BEng Biomedical Engineering
Object-Oriented Programming

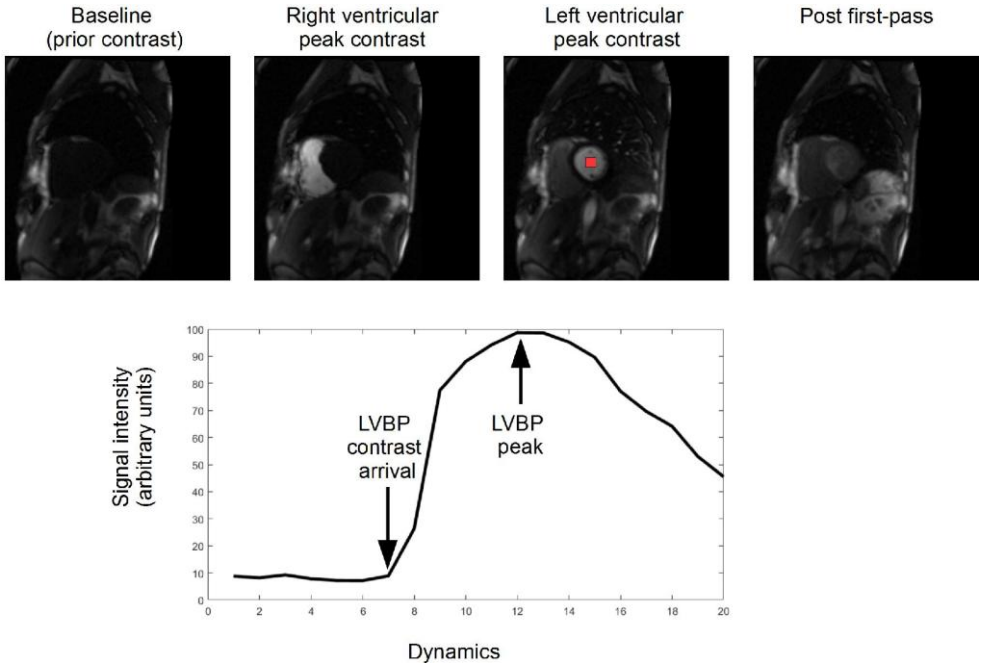
Coursework 3

Objective

To gain practical experience of C++ coding to design and implement a project to solve a more complex problem.

Introduction

Magnetic resonance imaging (MRI) is used clinically for the assessment of myocardial perfusion. In this protocol, a contrast agent is injected intravenously into the patient and images of the heart are acquired in each heartbeat over a period of 1-2 min to observe the first pass of the contrast agent in the myocardium. This protocol therefore involves acquiring images of the heart at regular time points, with a time interval of a few seconds. Any region in the myocardium with reduced contrast uptake (reflecting a perfusion defect) can then be identified as having reduced signal intensity compared to the rest of the myocardium during the first pass of the contrast agent in the myocardium. Typical images from a cardiac perfusion MRI scan are shown below (top row), together with the temporal profile of the signal in the left ventricular blood pool (LVBP) (bottom row).



A research team is developing software to facilitate the analysis of cardiac MRI perfusion data.

This involves the calculation of different parameters related to the temporal evolution of the contrast agent. The initial processing pipeline should implement the following functionalities:

