Oct. 24, 2013

CEG4316
Lab Assignment #2
Filtering, Subsampling, Interpolation

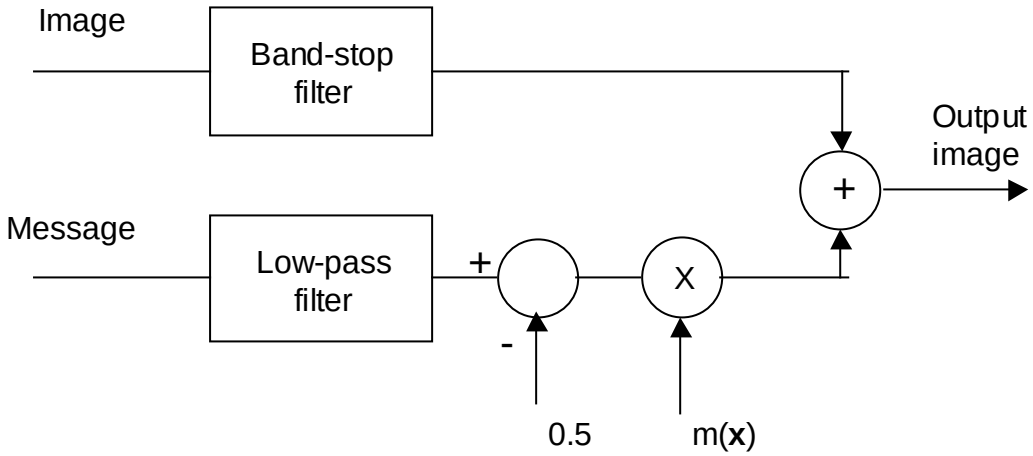
Due date: Your report is due on November 11 on the Blackboard Learn site. Results will be demonstrated during the lab periods on October 24, October 31 and November 7 in SITE 2060.

Objectives:

1. Decode a hidden image using filtering and demodulation.
2. Subsample and interpolate color image.

Procedure:

1. An image with another image hidden within it (message) has been created using the following system:



The modulation function is $m[n_1X, n_2X] = 0.0625\cos(2\pi(u_1n_1X + v_1n_2X))$ for a given modulation frequency (u_1, v_1). Both u_1X and v_1X are a multiple of 0.05. The image and the hidden image (message) are both sampled on a square lattice with spacing *X*, and are of size 512 by 512. The low-pass filter is non-separable with diamond-shaped pass-band and is designed using the window method. The band-stop filter has a stop-band centered at the frequency (u_1, v_1) and has a slightly larger bandwidth than the low pass filter. The output image is `portrait_code_2013.tif` .

Your task: Recover the hidden image, at baseband, from the output image. You first need to determine the modulation frequency using the power density spectrum estimation program `WelchPSD_comp.m`
In my encoder, I have used the two-dimensional low-pass filter $h_{LP;}$ the MATLAB code to generate it is given on the course page. Note that modulating a two-dimensional low-pass

filter unit-sample response with similar modulation functions (but different amplitudes) yields suitable band-pass filter unit-sample responses.

Can you see the effect of the hidden image in the original coded image? Can you remove that effect?

In your report, you should give a complete block diagram of your decoder, explain the theory for each block, and give the MATLAB statements that implement that block. You may show any frequency responses or power spectra that you find relevant. Display the decoded hidden image in your report. Report on who the portrait is of, and what is his claim to fame.

2. This problem explores subsampling and interpolation, as applied to the red, green and blue components of a color image. The standard Kodak color image '`stream.tif`', available on the course web page, will be used for this exercise.

a) Read the color image and convert it to `double` format. Extract each of the three color components and view them as grayscale images (e.g. `stream_red = stream(:,:,1);`). Subsample the red, green and blue components according to the corresponding Bayer sampling structures, as given in Section 2.2.2 of the course notes, by inserting zeros in place of the dropped samples. Make sure to maintain the correct sampling phase, so that only one component is present at each location in the original sampling lattice. Add the three subsampled images together to get the Bayer CFA image, and display it as a gray-scale image, and save it to a file. Compute and view the power spectral density of the CFA image using the routine `WelchPSD_comp.m`.

b) In this part, you will reconstruct the original color image from the Bayer CFA image. First initialize three zero images for the red, green and blue components and write the corresponding red, green and blue samples from the CFA image in the correct locations. Now, interpolate the three images using simple bilinear interpolation as shown in the presentation … /CEG4316/slides/FD_demosaicking_2013.pdf
Compute, view and compare the power spectral densities of each original component, each subsampled component and the interpolated components. Include plots and commentary for the red component and the green component.

c) Combine the three interpolated images to form a color image (e.g., `stream_interp(:,:,1) = stream_red_interp;`). Comment on the quality of this reconstructed image. Use your programs developed above to do Bayer demosaicking of the CFA image '`anchor_CFA.tif`' from the course web page.

d) Going back to the 'stream' image, prefilter the red, green and blue components before subsampling and mosaicking. Use the following separable filter:

```
h_pre_1D = [ -1 0 9 16 9 0 -1]/32; h_pre_2D = h_pre_1D' * h_pre_1D;
```

How does this affect the quality of the final demosaicked image? Could this be used in a practical digital camera?

e) Compare the performance of bilinear demosaicking with the adaptive frequency domain method as provided in the zip file `demos_freq_adapt_SPL.zip`. Note at least five differences between the results of c) and e) for the two images, (and compared to the original for ‘stream’). It may be helpful to superimpose the image windows on the screen and switch between them in place.

Usage:
```
CFA=im2double(imread('XXX_CFA.tif'));
OUT = demos_freq_adapt_SPL(CFA);
```