Lab 6: Edge Detection in Image and Video

Professor Deepa Kundur

Objectives of this Lab

- This lab introduces students to the basic image and video-processing task of edge detection. Software and hardware implementation and testing are conducted.

Prelab

- Prior to beginning, you must carefully read over this lab in order to plan how to implement the tasks assigned. Please highlight all the parts you need to show or answer for the TA, so that you do not miss any graded points during the in-lab component or for the report.
- Please download and bring to the lab the input source files available on the course webpage: TMW2.jpg, Linus.jpg, vipmem_Y.avi.
- **In addition, please bring your own JPEG file to do testing on.** Everyone has to have his or her own distinct image.

Deliverables

- Please show the TA all requested in-lab components for full points.
- After the lab, each individual must submit a separate report answering all the questions requested by the TA and asked in this Lab (see Questions section). For full points, please make sure you address all the parts in the lab that are required for the report.

Grading and Due Date

Please note STRICT DEADLINE for report on the course web site.

Lab 6: Edge Detection in Image and Video

TA: Mr. Lin Lai, Instructor: Dr. Deepa Kundur

Introduction and Background

Advances in digital image processing and digital video processing have opened incredible possibilities for computer vision applications. Research and development into common image and signal processing algorithms often employ a combination of multidimensional signal processing, discrete mathematics, topology, color theory, human perception and application-based physics modeling, among others. Algorithms range from "low-level" techniques such as image and video restoration and enhancement to abstract methodologies based on artificial intelligence for pattern recognition. Common challenges in designing image and video processing applications include managing computational complexity and scalability, achieving real-time performance given the volume of data that must be processed, mathematical modeling of a broad class of image and video signals that can vary greatly for different applications, and reliability and robustness to a broad range of applications.

Many practical image and video processing applications such as those requiring automated object recognition process the visual signals in a series of stages. There are several reasons for this. Breaking up a complex image/video processing task into smaller more specialized stages allows one to more easily replace a component with a more advanced algorithm. In addition, it allows each component to be somewhat separately designed, tested and improved for efficiency. Finally, if each stage is designed to reduce the volume of data to be processed by the subsequent stage, then real-time performance improvements can be gained.

One common image and video processing stage is edge detection. Edge detection is the process of identifying sharp changes in image brightness. This is important because it detects physical changes in the objects imaged. The image formation process in many applications such as seismology, photography, radar, microscopy and ultrasound is such that changes in acquired image brightness can relate to severe changes in depth/distance, discontinuities in surface orientation, changes in material properties, variations on scene illumination and boundaries of an object. Thus identifying edges can lead to better image understanding while reducing data volume for more efficient subsequent processing.

Edge Detection: 1-D Example

Edge detection algorithms can be grouped into one of two categories: *gradient-based edge detection* and *zero-crossing-based edge detection*. To elucidate the concept of edge detection, we present the steps for a one-dimensional example that makes use of a gradient-based approach.

Figure 1(a) presents a one-dimensional signal $f(x)$ for which we would like to identify "edges" (i.e., sharp changes in amplitude). Taking the gradient (in this case corresponding to a one-dimensional continuous-time derivative with respect to the independent variable x) to give $f'(x)$ results in the signal of Figure 1(b). Taking the absolute value of $f'(x)$ and then thresholding results in the detection of edges as seen in Figures 1(c) and (d). If the threshold is high, then fewer edges are identified. If the threshold is too low, then spurious edges may be detected.

