****

***Jayawant Shikshan Prasarak Mandal***

**LAB MANUAL**

**OF**

**T E (COMP) 2012COURSE**

**Computer Engineering**

**1 Laboratory Equipment list**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Equipment details** | **Experiment performed** | **Cost** | **Status** |
| 1. | 8 GB RAM | NO | 5,500 Approx. | Not available |
| 2. | 500GB/1TB HDD | NO | 3,500 Approx. | Available |
| 3. | Web Camera | NO | 400 Approx. | Not available |
| 4. | ARM Cortex M4/A5 | NO | 600 Approx. | Not available |
| 5. | Oscilloscope | NO | 21350Approx. | Not available |
| 6. | Signal Generator | NO | 5,000Approx. | Not available |

**1.1 Laboratory Plan**

|  |  |  |
| --- | --- | --- |
| **Exp. No.** | **Title of Experiment** | **Session** |
|  | Menu Driven Image Processing Operations | I |
|  | Sampling a sign wave and reconstructing it using samples | II |
| **Mid semester assessment** | |  |
|  | Generate a square wave of programmable frequency | III |
|  | To capture a signal and perform various operations for analyzing it. | IV |
|  | To grab Image from camera and apply aged detection algorithm | V |
| **End semester assessment** | |  |
|  | **Mock oral** | XIII |

**2.3 EXPERIMENT NO.1**

**Menu Driven Image Processing Operations**

**Title:** Menu Driven Image Processing Operations

**Objectives**: To understand & implement various image processing operations on a remotely captured image.

**Aim:** Write C++ Program with GUI to capture using remotely placed camera and read

uncompressed TIFF Image to perform following functions (Menu Driven) Use of Overloading and Morphism is expected. Image Frame1 is used for displaying Original Image and Image Frame 2 is used for displaying the action performed.

* Sharpen the Image
* Blur the Image (Programmable rectangular Seed)
* Programmable image Contrast and Brightness
* Rotate image by programmable angle
* Convolution (overloading: FFT, Other)
* Histogram
* Mean and Standard Deviation of image
* PDF of a Signal acquired through ADC

**Requirements:**

**H/W Requirements:**

1. 8 GB RAM
2. 500GB/1TB HDD
3. Web Camera

**S/W Requirements:**

1. Latest version of 64 Bit Operating Systems Open Source Fedora-20.
2. Windows 8 with Multi core CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading

**Theory:**

* **TIFF (Tag Image File Format)**

TIFF (Tag Image File Format) is a common format for exchanging [raster graphics](http://searchcio-midmarket.techtarget.com/definition/raster-graphics) ([bitmap](http://searchcio-midmarket.techtarget.com/definition/bit-map)) images between applications programs, including those used for [scanner](http://whatis.techtarget.com/definition/scanner) images. A TIFF file can be identified as a file with a ".tiff" or ".tif" file name suffix. The TIFF format was developed in 1986 by an industry committee chaired by the Aldus Corporation (now part of Adobe Software). Microsoft and Hewlett-Packard were among the contributors to the format. One of the most common graphic image formats, TIFF files are commonly used in desktop publishing, faxing, 3-D applications, and medical imaging applications.

TIFF files can be in any of several classes, including gray scale, color [palette](http://searchsoa.techtarget.com/definition/palette), or RGB full color, and can include files with [JPEG](http://searchsoa.techtarget.com/definition/JPEG), LZW, or CCITT Group 4 standard run-length [image compression](http://searchcio-midmarket.techtarget.com/definition/image-compression).

#### Compression

Baseline TIFF readers must handle the following three compression schemes:[[8]](http://en.wikipedia.org/wiki/Tagged_Image_File_Format" \l "cite_note-tiff6-8)

* No compression
* [CCITT](http://en.wikipedia.org/wiki/ITU-T) Group 3 1-Dimensional Modified [Huffman](http://en.wikipedia.org/wiki/Huffman_coding) RLE
* [Pack Bits](http://en.wikipedia.org/wiki/PackBits) compression - a form of [run-length encoding](http://en.wikipedia.org/wiki/Run-length_encoding)

**TIFF Extensions**

Many TIFF readers support tags additional to those in Baseline TIFF, but not every reader supports every extension. As a consequence, Baseline TIFF features became the [lowest common denominator](http://en.wikipedia.org/wiki/Lowest_common_denominator) for TIFF format. Baseline TIFF features are extended in TIFF Extensions (defined in the TIFF 6.0 Part 2 specification) but extensions can also be defined in private tags.

The TIFF Extensions are formally known as TIFF 6.0, Part 2: TIFF Extensions. Here are some examples of TIFF extensions defined in TIFF 6.0 specification:

#### Compression

* CCITT T.4 bi-level encoding
* CCITT T.6 bi-level encoding
* LZW Compression scheme
* JPEG-based compression (TIFF compression scheme 7) uses the DCT (Discrete Cosine Transform) introduced in 1974 by [N. Ahmed](http://en.wikipedia.org/wiki/N._Ahmed), T.Natarajan and K.R. Rao; see Reference 1 in [Discrete cosine transform](http://en.wikipedia.org/wiki/Discrete_cosine_transform). [For more details see Adobe document](http://partners.adobe.com/public/developer/en/tiff/TIFFphotoshop.pdf).

#### Image types

* [CMYK](http://en.wikipedia.org/wiki/CMYK) Images
* [YCbCr](http://en.wikipedia.org/wiki/YCbCr) Images
* HalftoneHints
* Tiled Images
* [CIE L\*a\*b\*](http://en.wikipedia.org/wiki/CIE_Lab) Images

Many used TIFF images contain only uncompressed 32-bit [CMYK](http://en.wikipedia.org/wiki/CMYK_color_model) or 24-bit [RGB](http://en.wikipedia.org/wiki/RGB_color_model) images.

#### Image Trees

A baseline TIFF file can contain a sequence of images (IFD). Typically, all the images are related but represent different data, such as the pages of a document. In order to explicitly support multiple views of the same data, the SubIFD tag was introduced.[[17]](http://en.wikipedia.org/wiki/Tagged_Image_File_Format#cite_note-17) This allows the images to be defined along a [tree structure](http://en.wikipedia.org/wiki/Tree_structure). Each image can have a sequence of children, each child being itself an image. The typical usage is to provide thumbnails or several versions of an image in different color spaces.

#### Other extensions

According to TIFF 6.0 specification (Introduction), all TIFF files using proposed TIFF extensions that are not approved by Adobe as part of Baseline TIFF (typically for specialized uses of TIFF that do not fall within the domain of publishing or general graphics or picture interchange) should be either not called TIFF files or should be marked some way so that they will not be confused with mainstream TIFF files.

**Private tags**

Developers can apply for a block of "private tags" to enable them to include their own proprietary information inside a TIFF file without causing problems for file interchange. TIFF readers are required to ignore tags that they do not recognize, and a registered developer's private tags are guaranteed not to clash with anyone else's tags or with the standard set of tags defined in the specification.

TIFF Tags numbered 32768 or higher, sometimes called private tags, are reserved for information meaningful only for some organization or for experiments with a new compression scheme within TIFF. Upon request, the TIFF administrator (Adobe) will allocate and register one or more private tags for an organization, to avoid possible conflicts with other organizations. Organizations and developers are discouraged from choosing their own tag numbers, because doing so could cause serious compatibility problems. However, if there is little or no chance that TIFF files will escape a private environment, organizations and developers are encouraged to consider using TIFF tags in the "reusable" 65000-65535 range. There is no need to contact Adobe when using numbers in this range.

**TIFF Compression Tag**

The TIFF Tag 259 (010316) stores the information about the Compression method. The default value is 1 = no compression.

Most of TIFF writers and TIFF readers support only some of existing TIFF compression schemes. Here are some examples of used TIFF compression schemes:

**Overloading**

It allows [functions](http://cplus.about.com/od/introductiontoprogramming/g/functiondefn.htm) in computer languages such as [C++](http://cplus.about.com/od/glossary/g/Cppdefn.htm) and [C#](http://cplus.about.com/od/introductiontoprogramming/g/csharpdefn.htm) to have the same name but with different parameters.

For example, rather than have a differently named function to sort each type of array

Sort\_Int(Int Array Type) ;

Sort Doubles ( Double Array Type) ;

Instead use the same name with different parameter types.

Sort(Int Array Type ) ;

Sort( Double Array Type ) ;

The Complier is able to call the appropriate function depending on the [parameter](http://cplus.about.com/od/glossar1/g/parametersdefn.htm) [type](http://cplus.about.com/od/introductiontoprogramming/g/typedefn.htm)

**Morphism**

In programming languages, polymorphism means that some code or operations or objects behave differently in different contexts.

For example, the + (plus) operator in C++:

4 + 5 <-- integer addition

3.14 + 2.0 <-- floating point addition

s1 + "bar" <-- string concatenation!

In C++, that type of polymorphism is called overloading.

Typically, when the term polymorphism is used with C++, however, it refers to using virtual methods, which we'll discuss shortly.

**Procedure:**

* 1. Make connection between local machine and web camera.
  2. Capture TIFF Image with Web camera remotely.
  3. Display captured image in Image Frame 1
  4. Perform Menu Driven Operations on Image Frame 2
  5. Apply different operations on image like

1. Sharpen the Image
2. Blur the Image (Programmable rectangular Seed)
3. Programmable image Contrast and Brightness
4. Rotate image by programmable angle
5. Convolution (overloading: FFT, Other)
6. Histogram
7. Mean and Standard Deviation of image
8. PDF of a Signal acquired through ADC
   1. Analyze the both Images of frame 1and frame 2.

**Mathematical Model:**

1. Let S be a system that describes the Different Operations on Image.

S = {…..}

2. Identify input as I

S = {I,N,…..}

Ii € I

Ii = The inputs to the system such compressed TIFF image.

N =Number of Images

3. Identify Output as O.

S = {I,N,O,…..}

O = Depend upon the respective operation selected.

4. Identify the process as P.

S = {I,N,O, P,……}

P ={Ps, Pb, Pc,Pr }

Ps = The process of sharpening the image.

Pb = The process of Bluring the image..

Pc = The process of Contrast and brightness on the image.

Pr = The process of rotating the image.

5. Identify success as Su

S = {I,N,O, P, Su, F……}

Su = Success is defined when Proper operation takes place as on requirement.

F = When improper operation is done on image.

6. Identify Initial condition Si

S = {I, N, O, P, Su, Si,….}

The user will capture image properly and will be able to do menu driven operations on Image

S = I U N U O U P

**Observations:**

1. Sharpening of the Image is done
2. Blur the Image (Programmable rectangular Seed) is done
3. Programmable image Contrast and Brightness is done
4. Rotate image by programmable angle is done
5. Convolution (overloading: FFT, Other) is done
6. Histogram is done
7. Mean and Standard Deviation of image is done
8. PDF of a Signal acquired through ADC is done

**Results:**

Sharpen the Image

****

1. Blur the Image

****

1. **Contrast and Brightness of Image**

****

**Conclusion:**

Understanding &Implementation of various image processing operations on a remotely captured image is done successfully.

