ECSE 436 **McGill University**
Prof. Bajcsy **Fall 2016**

Individual Final Projects – Lab Exam

Due date: As scheduled by McGill, the (lab) final exam is to be held on **Dec. 15** in TR 4180.

Description: You will be presenting your lab implementation and doing demonstration of its hardware/software aspects, while answering accompanying questions. This will tests your acquired skills to implement a specific signal processing technique using signal processing hardware and software you have learned to use during this semester in the course.

Notes: (1) December 16 and 19 are the backup dates for people who want to present late.

(2) Your full final project report, printed copy as well as a PDF file, are due on Dec. 19 at midnight. There is a 15 percent penalty for each day you are late up to 2 days.

!!! Reports submitted after December 21 (midnight EST) will NOT be graded **!!!**

(3) **If your lab group has 3 members, you have to do all three projects, one per person. If your group has 2 people, you can choose two out of the following three projects. Do each implementation individually, fully acknowledging references used.**

1. Real-time Spectrum Analyzer

Implement a real-time spectrum analyzer that runs FFT (Fast Fourier Transform) algorithm on short windows of audio data using the cyclone FPGA board. In addition, it is to display the magnitude of the signal spectrum on the computer screen connected to the board.

(a) Algorithm: First perform your implementation in Matlab to analyze a given speech waveform. Use built-in FFT first, then write your DFT and eventually implement length 16 and 64 FFT on your own. Display the evaluated spectrum value-by-value on the screen, i.e., without using the *plot*, *stem* or *semilog* Matlab commands.

(b) Graphic interface: Using the FPGA board, generate an artificial image – a bar graph with 4 bars, each bar with a prescribed height. Consequently, output these onto the computer screen. Use board switches to adjust the heights of the bars.

(c) Real-Time Implementation: Port your FFT implementation from **(a)** into your FPGA board, so you can analyze samples in sub-frames of real-time speech data coming via the audio port from a microphone and/or desktop computer.

(d) Implement a routine to properly display the calculated spectrum values on the computer screen, label your axes appropriately and refresh the displayed spectrum frame by frame.

(e) Document your design in a write-up and prepare/perform the final demonstration of your system.

2. Image Processing Routine for Red-Eye Removal

This project is a bit more research oriented and involves Matlab implementation only. In particular, implement a red eye reduction routine for digital images that were taken with intense camera flash. Such images are affected by unnatural "red-eyes" (pupils) in one or more persons.

(a) Research the "red eye" reduction problem and its elimination methods in the literature. Select or design and implement a specific method that automatically detects red eyes in digital test images. Be careful about "misses" and "false" detection. (See notes below for additional details.)

(b) Once the red-eye/s in the image are detected, implement a follow-up algorithm/routine that replaces the un-natural red eyes in the image. The removal/replacement should be done automatically in such a way, that the image looks natural. (I.e., do not just use "0" value for R/G/B intensities.)

(c) Test and fine-tune/optimize your implementation on different test images containing faces only.

(d) Consequently, test and modify the routine to detect and correct red eyes in different images that include full person, not just a face.

(e) Fully document your system design in a write-up and prepare/perform the final demonstration of your system on various images. Summarize the obtained results of testing and explain its strengths and potential weaknesses of your method/implementation.

Approaches:

/i/ You can research through IEEE and other relevant image processing literature to find reference paper/s with specific red-eye reduction techniques. Then implement a chosen method, properly explaining its methodology, why and how it works, learning details of each of its sub-routines/algorithms. Explain what you have learned and implemented, plus what you have built. You can use support algorithms from Matlab image processing toolbox, but reference them as such and do understand how they work in order to answer questions about their functionality and origin.

/ii/ Alternatively, you can try to design your own method based on first detecting small circular, intense red objects in the image through appropriately configured adaptive thresholding and/or 2-D autocorrelation. (You can start using a fixed image of a face to detect small red circles of a given radius corresponding to eyes in this fixed image, then generalize the approach to unknown small radius.) Having detected a red eye, then write a separate routine to compensate/replace the unwanted redness due to the flash effect with appropriate image segment that appears natural.

3. Real-Time Error Control Decoding in Hardware

Consider the (16,5,8) Reed-Muller code discussed in class and used in Mariner Mars probes. Its decoding is to be done in the presence of both errors and erasures due to thresholded effects of AWGN noise.

(a) In Matlab, first implement the encoder and, in the presence of binary errors only, build a syndrome decoder for the (16,5,8) Reed-Muller code.

(b) Extend the architecture from **(a)** so it can handle both errors and erasures during the decoding step. First 'remove' symbols affected by erasures and syndrome decode errors in the reduced code. Consequently, after the errors were fixed, resolve the erasures as done in Lab 2.

(c) Test the overall system architecture from (a) and (b) to obtain bit-error vs. SNR curves for thresholded AWGN noise.

(d) Extend your implementation to the Altera FPGA board, so it can decode a stream of packets real-time corrupted by thresholded AWGN noise. Optimize your design to maximize the processing speed, in bits per second, when your board decodes arriving noisy packets.

(e) Document your design in a write-up and prepare/perform the final real-time demonstration of your FPGA system (e.g., with audio data).

NOTE: You will be provided you with the AWGN noise generating routine for your FPGA board. You will be responsible for writing the other individual 'matrix' modules and integrating them together.