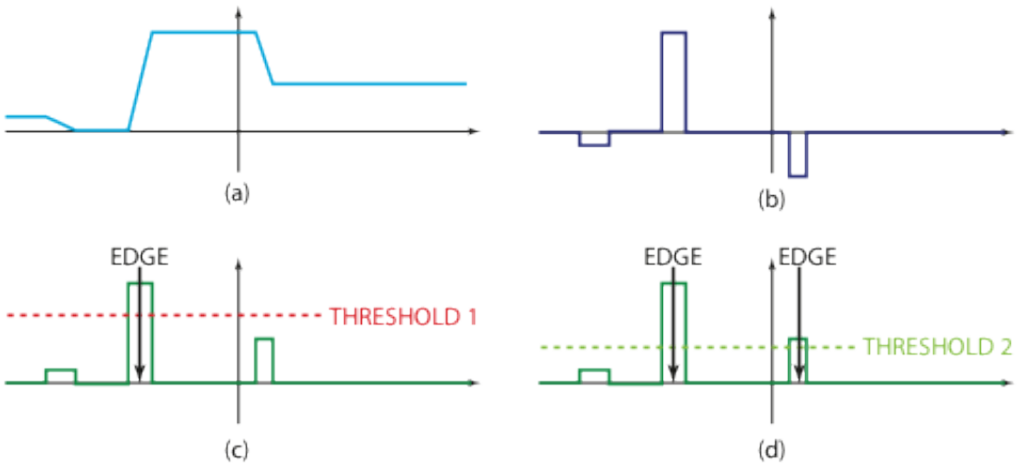

Figure 1: (a) One-dimensional signal denoted $f(x)$, (b) Gradient signal $f'(x)$, (c) Absolute value of gradient, $|f'(x)|$ and edge detection using Threshold 1, (d) Absolute value of gradient, $|f'(x)|$ and edge detection using Threshold 2.

Edge Detection in Images and Video

The procedure outlined for the one-dimensional case can be extended to two-dimensions (for digital images) or even three-dimensions (for digital video). Edge detection in digital images is typically conducted in the three steps outlined below:

1. Compute the gradient of the image frame. The gradient gives a measure of the difference in values amongst neighboring pixels better characterizing color changes in a localized region.
2. Compute the *edge strength* by taking the magnitude of the gradient. The gradient is more pronounced when the difference amongst adjacent pixel values is larger.
3. Threshold the result to identify the image edges. A smaller/larger threshold will detect fewer/greater edges. This "cleans up" the image resulting in a binary result that is easy to process for subsequent image interpretation stages.

An optional *thinning* stage may be applied after Step 3 to make sure that the edges are only a single pixel in width. The different edge detection algorithms used in practice differ in how the details of each step are implemented. For video processing, the edge detection is often conducted on a frame-by-frame basis independently.

Design and Implementation

In this lab, we shall implement edge detection, first on digital images, and then on digital video. The Sobel, Prewitt and Roberts edge detection approaches will be implemented and tested.

Edge Detection in Digital Images

The goal of this lab is to, in part, build and test the following Simulink model shown in Figure 2.

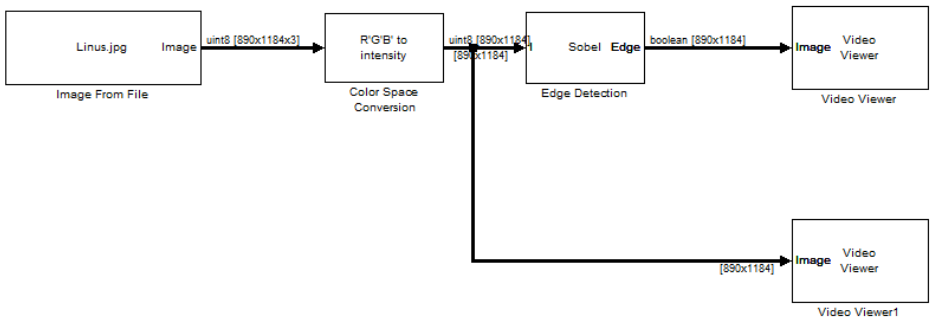

Figure 2: Edge detection Simulink Model to be implemented.

Building the Simulink Model

1. Add a source file for the edge detection from Video and Image Processing Blockset → Sources → Image From File. Set the filename to "TMW2.jpg", the sample time to "inf", the image signal to "One multidimensional signal", the output prot labels to "R|G|B" and the output data type to "Inherit from the input image."
2. Next convert the input color image from the file "TMW2.jpg" to grayscale using Video and Image Processing Blockset → Conversions → Color Space Conversion. Set the conversion to "R'G'B' to intensity ". Set the image signal to "Separate color signals".
3. Next add the Edge Detection block such that it takes input from the output of the Color Space Conversion block. Initially, set the method to "Sobel." Set the output type to "Binary image" and uncheck "User-defined threshold" and "Edge thinning". Set the Threshold scale factor to "1", the rounding mode to "Floor" and the overflow mode to "Wrap". Assign the product output to "Binary point scaling" model, the word length to "32" and fraction length to "8". Set the accumulator to "Same as product output". Finally, uncheck "Lock scaling against changes by the autoscaling tool".
4. Add two Video Viewer blocks from the Video and Image Processing Blockset and link them after Color Space Conversion block and Edge Detection block, separately, as shown in Figure 2.
5. Run the model. As you can see a processed image is shown where the image edges are displayed.
6. Now repeat the simulation here using the Linus image to experiment with the various algorithms using different thresholds.
    a. Change the edge detection method from Sobel to "Prewitt" and repeat the above steps for the two images and different thresholds. Please place you results in the report highlighting the thresholds used.
    b. Change the edge detection method to "Roberts" and repeat the above steps for the two images and different thresholds. Please place you results in the report highlighting the thresholds used.
7. Repeat the above steps with your own JPEG image.