1. Load one time series of cardiac MRI perfusion images from a series of PGM image files (saved as "mri-01.pgm", "mri-02.pgm", etc).
2. Load the contrast agent information from a separate file called "contrast_info.txt".
3. Create a 2D binary image (i.e. a *mask*, where pixels can only take values 0 or 1) of the same size as the original image, where all pixels are set to 1 if inside an LVBP, or 0 otherwise. The LVBP region should be defined as a 5x5 pixel square centred on the pixel at position (x, y) = (74, 90) in the given example.
4. Calculate the average signal *S(d)* in the LVBP region for each time frame *d*, and plot this timecourse on the terminal.
5. Identify the time frame of peak contrast concentration ($d_{peakBlood}$) in the LVBP region and the corresponding mean LVBP signal ($S_{peakBlood}$) in the image at that time frame. Display the image frame at peak contrast concentration in the terminal.
6. Identify the time frame of contrast arrival $d_{arrival}$ in LVBP and the corresponding LVBP signal ($S_{arrival}$) in the image at that time frame. This should be computed as the earliest time frame (prior to $d_{peakBlood}$) at which the temporal gradient of *S(d)* (the signal within the LVBP region) exceeds the threshold value of 10. The temporal gradient should be computed as *S(d+1)-S(d)*. Plot the gradient timecourse on the terminal.
7. Compute the temporal gradient (G) of the signal during contrast uptake as: $G = (S_{peakBlood} - S_{arrival})/(d_{peakBlood} - d_{arrival})$.
8. Display the contrast agent info associated with this scan as well as $d_{arrival}$, $S_{arrival}$, $d_{peakBlood}$, $S_{peakBlood}$, and G.