**Oral Question Bank**

**Theory Question Bank**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Q. No** | **Description** | **Attainment of** | | | |
| **CO** | **PO** | **BL\*** | **GA** |
| **1.** | Distinguish DTFT and DFT | **CO4** | **PO1** | Description: DB13.bmp |  |
| **2.** | Distinguish between DIT and DIF algorithm. | **CO4** | **PO1** | Description: DB13.bmp |  |
| **3.** | Evaluate how many stages are there for 8 point DFT? | **CO4** | **PO3** | Description: DB13.bmp |  |
| **4.** | Evaluate how many multiplication terms are required for doing DFT by expressional method and FFT method? | **CO4** | **PO4** | Description: DB11.bmp |  |
| **5.** | Define histogram. | **CO3** | **PO5** | Description: DB11.bmp |  |
| **6.** | Describe convolution of the signals? | **CO4** | **PO4** | Description: DB13.bmp |  |
| **7.** | Define what do you mean by periodic and non-periodic signals | **CO4** | **PO3** | Description: DB11.bmp |  |
| **8.** | Define the properties of a system? | **CO4** | **PO3** | Description: DB11.bmp |  |
| **9.** | Define image compression | **CO2** | **PO3** | Description: DB11.bmp |  |
| **10.** | Define different Compression Methods | **CO2** | **PO4** | Description: DB11.bmp |  |
| **11.** | Define image compression? Explain any four variable length coding compression schemes. | **CO2** | **PO4** | Description: DB11.bmp |  |
| **12.** | Describe image compression | **CO2** | **PO5** | Description: DB13.bmp |  |
| **13.** | Explain warping effect? | **CO2** | **PO5** | Description: DB13.bmp |  |
| **14.** | Define pre warping. | **CO2** | **PO3** |  |  |
| **15.** | Explain how phase distortion and delay distortion are introduced | **CO2** | **PO3** | Description: DB13.bmp |  |
| **16** | Define correlation | **CO4** | **PO3** | Description: DB11.bmp |  |
| **17** | Define aliasing effect | **CO4** | **PO4** | Description: DB11.bmp |  |
| **18** | Define symmetric and anti-symmetric signals. How do you prevent aliasing while sampling a CT signal? | **CO4** | **PO5** | Description: DB11.bmp |  |
| **19** | Define linear convolution of two DT signals | **CO4** | **PO6** | Description: DB11.bmp |  |
| **20** | Define system function and stability of DT system | **CO4** | **PO3** | Description: DB11.bmp |  |
| **21** | Define the following (a) System (b) Discrete-time system | **CO4** | **PO4** | Description: DB11.bmp |  |
| **22** | Define Overloading | **CO4** | **PO5** | Description: DB11.bmp |  |
| **23** | Define morphism | **CO4** | **PO3** | Description: DB11.bmp |  |
| **24** | Define Mean | **CO4** | **PO4** | Description: DB11.bmp |  |
| **25** | Define Standard Deviation | **CO4** | **PO5** | Description: DB11.bmp |  |
| **26** | Cite the classifications of discrete-time system. | **CO4** | **PO6** | Description: DB13.bmp |  |
| **27** | Describe property of shift-invariant system? | **CO4** | **PO5** | Description: DB13.bmp |  |
| **28** | Define (a) Static system (b) Dynamic system? | **CO4** | **PO5** | Description: DB11.bmp |  |
| **29** | Describe what is an anti-aliasing filter? | **CO4** | **PO4** | Description: DB13.bmp |  |
| **30** | State the condition for a digital filter to be causal and stable | **CO4** | **PO4** | Description: DB11.bmp |  |
| **31** | Describe zero padding? What are its uses? | **CO4** | **PO4** | Description: DB13.bmp |  |
| **32** | Describe A/D conversion noise? | **CO4** | **PO4** | Description: DB13.bmp |  |
| **33** | Define circular convolution | **CO4** | **PO3** | Description: DB11.bmp |  |
| **34** | Distinguish between linear convolution and Circular Convolution. | **CO4** | **PO3** | Description: DB13.bmp |  |
| **35** | State the circular time shifting and circular frequency shifting properties of DFT | **CO4** | **PO2** | Description: DB11.bmp |  |
| **36** | State the time reversal property of DFT | **CO4** | **PO2** | Description: DB11.bmp |  |
| **37** | Cite advantage of direct and cascade structures | **CO4** | **PO1** | Description: DB13.bmp |  |
| **38** | Discuss the properties of DFT | **CO4** | **PO1** | Description: DB13.bmp |  |
| **39** | Explain the use of FFT algorithms in linear filtering and correlation | **CO4** | **PO2** | Description: DB13.bmp |  |
| **40** | Describe A/D conversion mode | **CO4** | **PO5** | Description: DB13.bmp |  |
| **41** | Describe product quantization error | **CO4** | **PO5** | Description: DB13.bmp |  |
| **42** | Cite the effect of quantization on pole locations | **CO4** | **PO5** | Description: DB13.bmp |  |
| **43** | Explain zero input limit cycle overflow oscillation | **CO4** | **PO5** | Description: DB13.bmp |  |
| **44** | Describe what is meant by rounding? Draw the pdf of round off error | **CO4** | **PO5** | Description: DB13.bmp |  |
| **45** | Define What is meant by truncation? Draw the pdf of round off error | **CO4** | **PO4** | Description: DB11.bmp |  |
| **46** | Explain what do you mean by quantization step size? | **CO4** | **PO4** | Description: DB13.bmp |  |
| **47** | Cite different types of frequency domain coding | **CO4** | **PO4** | Description: DB13.bmp |  |
| **48** | Describe subband coding? | **CO4** | **PO3** | Description: DB13.bmp |  |
| **49** | Cite properties of DT sinusoids | **CO4** | **PO3** | Description: DB13.bmp |  |
| **50** | Explain what is the causality condition for an LTI system? | **CO4** | **PO3** | Description: DB13.bmp |  |

**EXPERIMENT NO.2**

**Sampling a sign wave and reconstructing it using samples**

**Title:** Sampling a sine wave and reconstructing it using samples

**Objectives:** To understand & implement Sampling of a sine wave and reconstructing it using samples

**Aim:** Write a C++/ Python program to generate a sine wave of Programmable frequency and capture samples at programmable frequency (Max up as per Nyquist Sampling Theorem) and reconstruct the sine wave using collected Samples using ARM Cortex A5/A9. Use oscilloscope to calculate signal frequency. Write your observations .Store a Data file in SAN (BIGDATA)

**Requirements:**

**H/W Requirements:**

1. 8 GB RAM
2. 500GB/1TB HDD
3. ARM Cortex M4/A5
4. Oscilloscope
5. Signal Generator

**S/W Requirements:**

1. Latest version of 64 Bit Operating Systems Open Source Fedora-20.
2. Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading

**Theory:**

Nyquist-Shannon sampling theorem:

In the field of [digital signal processing](http://en.wikipedia.org/wiki/Digital_signal_processing), the **sampling theorem** is a fundamental bridge between [continuous signals](http://en.wikipedia.org/wiki/Continuous_signal) (*analog* domain) and [discrete signals](http://en.wikipedia.org/wiki/Discrete_signal) (*digital* domain). Strictly speaking, it only applies to a class of [mathematical functions](http://en.wikipedia.org/wiki/Mathematical_function) whose [Fourier transforms](http://en.wikipedia.org/wiki/Continuous_Fourier_transform) are zero outside of a finite region of frequencies (see Fig 1). The analytical extension to actual [signals](http://en.wikipedia.org/wiki/Signal_(electrical_engineering)), which can only approximate that condition, is provided by the [discrete-time Fourier transform](http://en.wikipedia.org/wiki/Discrete-time_Fourier_transform), a version of the [Poisson summation formula](http://en.wikipedia.org/wiki/Poisson_summation_formula).  Intuitively we expect that when one reduces a continuous function to a discrete sequence (called *samples*) and [interpolates](http://en.wikipedia.org/wiki/Interpolates) back to a continuous function, the fidelity of the result depends on the density (or [sample-rate](http://en.wikipedia.org/wiki/Sampling_(signal_processing))) of the original samples. The sampling theorem introduces the concept of a sample-rate that is sufficient for perfect fidelity for the class of [band limited](http://en.wikipedia.org/wiki/Bandlimited) functions; no actual "information" is lost during the sampling process. It expresses the sample-rate in terms of the function's bandwidth. The theorem also leads to a formula for the mathematically *ideal* interpolation algorithm.

**Sampling**

So what is sampling, and what does it do? Sampling is the process by which continuous time signals, such as voltages or water levels or altitudes, are turned into discrete time signals. This is usually done by translating the signal in question into a voltage, then using Tim Wescott 1 Westcott Design Services. What Nyquist Didn’t Say, and What to Do About It

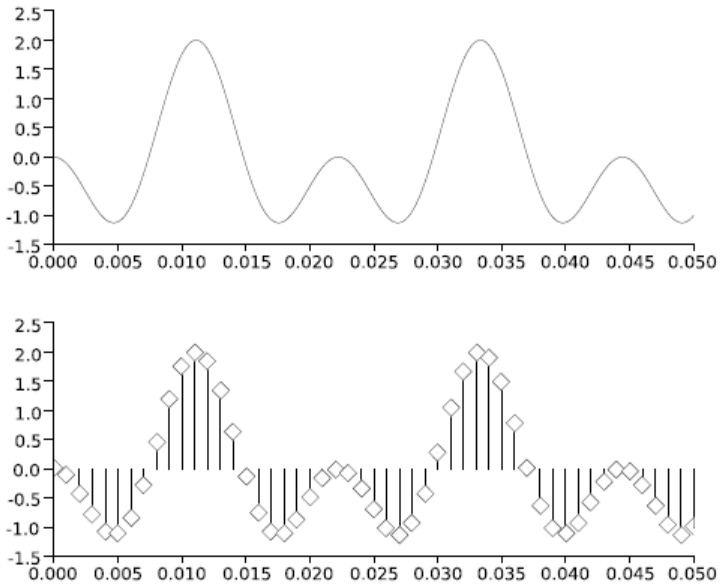


Figure 1: The results of Sampling an analog to digital converter (ADC) to turn this continuous, analog signal into a discrete, digital one. The ADC both samples1 the voltage and converts it to a digital signal. The sampling process itself is easy to represent mathematically: given a continuous signal x(t) to be sampled, and a sample interval T, the sampled version of x is simply the continuous version of x taken at integer intervals of T:

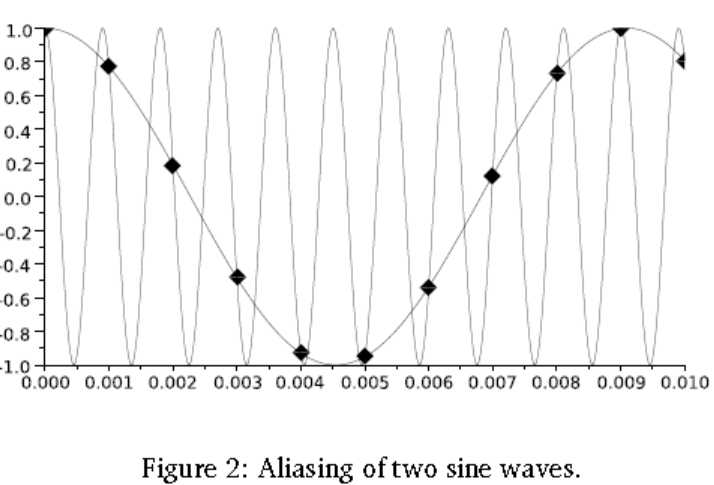


Figure 1 shows the result of sampling a signal. The upper trace is the continuous-time signal, while the lower trace shows the signal after being sampled once per millisecond. You may wonder why the lower trace shows no signal between samples. This is because after sampling there is no signal between samples — all the information that existed between the samples in the original signal is irretrievably lost in the sampling process.

**Aliasing**

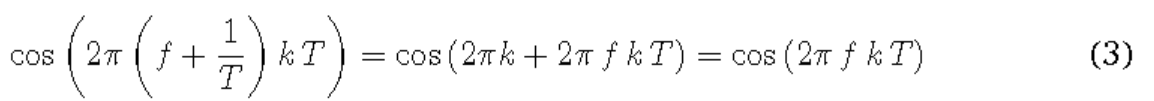
By ignoring anything that goes on between samples the sampling process throws away information about the original signal2. This information loss must be taken into account during system design. Most of the time, when folks are designing systems they are doing their analysis in the frequency domain. When you are doing your design from this point of view you call this effect aliasing, and you can easily express it and model it as a frequency domain phenomenon. To understand aliasing, consider a signal that is a pure sinusoid, and look at it’s sampled version:





xk = cos (w k T) (2)

If you know the frequency of the original sine wave you’ll be able to exactly predict the sampled signal. This is a concept is easy to grasp and apply. But the sampled signal won’t necessarily seem to be at the same frequency as the original signal: there is an ambiguity in the signal frequency equal to the sampling rate. This can be seen if you consider two signals, one at frequency f and one at frequency f + 1=T. Using trigonometry, you can see that the sampled version of these two signals will be exactly the same:



This means that given a pair of sampled versions of the signals, one of the lower frequency sinusoid and one of the higher, you will have no way of distinguishing these signals from one another. This ambiguity between two signals of different frequencies (or two components of one signal) is aliasing, and it is happening all the time in the real world, anywhere that a real-world signal is being sampled.

Figure 2 shows an example of aliasing. Two possible input sine waves are shown: one has a frequency of 110Hz, the other has a frequency of 1110Hz. Both are sampled at 1000Hz.

