Mini-Project 3

Image Deblurring

3.1 Synopsis

Photographic image capture requires a finite amount of time to gather sufficient light energy to expose the image sensor. Relative motion between the camera and the subject during exposure causes motion blur that smears image details. To the casual observer the blur renders the captured image useless. However, *deconvolution* of the blurring mechanism can recover a close approximation of the original non-blurred image.

In this project you will be presented with an image that suffers from *uniform motion blur*, the special case of constant relative velocity between the camera and subject. With knowledge of the blurring mechanism you will create and implement an *inverse filter* to deblur the image.

Ulaby/Yagle Sections 8-4

3.2 Objectives

1. Work with image files; process and display images
2. Estimate motion blur with a mouse cursor measurement
3. Implement a stable inverse filter for a non-minimum phase system
4. Deblur an image with deconvolution

3.3 Deliverables

1. Hardcopy of all LabVIEW block diagrams and front panels that you create
2. Screen shots of requested images and plots
3. Deblurred image file (PNG format)
4. Lab report or notebook formatted according to instructor's requirements

3.4 Required Resources

1. NI LabVIEW with MathScript RT Module
2. Blurred image supplied by instructor (PNG format)

3.5 Preparation

Image handling in LabVIEW

Images are two-dimensional signals with horizontal and vertical dimensions. LabVIEW can work with images as either 2-D or 1-D arrays. For the sake of simplicity in this project the image signal will be treated as a 1-D signal — much like an audio signal — in which the image rows are laid end to end beginning at the top of the image. In this scheme the first array element is the upper left corner image pixel and the last array element is the lower right corner image pixel.

► Study the tutorial video `http://youtu.be/Z_81P1d_rCg` on image handling in LabVIEW; topics include reading and writing PNG image files, image display, obtaining image information, and extracting the image pixels as a 1-D array. You need only watch through time 5:00.

► Study the tutorial video `http://youtu.be/pw7-smkZh_U` to learn how to make the picture indicator automatically size itself to the image.

Image blurring

Your instructor has supplied you with a blurred image of a subject that was originally sharp and detailed. The blurred image resulted from uniform motion in the horizontal direction by an unknown distance of D pixels where D is an integer. Uniform motion blur is identical to the *moving average* (MA) system described in Section 7-3.4 of your text; see Equation 7.34 for the difference equation of the MA system. Substituting blurring distance D for the constant M, setting the b_i coefficients to 1, and $\mathbf{z}$-transforming this equation leads to the blurring system

$$\mathbf{H}(\mathbf{z}) = \frac{1}{D+1} \sum_{i=0}^{D} \mathbf{z}^{-i}. \tag{3.1}$$

Dividing by $D + 1$ ensures that the system has unity DC gain to keep the blurred image at the same average intensity as the original image.

Image analysis

The motion blur distance can be estimated by measuring the length of streaks traced out by the blur of small but distinct features.

► Create a new LabVIEW VI as a starting point for your image analysis application, and then load and display the blurred image provided by your instructor.

► Study the tutorial video `http://youtu.be/nVIOnz2WYyQ` to learn how to display mouse coordinates. Note that the while-loop and event structure need not enclose your existing code that reads the PNG image, but should instead be placed in its own area of the block diagram.

$\square_1$ Update your VI to implement the mouse coordinate display, and then estimate and report the blurring distance D.

3.6 Inverse Filter Design

The inverse filter (also called image deblurring filter) seeks to undo the effects of the image blurring filter. The inverse filter $\mathbf{G}(\mathbf{z})$ satisfies the property $\mathbf{G}(\mathbf{z})\mathbf{H}(\mathbf{z})$=1.

Basic inverse filter

► Create a new LabVIEW VI as a starting point for your image deblurring application, and then load and display the blurred image provided by your instructor.

$\square_2$ Study the tutorial video `http://youtu.be/JJZ-shHiayU` to learn how to write the blurring system $\mathbf{H}(\mathbf{z})$ (Equation 3.1 on the preceding page in closed form, i.e., without a summation symbol.

$\square_3$ Write the inverse filter system $\mathbf{G}(\mathbf{z})$ for your closed-form version of $\mathbf{H}(\mathbf{z})$.

