EE630 PROJECT D
SNAKE DETECTION USING FIR AND IIR FILTERS
by LGH
Updated 7-25-2013:

You will be graded on performing steps and answering questions (indicated by ?).

1. INPUT DATA

Given a bmp image HOLDPATFRED.bmp.

Figure 1: Mannequin Fred with triangular stripe patter projected at an angle from the camera’s optical axis.

Step 1: The image can be brought into MATLAB with the following code:

A_bmp=double(imread('HOLDPATFRED.bmp')); % load HOLDPATFRED.jpg image
Ar=A_bmp(:,:,1);
Ag=A_bmp(:,:,2);
Ab=A_bmp(:,:,3);

The image is B&W so the Ar=Ag=Ab. To plot Ar use imagesc(Ar) followed by colormap gray.

Step 1.1: Plot image of input data file.

2. SELECT SIGNAL COLUMN
Step 2.1: Select and plot, the $n_x$ column to represent a signal. Indicate which $n_x$ you used. The size of the image is $[M_y, N_x]$ = size(Ar); A 1-D vector would be $M_y$ by 1. $s_{c} = Ar(1:M_y, nx);$

Figure 2: The middle column intensity of Fig. 1.

Step 2.2 DFT the 1-D signal and isolate the sine wave region. That is, determine the boundaries of the region/s in the DFT domain that you believe represent the energy contribution of the sine wave projection. Plot the magnitude DFT spectrum (you may suppress dc). Indicate your boundary regions which you will be designing filters for in section 3.

3. DISCRETE TIME FILTER DESIGN

For all designs use the 1-D data in section 2 as input. Show results for each filter design in both time and frequency domains. Remember to design a lowpass and then frequency translate it to be a band pass filter.

3.1 Design an IIR using impulse invariance of a butterworth filter. Define your specifications and show design process of butterworth and why you chose the specific order $N$?

3.2 Design an FIR using the Kaiser window. Optimize to most closely match the spectral shape found in question 2.1?

3.3 Design an FIR using “frequency sampling approximation” of spectral region in question 2.1? What I mean by this is take the magnitude of the spectrum that represents the striping contribution and use it as a discrete time filter. Just window or section this region out and inverse DFT it to get the discrete-time impulse response or just use as is in the frequency domain.
3.4 Choose one of the filter designs and implement on entire 2-D image. Show results in both space and frequency domains. Take log of frequency domain to better visualize the response.

4. DFT SUBPIXEL TRANSLATION
No longer necessary

5. RECONSTRUCTION OF 3D SURFACE

Input the pattern set and G.byf files into MATLAB. Read about the mat5 format on LGH website under software-software 3D. See appendix for MATLAB code to do this.

Let the $k$th frequency, $n$th shifted pattern image captured by the camera is represented as $I_{k,n}(x,y)$ where $x$ and $y$ are the column and row pixel locations, respectively. The indices are $k = 0, (N_f-1)$ and $n = 0, 1, \ldots (N-1)$ where $N_f=2$ and $N=32$. The two frequencies are $k_0=1$ and $k_1=16$.

5.1 From the pattern set decode the albedo image C.bmp, use that as the I.bmp image and obtain the phase image YP.byf of the pattern set. To find the albedo image C.bmp, average the pattern images together and scale between 0 and 255;

$$I_C(x,y) = I_I(x,y) = \frac{1}{N_f N} \sum_{k=0}^{N_f-1} \sum_{n=0}^{N-1} I_{k,n}(x,y)$$  \hspace{1cm} (1a)

The performance matrix $I$, in I.bmp, is based on the modulation amplitude [1] of the patterns such that:

$$I_I(x,y) = \left[ \sum_{n=0}^{N-1} I_{k,n}(x,y)\cos(2\pi n/N) \right]^2 + \left[ \sum_{n=0}^{N-1} I_{k,n}(x,y)\sin(2\pi n/N) \right]^2$$  \hspace{1cm} (1b)

Let the $k$ value be that of the highest frequency pattern set.

An example is shown in Fig. 5.1.
Figure 5.1: Average value of all captured pattern images.

The phase image for the baseline \( k=0 \) and the high frequency \( k=1 \) are determined by

\[
\Phi_k(x, y) = \arctan \left[ \frac{\sum_{n=0}^{N-1} I_{k,n}(x, y) \sin(2\pi n / N)}{\sum_{n=0}^{N-1} I_{k,n}(x, y) \cos(2\pi n / N)} \right] + \pi
\]  

(2)

For implementation, you should use \( \text{atan2}(y,x) \) instead of \( \text{atan}(y/x) \) because \( \text{atan}() \) only yields values for 2 quadrants where as \( \text{atan2}() \) yields angles spanning all 4 quadrants. The \( \pi \) is added to make the phase go from 0 to \( 2\pi \) rather than \( -\pi \) to \( \pi \). The baseline phase is shown in Fig. 5.2 (left) and the high frequency phase is shown in Fig. 5.2 (right).