The dots show the value of the sine waves at the sampling instants. As indicated by (1) these two possible inputs both result in exactly the same output: after sampling you cannot tell these two signals apart. It is rare, however, for real-world signals to resemble pure sine waves. In general, real world continuous-time signals are more complex than simple sine waves. But we can use what we learn about the system’s response to pure sine wave input to predict the behavior of a system that is presented with a more complex signal. This is because more complex continuous-time signals can be represented as sums of collections of sine waves at different frequencies and amplitudes. For many systems we can break the signal down into it’s component parts, analyze the system’s response to each part separately, then add these responses back together to get the system’s response. When you break a signal down into its component sine waves, you see that the signal’s energy is distributed as a function of frequency. This distribution of a signal’s energy over frequency can be shown as a plot of spectral density vs. frequency, such as the solid plot in the center of Figure 3. When you have a signal such as the one mentioned above, and you sample it, aliasing will cause the signal components will be replicated endlessly. These replica signals are the signal’s aliases. The spacing between these aliases will be even and equal to the sampling rate. These aliases are indistinguishable from ‘real’ signals spaced an integer number of sampling rates away: there is no way, once the signal is sampled, to know which parts of it are ‘real’ and which parts are aliased. To compound our trouble, any real-world signal will have a power spectrum that’s symmetrical around zero frequency, with ‘negative’ frequency components; after sampling these components of the original signal will appear at frequencies that are lower than the sample rate. It shows this effect. The central frequency density plot is the signal that’s being sampled; the others are the signal’s aliases in sampled time. If you sampled this signal as shown, then after sampling the signal energy would appear to “fold back” at 1=2 the sampling rate. This can be used to demonstrate part of the Nyquist-Shannon sampling

theorem: if the original signal were band limited to 1=2 the sampling rate then after aliasing there would be no overlapping energy, and thus no ambiguity caused by aliasing

**ARM® cortex®**

ARM cortex-A5 processor is the smallest, lowest, and lowest power ARMv7 application processor, ideal as a standalone processor within current and future generation of smart wearable devices. It is capable of delivering the internet to the widest possible range of devices, from smart devices like wearable, feature phones and low cost, entry-level smart phones, to a range of pervasive embedded, consumer and industrial devices.

**Overview**

The Cortex-A5 processor is the most mature, most configurable, smallest and lowest power ARMv7-A CPU. It provides a high-value migration path for existing [ARM926EJ-S™](http://www.arm.com/products/processors/classic/arm9/arm926.php) and [ARM1176JZ-S™](http://www.arm.com/products/processors/classic/arm11/arm1176.php) processor designs. It achieves better performance than the [ARM1176JZ-S](http://www.arm.com/products/processors/classic/arm11/arm1176.php) processor, better power and energy efficiency than the [ARM926EJ-S](http://www.arm.com/products/processors/classic/arm9/arm926.php), and 100% [Cortex-HYPERLINK "http://www.arm.com/products/processors/cortex-a/index.php"A](http://www.arm.com/products/processors/cortex-a/index.php) compatibility.

These processors deliver high-end features to power and cost-sensitive applications, featuring:

* Multiprocessing capability for scalable, energy-efficient performance
* Optional [Floating Point UnitHYPERLINK "http://www.arm.com/products/processors/technologies/vector-floating-point.php"](http://www.arm.com/products/processors/technologies/vector-floating-point.php)(FPU) or [NEON™HYPERLINK "http://www.arm.com/products/processors/technologies/neon.php"](http://www.arm.com/products/processors/technologies/neon.php)units for media and signal processing
* Full application compatibility with the [Cortex-A8](http://www.arm.com/products/processors/cortex-a/cortex-a8.php), [Cortex-A9](http://www.arm.com/products/processors/cortex-a/cortex-a9.php), and [Classic](http://www.arm.com/products/processors/classic/index.php) ARM processors
* High performance memory system including caches and Memory Management Unit (MMU)

**Applications**

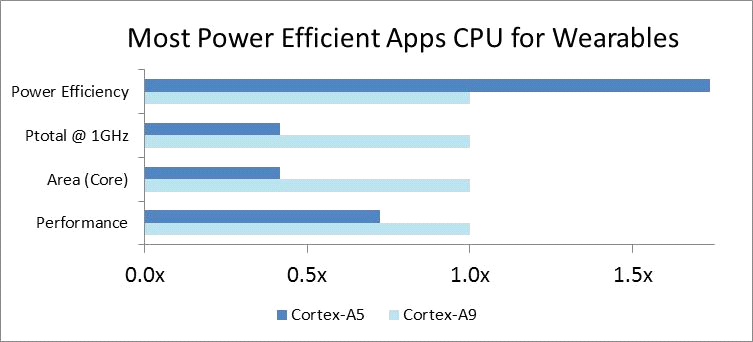
The Cortex-A5 is designed for applications that require virtual memory management for high-level operating systems within an extremely low power profile.

|  |  |
| --- | --- |
| **Product Type** | **Application** |
| [Mobile](http://www.arm.com/markets/mobile/index.php) | [Entry-level Smart phones](http://www.arm.com/markets/mobile/smartphones.php), [Feature Phones](http://www.arm.com/markets/mobile/feature-phones.php), [Mobile Payments](http://www.arm.com/markets/mobile/trustzone-and-mobile-payments.php), Audio |
| [Home/Consumer](http://www.arm.com/markets/home/index.php) | [Digital TV](http://www.arm.com/markets/home/digital-tv.php), [DVD](http://www.arm.com/markets/home/blu-ray-and-dvd.php) |
| [Embedded/Industrial](http://www.arm.com/markets/embedded/index.php) | [MPU](http://www.arm.com/markets/embedded/mcu.php), [Smart Meters](http://www.arm.com/markets/embedded/smart-meter.php), IoT, [Wearable Devices](http://www.arm.com/markets/wearables.php) |

**Area and Energy Efficiency**

The Cortex-A5 is the smallest and lowest power applications processor, delivering rich functionality to high-performance wearable’s As ARM's most energy-efficient ARMv7 applications processor, the Cortex-A5 gets more work done per unit of energy. This corresponds to longer battery life and less heat dissipation in wearable and mobile devices.

* The tiny size of Cortex-A5 offers the following advantages:
* Lowers manufacturing cost
* Allows more low-cost integration
* Reduces leakage



**Compatibility**

The Cortex-A5 processor provides full instruction and feature compatibility with the higher performance Cortex-A8 and Cortex-A9 processors at one-third of the area and power. The Cortex-A5 processor also maintains backwards application compatibility with Classic ARM processors including the ARM926EJ-S, ARM1176JZ-S, and ARM7TDMI®.

**Procedure:**

1. Generate a Sign wave of Programmable frequency.
2. Capture samples at programmable frequency (Max up as per Nyquist Sampling Theorem).
3. Reconstruct the Sign wave using collected Samples using ARM Cortex A5/A9.
4. Using oscilloscope calculate signal’s frequency.
5. Write observations.

**Mathematical Model:**

1. Let S be a system that describes the operations on sine wave.

S = {…..}

2. Identify input as I

S = {I,N,…..}

Ii € I

Ii = The inputs to the system such as frequency of sine wave

N= No. of samples.

3. Identify Output as O.

S = {I,N,O,…..}

O = Sine wave of specified frequency.

4. Identify the process as P.

S = {I,N, O, P,……}

P ={Ps, Pb, Pc,Pr }

Ps = The process of generation of sine wave.

Pb = The process of sampling of sine wave .

Pc = The process displaying sine wave.

Pr = The process of reconstructing of sine wave using samples.

5. Identify success as Su

S = {I, N, O, P, Su, F……}

Su = Success is defined when Proper operation takes place as on requirement.

F = When improper operation is done .

6. Identify Initial condition Si

S = {I, N, O, P, Su, Si,….}

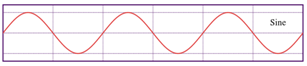
The user will capture generate the sine wave of required frequency and pole-zero diagram.

S = I U N U O U P

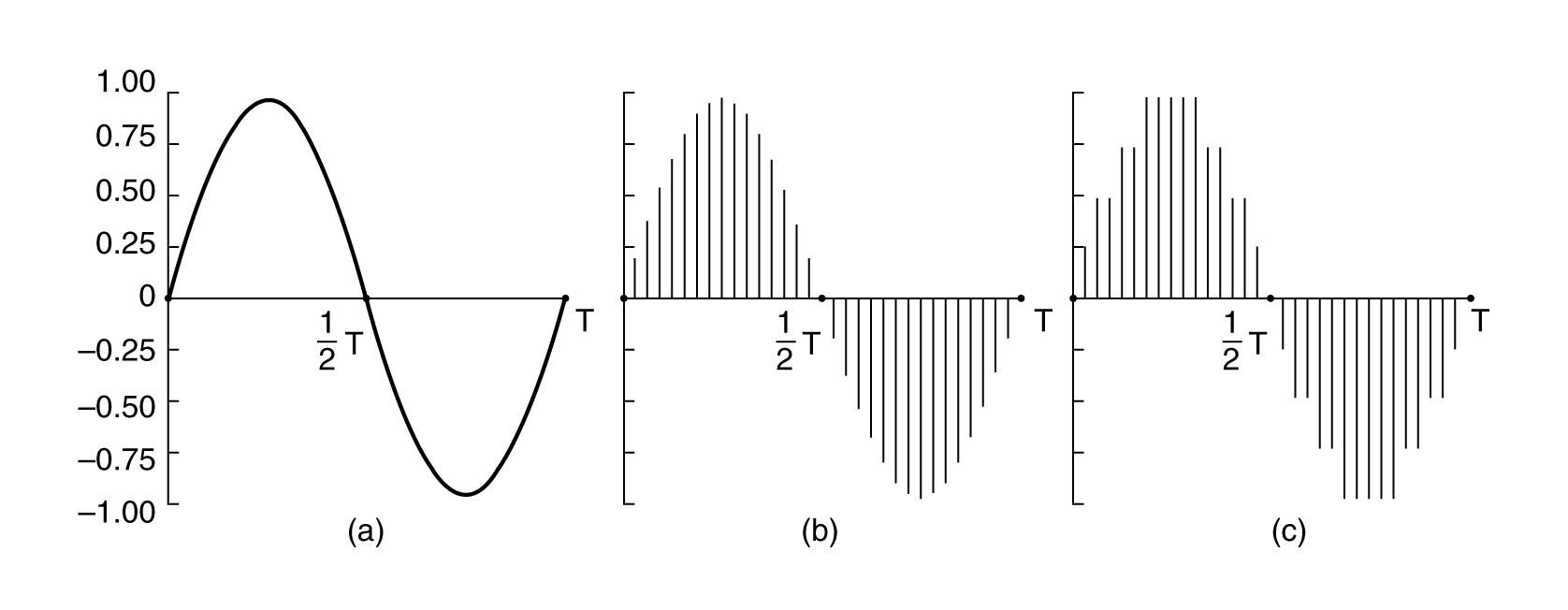
**Observations:**

* 1. The sine wave is generated of given programmable frequency.
  2. Using Samples reconstruction is done again.

**Results:**

****

**Fig: Sine Wave**



**Fig : Sampling of sine wave.**

**Conclusion:** Understanding &Implementation sine wave using Nyquist theorem is done. And

Reconstruction of sine wave is done with the help of samples successfully.

**Oral Question Bank**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Q. No** | **Description** | **Attainment of** | | | |
| **CO** | **PO** | **BL\*** | **GA** |
| **1.** | What is sign wave programmable frequency? | **CO4** | **PO1** |  |  |
| **2.** | Explain the sampling theorem?. | **CO4** | **PO1** |  |  |
| **3.** | What is Nyquist sampling theorem? | **CO4** | **PO3** |  |  |
| **4.** | Explain Nyquist sampling theorem with suitable examples. | **CO4** | **PO4** |  |  |
| **5.** | What is the exact role of aliasineg in the sampling theorem? | **CO3** | **PO5** |  |  |
| **6.** | Explain the aliasing of two sign waves. | **CO4** | **PO4** |  |  |
| **7.** | Prove that the Reconstruction is the opposite process of sampling with example. | **CO4** | **PO3** |  |  |
| **8.** | Explain the Aliasing of a signal’s spectrum in the frequency domain? | **CO4** | **PO3** |  |  |
| **9.** | Elaborate in detail about the signal and its reconstructed signal? | **CO2** | **PO3** |  |  |
| **10.** | Define the sampling with a cut-off frequency of 1/T. | **CO2** | **PO4** |  |  |
| **11.** | Explain the ARM® cortex®. | **CO2** | **PO4** |  |  |
| **12.** | Ellaborate whole architecture of the ARM cortex. | **CO2** | **PO5** |  |  |
| **13.** | What are the key instructions from A32 ARM | **CO2** | **PO5** |  |  |
| **14.** | Enlist ARM v7 instructions set | **CO2** | **PO3** |  |  |
| **15.** | List of ARM microprocessor cores | **CO2** | **PO3** |  |  |
| **16** | Give the Applications on ARM microprocessors. | **CO4** | **PO3** |  |  |

**EXPERIMENT NO.3**

**To Generate a square wave of programmable frequency**

**Title:** To Generate a square wave of programmable frequency

**Objectives**: To understand & generate a Square wave programmable frequency and also study different function to generate Pole-Zero Diagram using multi core programming.