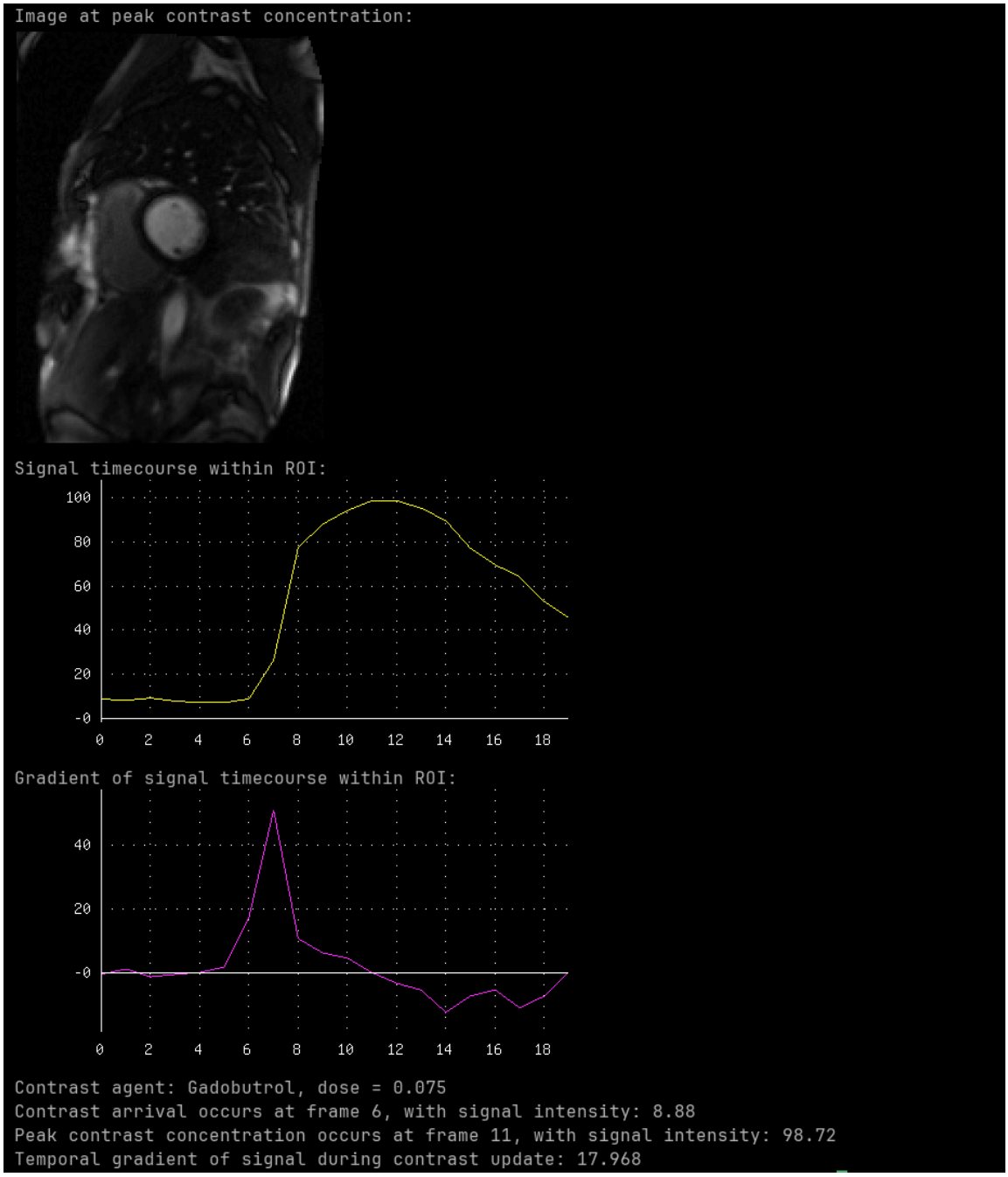
Instructions

Your task in this coursework is to write a C++ program which meets the requirements described above.

A time series of images in plain PGM format is stored in the "data" folder. A full description of this file format is available online. In brief, the first line contains a string ("P2" in our case), followed by three numbers: the image dimensions along X and Y, followed by the maximum pixel value (255 in this case). The rest of the file contains the signal intensities of all pixels, where all pixels values have been stored one after the other, in row-major order.

The contrast agent information is also stored in a separate file called "contrast_info.txt", which contains two lines: the first line contains the name of the contrast agent, the second line represents the contrast dose used during the perfusion scan (a floating-point number in mmol/kg).

An illustrative example output is shown on the next page, for the data files that you are provided with.



All indices in these instructions (time point and pixel coordinates) are expressed using zero-based array indexing (i.e. starting as 0).

Note that a perfusion image dataset is a specific type of temporal dataset. In the future, the research team would also like to model other types of temporal datasets, which may be acquired in the absence of contrast agent but may have other additional properties. Given this wider context, the functionality developed in this project should be designed to maximise its potential for re-use, whether as part of this project or for the analysis of other types of images.

Reporting Requirements

Your submission should consist of two parts:

1. A **short report** explaining your object-oriented design. The report should include a short explanation of the purpose of all classes and relationships, attributes and behaviours, as well as a brief description of the object-oriented design process that you went through to produce your design. Your report should not need to be longer than two pages of A4, and can be shorter.
2. **C++ code** that matches your design and meets as many of the requirements as possible. Try to use variable/function naming, comments and indentation to make your program as easy to understand as possible. Also try to make your program as resilient to runtime errors as possible.

Submission will be via the KEATS system. The submission portal will allow you to upload two files: one for your report (in MicroSoft Word or PDF format), and one for your C++ code (as a single ZIP file). **Please only include your .cpp & .h files in the ZIP file** – you can use the "oop_build clean" command to remove all intermediate outputs prior to creating your ZIP file.

The hand-in date is Thursday, 10 April 2025 at 4pm. Late submissions will be treated according to the relevant University regulations for late submission.

If your program does not meet all requirements, please submit what you have written by the deadline.

Assessment

Your coursework will be marked as follows:

Written report	15%
Does the program work? Does it meet all requirements? Has it been tested extensively?	50%
Program design and appropriate use of C++ language features, e.g. functions, classes, composition/aggregation/inheritance as appropriate, etc.	30%
Use of comments, indentation and variable/function names to make code easy to understand	5%

The overall mark for this coursework will make up 30% of your total mark for this module.

This is an individual assignment. **You are not permitted to work together with any other student. Code generated using AI assistants, etc. is also not permitted.** Note that general discussions about design decisions and/or coding strategies are permitted, and such discussions can be a useful learning experience for you. But you should not, under any circumstances, share details of designs or code.

If you have any questions about this coursework, please contact:

- J-Donald Tournier (jacques-donald.tournier@kcl.ac.uk)
- Michela Antonelli (michela.antonelli@kcl.ac.uk)