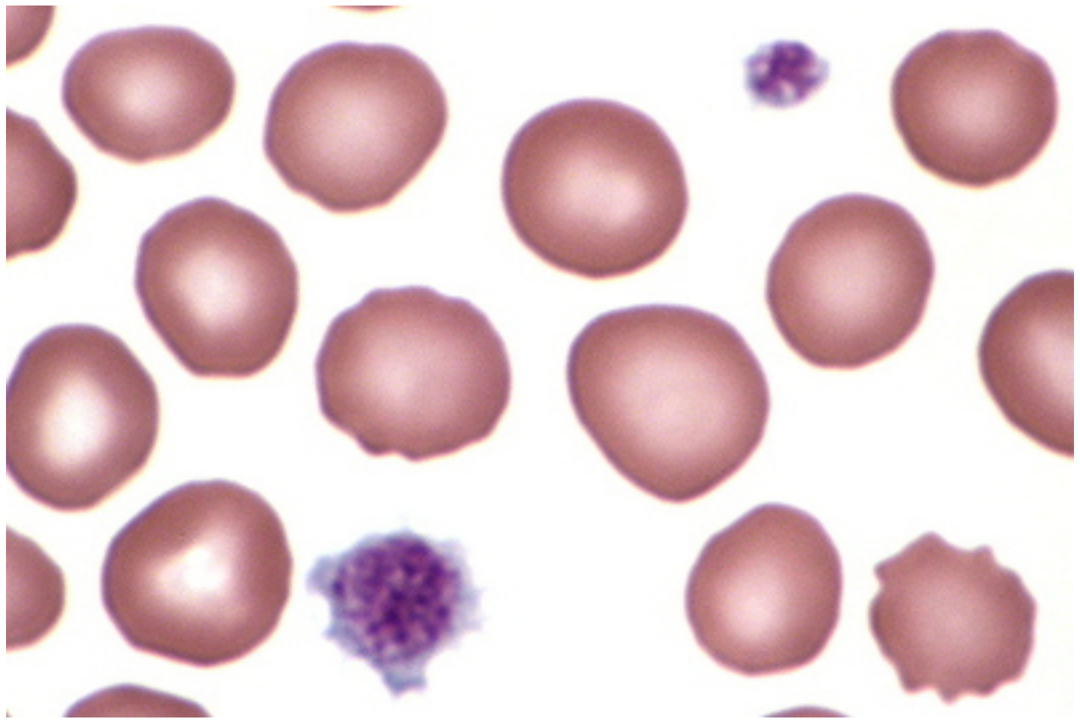
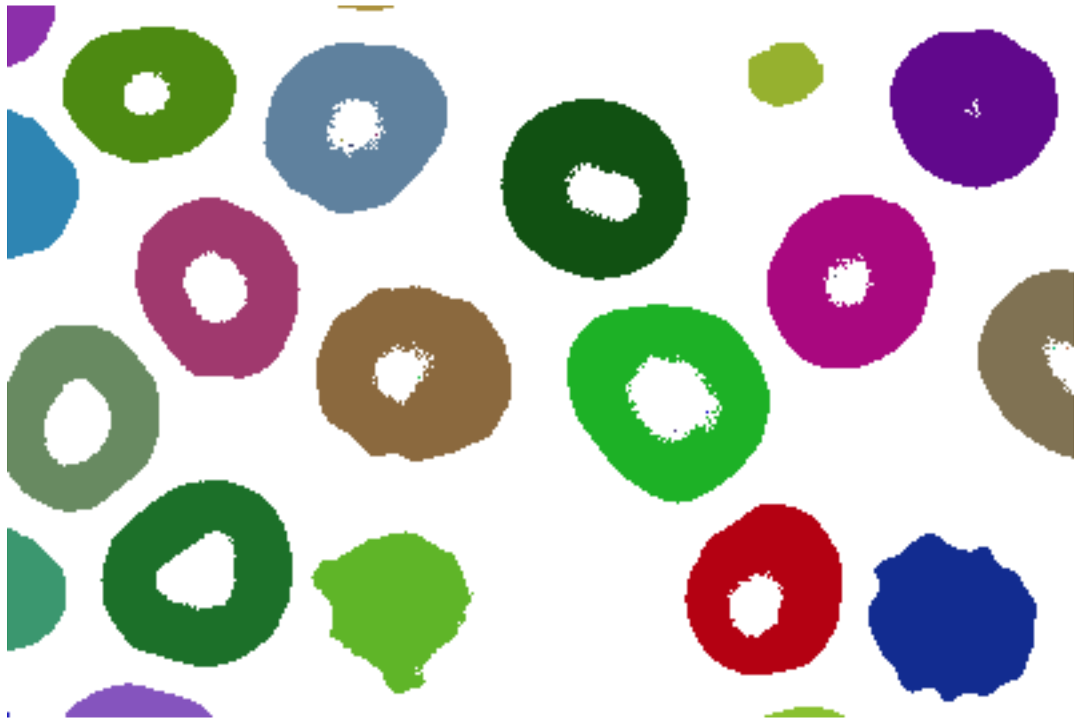
Project OpenCL: counting cells