**Aim:** Write a C++/ Python program to generate a Square wave of programmable frequency. Write a function to generate Pole-Zero Diagram using multi core programming.

**Requirements:**

**H/W Requirements:**

1. 8 GB RAM
2. 500GB/1TB HDD
3. Oscilloscope
4. Signal Generator

**S/W Requirements:**

1. Latest version of 64 Bit Operating Systems Open Source Fedora-20.
2. Windows 8 with Multi core CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading

**Theory:**

1. **Pole-Zero Diagrams:**

In [mathematics](http://en.wikipedia.org/wiki/Mathematics), [signal processing](http://en.wikipedia.org/wiki/Signal_processing) and [control theory](http://en.wikipedia.org/wiki/Control_theory), a **pole–zero plot** is a graphical representation of a [rational](http://en.wikipedia.org/wiki/Rational_function) [transfer function](http://en.wikipedia.org/wiki/Transfer_function) in the complex plane which helps to convey certain properties of the system such as:

* [Stability](http://en.wikipedia.org/wiki/BIBO_stability)
* [Causal system](http://en.wikipedia.org/wiki/Causal_system) / [anti-causal system](http://en.wikipedia.org/wiki/Anticausal_system)
* [Region of convergence (ROC)](http://en.wikipedia.org/wiki/Radius_of_convergence)
* [Minimum phase](http://en.wikipedia.org/wiki/Minimum_phase) / non minimum phase

**Stability:**

In [signal processing](http://en.wikipedia.org/wiki/Signal_processing), specifically [control theory](http://en.wikipedia.org/wiki/Control_theory), BIBO stability is a form of [stability](http://en.wikipedia.org/wiki/Control_theory#Stability) for [linear](http://en.wikipedia.org/wiki/Linear_system) [signals](http://en.wikipedia.org/wiki/Signal_(information_theory)) and systems that take inputs. BIBO stands for *bounded-input, bounded-output*. If a system is BIBO stable, then the output will be bounded for every input to the system that is bounded.

* A signal is bounded if there is a finite value B > 0 such that the signal magnitude never exceeds B, that is
* \ |y[n]| \leq B \quad \forall n \in \mathbb{Z} for discrete-time signals, or
* \ |y(t)| \leq B \quad \forall t \in \mathbb{R} for continuous-time signals.

[**Causal system**](http://en.wikipedia.org/wiki/Causal_system)**/ [anticausal system](http://en.wikipedia.org/wiki/Anticausal_system" \o "Anticausal system):**

An **anti-causal system** is a [hypothetical](http://en.wikipedia.org/wiki/Hypothetical) [system](http://en.wikipedia.org/wiki/System) with outputs and internal states that depend *solely* on future input values. Some textbooks and published research literature might define an anti-causal system to be one that does not depend on past input values, allowing also for the dependence on present input values. An **causal system** is a system that is not a [causal system](http://en.wikipedia.org/wiki/Causal_system), that is one that depends on some future input values and possibly on some input values from the past or present. This is in contrast to a causal system which depends only on current and/or past input values.[[2]](http://en.wikipedia.org/wiki/Anticausal_system#cite_note-2) This is often a topic of [control theory](http://en.wikipedia.org/wiki/Control_theory) and [digital signal processing](http://en.wikipedia.org/wiki/Digital_signal_processing) (DSP).Anti-causal systems are also casual, but the converse is not always true. An causal system that has any dependence on past input values is not anti-causal.

An example of a causal signal processing is the production of an output signal that is processed from another input signal that is recorded by looking at input values both forward and backward in time from a predefined time arbitrarily denoted as the "present" time. (In reality, that "present" time input, as well as the "future" time input values, have been recorded at some time in the past, but conceptually it can be called the "present" or "future" input values in this a causal process.) This type of processing cannot be done in [real time](http://en.wikipedia.org/wiki/Real-time_computing) as future input values are not yet known, but is done after the input signal has been recorded and is post-processed.

[**Region of convergence (ROC)**](http://en.wikipedia.org/wiki/Radius_of_convergence)

In [mathematics](http://en.wikipedia.org/wiki/Mathematics), the **radius of convergence** of a [power series](http://en.wikipedia.org/wiki/Power_series) is the radius of the largest [disk](http://en.wikipedia.org/wiki/Disk_(mathematics)) in which the [series](http://en.wikipedia.org/wiki/Power_series) [converges](http://en.wikipedia.org/wiki/Convergent_series). It is either a non-negative real number or ∞. When it is positive, the power series [converges absolutely](http://en.wikipedia.org/wiki/Absolute_convergence) and [uniformly on compact sets](http://en.wikipedia.org/wiki/Compact_convergence) inside the open disk of radius equal to the radius of convergence, and it is the [Taylor series](http://en.wikipedia.org/wiki/Taylor_series) of the [analytic function](http://en.wikipedia.org/wiki/Analytic_function) to which it converges.

For a power series *ƒ* defined as:

f(z) =  \sum_{n=0}^\infty c_n (z-a)^n, 

Where,

*a* is a [complex](http://en.wikipedia.org/wiki/Complex_number) constant, the center of the [disk](http://en.wikipedia.org/wiki/Disk_(mathematics)) of convergence,

*cn* is the *n*th complex coefficient, and

*z* is a complex variable.

The radius of convergence *r* is a nonnegative real number or ∞ such that the series converges if

|z-a| < r\,

And diverges if

|z-a| > r.\,

In other words, the series converges if *z* is close enough to the center and diverges if it is too far away. The radius of convergence specifies how close is close enough. On the boundary, that is, where |*z* − *a*| = *r*, the behavior of the power series may be complicated, and the series may converge for some values of *z* and diverge for others. The radius of convergence is infinite if the series converges for all [complex numbers](http://en.wikipedia.org/wiki/Complex_number) *z*.

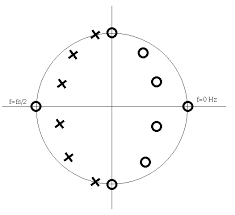
[**Minimum phase**](http://en.wikipedia.org/wiki/Minimum_phase)**/ non minimum phase**

In [control theory](http://en.wikipedia.org/wiki/Control_theory) and [signal processing](http://en.wikipedia.org/wiki/Signal_processing), a [linear, time-invariant](http://en.wikipedia.org/wiki/LTI_system_theory) system is said to be **minimum-phase** if the system and its [inverse](http://en.wikipedia.org/wiki/Inverse_(mathematics)) are [causal](http://en.wikipedia.org/wiki/Causal_system) and [stable](http://en.wikipedia.org/wiki/BIBO_stability).

For example, a discrete-time system with [rational](http://en.wikipedia.org/wiki/Rational_function) [transfer function](http://en.wikipedia.org/wiki/Transfer_function) H(z) can only satisfy [causality](http://en.wikipedia.org/wiki/Causal#Engineering) and [stability](http://en.wikipedia.org/wiki/BIBO_stability) requirements if all of its [poles](http://en.wikipedia.org/wiki/Pole_(complex_analysis)) are inside the [unit circle](http://en.wikipedia.org/wiki/Unit_circle). However, we are free to choose whether the [zeros](http://en.wikipedia.org/wiki/Zero_(complex_analysis)) of the system are inside or outside the [unit circle](http://en.wikipedia.org/wiki/Unit_circle). A system with rational transfer function is minimum-phase if all its zeros are also inside the unit circle. Insight is given below as to why this system is called minimum-phase.

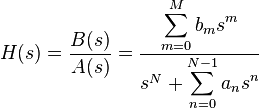
**A pole-zero plots** shows the location in the complex plane of the poles and zeros of the [transfer function](http://en.wikipedia.org/wiki/Transfer_function) of a [dynamic system](http://en.wikipedia.org/wiki/Dynamic_system), such as a controller, compensator, sensor, equalizer, filter, or communications channel. By convention, the poles of the system are indicated in the plot by an X while the zeroes are indicated by a circle or O.

A pole-zero plots can represent either a continuous-time (CT) or a discrete-time (DT) system. For a CT system, the plane in which the poles and zeros appear is the [s plane](http://en.wikipedia.org/wiki/S_plane) of the [Laplace transform](http://en.wikipedia.org/wiki/Laplace_transform). In this context, the parameter *s* represents the[complex](http://en.wikipedia.org/wiki/Complex_number) [angular frequency](http://en.wikipedia.org/wiki/Angular_frequency), which is the domain of the CT transfer function. For a DT system, the plane is the z plane, where *z* represents the domain of the [Z-transform](http://en.wikipedia.org/wiki/Z-transform).



Continuous-time system

In general, a [rational](http://en.wikipedia.org/wiki/Rational_function) transfer function for a continuous-time [LTI system](http://en.wikipedia.org/wiki/LTI_system_theory) has the form:



Where,

* B and A are polynomials in s,
* M is the order of the numerator polynomial,
* b_m is the *m*-th coefficient of the numerator polynomial,
* N is the order of the denominator polynomial, and
* a_n is the *n*-th coefficient of the denominator polynomial.

Either M or N or both may be zero, but in real systems, it should be the case thatM \le N; otherwise the gain would be unbounded at high frequencies.

**Poles and zeros**

* the [zeros](http://en.wikipedia.org/wiki/Zero_(complex_analysis)) of the system are roots of the numerator polynomial:

s = \{ \beta_m | m \in 1, \ldots M \}such that B(s)|_{s = \beta_m} = 0 

* the [poles](http://en.wikipedia.org/wiki/Pole_(complex_analysis)) of the system are roots of the denominator polynomial:

s = \{ \alpha_n | n \in 1, \ldots N \}such that A(s)|_{s = \alpha_n} = 0 .

**Region of convergence**

The [region of convergence](http://en.wikipedia.org/wiki/Region_of_convergence) (ROC) for a given CT transfer function is a half-plane or vertical strip, either of which contains no poles. In general, the ROC is not unique, and the particular ROC in any given case depends on whether the system is[causal](http://en.wikipedia.org/wiki/Causal_system) or anti-causal.

* If the ROC includes the [imaginary axis](http://en.wikipedia.org/wiki/Imaginary_axis), then the system is [bounded-input, bounded-output (BIBO) stable](http://en.wikipedia.org/wiki/BIBO_stability).
* If the ROC extends rightward from the pole with the largest [real-part](http://en.wikipedia.org/wiki/Complex_number#Definition) (but not at infinity), then the system is causal.
* If the ROC extends leftward from the pole with the smallest real-part (but not at negative infinity), then the system is anti-causal.

The ROC is usually chosen to include the imaginary axis since it is important for most practical systems to have [BIBO stability](http://en.wikipedia.org/wiki/BIBO_stability).

**Example**

H(s) = {    25    \over  s^2  +  6s   +   25     }     

This system has no (finite) zeros and two poles:

  s = \alpha_1  =  -3 + 4j    

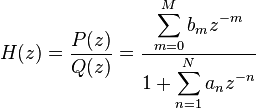
and

   s = \alpha_2   =   -3 - 4j    

Notice that these two poles are [complex conjugates](http://en.wikipedia.org/wiki/Complex_conjugate), which is the necessary and sufficient condition to have real-valued coefficients in the differential equation representing the system.

**Discrete-time systems**

In general, a rational transfer function for a discrete-time [LTI system](http://en.wikipedia.org/wiki/LTI_system_theory) has the form:



Where,

* M is the order of the numerator polynomial,
* b_m is the *m*-th coefficient of the numerator polynomial,
* N is the order of the denominator polynomial, and
* a_n is the *n*-th coefficient of the denominator polynomial.
* Either M or N or both may be zero.