Edge Detection in Digital Video

Now, we will extend our edge detection simulations to digital video case studies.

Building the Simulink Model

1. Start with your Simulink Model for digital image edge detection and replace the Image From File and Color Space Conversion blocks with an Add From Multimedia File block from the Video and Image Processing Blockset. Set the filename to "vipmem_Y.avi", check the "Inherit sample time from file", set the number of times to play file to "inf", uncheck "Output end-of-file indicator", set the image color space to "Intensity" and finally set the video output data type to "single".
2. Change back the edge detection method back to "Sobel". Use the same settings as for the Sobel image edge detection.
3. Replace the Video Viewer blocks each with a To Video Display block from the Video and Image Processing Blockset and make sure one is linked to the From Multimedia File block and the other to the Edge Detection block, separately.
4. Go to the Simulation→Configuration Parameters, change the solver options type to "variable steps."
5. Run the model to view the results.
6. Repeat the test with the "Prewitt" and "Roberts" edge detection methods.

Edge Detection in Digital Video Hardware Implementation

1. Go to the Simulation→Configuration Parameters, change the solver options type back to "Fixed-step", solver to "discrete (no continuous states)."
2. Start with your Simulink Model for digital video edge detection and delete all the input and output blocks.
3. Under Target Support Package→Supported Processors→TI C6000→Board Support→DM6437 EVM, found Video Capture block. Set Sample Time to -1.
4. Add two Video Display blocks. For the first one, set Video Window to "Video 0", Video Window Position to [180, 0, 360, 480], Horizontal Zoom to 2x, Vertical Zoom to 2x; for the second one, Set Video Window to "Video 1", Video Window Position to [0, 240, 360, 240], Horizontal Zoom to 2x, Vertical Zoom to 2x.
5. Add a Deinterleave block, and two Interleave blocks from DM6437 EVM Board Support. Link the Video Capture block to the Deinterleave block. And Link the two Interleave blocks each with a Video Display.
6. Add an Image Data Type Conversion block from the Video and Image Processing Blockset →Conversions. Set the output data type to "uint 8", and link its input with the output of the Edge Detection block.
7. Link the Edge detection Block and the Image Data Type Conversion block between the Y channel of Deinterleave block and the Interleave block for Video 0. And also link the output Y channel of Deinterleave block with the Interleave block for Video 1.
8. Link Cb and Cr channels between Deinterleave block and both of the Interleave blocks, as shown in Figure 3.
9. Adding a DM6437EVM Block under Target Support Package TI C6000→Target Preferences.

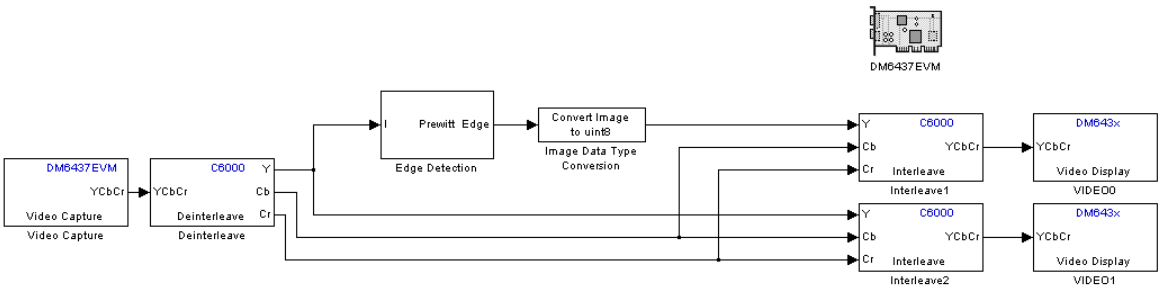

Figure 3: Edge detection in Digital Video Simulink Model to be implemented.

10. Connect the video output (the yellow port) of the camera with the video input of the board, and the output of the board with the monitor.
11. Go to Tools->Real Time Workshop -> Build the model.
12. Repeat the test with the “Prewitt” and “Roberts” edge detection methods.

Questions

The answers to these questions should be provided in the report for grading.

1. This questions requires you to research (i.e., look up) different edge detection algorithms. Compare and contrast the similarities and differences between the Sobel, Prewitt and Roberts edge detection algorithms.

2. Discuss your image and video simulation results of these three methods and discuss how they differ or are similar. Does this make sense?