**Computer Vision – Assignment:**

**Task:**

A museum is developing a virtual reality (VR) reconstruction of a region of London, showing how it appeared in about 1900 — see Figure 1. They want visitors to be able to place a pointer on a map of London and use the position and orientation of the pointer to determine the location and direction of the view in the reconstruction, as shown in Figure 2. You might be interested to learn that this reconstruction can be experienced in my research lab, which has a whole-wall (4.2×2.5 m) stereoscopic display, using interaction modalities that include gestures and a bicycle!

Figure 1: VR reconstruction of Paternoster Row in London

The map to be used is Horwood’s hand-drawn one from about 1792. It is actually two of his maps joined together; pretty accurate for something hand-drawn that long ago.



Figure 2: Horwood’s map with blue background and red pointer; the view in Figure 1 corresponds to the location and direction of the pointer.

In the museum, a camera is arranged above a dark blue table, looking directly downwards. The idea is that visitors put their copy of Horwood’s map roughly in the middle of the table and then place the red pointer on it. The people who are developing the exhibit clearly designed things with care to make the computer vision task easier: the lighting illuminates the table fairly uniformly, the light-coloured paper on which the map is printed is a good contrast to the dark background and the red pointer is a very different colour from the paper. The map has a green arrow showing the direction of north. The tip of the red pointer determines the viewpoint and the direction in which it is pointing determines the view direction; you have to print these out from your solution as detailed in Section 2.

The general approach to the assignment should be fairly clear:

1. Segment the map from the blue background in the original image and extract it into a separate image in a way that makes the map edges match those of the extracted image.

2. Segment the red pointer

3. Locate the tip of the pointer to determine the location.

4. Determine the orientation of the pointer, convert it to a bearing and output it.

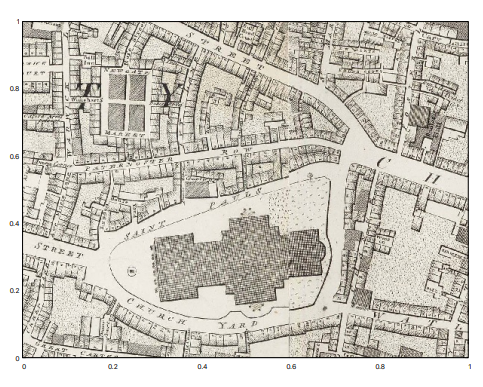


Figure 3: The coordinate system for the map

2. Operation and evaluation

Your solution should be written in Python. You are free to use OpenCV and/or numpy functionality in your solution. You should not use any modules that are not already installed on these systems.

Your program must accept precisely one argument from the command line, the name of the image to be processed, as in:

python3 mapreader.py develop/develop-001.png

It should output two lines in the following format:

POSITION 0.673 0.212

BEARING 316.4

Any other output it generates is ignored. Your submitted program must not display any images. A template version of mapreader.py is supplied, which does nothing more than handle the command line and print output in the required format; you are advised to use this as a starting point for your own solution.

When the map is oriented correctly with north pointing upwards, the two numbers following POSITION represent the location of the point of the red pointer, being respectively the distance along the bottom of the map and the distance up its side, with the origin at the bottom left-hand (south west) corner. These numbers should both be in the range 0–1, as illustrated in Figure 3. The number following BEARING should be the angle in which the red pointer is pointing, given in degrees measured clockwise from north.

In accordance with good testing in computer vision, the imagery I shall test your program with is different from that you are using to develop it. It does however have identical characteristics, so if your program works on the supplied imagery it should also work on my unseen test imagery.