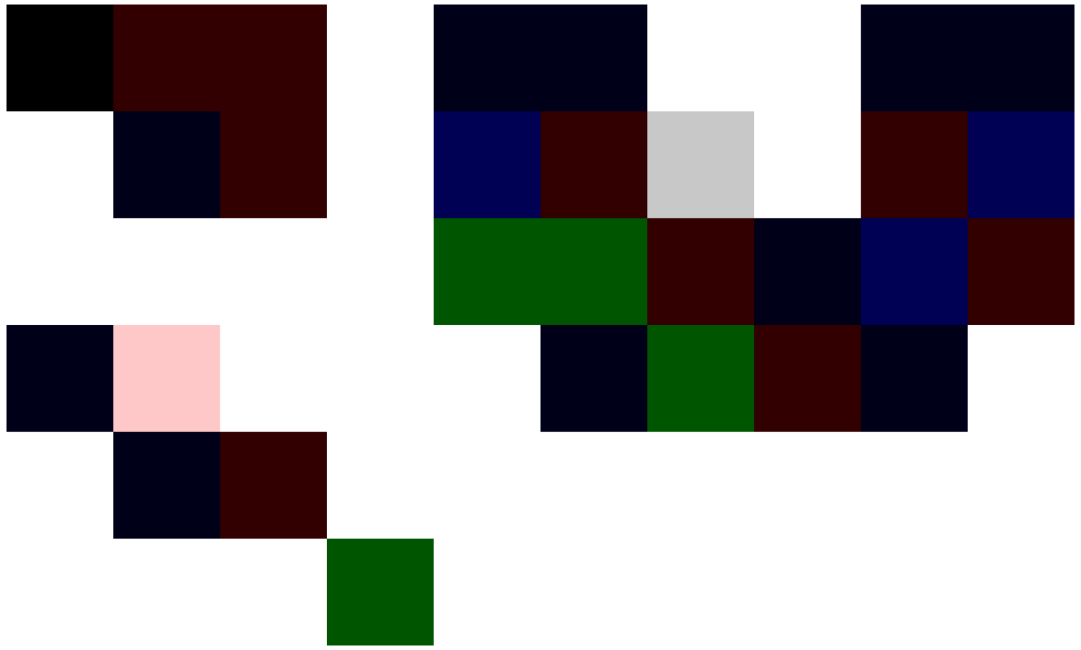
Multicore Programming

Count the number of platelets in a blood sample