Figure 5.2: (left) Phase \( k=0 \) and (right) phase \( k=1 \) (\( k_c=16 \)).

If we take a cross section of both phase images in Fig. 5.2, we plot together, shown in Fig. 5.3, we can see that the \( k=0 \) phase is non-ambiguous where as the \( k=1 \) is wrapped. Both vary from 0 to \( 2\pi \).
In preparation to unwrap k=1, we normalize it by dividing by its pattern spatial frequency of kc = 16.

As seen in Fig. 5.4, the phase gradients of the (k=0) kc0=1 and (k=1) kc1=16 are about the same when normalized. We need to use k=0 phase to estimate how many integer multiples of 2\pi to align the k=1 phase with the k=0 phase. Once that is done, the k=0 phase is not included in the final phase estimate, thus removing its noise contribution.

The first step [2] is to obtain the integer number of phase unwrapping wavelengths (unwrapping cycles) that are at or below the desired value such that

$$\lambda(x, y) = \text{floor}(\frac{(k_1 \Phi_0(x, y) - \Phi_1(x, y))}{2\pi})$$

(3)
In units of cycles, we find the difference between the baseline phase and the unwrapping cycles such that

\[ r(x, y) = (k \cdot 1) \Phi_0(x, y) - \Phi_1(x, y) - 2\pi \lambda(x, y)/2\pi \]  

(4)

A final adjustment of +/- 1 unwrapping cycle is done by evaluating the remainder such that

\[ \lambda_u(x, y) = \begin{cases} 
\lambda(x, y) + 1 & \text{for } r(x, y) > 0.5 \\
\lambda(x, y) - 1 & \text{for } r(x, y) < -0.5 
\end{cases} \]  

(5)

The final phase image is then the unwrapping cycles plus the higher frequency phase such that

\[ \Phi(x, y) = (2\pi \lambda_u(x, y) + \Phi_1(x, y))/k \cdot 1 \]  

(6)

Thus, the \( k=1 \) phase is unwrapped and shown in Figs. 5.5 and 5.6.

![Figure 5.5: Unwrapped phase (green) superimposed on \( k=0 \) phase.](image)

![Figure 5.6: Unwrapped phase image.](image)

5.2 Choose a single spatial point, \( \{x, y\} \), and perform DFT analysis along the temporal dimension, \( n \), of the pattern set, both for the base frequency and the high frequency. See if you
can manually estimate a correction gamma for the spatial point you chose. Show mean square error of result (ie., generate a sine wave with estimated ,dc, amplitude, gamma and phase, then average the sum of the square difference between the estimated and the actual). Try to choose a point that contains a wide variation of values but does not saturate above 255. Ideally the best point is one that varies between 0 and 255.

Assume a model of
\[ I_{x,y}(n) = \beta_{x,y} \left( A_{x,y} + B_{x,y} \cos\left(2\pi k c n / N + \phi_{x,y}\right)\right)^\Gamma + I_{\text{ambient}} \]

Assume \( k_c=1 \) and \( I_{\text{ambient}} \) is the minimum value of your sequence and subtract it out such that
\[ I_{x,y}(n) = \left( A'_{x,y} + B'_{x,y} \cos\left(2\pi k c n / N + \phi_{x,y}\right)\right)^\Gamma \]

Where \( \beta \) is brought in as a multiplier by \( A \) and \( B \) (ie., you don’t have to know its value). The Gamma correction can be done after the fact by
\[ I_{x,y,\text{corrected}}(n) = \left( I_{x,y}(n)\right)^{1/\Gamma_{\text{correct}}} \]

Just DFT \( I_{x,y,\text{corrected}}(n) \) and vary \( 1<\Gamma_{\text{correct}}<4 \) by 0.001 to minimize the harmonic error in the DFT domain. The harmonics are where \( k \) is not equal to 2 or 32. Also, the harmonics should be normalized such that \( k=2 \) and \( k=32 \) have the same magnitude in your model and the pixel sequence. And \( k=1=dc \) should be zeroed out. Also note that \( A=B \). Another approach to gamma estimation is presented in reference [3]. From reference [3], the gamma is estimated from the harmonics as
\[ \Gamma = \frac{B_1 + 2B_2}{B_1 - B_2} \]

Where \( B_1 \) is the first harmonic amplitude and \( B_2 \) is the second harmonic amplitude. This approach is sensitive to noise but simple to implement.

