaps extract useful regions form BOARD3-1. If you choose to use the annotation data, you **must** show your method works on different BOARD images!

Submission of Code, Report and GAIT declaration

1. Code: for each part submit a Jupyter notebook named: `part-1.ipynb`, `part-2.ipynb`, and `part-3.ipynb`. The code should read data from a directory specified by the string variable `data_dir`.

You may submit your own python files (`*.py`) which contain classess/functions which are re-used in the three notebooks. All code should be zipped into a **single** archive called `coursework.zip`.

Do not submit the coursework data files provided.

All code should be well documented and the steps to run the notebooks should be clear. You may wish to use markdown cells as well as code cells to explain your code. The code should be tested to run without errors on the DCS machines and not require the installation of other libraries or python code. We will run and test each submission.

2. Report: Submit a report (`report.PDF`) which describes your code and methods. It **should not exceed 1250** words in length. You may use this to present and discuss the results of your methods. The report should use **1in margins** on all sides, minimum **12-point text**, formatted in **single column** and **single spaced text**. You should give the total number of words used at the end of the document used and submit it as a **PDF**. Please do not exceed the word-length. *Any reports not in the correct format will not be marked.*

3. GAIT declaration: a single page **PDF** format file (`GAIT.PDF`) containing a declaration on use of GAIT (see below).

Tabula: You should submit a **ZIP** of your code, and **PDFs** of your report and declaration before the submission deadline. Late submissions will incur a 5% per day penalty.

Plagiarism and Academic Integrity

You must strictly abide by the University's code of conduct for coursework submission, integrity codes and rules about cheating and plagiarism: Regulation 11 on Academic Integrity
https://warwick.ac.uk/services/gov/calendar/section2/regulations/academic_integrity/

Do not use any external sources of code e.g. GitHub, machine learning blogs, or other internet sources aimed at solving a similar problem.

Do not copy/share code between each other.

Do design your own solutions, write your own code and, to the best of your own ability, make your own attempts to solve the problem. We will use plagiarism detection software and any abuse of the rules will be penalised.

GAIT Declaration

Read carefully the guidance on GAIT given in the Student Handbook. Each submission must be accompanied by a **declaration** on any use of GAIT as a separate document file, `GAIT.PDF`.

Where you have used GAITs more broadly, this should be **clearly** acknowledged using one or more of the following disclaimers:

- **[GAIT Name] has assisted in structuring this assessment. This was used to help me organise my thoughts, the underlying work remains my own. I provide specifics in the document/source code.**
- **[GAIT Name] was used to generate alternative streams of thought. It was used to generate ideas to help me start my assessment. I have attributed the ideas it generated and my further development of them remains my own. I provide specifics in the document/source code.**
- **[GAIT NAME] has been used to proofread, edit, and refine spelling and grammar in the following sections. This was used because I am not confident in the use of English tenses. I provide specifics in the document/source code.**
- In specific cases where code or text has been generated by GAITs, please cite those instances as "**personal communication with [GAIT NAME]**" (because it is based on asking a question or giving a prompt and receiving an answer). This is usually an in-text only citation.

Marks and Feedback

The code and report will be assessed in three categories: (1) quality and completion of tasks; (2) novelty and sophistication of code and methods; (3) clarity of communication and critical analysis.

We will run and test each program submitted. We will **not** attempt to fix buggy or non-working code.

Marking scales: we will aim to assess the overall quality of coursework using the University 20-point marking scale: https://warwick.ac.uk/fac/sci/dcs/teaching/handbook/assessment

Note that because of the number and variety of solutions expected, short marking times, any individual feedback given may be quite limited.

Deadline

You should submit your solution via Tabula before **12pm on Monday 15th January, 2024**.

Abhir Bhalerao, November 2023