**Poles and zeros**

* z = \beta_m such that P(z)|_{z = \beta_m} = 0  are the [zeros](http://en.wikipedia.org/wiki/Zero_(complex_analysis)) of the system
* z = \alpha_n Such that Q(z)|_{z = \alpha_n} = 0 are the [poles](http://en.wikipedia.org/wiki/Pole_(complex_analysis)) of the system.

**Region of convergence**

The [region of convergence](http://en.wikipedia.org/wiki/Region_of_convergence) (ROC) for a given DT transfer function is a [disk](http://en.wikipedia.org/wiki/Disk_(mathematics)) or [annulus](http://en.wikipedia.org/wiki/Annulus_(mathematics)) which contains no poles. In general, the ROC is not unique, and the particular ROC in any given case depends on whether the system is [causal](http://en.wikipedia.org/wiki/Causal_system) or anti-causal.

* If the ROC includes the [unit circle](http://en.wikipedia.org/wiki/Unit_circle), then the system is [bounded-input, bounded-output (BIBO) stable](http://en.wikipedia.org/wiki/BIBO_stability).
* If the ROC extends outward from the pole with the largest (but not infinite) magnitude, then the system has a right-sided impulse response. If the ROC extends outward from the pole with the largest magnitude and there is no pole at infinity, then the system is causal.
* If the ROC extends inward from the pole with the smallest (nonzero) magnitude, then the system is anti-causal.
* The ROC is usually chosen to include the unit circle since it is important for most practical systems to have [BIBO stability](http://en.wikipedia.org/wiki/BIBO_stability).

**Example**

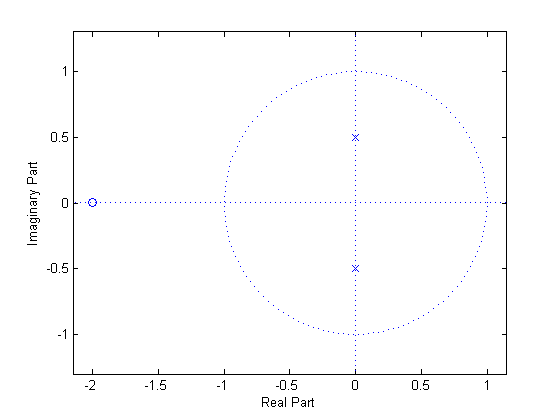
If P(z) and Q(z) are completely factored, their solution can be easily plotted in the z-plane. For example, given the following transfer function:

H(z) = \frac{z^1+2}{z^2+\frac{1}{4}}

The only (finite) zero is located at: z = -2, and the two poles are located at:

z = \pm\frac{j}{2}, where *j* is the imaginary unit.

The pole–zero plot would be:

[](http://en.wikipedia.org/wiki/File:PoleZeroPlot.png)

1. **What Is a Multicore?**

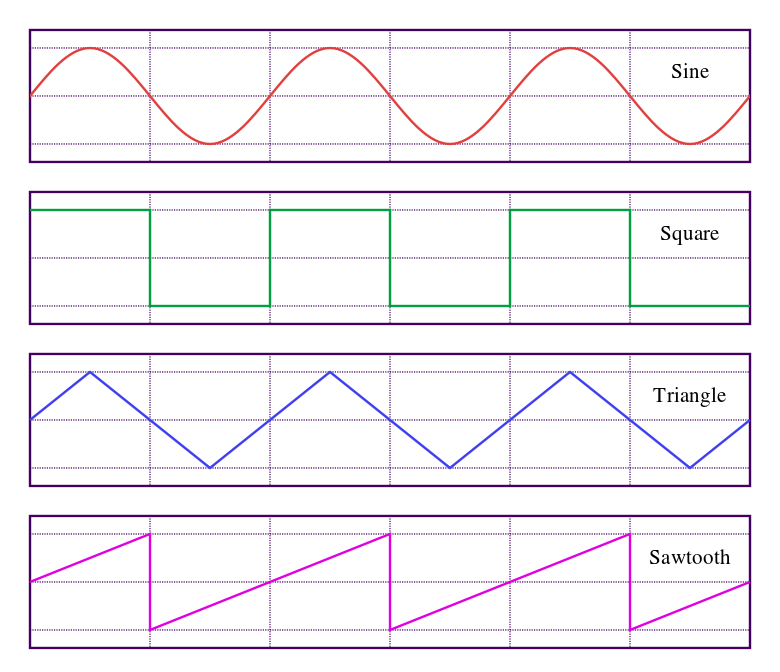
* A multicore is an architecture design that places multiple processors on a single die (computer chip). Each processor is called a core. As chip capacity increased, placing multiple processors on a single chip became practical. These designs are known as Chip Multiprocessors (CMPs) because they allow for single chip multiprocessing. Multicore is simply a popular name for CMP or single chip multiprocessors. The concept of single chip multiprocessing is not new, and chip manufacturers have been exploring the idea of multiple cores on a uniprocessor since the early 1990s. Recently, the CMP has become the preferred method of improving overall system performance. This is a departure from the approach of increasing the clock frequency or processor speed to achieve gains in overall system performance. Increasing the clock frequency has started to hit its limits in terms of cost - effectiveness. Higher frequency requires more power, making it harder and more expensive to cool the system. This also affects sizing and packaging considerations. So, instead of trying to make the processor faster to gain performance, the response is now just to add more processors. The simple realization that this approach is better has prompted the multicore revolution. Multicore architectures are now center stage in terms of improving overall system performance. For software developers who are familiar with multiprocessing, multicore development will be familiar. From a logical point of view, there is no real significant difference between programming for multiple processors in separate packages and programming for multiple processors contained in a single package on a single chip. There may be performance differences, however, because the new CMPs are using advances in bus architectures and in connections between processors. In some circumstances, this may cause an application that was originally written for multiple processors to run faster when executed on a CMP. Aside from the potential performance gains, the design and implementation are very similar. We discuss minor differences throughout the book. For developers who are only familiar with sequential programming and single core development, the multicore approach offers many new software development paradigms.
* **Multicore Architectures**

CMPs come in multiple flavors: two processors (dual core), four processors (quad core), and eight processors (octa - core) configurations. Some configurations are multithreaded; some are not. There are several variations in how cache and memory are approached in the new CMPs. The approaches to processor - to - processor communication vary among different implementations. The CMP implementations from the major chip manufacturers each handle the I/O bus and the Front Side Bus (FSB) differently Again, most of these differences are not visible when looking strictly at the logical view of an application that is being designed to take advantage of a multicore architecture. Figure 1 - 1 illustrates three common configurations that support multiprocessing.

1. **Square wave of programmable frequency**

* **Square wave**

A **square wave** is a [non-sinusoidal periodic waveform](http://en.wikipedia.org/wiki/Non-sinusoidal_waveform)(which can be represented as an infinite summation of sinusoidal waves), in which the amplitude alternates at a steady [frequency](http://en.wikipedia.org/wiki/Frequency) between fixed minimum and maximum values, with the same duration at minimum and maximum. The transition between minimum to maximum is instantaneous for an ideal square wave; this is not realizable in physical systems. Square waves are often encountered in [electronics](http://en.wikipedia.org/wiki/Electronics) and [signal processing](http://en.wikipedia.org/wiki/Signal_processing). Its stochastic counterpart is a [two-state trajectory](http://en.wikipedia.org/wiki/Two-state_trajectory). A similar but not necessarily symmetrical wave, with arbitrary durations at minimum and maximum, is called a [rectangular wave](http://en.wikipedia.org/wiki/Rectangular_wave) (of which the square wave is a special case).

****

**Figure 2. Square waves**

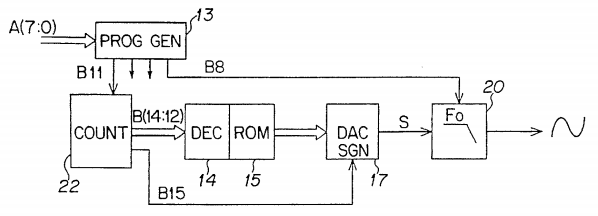


Figure 3.Square wave of programmable frequency

FIG. 3 shows again generator 13 generating square-wave signals at a programmable frequency, decoder 14 associated with ROM 15, and the DAC 17. According to the invention, the output signal S of DAC 17 is filtered by a switched-capacitor filter 20 with a control frequency, or sampling frequency that is proportional to the frequency of the sine wave signal to be generated. Thus, the cut-of frequency F0 of the switched-capacitor filter 20 is proportional to the frequency of the sine wave signal. The components of filter 20 are selected so that the cut-off frequency F0 is slightly higher than the frequency of the sine wave signal. A signal provided by the programmable generator 13 is used as the control signal of filter 20. In order that filter 20 operates correctly, the frequency of its control signal must be high with respect to the frequency of signal S to be filtered, the frequency of signal S being substantially equal to the cut-off frequency F0. To achieve this purpose, signal B8, which has the highest frequency, is chosen as a control signal from among signals B8—Bll provided by generator 13. In addition, instead of providing the outputs of the programmable generator 13 directly to decoder 14, the output signal B11, which has the lowest frequency, is provided to a counter 22. Counter 22 provides values to decoder 14 at a frequency sufficiently low with respect to the control frequency of filter 20. In the given example, with a 4\_bit (Bl2-Bl5) counter 22, a sufficient ratio (128) is obtained between the frequency of the control signal B8 and the frequency of signal B15, which is the frequency of the generated signal.

**Mathematical Model:**

1. Let S be a system that describes the operations on square wave.

S = {…..}

2. Identify input as I

S = {I,…..}

Ii € I

Ii = The inputs to the system such as frequency of square wave.

3. Identify Output as O.

S = {I,O,…..}

O = Square wave of specified frequency. And pole-Zero diagram.

4. Identify the process as P.

S = {I, O, P,……}

P ={Ps, Pb, Pc,Pr }

Ps = The process of generation of square wave.

Pb = The process of plotting of pole-zero diagram using multicore programming.

Pc = The process displaying square wave.

Pr = The process of displaying pole-zero diagram.

5. Identify success as Su

S = {I,O, P, Su, F……}

Su = Success is defined when Proper operation takes place as on requirement.

F = When improper operation is done .

6. Identify Initial condition Si

S = {I, O, P, Su, Si,….}

The user will capture generate the square wave of required frequency and pole-zero diagram.

S = I U O U P

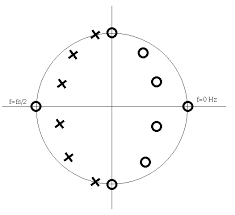
**Observations:**

* 1. The square wave is generated of given programmable frequency.
  2. Using proper function pole-zero diagrams is plotted.

**Results:**

****

**Fig: Square Wave**

****

**Fig. Pole-Zero Diagram**

**Conclusion:**

Understanding & Generating a Square wave of programmable frequency is done successfully. Studied and implementation of different function to generate Pole-Zero Diagram using multicore programming is done successfully.

**Oral Question Bank**

**Theory Question Bank**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Q. No** | **Description** | **Attainment of** | | | |
| **CO** | **PO** | **BL\*** | **GA** |
| 1. | Define is Multicore Programming? | CO4 | PO1 | Description: DB13.bmp |  |
| 2. | Explain the architecture of Multicore Programming? | CO4 | PO1 | Description: DB13.bmp |  |
| 3. | Explain are the challenges of multi core programming | CO4 | PO3 | Description: DB13.bmp |  |
| 4. | Explain Pole-Zero Diagrams? | CO4 | PO4 | Description: DB11.bmp |  |
| 5. | Explain are the different properties of Pole-Zero Plot? | CO3 | PO5 | Description: DB11.bmp |  |
| 6. | Explain pole-zero cancellation? | CO2 | PO4 | Description: DB11.bmp |  |
| 7. | Define “gain” of z-transform? | CO4 | PO4 | Description: DB13.bmp |  |
| 8. | Explain the term H (z)? : Give the equation for System Function of LTI system? | CO4 | PO3 | Description: DB11.bmp |  |
| 9. | Explain the condition for Causality of LTI system in terms of z-transform? | CO4 | PO3 | Description: DB11.bmp |  |
| 10. | Define is ROC? | CO2 | PO3 | Description: DB11.bmp |  |
| 11. | Define is Programmable frequency? | CO2 | PO4 | Description: DB11.bmp |  |
| 12. | Describe sine wave generator? | CO4 | PO5 | Description: DB11.bmp |  |
| 13. | Explain the challenges of Multicore Programming? | CO4 | PO3 | Description: DB11.bmp |  |
| **14.** | Cite the effect of quantization on pole locations | **CO4** | **PO5** | Description: DB13.bmp |  |
| **15.** | Explain zero input limit cycle overflow oscillation | **CO4** | **PO5** | Description: DB13.bmp |  |
| **16.** | Explain about causality condition for an LTI system? | **CO4** | **PO3** | Description: DB13.bmp |  |

**Experiment No. 4**

**To capture a signal and perform various operations for analyzing it.**

**Title:** To capture a signal and perform various operations for analyzing it.

**Objectives**: To generate a square/sine wave of programmable frequency and emulate a RC filter and understanding the response curves.

