Task 4 – Fourier Transform

Part A. FT Magnitude/Phase Mixer or Emphasizer

Design and implement a desktop program that explains the relative importance of the magnitude and phase components as well emphasizing the different contributions of each component in the signal. We will do this task on a 2D signal (i.e. image) just for clarity but the same concept applies to any signal. Your software should have the following features:

Images Viewers:

- Ability to open and view four gray scale images, each in one "viewport".
  a. **No Colored:** If the user opens a colored image, the program should convert it to grayscale.
  b. **One Size:** At any point of time, the sizes of the opened images have to be unified on the smallest size of all of them.
  c. **FT Components:** For each image, the program should have two displays (one fixed display for the image, while the second display can show several components based on a combo-box/drop-menu selection of 1) FT Magnitude, 2) FT Phase, 3) FT Real, and 4) FT Imaginary components.
  d. **Easy Browse:** The user can change any of the images by just double clicking on its viewer (Browse function for this specific image).
- **Two Output Ports:** The mixer result can be shown in one of two output viewports. Each output viewport is exactly similar to the input image viewport. The user should control in which viewport the new mixer result will show/resides.
- **Brightness/Contrast:** In any image viewport, the user can change the brightness/contrast (i.e. window/level) of the image via mouse dragging (up/down and left/right). This should be doable on any of the four components as well.

Components Mixer:

- Any output image is the ifft of a weight average of the FT of the input four images. The user should be able to customize the weights of each image FT via sliders. Think of how to do such customized weights for two components (e.g. magnitude and phase, or real and imaginary) in an intuitive user interface.

Regions Mixer:

- For each FT component, the user should be able to pick which region that will be taken for the output, inner region (i.e. low frequencies) or out region (i.e. high frequencies). This should be done by rectangle drawn on each FT, with an option for the user to include the inner or outer region, and the selected region highlighted (via semi-transparent coloring or hashing) to represent the selection. The size (or percentage) of region rectangle should be customizable for the user via a slider or resize handles. Note that this region should be unified for all the four images.

Realtime Mixing:

- The mixing process requires an ifft operation which takes some time to be performed. Such lengthy operation usually requires a progress bar shown for the user to indicate the update of the process.
- If the user changes the mixing settings and requests an update while the previous operation is still running, the program should just cancel the previous operation and start the new request one (Check *Threads*!)

Part B. Beamforming Simulator

Beamforming is a fundamental concept in nowadays modern technologies starting from wireless communications, 5G, radar, sonar to biomedical applications like ultrasound and tumor ablations. The core ideas of beamforming are delays/ phase-shifts and constructive/destructive interference. Do your own research and/or check the tutorials below (tutorial 1, tutorial 2, tutorial 3) to grasp the idea and inspired by the relevant available toolboxes (e.g., Matlab's and others) develop your own 2D beamforming simulator where the user can:

- Customize the parameters below to be able to steer the beam direction in real time.
    - Different system parameters like number of transmitters/receivers, applied delays/shifts, number of operating frequencies along with their values in real time.
    - The geometry of the phased array in terms of linear or curved (the curvature parameters should be customizable).
- Visualize the constructive/destructive map along with the beam profile in different synchronized viewers.
- Add multiple phased array units to the system and customize the location and parameters of each of them.

Equip your simulator with at least three different scenarios through parameter settings files that the user can directly open, visualize and customize or fine-tune. Your scenarios should be inspired by ideas from 5G, Ultrasound, tumor ablation.

Code Practice:

- Same practices from previous tasks (i.e. proper variable names & No code repetition) will continue in this task.
- **Apply OOP Concepts:**
    - Think of in terms of OOP. If you start implementing each image and its components, or each phased array, the code will have a lot of repetitions.
    - Apply the encapsulation concepts of OOP. Do not create a class for the image and its display and then start implementing everything outside! A very few codes in expected to show in your main function. Most of the code should be inside the image/display class or the phased array class! No mathematical manipulation for the image or the phased array should be handled outside the image class!
- **Logging:** Logging is a very important concept you should get used to. It helps you to track how problems happen and figure out their resolution much quicker. Python has a logging library. All you need to do is to import it and start logging the main user interactions and main steps into some text file. You need to show how logging has helped you to debug the problems you faced during your development. i.e. Do NOT just throw any trash variables to the log file to show you did it. You will be asked why you logged this variable or that.