So if \( I_{A,x,y}(n) \) is the real data and \( I_{x,y}(n) \) is your model data then the harmonic error is
\[ \varepsilon^2 = \frac{1}{32 - 3} \frac{\sum_{k=3}^{31} \left| I_{A,x,y}^{1}(k) - I_{x,y}^{1}(k) \right|^2}{\frac{1}{2} \left( \left| I_{A,x,y}^{1}(2) \right|^2 + \left| I_{A,x,y}^{1}(32) \right|^2 \right)} \]

5.4 Use the G file to determine the M matrix of coefficients (we will provide this for you). A linear triangulation paper will be provided with the assignment for explanation. Include M values in your report.

5.5 Use the M matrix of coefficients to reconstruct the 3-D surface (we provide this MATLAB code as well), save to mat5 format and then view in mat5 viewer. Use print screen to capture image for this report.
REFERENCES


APPENDIX: MATLAB

READ IN MAT5 (mat5read.m)

```matlab
% Veeraganesh Yalla
% template script to read the
% MAT5 files
% Date: Jan 30 2004
function [xw,yw,zw,imageI,imageC] = mat5read(matfile);
% open the world coordinate
```
% files
% the x,y,z are 1-D arrays
% need to be reshaped based on
% the dimensions of I and C images
fnamex = strcat(matfile,'X.byt');fnamey = strcat(matfile,'Y.byt');
fnamez = strcat(matfile,'Z.byt');fnameC = strcat(matfile,'C.bmp');
fnameI = strcat(matfile,'I.bmp');

fpx = fopen(fnamex,'rb');x = fread(fpx,'float');fclose(fpx);

fpy = fopen(fnamey,'rb');y = fread(fpy,'float');fclose(fpy);

fpz = fopen(fnamez,'rb');z = fread(fpz,'float');fclose(fpz);

%open the I and C images
imageC = imread(fnameC);imageI = imread(fnameI);

%get the dimensions
[my,nx,pz] = size(imageI);

%reshape the 1D world coordinate vectors
xw = reshape(x,nx,my)';
yw = reshape(y,nx,my)';
zw = reshape(z,nx,my)';

READ IN CALIBRATION FILE *G.byt (read_calib_data.m)

% read the calibration data
% function needs the input calibration grid data file '.byt'
% Npoints number of calibration points
function [coods] = read_calib_data(infname,Npoints)

fp = fopen(infname,'rb');
calib_data = fread(fp,'float');
close(fp);

%read the coordinate information
[m,n] = size(calib_data);
coods = calib_data(2:m);
coods = reshape(coods,7,Npoints)';

ASCII FORMAT OF calgridG.byt
Note that the data below is for a grid that has “Npoint” = 28 points. The “npoint” indicates the
last point set being edited and carries no information in itself. Each line of data or point set has 7
floating point values. Going from left to right, they are \(X_w, Y_w, Z_w, X_c, Y_c, X_p, Y_p\) where “w”
indicates a world coordinate, “c” indicates a camera coordinate and “p” indicates a projector
coordinate. Notice that \(X_p\)’s are all 0 because this is not used in the reconstruction of the 3-D data
for most applications.

Npoint=28
npoint=0
0.000000 0.000000 0.000000 937.007996 1490.270020 0.000000 4.174830
0.000000 15.875000 0.000000 935.841003 1260.900024 0.000000 3.619083
0.000000 31.750000 0.000000 934.781982 1027.890015 0.000000 3.059905
0.000000 47.625000 0.000000 933.473999 796.344971 0.000000 2.512760
15.875000 0.000000 12.700000 1178.099976 1512.079956 0.000000 3.981830
15.875000 15.875000 12.700000 1176.050049 1278.420044 0.000000 3.413457
15.875000 31.750000 12.700000 1178.459961 1040.170044 0.000000 2.843287
function [result] = mat5write(matfile,xw,yw,zw,imageI,imageC);

% open the world coordinate files
fnamex = strcat(matfile,'X.byt');fnamey = strcat(matfile,'Y.byt');
fnamez = strcat(matfile,'Z.byt');fnameC = strcat(matfile,'C.bmp');
fnameI = strcat(matfile,'I.bmp');
%get the dimensions
[my,nx,pz] = size(imageI);
%reshape the 1D world coordinate vectors
x = reshape(xw',1,my*nx)';y = reshape(yw',1,nx*my)';z = 
reshape(zw',1,nx*my)';
%xw
fpx = fopen(fnamex,'wb');fwrite(fpx,x,'float32');fclose(fpx);
%yw
fpy = fopen(fnamey,'wb');fwrite(fpy,y,'float32');fclose(fpy);
%zw
fpz = fopen(fnamez,'wb');fwrite(fpz,z,'float32');fclose(fpz);
%C
imwrite(imageC,fnameC,'bmp');imwrite(imageI,fnameI,'bmp');
%result = 1;