The reason for being so prescriptive about the way your program is executed and the appearance of its output is because its functionality will be checked using a test harness, a piece of software that runs it and analyses its output. You will lose marks if your submission doesn’t work with the harness. There is nothing secret about the test harness, it is included as part of the assignment’s zip file. You run it as follows:

python3 harness.py

3 Guidance for not copying things from the web:

Let’s say you cannot remember how to work out the roots of the quadratic ax2 + bx + c = 0 and you do a web search for:

python roots of a quadratic equation

# The following two lines are taken from

# <https://www.w3resource.com/python-exercises/math/python-math-exercise-30.php>

x1 = (((-b) + sqrt(r))/(2\*a))

x2 = (((-b) - sqrt(r))/(2\*a))

Studying the code you might see that, at least in Python 2, this could give the wrong answer if a, b and c were all integers; and there are far too many parentheses. If you take that code but correct it and change it to fit into your algorithm, you should write in your program:

# The following two lines are adapted from

# <https://www.w3resource.com/python-exercises/math/python-math-exercise-30.php>

x1 = (-b + math.sqrt(r)) / 2.0 / a

x2 = (-b + math.sqrt(r)) / 2.0 / a

4 Marking criteria:

**Algorithms**. Underlying every program is an algorithm or, more commonly, several algorithms. An algorithm is a series of steps that transform some input to an output. The questions I ask when marking submissions are:

• When there are several alternative algorithms, has the most suitable one been chosen? Are there comments that justify the choice?

• When an algorithm has had to be developed from scratch, does it do the job it is intended to do? Has it been implemented correctly?

• A program that runs in one second is better than an equivalent program that takes ten seconds to run, and that is rewarded under this heading. Conversely, attempting to squeeze every nanosecond of performance from a single line of code that is executed only once and takes little time to run is pointless and will be penalised. It is important to know when to improve performance as well as how and where to improve it.

**Coding style.** Although an algorithm is the core of a piece of code, the way in which it is implemented is important. When reviewing submissions, I ask the following questions:

• Is your code easy for someone to read? You might know that, say, a particular numpy (numerical Python) expression can be made to execute very quickly; but if it is hard to understand and not in the most time-critical part of a program, it is probably better written to be easier to understand, or at least well-explained in a comment.

• Does the way the program is structured help its maintenance? In the Real World1 , maintaining and extending existing code is much more common than writing code from scratch, so a good programmer will make sure there is enough explanation for someone coming to the program can maintain it.

• Does your program structure aid re-use of parts of the code? This could be by creating useful classes, modules or procedures that are general enough to be used elsewhere, for example. Similarly, do you use standard procedures rather than writing your own?

• Are you careful in your use of resources, especially imaged and large arrays? Programs that are wasteful in terms of memory usage are difficult to port to small computers or embedded platforms.

• Code that carefully arranges data to be organized so that a series of processing steps can be performed quickly and easily gains more credit than brute-force processing. It is not always straightforward to make code elegant in this way but it is easy to recognize it when you see it.

Clearly, there can be trade-offs between computing things several times and storing the results of calculations for subsequent use, and a good programmer will discuss this topic briefly in the code.

**Results**. This section catches all the other aspects of the program — principally whether it actually is a bug-free solution to the assignment, but also other factors such as whether or not the code provides useful output when it encounters problems.

• Is the program invoked as per the specification? Failure of a program to meet its specification is a problem in the Real World as it will often end up as a component of a larger system. By the same token, writing a program that does what you want rather than what was specified will yield a substantially lower mark than one that does meet the specification.

• Is the program buggy? All bugs that you can find should be exterminated before submitting your program.

• Does the program describe what should be produced from a specific input? How would a person receiving your program know that it works? The usual way is to include example inputs and the corresponding outputs in a block of comments at the top of the program. Python provides really cute doctest functionality, for example.

• Does the program produce helpful output if invoked incorrectly? If a program requires an argument on the command line, it should not crash but should instead output (on the standard error stream) a message that explains how the program should be used. Similarly, if the program reads files with specific names as part of its operation, you need to produce a useful message if they are not present.

• Are the results as expected? Does the program produce results that are consistent with what you expect? If not, say so in your program’s comments as I can award you some credit for having spotted that your program is not giving the correct answers.