Junior-Prof. Dr. Olaf Ronneberger,
Image Analysis Lab, Institute of Computer Science, Albert-Ludwigs-University Freiburg

Exercises for 3D Image Analysis

Summer term 2015

Exercise 1 (Issue Date: Fri, 24.04.2015, Due Date: Tue, 12.05.2015)

Maximum and average intensity projections

1) Write a C++ program “slices” that extracts the middle slices (xy), (xz) and (yz) of the given data set
2) Write a program “mip” that computes the maximum intensity projection of the data set in z-direction
3) Write a program “aip” that computes the average intensity projection of the data set in z-direction

The data set is provided at *http://lmb.informatik.uni-freiburg.de/lectures/3D_image_analysis/exercises/material* as “whatisit_124x216x181_8bit.raw” in raw data format, 8bit. The dimensions are 124 levels, 216 rows and 181 columns. The programs shall produce raw PGM images at output format (which can be read by nearly every image viewer). PGM format is defined as follows:

- Header “P5\n”
- width height and maximum gray value, e.g. “181 216 255\n”
- contiguous raw image data

General hints

- Course information is available online:

  http://lmb.informatik.uni-freiburg.de/lectures/3D_image_analysis/index.en.html
  - Up-to-date course slides after the lectures
  - Exercise sheets and additional material for download not later than Friday

Hints

- The program only needs one main() function
- use **unsigned char** as data type
- specify filenames and dimensions hard-coded in the program
- use std::ifstream::read() and std::ofstream::write() for raw data input and output. These methods want a pointer to char, so you have to cast your memory with **reinterpret_cast<char*>(data)**

- for compilation just use "**g++ slices.cc -o slices**" or better with all Warnings: "**g++ -Wall slices.cc -o slices**" (Only use the filenames given in this compiler string. We must be able to compile your programs given this compiler string!)
- only use **relative paths**, instead of absolute paths!
- don't use width,height,depth x,y,z as variable names. Better use nLevels, nRows, nCols and level, row, col – which makes the program much more readable.
- Element wise data access via data[level * nRows * nCols + row * nCols + col]
- you can make use of the analysis tool "valgrind" (see http://valgrind.org/) to check you program for many memory management and threading bugs before you hand in your solutions. In this way you are able to discover memory leaks, that may remain undiscovered otherwise.

Further Information

Some more details about the PGM format: http://netpbm.sourceforge.net/doc/pgm.html