**Aim:** Write a C++/ Python program to capture signal using ARM

Cortex A5/A9/M4 ADC and signal generator, generate/construct a Square/Sine wave of

programmable frequency and voltage Draw Voltage (y-axis) and Time (x-axis) graph. Write a

function to emulate simple RC filter with R being Trim-pot(GUI meter) of 10K and C = 0.1

microFarad. Write a program to generate a Voltage-Time response curve with reference to

change in R. Draw the resultant outcome graph. Store the data in SAN (BIGDATA)

**Requirements:**

**H/W Requirements:**

8 GB RAM

500GB/1TB HDD

Signal Generator

ARM Cortex A5.

SAN

**S/W Requirements:**

Latest version of 64 Bit Operating Systems Open Source Fedora-20.

Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading

**Theory:**

**ARM**

Conceptually the Cortex-M4 is a Cortex-M3 plus DSP Instructions, and optional floating-point unit (FPU). If a core contains an FPU, it is known as a Cortex-M4F, otherwise it is a Cortex-M4. Key features of the Cortex-M4 core are:

* ARMv7E-M architecture
* Instruction sets
  + Thumb (entire)
  + Thumb-2 (entire)
  + 1-cycle 32-bit hardware multiply, 2-12 cycle 32-bit hardware divide, saturated math support
  + DSP extension: Single cycle 16/32-bit MAC, single cycle dual 16-bit MAC, 8/16-bit SIMD arithmetic
* 3-stage pipeline with branch speculation
* 1 to 240 physical interrupts, plus NMI
* 12 cycle interrupt latency
* Integrated sleep modes

Silicon options:

* Optional Floating-Point Unit (FPU): single-precision only IEEE-754 compliant. This is called the FPv4-SP extension.
* Optional Memory Protection Unit (MPU): 0 or 8 regions

### Chips

The following microcontrollers are based on the Cortex-M4 core:

* Atmel SAM4L, SAM4N, SAM4S
* Freescale Kinetis K

The following microcontrollers are based on the Cortex-M4F (M4 + FPU) core:

* Atmel SAM4C (dual core), SAM4E, SAMG
* Energy Micro EFM32 Wonder
* Freescale Kinetis K
* Infineon XMC4000
* NXP LPC4000, LPC4300(one Cortex-M4F + one Cortex-M0)
* STMicroelectronics STM32 F3, F4
* Texas Instruments LM4F, TM4C
* Spansion FM4F
* Toshiba TX04

The following chips have either a Cortex-M4 or M4F as a secondary core:

* Freescale Vybrid VF6 (one Cortex-A5 + one Cortex-M4F)
* Texas Instruments OMAP 5 (one dual-core Cortex-A15 + two Cortex-M4)

Square wave

A **square wave** is a non-sinusoidal periodic waveform (which can be represented as an infinite summation of sinusoidal waves), in which the amplitude alternates at a steady frequency between fixed minimum and maximum values, with the same duration at minimum and maximum. The transition between minimum to maximum is instantaneous for an ideal square wave; this is not realizable in physical systems. Square waves are often encountered in electronics and signal processing. Its stochastic counterpart is a two-state trajectory. A similar but not necessarily symmetrical wave, with arbitrary durations at minimum and maximum, is called a rectangular wave (of which the square wave is a special case).

Square waves are universally encountered in digital switching circuits and are naturally generated by binary (two-level) logic devices. They are used as timing references or "clock signals", because their fast transitions are suitable for triggering synchronous logic circuits at precisely determined intervals. However, as the frequency-domain graph shows, square waves contain a wide range of harmonics; these can generate electromagnetic radiation or pulses of current that interfere with other nearby circuits, causing noise or errors. To avoid this problem in very sensitive circuits such as precision analog-to-digital converters, sine waves are used instead of square waves as timing references.

In musical terms, they are often described as sounding hollow, and are therefore used as the basis for wind instrument sounds created using subtractive synthesis. Additionally, the distortion effect used on electric guitars clips the outermost regions of the waveform, causing it to increasingly resemble a square wave as more distortion is applied.

The **sine wave** or **sinusoid** is a mathematical curve that describes a smooth repetitive oscillation. It is named after the function sine, of which it is the graph. It occurs often in pure and applied mathematics, as well as physics, engineering, signal processing and many other fields. Its most basic form as a function of time (*t*) is**:**

y(t) = A\sin(2 \pi f t + \varphi) = A\sin(\omega t + \varphi)

The sine wave is important in physics because it retains its wave shape when added to another sine wave of the same frequency and arbitrary phase and magnitude. It is the only periodic waveform that has this property. This property leads to its importance in Fourier analysis and makes it acoustically unique.

A **storage area network** (**SAN**) is a dedicated network that provides access to consolidated, block level data storage. SANs are primarily used to enhance storage devices, such as disk arrays, tape libraries, and optical jukeboxes, accessible to servers so that the devices appear like locally attached devices to the operating system. A SAN typically has its own network of storage devices that are generally not accessible through the local area network (LAN) by other devices. The cost and complexity of SANs dropped in the early 2000s to levels allowing wider adoption across both enterprise and small to medium sized business environments.

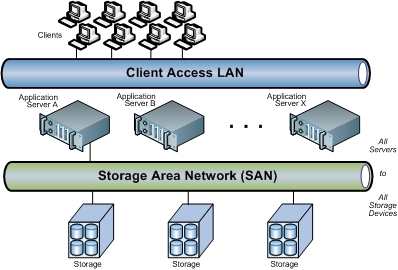
A SAN does not provide file abstraction, only block-level operations. However, file systems built on top of SANs do provide file-level access, and are known as *SAN filesystems* or shared disk file systems.

Sharing storage usually simplifies storage administration and adds flexibility since cables and storage devices do not have to be physically moved to shift storage from one server to another.

Other benefits include the ability to allow servers to boot from the SAN itself. This allows for a quick and easy replacement of faulty servers since the SAN can be reconfigured so that a replacement server can use the LUN of the faulty server. While this area of technology is still new, many view it as being the future of the enterprise datacenter.

SANs also tend to enable more effective disaster recovery processes. A SAN could span a distant location containing a secondary storage array. This enables storage replication either implemented by disk array controllers, by server software, or by specialized SAN devices. Since IP WANs are often the least costly method of long-distance transport, the Fibre Channel over IP (FCIP) and iSCSI protocols have been developed to allow SAN extension over IP networks. The traditional physical SCSI layer could only support a few meters of distance - not nearly enough to ensure business continuance in a disaster.

The economic consolidation of disk arrays has accelerated the advancement of several features including I/O caching, snap shotting, and volume cloning (Business Continuance Volumes or BCVs).



**Procedure:-**

* 1. Generate a signal using signal generator.
  2. Construct a square wave of signal frequency.
  3. Plot the signal on a VT
  4. Emulate a RC filter and trim by 10k
  5. Again plot the signal on VT.
  6. Store the observations in a SAN.

**Mathematical Model:**

1. Let S be a system that describes the generation of signal along with an emulated signal.

S = {…..}

2. Identify input as I

S = {I,N,…..}

Ii € I

Ii = The inputs to the system such as a square or sine wave of programmable frequency.

N =Number of waves

3. Identify Output as O.

S = {I,N,O,…..}

O = VT response curve i.e. Graph of voltage (y-axis) and Time (x-axis) graph

4. Identify the process as P.

S = {I,N,O, P,……}

P ={Ps, Pb, Pc,Pr }

Ps = The process of capturing signals.

Pb = The process of plotting signals on VT.

Pc = The process of using a RC filter.

Pr = The process of storing observations on SAN.

5. Identify success as Su

S = {I,N,O, P, Su, F……}

Su = Success is defined when Proper operation takes place as on requirement.

F = When improper operation is done .

6. Identify Initial condition Si

S = {I, N, O, P, Su, Si,….}

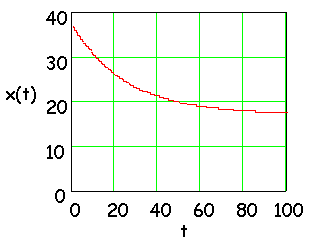
The user will capture signal and filter it and compare it while storing it in SAN.

S = I U N U O U P

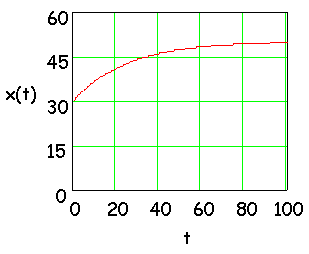
**Observations:**

1. Signal is generated of programmable frequency.
2. RC filter is able to trim by 10K
3. VT curve shows the changes in R
4. Data is stored in SAN

**Results:**



**Fig: VT curve before RC filter**



**Fig: VT curve after RC filter**

**Conclusion:** Thus we have successfully done trimming using RC filter on a sign wave & square

wave and have seen changes in VT plot accordingly.

**Oral Question Bank**

**Theory Question Bank**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Q. No** | **Description** | **Attainment of** | | | |
| **CO** | **PO** | **BL\*** | **GA** |
| 1. | Explain what is ARM | CO4 | PO1 | Description: DB13.bmp |  |
| 2. | Explain what is ARM Cortex M4?? | CO4 | PO1 | Description: DB13.bmp |  |
| 3. | Describe RC filter | CO4 | PO3 | Description: DB13.bmp |  |
| 4. | Describe what is the importance of VT | CO4 | PO4 | Description: DB11.bmp |  |
| 5. | Cite the features of SAN | CO3 | PO5 | Description: DB11.bmp |  |
| 6. | Explain pole-zero cancellation? | CO2 | PO4 | Description: DB11.bmp |  |
| 7. | Define “gain” of z-transform? | CO4 | PO4 | Description: DB13.bmp |  |
| 8. | Explain the term H (z)? : Give the equation for System Function of LTI system? | CO4 | PO3 | Description: DB11.bmp |  |
| 9. | Explain the condition for Causality of LTI system in terms of z-transform? | CO4 | PO3 | Description: DB11.bmp |  |
| 10. | What is ROC? | CO2 | PO3 | Description: DB11.bmp |  |
| 11. | What is Programmable frequency? | CO2 | PO4 | Description: DB11.bmp |  |
| 12. | Describe sine wave generator? | CO4 | PO5 | Description: DB11.bmp |  |
| 13. | Explain the challenges of Multicore Programming? | CO4 | PO3 | Description: DB11.bmp |  |
| **14.** | Cite the effect of quantization on pole locations | **CO4** | **PO5** | Description: DB13.bmp |  |
| **15.** | Explain zero input limit cycle overflow oscillation | **CO4** | **PO5** | Description: DB13.bmp |  |
| **16.** | Explain what is the causality condition for an LTI system? | **CO4** | **PO3** | Description: DB13.bmp |  |
| **21** | Describe what is meant by rounding? Draw the pdf of round off error | **CO4** | **PO5** | Description: DB13.bmp |  |
| **22** | Define What is meant by truncation? Draw the pdf of round off error | **CO4** | **PO4** | Description: DB11.bmp |  |
| **23** | Explain what do you mean by quantization step size? | **CO4** | **PO4** | Description: DB13.bmp |  |
| **24** | Cite different types of frequency domain coding | **CO4** | **PO4** | Description: DB13.bmp |  |
| **25** | Describe subband coding? | **CO4** | **PO3** | Description: DB13.bmp |  |
| **26** | Cite properties of DT sinusoids | **CO4** | **PO3** | Description: DB13.bmp |  |
| **27** | State the condition for a digital filter to be causal and stable | **CO4** | **PO4** | Description: DB11.bmp |  |
| **28** | Describe zero padding? What are its uses? | **CO4** | **PO4** | Description: DB13.bmp |  |
| **29** | Describe A/D conversion noise? | **CO4** | **PO4** | Description: DB13.bmp |  |
| **30** | Define circular convolution | **CO4** | **PO3** | Description: DB11.bmp |  |
| **31** | Distinguish between linear convolution and Circular Convolution. | **CO4** | **PO3** | Description: DB13.bmp |  |
| **32** | State the circular time shifting and circular frequency shifting properties of DFT | **CO4** | **PO2** | Description: DB11.bmp |  |
| **33** | State the time reversal property of DFT | **CO4** | **PO2** | Description: DB11.bmp |  |
| **34** | State the condition for a digital filter to be causal and stable | **CO4** | **PO4** | Description: DB11.bmp |  |

