Project 2: CT-Reconstruction

Motivation

CT-Reconstruction is an important quantitating and visualizing tool for all natural sciences.

To learn the basics we use a cone-beam step-and-shoot method based on the inverse-radon transform that is a mathematical model for transforming a series of 2D images into a 3D volume.

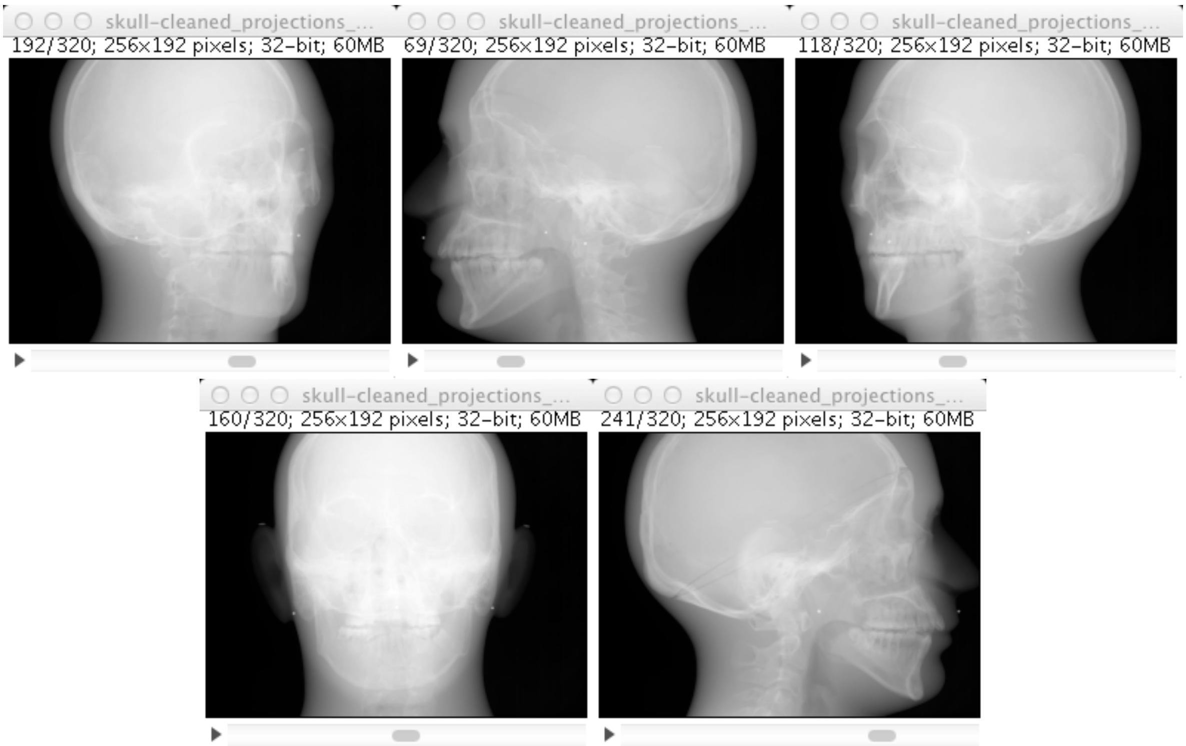

Figure 1: ***X-Ray 2D images***

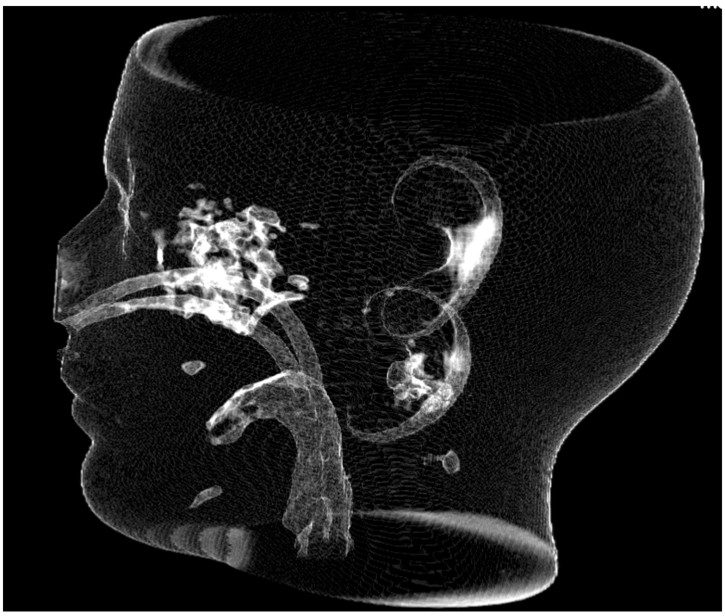

Figure 2: ***Reconstructed 3D Volume rendered for soft tissue***

CT-Reconstruction steps

Reconstructing a 3D volume from a series of 2D images requires several steps, in this assignment we will only look at the core steps. In order to simplify the task a number of pre-calculated input files are provided with the assignment:

1) projections.bin
    1. Matrix with the pre-processed 2D X-Ray images
2) combined.bin
    1. Matrix with all combinations of all X,Y coordinates for the 3D volume
3) z_voxel_coords.bin
    1. Matrix with Z coordinates for the 3D volume
4) transform.bin
    1. Matrix used to map 3D coordinates to 2D coordinates
5) volumeweight.bin
    1. Post weight to compensate for the cone effect of the X-Ray beam

The sequential code for the core reconstruction steps become:

```
1: for p in xrange(nr_projections):
2:       for z in xrange(z_voxels):
3:           combined_matrix[2, :] = z_voxel_coords[z]
4:           vol_det_map = dot(transform_matrix[p], combined_matrix)
5:           map_cols = rint(divide(vol_det_map[0, :], vol_det_map[2, :])).astype(int32)
6:           map_rows = rint(divide(vol_det_map[1, :], vol_det_map[2, :])).astype(int32)
7:           mask = (map_cols >= 0) & (map_rows >= 0) & (map_cols
                    < detector_columns) & (map_rows < detector_rows)
8:           proj_indexs = map_cols * mask + map_rows * mask \
                * detector_columns
9:            recon_volume[z].flat += flat_proj_data[proj_indexs] \
                * volume_weight[p] * mask
```

Explanation:

Line 1: Loop over the set of all recorded 2D images.

Line 2: Loop over the 3D volume Z coordinates

Line 3: Copy Z coordinates into the combined matrix

Line 4-6: Transforms 3D voxel[1] coordinates into 2D pixel coordinates

Line 7: Test if the calculated 2D pixel coordinates are within the 2D image

Line 8: Calculate 2D image flat indexes

Line 9: Weight and add pixels from the 2D image to the reconstructed 3D volume

---

1 A voxel is a 3D pixel

Programing Task

You need to parallelize the provided sequential code using Python's multiprocessing module and shared memory.

Report

Your report, which should be submitted through Absalon, should be no longer than 2 pages in total, please upload the report as PDF document and the code as a zip file but keep them as two different files. You should explain how you have divided the workload amongst the CPUs. You should provide experiment results with all four provided datasets, i.e. 64, 128, and 256, cubed reconstructions, where you compare the performance of the original version to your parallelized solution. The results should be presented as an easy the read graph, which includes the absolute and relative performance before and after transformation of the code.