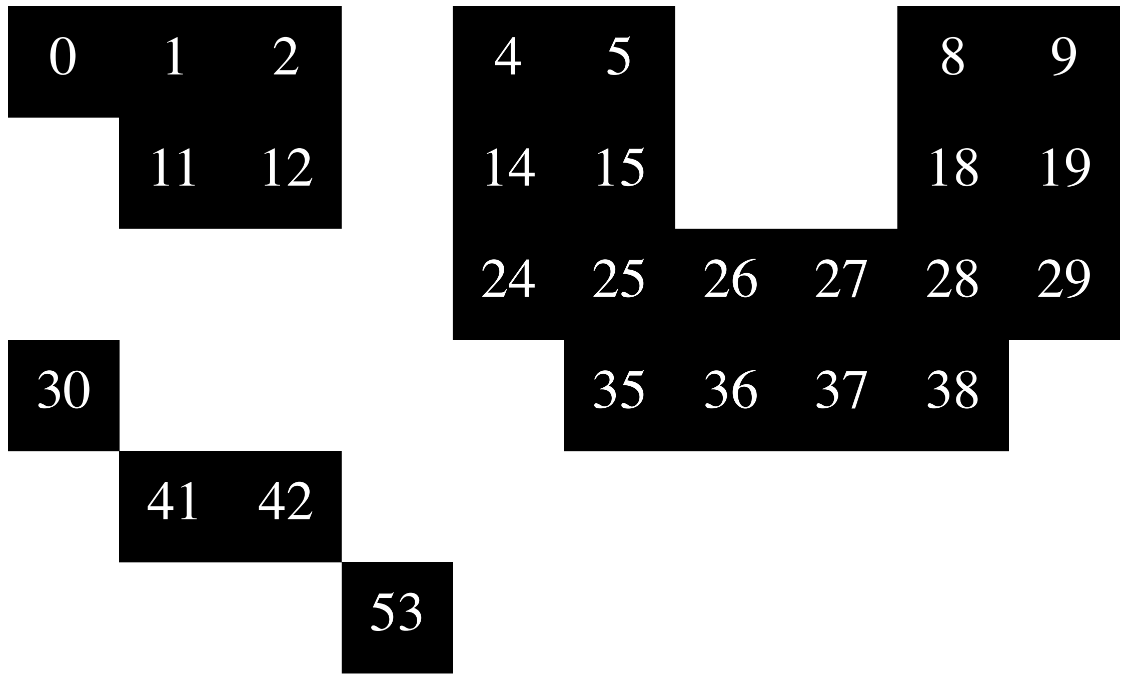




Algorithm

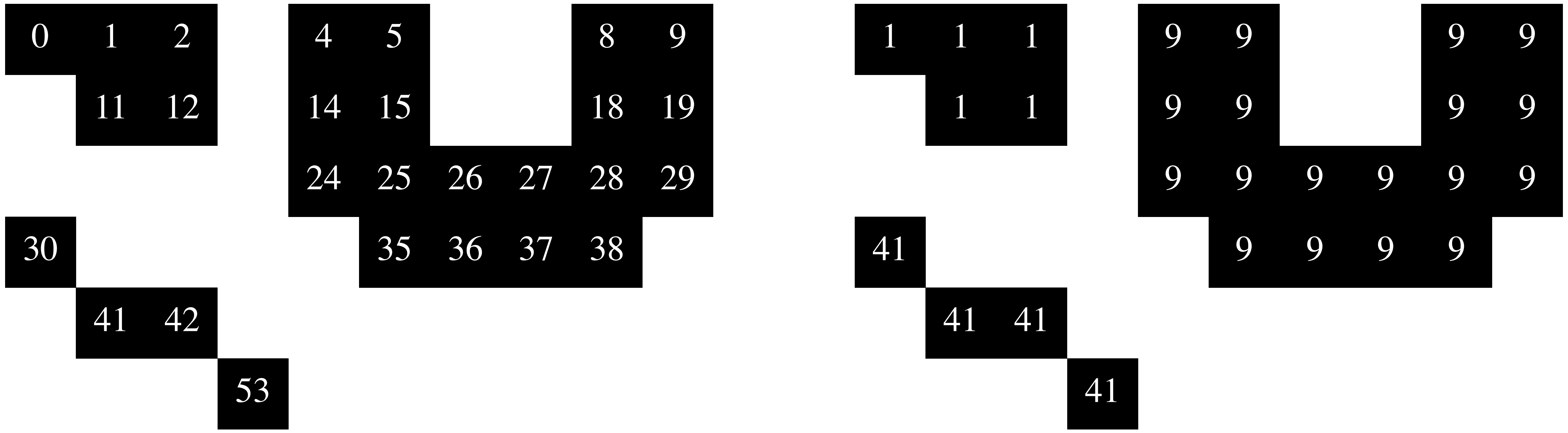

This is a tiny input image.

