Parallel and Distributed Computing

DIGITAL ASSIGNMENT-1 REPORT FOR THE COURSE 'CSE4001'

Submitted To:

Prof. Sairabanu J.

Assistant Professor,

School of Computer Engineering (SCOPE)

Submitted By:

BRIJGOPAL BHARADWAJ | 17BCE2399 | B. Tech. (CSE) | Slot : C1

ASSIGNMENT TITLE:

Spatial Enhancement of Full-Body Bone Scans using the Parallel Programming Paradigms of OpenMP and OpenCV

LANGUAGE AND HARDWARE DETAILS:

Programming Languages:	**C++**
External Libraries:	**OpenCV, OpenMP**
Graphics-Card:	**Intel(R) HD Graphics 5500**
Manufacturer:	**Intel Corporation**
Chip type:	**Intel(R) HD Graphics Family**
Operating System:	**Windows 10 Pro 64-bit (10.0, Build 17763)**
System Manufacturer:	**Hewlett-Packard**
System Model:	**HP Spectre Pro x360 G1**
Processor:	**Intel(R) Core i7-5600U CPU @ 2.60GHz (4 CPUs)**
Memory:	**8192MB RAM**
Available OS Memory:	**8074MB RAM**
Page File:	**5319MB used, 4760MB available**

APPLICATION DETAILS AND DESIGN DECISION:

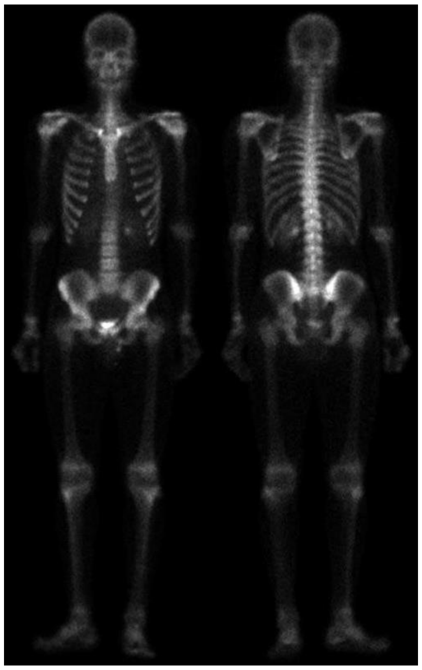
Bone scans are a class of imaging tests, that play a crucial role in the diagnosis of a variety of defects, damages, and problems with the bone-structure. They are carried out by introducing a very small amount of a special type of radioactive drug, called a radiopharmaceutical, in the system of the subject, and then recording the recording the radioactive emissions from the subject's body. When these emissions come out of the subject's body, they vary in intensity, with respect to the amount and type of the body-organs and other organic structures present in their way. This results in the development of an image on the photographic film kept in front of the subject, which shows a one-to-one correspondence with the internal structure of the subject's body.

It is to be noted that, these emissions are only affected by dense objects in their path, like the bones, and thus other internal organs are transparent to them. This gives us the added benefit of obtaining an image that only contains the bone-structure of the body. Thus, bone scans are performed to uncover the problems associated with bones only, like bone-metabolism, deep-fractures, etc. It can also be used to detect if cancer has spread to the bones from another area.

But these applications require a thorough human evaluations by a doctor. Thus, it is essential that the obtained bone-scan is visually clear to see and understand, for better diagnosis. This is often not t he case with the unprocessed bone-scans, due to the operations of various physical phenomena like diffraction and refraction on the radiation. In this project, the aim is to develop an algorithm that will take the original bone-scan, and enhance it to be easily interpreted, via the use of image-processing concept. This program will be developed with the help of OpenCV, and will also use OpenMP for obtaining better performance for the same results.

The interesting aspect of this application is the fact that at each step of the process, one can observe the current state of the input image, and obtain a better understanding about what the underlying mathematical concepts signify in a real-world scenario. It is also interesting to observe how the application of parallel paradigms affects the execution-time and efficiency of the given sequential code, thus providing a reinforced proof of relevency to the parallel programming concepts, and show why they should be adopted in any computationally intensive task.