**EXPERIMENT NO.5**

**To grab Image from camera and apply aged detection algorithm**

**Title:** To grab Image from camera and apply edge detection algorithm.

**Objectives**: To understand & implement various image processing operations on a remotely captured image.

**Aim:** Write a Python program to grab the image from Camera and apply the edge detection algorithm(overloaded with Sobel variants, Others) to find the edges use BBB / ARM Cortex A5/A9/M4 Mobile Boards. Store the Images in SAN (for BIGDATA analitics)

**Requirements:**

**H/W Requirements:**

1. 8 GB RAM
2. 500GB/1TB HDD
3. Web Camera

**S/W Requirements:**

1. Latest version of 64 Bit Operating Systems Open Source Fedora-20.
2. Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading

**Theory:**

The edges extracted from a two-dimensional image of a three-dimensional scene can be classified as either viewpoint dependent or viewpoint independent. A *viewpoint independent edge* typically reflects inherent properties of the three-dimensional objects, such as surface markings and surface shape. A *viewpoint dependent edge* may change as the viewpoint changes, and typically reflects the geometry of the scene, such as objects occluding one another.

A typical edge might for instance be the border between a block of red color and a block of yellow. In contrast a **line** (as can be extracted by a ridge detector) can be a small number of pixels of a different color on an otherwise unchanging background. For a line, there may therefore usually be one edge on each side of the line.

Although certain literature has considered the detection of ideal step edges, the edges obtained from natural images are usually not at all ideal step edges. Instead they are normally affected by one or several of the following effects:

* focal blur caused by a finite depth-of-field and finite point spread function.
* penumbral blur caused by shadows created by light sources of non-zero radius.
* shading at a smooth object

A number of researchers have used a Gaussian smoothed step edge (an error function) as the simplest extension of the ideal step edge model for modeling the effects of edge blur in practical applications. Thus, a one-dimensional image fwhich has exactly one edge placed at x = 0may be modeled as:

f(x) = \frac{I_r - I_l}{2} \left( \operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + 1\right) + I_l.

At the left side of the edge, the intensity is I_l = \lim_{x \rightarrow -\infty} f(x), and right of the edge it is I_r = \lim_{x \rightarrow \infty} f(x). The scale parameter \sigmais called the blur scale of the edge.

To illustrate why edge detection is not a trivial task, consider the problem of detecting edges in the following one-dimensional signal. Here, we may intuitively say that there should be an edge between the 4th and 5th pixels.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 7 | 6 | 4 | 152 | 148 | 149 |
|  |  |  |  |  |  |  |

If the intensity difference were smaller between the 4th and the 5th pixels and if the intensity differences between the adjacent neighboring pixels were higher, it would not be as easy to say that there should be an edge in the corresponding region. Moreover, one could argue that this case is one in which there are several edges.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 7 | 6 | 41 | 113 | 148 | 149 |
|  |  |  |  |  |  |  |

Hence, to firmly state a specific threshold on how large the intensity change between two neighboring pixels must be for us to say that there should be an edge between these pixels is not always simple. Indeed, this is one of the reasons why edge detection may be a non-trivial problem unless the objects in the scene are particularly simple and the illumination conditions can be well controlled (see for example, the edges extracted from the image with the girl above).

## Approaches

There are many methods for edge detection, but most of them can be grouped into two categories, search-based and [zero-crossing](http://en.wikipedia.org/wiki/Zero_crossing) based. The search-based methods detect edges by first computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction. The zero-crossing based methods search for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the [Laplacian](http://en.wikipedia.org/wiki/Laplacian) or the zero-crossings of a non-linear differential expression. As a pre-processing step to edge detection, a smoothing stage, typically Gaussian smoothing, is almost always applied (see also [noise reduction](http://en.wikipedia.org/wiki/Noise_reduction)).

The edge detection methods that have been published mainly differ in the types of smoothing filters that are applied and the way the measures of edge strength are computed. As many edge detection methods rely on the computation of image gradients, they also differ in the types of filters used for computing gradient estimates in the x- and y-directions.

A survey of a number of different edge detection methods can be found in (Ziou and Tabbone 1998) see also the encyclopedia articles on edge detection in *Encyclopedia of Mathematics* and Encyclopedia of Computer Science and Engineering.

John Canny considered the mathematical problem of deriving an optimal smoothing filter given the criteria of detection, localization and minimizing multiple responses to a single edge. He showed that the optimal filter given these assumptions is a sum of four exponential terms. He also showed that this filter can be well approximated by first-order derivatives of Gaussians. Canny also introduced the notion of non-maximum suppression, which means that given the pre smoothing filters, edge points are defined as points where the gradient magnitude assumes a local maximum in the gradient direction. Looking for the zero crossing of the 2nd derivative along the gradient direction was first proposed by Haralick.It took less than two decades to find a modern geometric variation meaning for that operator that links it to the [Marr–Hildreth](http://en.wikipedia.org/wiki/Marr-Hildreth_algorithm) (zero crossing of the Laplacian) edge detector. That observation was presented by [Ron Kimmel](http://en.wikipedia.org/wiki/Ron_Kimmel) and [Alfred Bruckstein](http://en.wikipedia.org/w/index.php?title=Alfred_Bruckstein&action=edit&redlink=1).[[10]](http://en.wikipedia.org/wiki/Edge_detection#cite_note-10)

Although his work was done in the early days of computer vision, the [Canny edge detector](http://en.wikipedia.org/wiki/Canny_edge_detector) (including its variations) is still a state-of-the-art edge detector.[[11]](http://en.wikipedia.org/wiki/Edge_detection#cite_note-11) Unless the preconditions are particularly suitable, it is hard to find an edge detector that performs significantly better than the Canny edge detector.

The Canny-Deriche detector was derived from similar mathematical criteria as the Canny edge detector, although starting from a discrete viewpoint and then leading to a set of recursive filters for image smoothing instead of [exponential filters](http://en.wikipedia.org/w/index.php?title=Exponential_filter&action=edit&redlink=1) or Gaussian filters.[[12]](http://en.wikipedia.org/wiki/Edge_detection#cite_note-12)

The [differential edge detector](http://en.wikipedia.org/wiki/Edge_detection#Differential_edge_detection) described below can be seen as a reformulation of Canny's method from the viewpoint of differential invariants computed from a [scale space representation](http://en.wikipedia.org/wiki/Scale_space_representation) leading to a number of advantages in terms of both theoretical analysis and sub-pixel implementation.

### Other first-order methods

Different gradient operators can be applied to estimate image gradients from the input image or a smoothed version of it. The simplest approach is to use central differences:

L_x(x, y)=-1/2\cdot L(x-1, y) + 0 \cdot L(x, y) + 1/2 \cdot L(x+1, y)\,

L_y(x, y)=-1/2\cdot L(x, y-1) + 0 \cdot L(x, y) + 1/2 \cdot L(x, y+1),\,

corresponding to the application of the following filter masks to the image data:


L_x = \begin{bmatrix}
-1/2 & 0 & 1/2
\end{bmatrix} * L
\quad \mbox{and} \quad
L_y = \begin{bmatrix}
+1/2 \\
0 \\
-1/2
\end{bmatrix} * L.


The well-known and earlier [Sobel operator](http://en.wikipedia.org/wiki/Sobel_operator) is based on the following filters:


L_x = \begin{bmatrix}
-1 & 0 & +1 \\
-2 & 0 & +2 \\
-1 & 0 & +1
\end{bmatrix} * L
\quad \mbox{and} \quad
L_y = \begin{bmatrix}
+1 & +2 & +1  \\
0 & 0 & 0 \\
-1 & -2 & -1
\end{bmatrix} * L.


Given such estimates of first-order derivatives, the gradient magnitude is then computed as:

|\nabla L| = \sqrt{ L_x^2 + L_y^2}

while the gradient orientation can be estimated as

\theta = \operatorname{atan2}(L_y, L_x).

Other first-order difference operators for estimating image gradient have been proposed in the [Prewitt](http://en.wikipedia.org/wiki/Prewitt) operator, [Roberts cross](http://en.wikipedia.org/wiki/Roberts_Cross) and [Frei-Chen](http://en.wikipedia.org/w/index.php?title=Frei-Chen&action=edit&redlink=1).

It is possible to extend filters dimension to avoid the issue of recognizing edge in low SNR image. The cost of this operation is loss in terms of resolution. Examples are Extended Prewitt 7x7 and [Abdou](http://en.wikipedia.org/wiki/Abdou)

### Thresholding and linking

Once we have computed a measure of edge strength (typically the gradient magnitude), the next stage is to apply a threshold, to decide whether edges are present or not at an image point. The lower the threshold, the more edges will be detected, and the result will be increasingly susceptible to [noise](http://en.wikipedia.org/wiki/Image_noise) and detecting edges of irrelevant features in the image. Conversely a high threshold may miss subtle edges, or result in fragmented edges.

If the edge thresholding is applied to just the gradient magnitude image, the resulting edges will in general be thick and some type of edge thinning post-processing is necessary. For edges detected with non-maximum suppression however, the edge curves are thin by definition and the edge pixels can be linked into edge polygon by an edge linking (edge tracking) procedure. On a discrete grid, the non-maximum suppression stage can be implemented by estimating the gradient direction using first-order derivatives, then rounding off the gradient direction to multiples of 45 degrees, and finally comparing the values of the gradient magnitude in the estimated gradient direction.

A commonly used approach to handle the problem of appropriate thresholds for thresholding is by using [thresholding](http://en.wikipedia.org/wiki/Adaptive_thresholding) with [hysteresis](http://en.wikipedia.org/wiki/Hysteresis). This method uses multiple thresholds to find edges. We begin by using the upper threshold to find the start of an edge. Once we have a start point, we then trace the path of the edge through the image pixel by pixel, marking an edge whenever we are above the lower threshold. We stop marking our edge only when the value falls below our lower threshold. This approach makes the assumption that edges are likely to be in continuous curves, and allows us to follow a faint section of an edge we have previously seen, without meaning that every noisy pixel in the image is marked down as an edge. Still, however, we have the problem of choosing appropriate thresholding parameters, and suitable thresholding values may vary over the image.

### Edge thinning

Edge thinning is a technique used to remove the unwanted spurious points on the edges in an image. This technique is employed after the image has been filtered for noise (using median, Gaussian filter etc.), the edge operator has been applied (like the ones described above) to detect the edges and after the edges have been smoothed using an appropriate threshold value. This removes all the unwanted points and if applied carefully, results in one pixel thick edge elements.

Advantages:

1. Sharp and thin edges lead to greater efficiency in object recognition.
2. If [Hough transforms](http://en.wikipedia.org/wiki/Hough_transform) are used to detect lines and ellipses, then thinning could give much better results.
3. If the edge happens to be the boundary of a region, then thinning could easily give the image parameters like perimeter without much algebra.

There are many popular algorithms used to do this, one such is described below:

1. Choose a type of connectivity, like 8, 6 or 4.
2. 8 connectivity is preferred, where all the immediate pixels surrounding a particular pixel are considered.
3. Remove points from North, south, east and west.
4. Do this in multiple passes, i.e. after the north pass, use the same semi processed image in the other passes and so on.

The number of passes across direction should be chosen according to the level of accuracy desired.

### Second-order approaches to edge detection

Some edge-detection operators are instead based upon second-order derivatives of the intensity. This essentially captures the [rate of change](http://en.wikipedia.org/wiki/Derivative) in the intensity gradient. Thus, in the ideal continuous case, detection of zero-crossings in the second derivative captures local maxima in the gradient.