► Study the video `http://youtu.be/IsbItFzzOdA` to learn how to translate a system transfer function into a MathScript structure; also study the section near the end of the video that shows how to connect the filter coefficient arrays for the a and b coefficient arrays to the LabVIEW `IIR_Filter.vi` subVI.

► Implement the system $\mathbf{G}(\mathbf{z})$ as a MathScript structure. Create an input for pixel distance D and two outputs for the a and b filter coefficient arrays. The MathScript `zeros` function comes in handy to implement the denominator coefficients; use the form `zeros(1,N)` to create a 1-D array of zeros of length N.

$\square_4$ Place the `Plot Pole-Zero` and `Plot Frequency Response` VIs (available at `https://decibel.ni.com/content/docs/DOC-24032` and `https://decibel.ni.com/content/docs/DOC-24021`) and connect these to the outputs of your MathScript structure for $\mathbf{G}(\mathbf{z})$. Study the video tutorials `http://youtu.be/S8WmBxIlnEc` and `http://youtu.be/IsjXautfUXs` to learn how to use these subVIs. Create default indicators for the frequency response magnitude and pole-zero plots of $\mathbf{G}(\mathbf{z})$. Study the effect of blurring distance D on the two plots and discuss your observations.

$\square_5$ Where are the poles of $\mathbf{G}(\mathbf{z})$ located in the pole-zero diagram? How do these pole locations manifest themselves in the frequency response plot?

Stable inverse filter

The basic inverse filter $\mathbf{G}(\mathbf{z})$ is not BIBO stable because the image blurring system $\mathbf{H}(\mathbf{z})$ is not minimum phase. Pulling the poles of $\mathbf{G}(\mathbf{z})$ slightly inside the unit circle makes the deblurring filter stable at the expense of introducing a slight mismatch between the original blurring filter and its inverse filter.

$\square_6$ Apply $\mathbf{z}$-transform property #5 ("Multiplication by a^n", also known as the $\mathbf{z}$-scaling property) of Table 7-6 in your text to $\mathbf{G}(\mathbf{z})$ by substituting each instance of $\mathbf{z}$ with $\mathbf{z}/R$ to form the stable inverse filter $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$ where R is the pole radius with $|R| < 1$.

► Update the MathScript structure to create the filter coefficients for $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$; add an additional input for R.

$\square_7$ Plot the pole-zero diagram and frequency response of $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$. Investigate values of R in the vicinity of 1.

Unit DC gain

The inverse filter requires a DC gain of 1 to ensure that the average brightness of the deblurred image remains the same as the original image.

$\square_8$ Calculate the DC gain of $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$, i.e., evaluate $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$ at $\mathbf{z} = e^{j0} = 1$.

► Update the MathScript structure to divide the existing $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$ by the DC gain you calculated in the previous step, thereby ensuring that $\mathbf{G}_{\mathrm{S}}(\mathbf{z})$ has unit DC gain for all values of D and R.

3.7 Image Deblurring

► Use the `IIR_Filter.vi` and the a and b coefficients produced by your MathScript structure to calculate the deblurred image.

► Place your deblurring system inside a combined event-structure and while-loop structure so that you can interactively adjust the blurring distance D and the pole radius R while observing the deblurred image. Refer to the tutorial video `http://youtu.be/eGlvOiqYVxg` for details.

► Begin with D set to your estimated blur distance and R set to 0.95. Iteratively adjust both parameters to deblur the image to the best of your ability. Strive for sharp (non-blurry) features, low noise, and minimum artifacts.

► Study the tutorial video `http://youtu.be/IDVHarNkJSU` to learn how to modify your event structure to save your image results to a file when you stop the while-loop. The video describes a general-purpose *load–process–save* design pattern that applies to a wide variety of signal processing tasks.

$\square_9$ Save your deblurred image as a PNG file, and report the values of D and R for this image.

3.8 Discussion

$\square_{10}$ Discuss your results, especially the nature of the noise and artifacts, and their relation to the parameters R and D. Consider including additional example images in your report to substantiate your discussion.

$\square_{11}$ Explain how the original image blurring process actually removes some information from the image, and how this impacts the design of the deblurring filter.