DISCUSSION:



The application under consideration was first developed sequentially, and then was parallelized according to the need and applicability. The image given to the left is used as the test-subject for the demonstration of the working of the given algorithm. It shows an un-processed full-body bone-scan of a human subject. When the program in question is executed, it reads the given input image, as a matrix containing the grayscale intensity values for each pixel. These values are then passed through the following processing pipeline, to obtain an enhanced version of the given image:

1. Action of the Laplacian Operator:

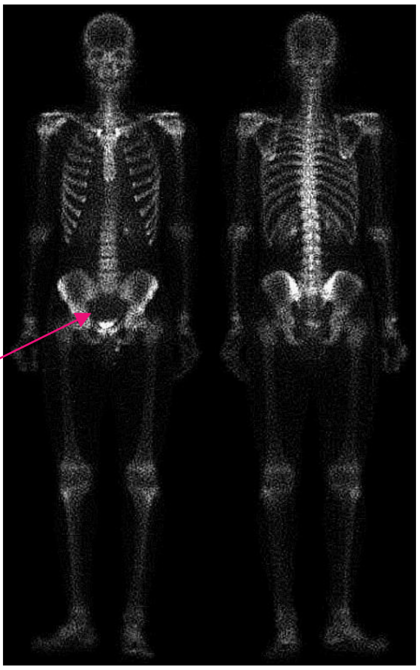
Here, an image containing the fine-details present in the original input image is obtained, by the operation of an 8-connected Laplacian Operator:

-1	-1	-1
-1	9	-1
-1	-1	-1

Laplacian operator

125	145	68
69	120	22
12	24	56

Part of Original Image



The obtained image is containing all the fine details, but is of pixelated nature.

2. Action of the Sobel Operator:

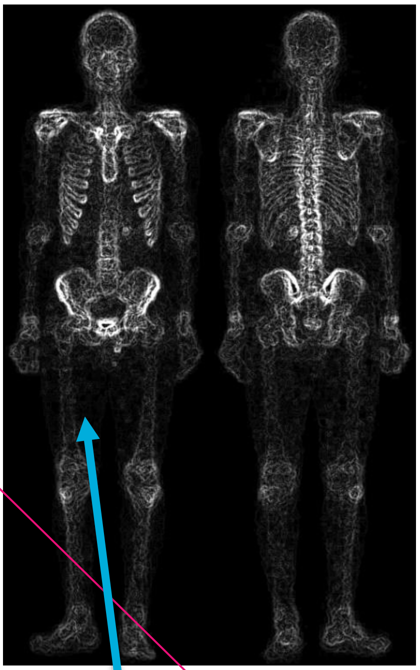
The Sobel operators are the tools for extracting the edges out of any given image. There is a matrix for each dimension of the image. Thus, for the case of our image, there are two matrices:

-1	0	1
-2	0	2
-1	0	1

G_x Sobel Operator

125	145	68
69	120	22
12	24	56

Part of Original Image



1	2	1
0	0	0
-1	-2	-1

G_y Sobel Operator

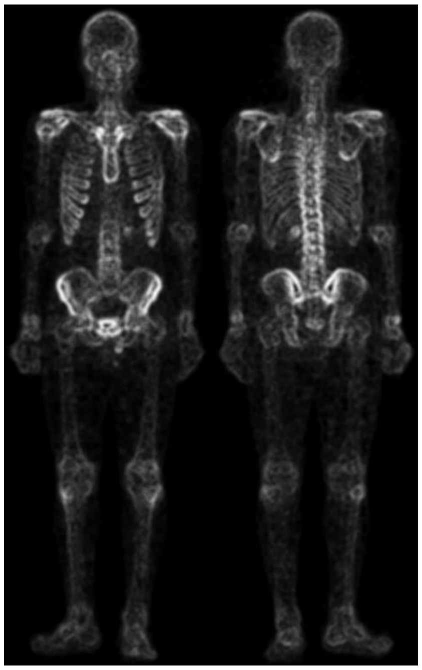
125	145	68
69	120	22
12	24	56

Part of Original Image

Pixel Value = $D_x + D_y$

3. Smoothened output of the Sobel Operator:

The output of the previous step provided us with the image that contains the edges present in the original image. But, in order to obtain the image mask, we need to apply an averaging smoothening filter on this image, so that the edges appear to blend-in with each other, and look a bit more realistic and continuous:

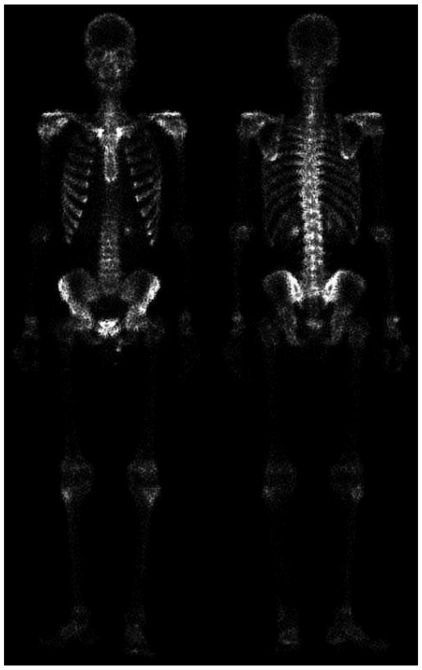


$$\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

19	123	150
225	100	34
33	89	17

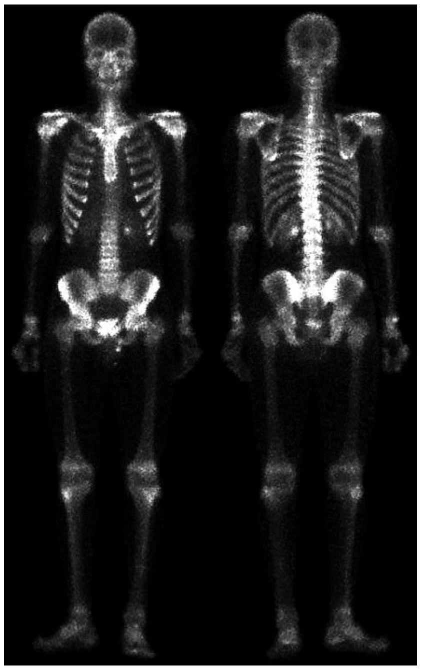
4. Obtaining the Image Mask:

The Image mask, which when added to the original image will return the enhanced image, is obtained by performing a pixelwise multiplication operation on the outputs of Step-2 and Step-4, i.e. the output of the Laplacian operator and the Smoothened output of the application of Sobel operator. The result, when scaled down appropriately, results in the image displayed alongside. As is clear from this image, it highlights all the places of interest, that contain the valuable details present in the original image, but in a brighter texture. These details are now ready to be placed back on the original image, in the next step.



5. Rendering the Enhanced Scan:

The image mask obtained in the previous step is overlaid on the input image, and the grayscale intensity values at each pixel for both the images are added together, to obtained the enhanced version of the input image. The obtained image is displayed alongside.