The early [Marr-Hildreth](http://en.wikipedia.org/wiki/Marr-Hildreth_algorithm) operator is based on the detection of zero-crossings of the Laplacian operator applied to a Gaussian-smoothed image. It can be shown, however, that this operator will also return false edges corresponding to local minima of the gradient magnitude. Moreover, this operator will give poor localization at curved edges. Hence, this operator is today mainly of historical interest.

#### Differential edge detection

A more refined second-order edge detection approach which automatically detects edges with sub-pixel accuracy, uses the following *differential approach* of detecting zero-crossings of the second-order directional derivative in the gradient direction:

Following the differential geometric way of expressing the requirement of non-maximum suppression proposed by Lindeberg,[[4]](http://en.wikipedia.org/wiki/Edge_detection" \l "cite_note-lin98-4)[[13]](http://en.wikipedia.org/wiki/Edge_detection#cite_note-lin93-13) let us introduce at every image point a local coordinate system (u, v), with the v-direction parallel to the gradient direction. Assuming that the image has been pre-smoothed by Gaussian smoothing and a [scale space representation](http://en.wikipedia.org/wiki/Scale_space_representation) L(x, y; t)at scale thas been computed, we can require that the gradient magnitude of the [scale space representation](http://en.wikipedia.org/wiki/Scale_space_representation), which is equal to the first-order directional derivative in the v-direction L_v, should have its first order directional derivative in the v-direction equal to zero

\partial_v(L_v) = 0

while the second-order directional derivative in the v-direction of L_vshould be negative, i.e.,

\partial_{vv}(L_v) \leq 0.

Written out as an explicit expression in terms of local partial derivatives L_x, L_y... L_{yyy}, this edge definition can be expressed as the zero-crossing curves of the differential invariant

L_v^2 L_{vv} = L_x^2 \, L_{xx} + 2 \, L_x \,  L_y \, L_{xy} + L_y^2 \, L_{yy} = 0,

that satisfy a sign-condition on the following differential invariant

L_v^3 L_{vvv} = L_x^3 \, L_{xxx} + 3 \, L_x^2 \, L_y \, L_{xxy} + 3 \, L_x \, L_y^2 \, L_{xyy} + L_y^3 \, L_{yyy} \leq 0

where L_x, L_y... L_{yyy}denote partial derivatives computed from a [scale space representation](http://en.wikipedia.org/wiki/Scale_space_representation) Lobtained by smoothing the original image with a Gaussian kernel. In this way, the edges will be automatically obtained as continuous curves with sub-pixel accuracy. Hysteresis thresholding can also be applied to these differential and subpixel edge segments.

In practice, first-order derivative approximations can be computed by central differences as described above, while second-order derivatives can be computed from the [scale space representation](http://en.wikipedia.org/wiki/Scale_space_representation) Laccording to:

L_{xx}(x, y) = L(x-1, y) - 2 L(x, y) + L(x+1, y).\,

L_{xy}(x, y) = (L(x-1, y-1) - L(x-1, y+1) - L(x+1, y-1) + L(x+1, y+1))/4, \,

L_{yy}(x, y) = L(x, y-1) - 2 L(x, y) + L(x, y+1).\,

corresponding to the following filter masks:


L_{xx} = \begin{bmatrix}
1 & -2 & 1
\end{bmatrix} * L
\quad \mbox{and} \quad
L_{xy} = \begin{bmatrix}
-1/4 & 0 & 1/4 \\
0 & 0 & 0\\
1/4 & 0 & -1/4
\end{bmatrix} * L
\quad \mbox{and} \quad
L_{yy} = \begin{bmatrix}
1 \\
-2 \\
1
\end{bmatrix} * L.


Higher-order derivatives for the third-order sign condition can be obtained in an analogous fashion.

### Phase congruency-based edge detection

A recent development in edge detection techniques takes a frequency domain approach to finding edge locations. [Phase congruency](http://en.wikipedia.org/wiki/Phase_congruency) (also known as phase coherence) methods attempt to find locations in an image where all sinusoids in the frequency domain are in phase. These locations will generally correspond to the location of a perceived edge, regardless of whether the edge is represented by a large change in intensity in the spatial domain. A key benefit of this technique is that it responds strongly to [Mach bands](http://en.wikipedia.org/wiki/Mach_bands), and avoids false positives typically found around [roof edges](http://en.wikipedia.org/w/index.php?title=Roof_edge&action=edit&redlink=1). A roof edge, is a discontinuity in the first order derivative of a grey-level profile.[[14]](http://en.wikipedia.org/wiki/Edge_detection#cite_note-14)

Conceptually the Cortex-M4 is a Cortex-M3 plus DSP Instructions, and optional floating-point unit (FPU). If a core contains an FPU, it is known as a Cortex-M4F, otherwise it is a Cortex-M4. Key features of the Cortex-M4 core are:

* ARMv7E-M architecture
* Instruction sets
  + Thumb (entire)
  + Thumb-2 (entire)
  + 1-cycle 32-bit hardware multiply, 2-12 cycle 32-bit hardware divide, saturated math support
  + DSP extension: Single cycle 16/32-bit MAC, single cycle dual 16-bit MAC, 8/16-bit SIMD arithmetic
* 3-stage pipeline with branch speculation
* 1 to 240 physical interrupts, plus NMI
* 12 cycle interrupt latency
* Integrated sleep modes

**Silicon options:**

* Optional Floating-Point Unit (FPU): single-precision only IEEE-754 compliant. This is called the FPv4-SP extension.
* Optional Memory Protection Unit (MPU): 0 or 8 regions

### Chips

The following microcontrollers are based on the Cortex-M4 core:

* Atmel SAM4L, SAM4N, SAM4S
* Freescale Kinetis K

The following microcontrollers are based on the Cortex-M4F (M4 + FPU) core:

* Atmel SAM4C (dual core), SAM4E, SAMG
* Energy Micro EFM32 Wonder
* Freescale Kinetis K
* Infineon XMC4000
* NXP LPC4000, LPC4300(one Cortex-M4F + one Cortex-M0)
* STMicroelectronics STM32 F3, F4
* Texas Instruments LM4F, TM4C
* Spansion FM4F
* Toshiba TX04

The following chips have either a Cortex-M4 or M4F as a secondary core:

* Freescale Vybrid VF6 (one Cortex-A5 + one Cortex-M4F)
* Texas Instruments OMAP 5 (one dual-core Cortex-A15 + two Cortex-M4)

**Square wave**

A **square wave** is a non-sinusoidal periodic waveform (which can be represented as an infinite summation of sinusoidal waves), in which the amplitude alternates at a steady frequency between fixed minimum and maximum values, with the same duration at minimum and maximum. The transition between minimum to maximum is instantaneous for an ideal square wave; this is not realizable in physical systems. Square waves are often encountered in electronics and signal processing. Its stochastic counterpart is a two-state trajectory. A similar but not necessarily symmetrical wave, with arbitrary durations at minimum and maximum, is called a rectangular wave (of which the square wave is a special case).

Square waves are universally encountered in digital switching circuits and are naturally generated by binary (two-level) logic devices. They are used as timing references or "clock signals", because their fast transitions are suitable for triggering synchronous logic circuits at precisely determined intervals. However, as the frequency-domain graph shows, square waves contain a wide range of harmonics; these can generate electromagnetic radiation or pulses of current that interfere with other nearby circuits, causing noise or errors. To avoid this problem in very sensitive circuits such as precision analog-to-digital converters, sine waves are used instead of square waves as timing references.

In musical terms, they are often described as sounding hollow, and are therefore used as the basis for wind instrument sounds created using subtractive synthesis. Additionally, the distortion effect used on electric guitars clips the outermost regions of the waveform, causing it to increasingly resemble a square wave as more distortion is applied.

The **sine wave** or **sinusoid** is a mathematical curve that describes a smooth repetitive oscillation. It is named after the function sine, of which it is the graph. It occurs often in pure and applied mathematics, as well as physics, engineering, signal processing and many other fields. Its most basic form as a function of time (*t*) is**:**

y(t) = A\sin(2 \pi f t + \varphi) = A\sin(\omega t + \varphi)

The sine wave is important in physics because it retains its wave shape when added to another sine wave of the same frequency and arbitrary phase and magnitude. It is the only periodic waveform that has this property. This property leads to its importance in Fourier analysis and makes it acoustically unique.

A **storage area network** (**SAN**) is a dedicated network that provides access to consolidated, block level data storage. SANs are primarily used to enhance storage devices, such as disk arrays, tape libraries, and optical jukeboxes, accessible to servers so that the devices appear like locally attached devices to the operating system. A SAN typically has its own network of storage devices that are generally not accessible through the local area network (LAN) by other devices. The cost and complexity of SANs dropped in the early 2000s to levels allowing wider adoption across both enterprise and small to medium sized business environments.

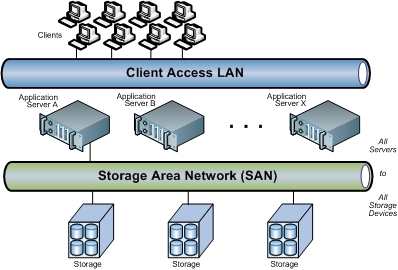
A SAN does not provide file abstraction, only block-level operations. However, file systems built on top of SANs do provide file-level access, and are known as *SAN filesystems* or shared disk file systems.

Sharing storage usually simplifies storage administration and adds flexibility since cables and storage devices do not have to be physically moved to shift storage from one server to another.

Other benefits include the ability to allow servers to boot from the SAN itself. This allows for a quick and easy replacement of faulty servers since the SAN can be reconfigured so that a replacement server can use the LUN of the faulty server. While this area of technology is still new, many view it as being the future of the enterprise datacenter.

SANs also tend to enable more effective disaster recovery processes. A SAN could span a distant location containing a secondary storage array. This enables storage replication either implemented by disk array controllers, by server software, or by specialized SAN devices. Since IP WANs are often the least costly method of long-distance transport, the Fibre Channel over IP (FCIP) and iSCSI protocols have been developed to allow SAN extension over IP networks. The traditional physical SCSI layer could only support a few meters of distance - not nearly enough to ensure business continuance in a disaster.

The economic consolidation of disk arrays has accelerated the advancement of several features including I/O caching, snap shotting, and volume cloning (Business Continuance Volumes or BCVs).



**Procedure:**

1. To grab the image from Camera.
2. Apply the edge detection algorithm on image
3. Store the Images in SAN.

**Mathematical Model:**

1. Let S be a system that describes that grabs the image and applies edge detection algorithm on it..

S = {…..}

2. Identify input as I

S = {I,N,…..}

Ii € I

Ii = The inputs to the system such as a a image

N =Number of image

3. Identify Output as O.

S = {I,N,O,…..}

O = Edges of the grabbed image.

4. Identify the process as P.

S = {I,N,O, P,……}

P ={Ps, Pb, Pc,Pr }

Ps = The process of capturing image.

Pb = The process of applying edge detection algorithm

Pc = The process of displaying output image

Pr = The process of storing image on SAN.

5. Identify success as Su

S = {I,N,O, P, Su, F……}

Su = Success is defined when Proper operation takes place as on requirement.

F = When improper operation is done .

6. Identify Initial condition Si

S = {I, N, O, P, Su, Si,….}

The user will capture image and finds edges using edge detection algorithm and store it in SAN.

S = I U N U O U P

**Observation:**

1. Capturing the image successful.
2. Edge detection algorithm detects all the edges of the images.

**Results:**

****

**Conclusion:** Understanding &Implementation of edge detection algorithm on captured image is done successfully. Stored it in SAN successfully.

**Oral Question Bank**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Q. No** | **Description** | **Attainment of** | | | |
| **CO** | **PO** | **BL\*** | **GA** |
| **1.** | Explain edge detection algorithm? | **CO4** | **PO1** | Description: DB13.bmp |  |
| **2.** | Describe the procedure for grabbing image from remote location | **CO4** | **PO1** | Description: DB13.bmp |  |
| **3.** | Cite various Edge detection algorithm | **CO4** | **PO3** | Description: DB13.bmp |  |
| **4.** | Analyze the use of edge detection in image processing | **CO4** | **PO4** | Description: DB11.bmp |  |
| **5.** | Discuss features of SAN can be utilized in edge detection. | **CO3** | **PO5** | Description: DB11.bmp |  |