On the optimization of imaging pipelines

Schoonhoven, R.A.

Summary

Today, many of our industrial and scientific tasks rely heavily on powerful computer systems that process huge amounts of data. These tasks are performed by computational pipelines that often consist of several algorithms that calculate the data, and pass the results on to each other. Conceptually, the process of designing a computational pipeline for these tasks has different stages.

- **The Idea Stage**: This is where we plan how the pipeline will work, deciding which algorithms to use and how they connect. Figure S3 shows a conceptual example of how we might design the workflow.
- **The Software Stage**: Here, we decide how each algorithm must perform its task. We try to write the code to make the algorithms as efficient and accurate as possible.
- **The Hardware Stage**: This is about deciding what computational hardware each algorithm will have available, and how to efficiently run the computation on that hardware.

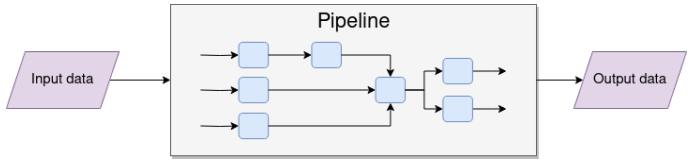

Figure S3: Example computational pipeline illustrating how input data is processed by different algorithms and passed on until the final result is computed.

Running such a pipeline can be challenging. We often need to tweak the way the algorithms work, make sure they're not wasting time and energy, upgrade their hardware, and sometimes, we need to redesign the whole pipeline to get the best result. Plus, with today's environmental challenges, we must also try to make our pipeline eco-friendly. To do this, in this thesis, we focus on all three stages to optimize our computational pipelines.

One of the applications where such challenging computational pipelines occur is in the field of X-ray computed tomography. Computed tomography (CT) is a technique for visualizing the interior of objects with some form of radiation (Figure S4). This makes it a powerful tool as we can study the internals of objects without damaging them, or opening them up. One of the computational challenges comes from the fact that CT is typically done in 3D, which means that large amounts of data are involved, and any computational step needs to work on a large batch of data. Furthermore, CT pipelines usually come with several different processing steps that we would like to optimize.

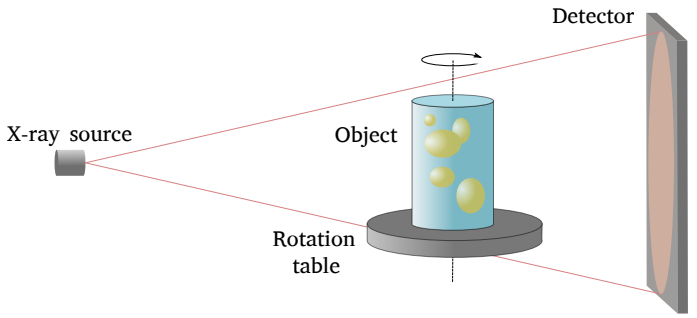

Figure S4: An example setup of an X-ray CT scanner that is common in laboratories.

Several settings of a CT scanner, such as the beam's energy and from which position it captures the X-ray photographs, need to be chosen correctly to get a sharp CT image with high contrast. This creates a challenge from an optimization point of view. Often, the desired result (or an undesirable image feature) is only visible at the very end, whereas it may be caused by a setting at the beginning of the CT workflow. For example. the alignment of the machine may be off, or we ran our algorithms with the wrong parameters. The problem is compounded because users often want to perform extra processing steps at the end of the reconstruction.

In Chapter 2, we attempt to add a so-called segmentation step to an X-ray CT workflow using AI, such that it can perform the computation in real time (meaning the operator gets a live view). In Chapter 3, we attempt to tackle the problem of optimizing algorithmic parameters along the whole pipeline for a final result. In Chapter 4, we revisit the AI that we used in Chapter 2, and attempt to speed it up significantly using a technique called "pruning". In Chapter 5, we look at the final hardware stage. In particular, we look at a type of chip called a Graphics Processing Unit (GPU), and which optimization algorithms are best for structuring these GPU computations. In Chapter 6, we look at GPU computations from an environmental and energy consumption perspective, and propose a model to make them more energy efficient. In the end, we look at how much energy our method saves for a large European network of radio telescopes called the Low-frequency Array (LOFAR).