125	30	25
128	39	23
140	22	5

Image Mask

+

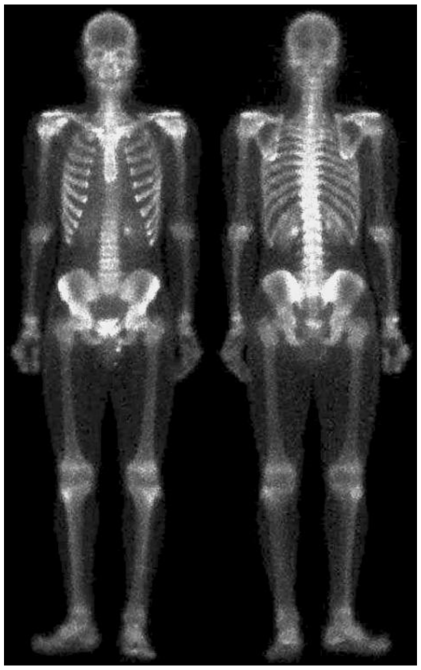
100	12	5
28	45	5
23	66	0

Original Image

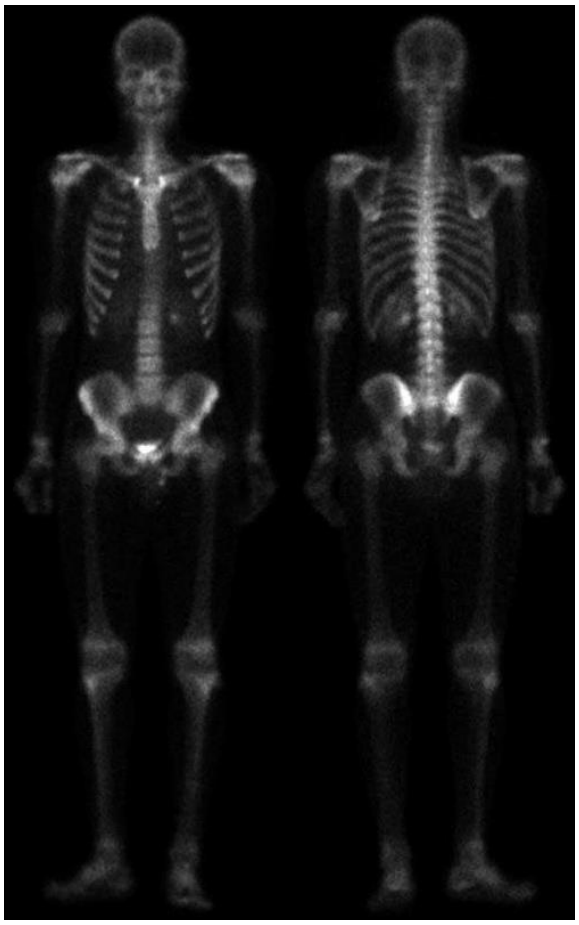
6. Performing the Power-Law Transformation:

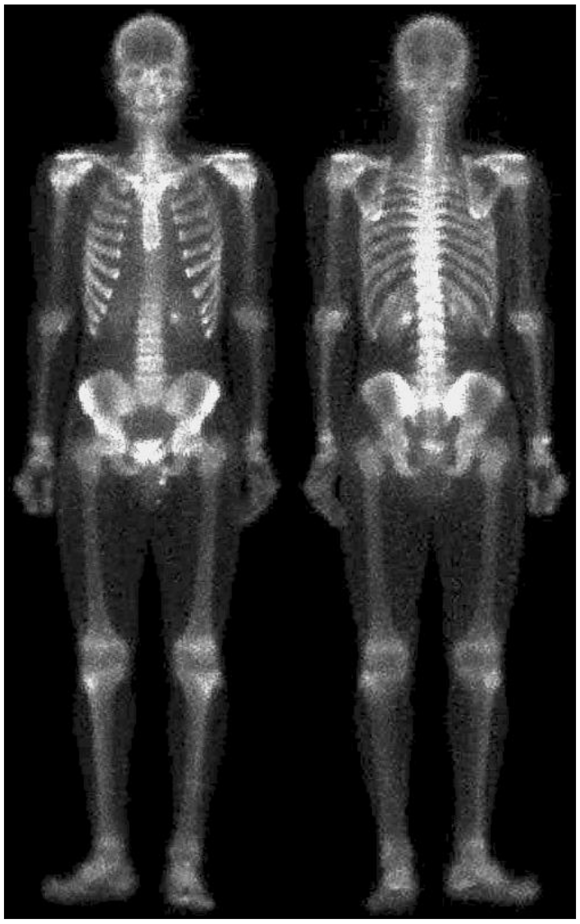
In this step, each pixel of the enhanced image is subjected to a power-law transformation, specifically designed to provide a wider range of values for the low-lit areas of the image, and conserve the brightness of the highly-lit areas. This further adds detail to the image, and reduces the dark regions.

$$\begin{matrix} Pixel \\ Value \end{matrix} = 15 \times \sqrt[2]{\begin{matrix} Input\ Pixel \\ value \end{matrix}}$$