Algorithm



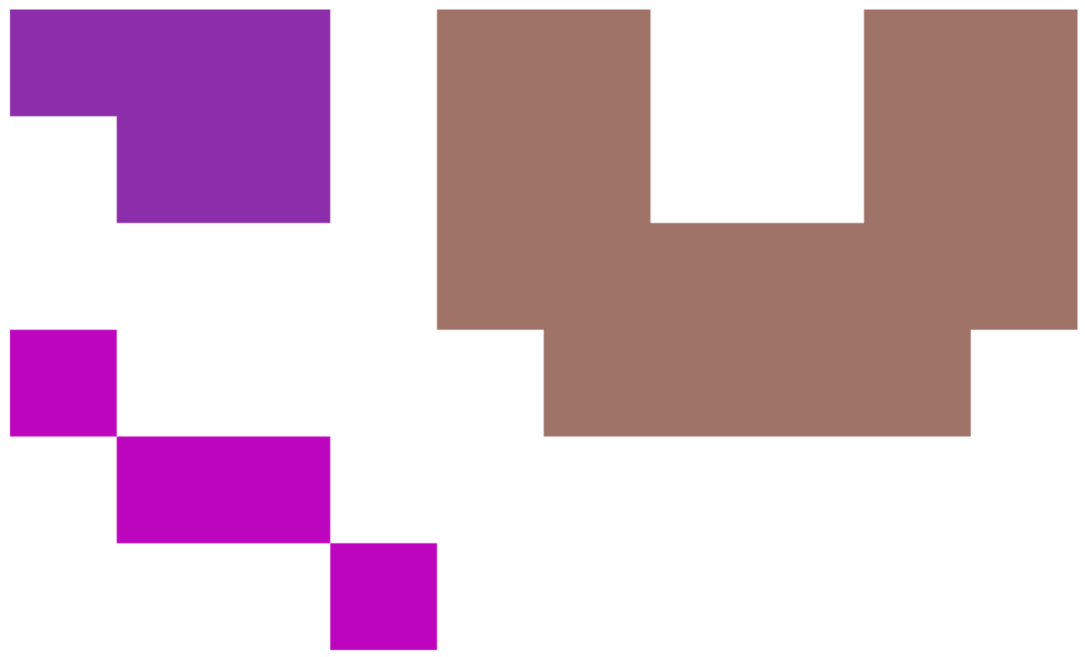
1. Convert to black/white:
   pixel brightness ≥ 200 = background = white
   pixel brightness < 200 = foreground = black
2. Give each pixel a unique number (= 'global index')

Algorithm



1. Convert to black/white
2. Give each pixel a unique number
3. Create a “disjoint-set” data structure.
   Each pixel is a separate ‘set’ containing only itself.
4. Perform a “union-find” algorithm to union/merge neighbouring pixels/sets.

Algorithm



1. Convert to black/white
2. Give each pixel a unique number
3. Create a “disjoint-set” data structure.
   Each pixel is a separate ‘set’ containing only itself.
4. Perform a “union-find” algorithm to merge/union neighbouring pixels/sets.
5. Give each set a unique number (and color), and count the number of sets.

Implementation (1)

Sequential implementation given in Python

Your assignment:
1. Port to OpenCL

→ this is the 'naive' version

⚠️ Be careful to avoid data races when joining sets

Implementation (2)

Your assignment:
2. Create optimized implementations

Many optimizations possible:

- Use of private/local memory
- Work group size
- How the work is divided into WIs/WGs (e.g. row/column, tiles)
- Data representation in memory (e.g. types), memory access patterns
- Changing/re-ordering steps of the algorithm (incl. combination of CPU and GPU)
- Vectorization

⇒ create at least 1 optimized version

Always optimize for GPU first

Evaluation: correctness

Result of naive and all optimized versions should be equal to result with sequential version in Python.

(If you change the algorithm, apply these changes back to the Python version.)

Some example images are given.

Evaluation: performance

Benchmarks on sample image(s)

Compare naive and optimized versions on GPU, for several images.

Optionally:

- Memory transfer time
- Different devices (incl CPU)

Evaluation: hardware

Run your experiments on “Dragonfly”, a machine at SOFT
with a high-end GPU
This is required!

You can also add experiments on your own machine or machines
in the computer rooms.
Especially if the machine has a dedicated GPU or Apple Silicon.
This is optional.

Report

Table of contents in assignment sheet:

- Implementation
  - Naive version: what does a work item / work group do?
  - What optimizations did you implement?
- Evaluation
  - Correctness
  - Performance evaluation

Details

Deadline: **Thursday, 8th of May**, 23:59

Submit code and report (ZIP) on Canvas

Project defense in June

⅓ of final grade

Assignment and Python implementation on Canvas