Measuring Processing time in your Matlab modules.

We will not be scrutinizing your functions frequently; however, we may ask you to show evidence of the processing times of certain parts of your program, during demonstration.
Especially, if we have the impression your program is running slow, we will surely ask you.

The processing time will be measured in average terms. A way to measure *elapsed* time is by using the functions tic() and toc(). This approach is good enough for measuring periods of time in order of milliseconds. However, it measures *elapsed time*, which will usually be higher than the actual CPU processing time used by your processing, because tic() and toc() do not consider if the operating system or the Matalb virtual machine gave your program the full time slice during the period of time between tic() and toc(). Still, on average, and if your PC is not very busy with other concurrent processes, tic() and toc() would be good enough for our purposes.
So, for measuring the processing time of certain module, e.g. when you call a function to process a LiDAR scan, you would use tic() and toc() in this way:

```
tic() ;
MyFunction(….)
T=toc()*1000;                %elapsed time, during execution of Function1()
%T = “processing time” , in milliseconds.
```

Processing times must not include processing dedicated to visualization matters (plots, etc.)

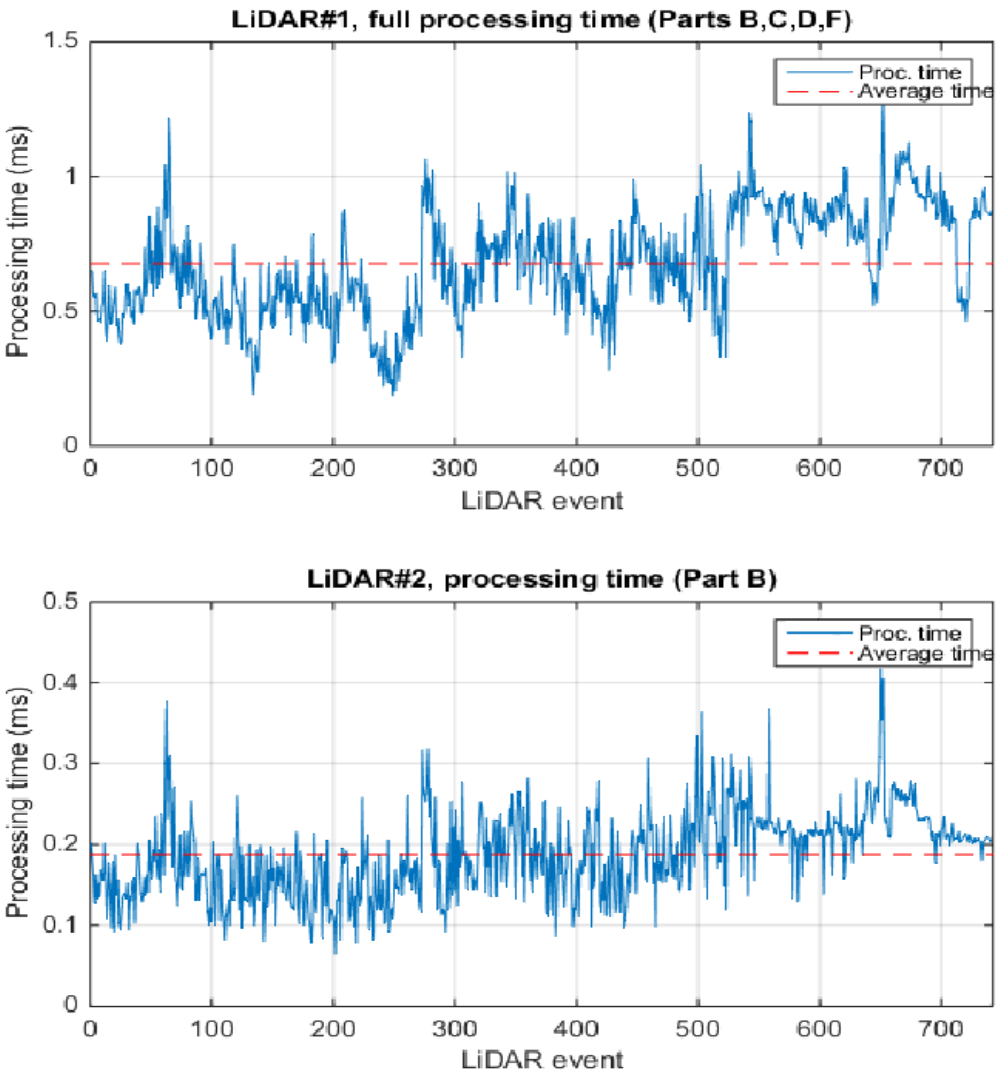

Figure 3. Example of “processing times” (excluded visualization), from the lecturer solution, in a laptop (i5, f=1.7Ghz, 4 Gb RAM, 2 cores, Matlab 2014b under Windows 7). The top subfigure shows the processing times for performing parts B,C,D and F applied to scans from LiDAR#1. The subfigure at the bottom shows the processing times for solving part B applied to scans from LiDAR #2. Both cases show pure processing times (i.e., excluding visualization).