As is evident from below, obtained output is far better for the purposes of analysis and diagnosis, due to its level of detail and better visual perception:

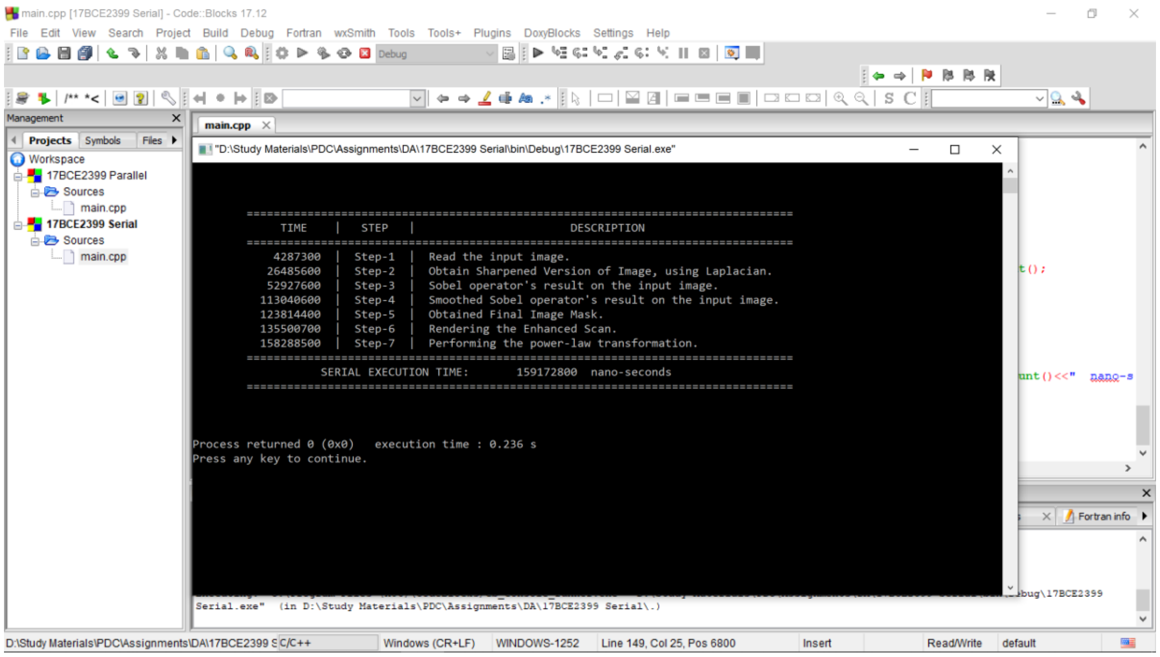

Original Image


Enhanced Image

PERFORMANCE ENHANCEMENT:

The program in question heavily deals with the image-processing concepts, which are historically known to mostly operate on each pixel value per image, without any correlation among the actions of any two pixels. Thus, they can be considered to be the ideal candidates for the parallel programming constructs, due to the sheer size of the problem domain, as for each operation, we are supposed to iterate through each-and-every pixel of all the input images, and perform the same operation. Thus, here we have used OpenMP to parallelize the programming iterations.

Each step is parallelized so that more than one pixels are processed at each clock cycle, the no. of which depends upon the no. of CPUs available in the given system. The system on which the program was executed possessed 4 CPUs, and the obtained performance is shown below:

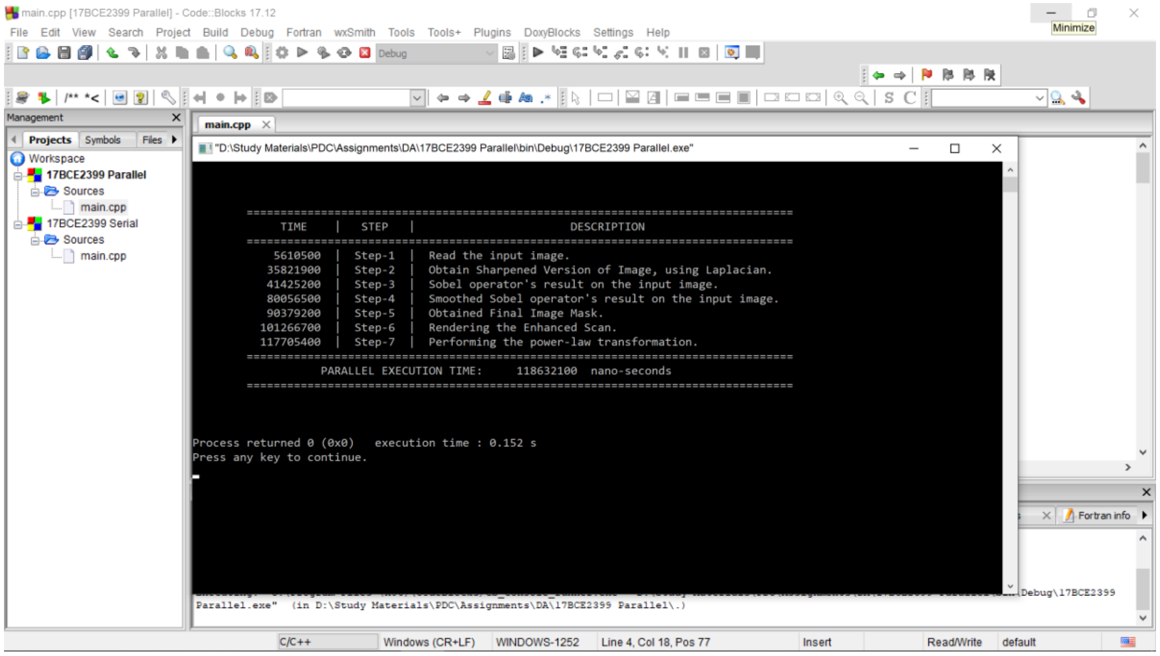


```
"D:\Study Materials\PDC\Assignments\DA\17BCE2399 Serial\bin\Debug\17BCE2399 Serial.exe"

        ================================================================================
             TIME      |   STEP    |                       DESCRIPTION
        ================================================================================
             4287300   |  Step-1   |  Read the input image.
            26485600   |  Step-2   |  Obtain Sharpened Version of Image, using Laplacian.
            52927600   |  Step-3   |  Sobel operator's result on the input image.
           113040600   |  Step-4   |  Smoothed Sobel operator's result on the input image.
           123814400   |  Step-5   |  Obtained Final Image Mask.
           135500700   |  Step-6   |  Rendering the Enhanced Scan.
           158288500   |  Step-7   |  Performing the power-law transformation.
        ================================================================================
                   SERIAL EXECUTION TIME:       159172800  nano-seconds
        ================================================================================


Process returned 0 (0x0)   execution time : 0.236 s
Press any key to continue.
```

Serial Execution



```
"D:\Study Materials\PDC\Assignments\DA\17BCE2399 Parallel\bin\Debug\17BCE2399 Parallel.exe"

        ================================================================================
             TIME      |   STEP    |                       DESCRIPTION
        ================================================================================
             5610500   |  Step-1   |  Read the input image.
            35821900   |  Step-2   |  Obtain Sharpened Version of Image, using Laplacian.
            41425200   |  Step-3   |  Sobel operator's result on the input image.
            80056500   |  Step-4   |  Smoothed Sobel operator's result on the input image.
            90379200   |  Step-5   |  Obtained Final Image Mask.
           101266700   |  Step-6   |  Rendering the Enhanced Scan.
           117705400   |  Step-7   |  Performing the power-law transformation.
        ================================================================================
                   PARALLEL EXECUTION TIME:     118632100  nano-seconds
        ================================================================================


Process returned 0 (0x0)   execution time : 0.152 s
Press any key to continue.
```

Parallel Execution

- **The Obtained Speedup:**

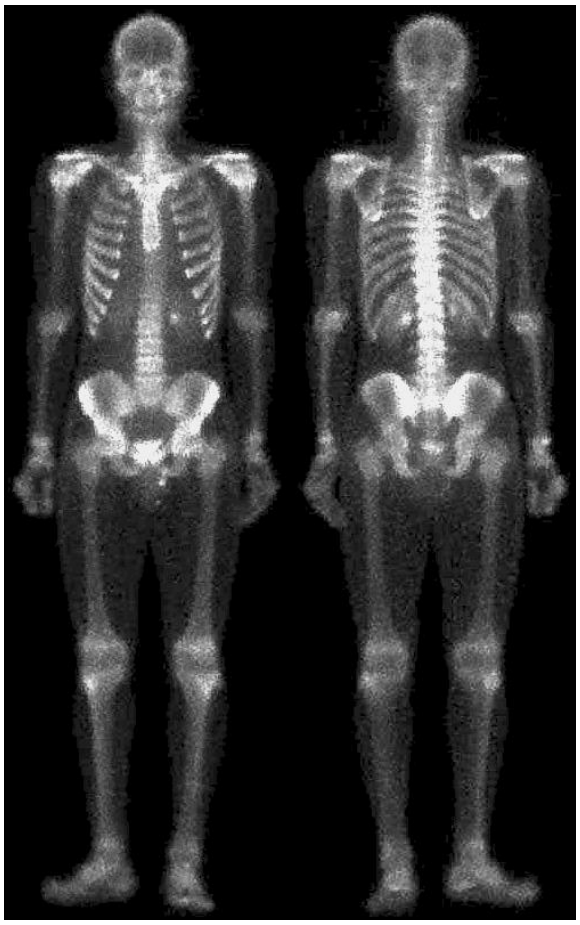
$$S = \frac{Serial\ Execution\ Time}{Parallel\ Execution\ Time} = \frac{159172800}{118632100} = \mathbf{1.34173465698}$$

- **Maximum Theoritical Speedup:**

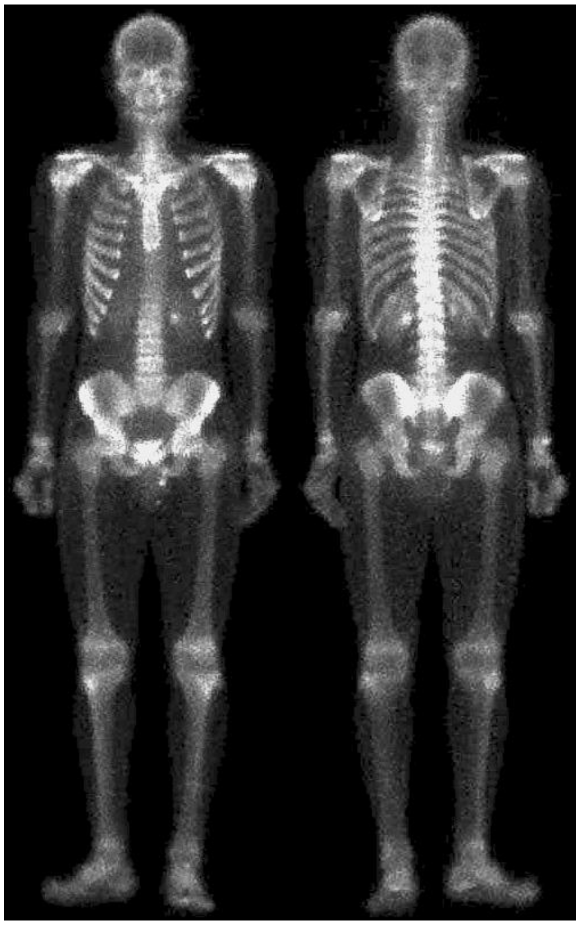
  By inspection of the source-code, it can be determined that around 40% of the code can be parallelized. Thus, the Maximum Theoretical Speedup for this program, according to the Amdahl's Law will be:

$$Speedup = \frac{1}{(1-p) + \ p/N} = \frac{1}{0.6\ +\ 0.4/4} = \mathbf{1.4285714}$$

Thus, we can see that the parallel version of the program is considerably better performance wise, and provides equivalent result:



Serial Execution Result



Parallel Execution Result