2014

Imager Library

Holger Findling

My CI Group

5/28/2014

Imager Library

Contents

Chapter 1: Imager Library 1
Chapter 2: Imager Library Classes and Methods 2
CImgLib 2
CComputerVision 3
Image Support Functions 3
Biometric Support Functions 3
Segmentation Support Functions 3
CEffects 4
Pixelation 4
Noise Degradation 4
Backgrounds 4
CImageEnhancement 5
Brightness and Contrast 5
Noise Reduction 5
Image Sharpening 5
CImageProcessing 6
Color Processing 6
Edge Trace Algorithms 6
Arithmetic Operations 6
Boolean Logic Operations 6
CFilters 7
Classical Method 7
Rank Order Filtering 7
CImgFunctions 8
Bitmap 8
Image 8
B&W or Binary 8
Normalization 9
Histograms 9
Arithmetic Operations 9
Logical Functions and Operators 10
Convolution 10
ColorSpaceConversion 11
Grayscale Conversions 11
RGB and HSL Data 11
CImgData 12

Chapter 1: Imager Library

Imager Project is open source software designed to provide a common platform to analyze and research image processing and computer vision algorithms. It is intended for a broad spectrum of topics ranging from photography to robotic vision. Students and researchers should find it relatively easy to start their projects using Imager and the associated library.

Usage of Imager and associated software is restricted to academic research. No component of this software shall be used for any commercial or governmental application without the written consent from My CI Group.

Imager and the Imager Library are designed to run on MS Windows 7, but with little effort it can be ported to any Operating System. Imager Library is a managed C++ dynamic link library and all of its associated classes are developed in native C++.

The class hierarchy is provided below.

CImgLib
:CComputerVision
:CEffects
:CImageEnhancements
:CImageProcessing
:CFilters
:CImgFunctions
:ColorSpaceConversion

:CImgData

Chapter 2: Imager Library Classes and Methods

CImgLib

CImgLib is a place for new user defined image algorithms.

CComputerVision

CComputerVision(void)

Image Support Functions

The image support functions are not fully implemented. Left as an exercise for student project to do.

void MergeImages(CImgData^ pI1, CImgData^ pR1, CImgData^ pI2, CImgData^ pR2);

Biometric Support Functions

void CenterFingerprint (CImgData^ pI, CImgData^ pR)

Segmentation Support Functions

void SelectObjects (CImgData^ pI, CImgData^ pR)
int FindThreshold (CImgData^ pI)

CEffects

CEffects(void)

Pixelation

```
void Mosaic(CImgData^ pI, int size);
```

Noise Degradation

```
void AddRandomNoise(CImgData^ pI, int RndNoise);
long seed;
```

Backgrounds

```
void SolidBackground(CImgData^ pI,int red, int green, int blue);
void GradientBackground(CImgData^ pI, int Color);

void Lines(CImgData^ pI,int red, int green, int blue, int pos, int thickness, int type);
void SuperImposeImg(CImgData^ pI1, CImgData^ pI2, int FrmSize, int type);
```

CImageEnhancement

CImageEnhancement(void)

Brightness and Contrast

Two functions are provided to adjust the Brightness levels and Contrast.

void AdjustBrightness (const CImgData^ pOrigImg, CImgData^ pI, int bVal);
void AdjustContrast (CImgData^ pI, int center, int range, int cVal);

Noise Reduction

void NoiseReductionI(CImgData^ pI,CImgData^ pR);
void NoiseReductionII(CImgData^ pI,CImgData^ pR);

Image Sharpening

void SharpeningI(CImgData^ pI,CImgData^ pR);
void SharpeningII(CImgData^ pI,CImgData^ pR);

CImageProcessing

CImageProcessing(void)

Color Processing

The set of functions provide for converting a color image to the red, green, or blue component. Since the process is not reversible, a copy of the bitmap object should be made prior to applying the filter. Applying any two color processing functions consecutivley to a bitmap object results in a black image void of any image content.

```
void Red_Component(Bitmap^ _Bmp);
void Green_Component(Bitmap^ _Bmp);
void Blue_Component(Bitmap^ _Bmp);
```

Edge Trace Algorithms

```
void Laplace_Edge_Detection (CImgData^  pI)
void Prewitt_Edge_Detection (CImgData^  pI)
void Sobel_Edge_Detection (CImgData^  pI)
void Canny_Edge_Detection (CImgData^  pI)

void FollowEdge (CImgData^ pR, int x, int y, int k)
void Magnitude_XY (CImgData^  pX, CImgData^  pY, CImgData^  pR)
```

Arithmetic Operations

```
void SuperImpose(CImgData^ pImg1, CImgData^ pImg2, int I1, int I2);
void SubtractImage(CImgData^ pImg1, CImgData^ pImg2);
```

Boolean Logic Operations

```
void Invert(CImgData^ pI);
void AND(CImgData^ pImg1, CImgData^ pImg2);
void OR(CImgData^ pImg1, CImgData^ pImg2);
void XOR(CImgData^ pImg1, CImgData^ pImg2);
int LogicThreshold;
```

CFilters

void CFilters(void)

Classical Method

This class of filters is most effective in processing Gaussian noise. A slight blurring of the image may be realized by applying the filters to the image.

void Box_Filter_3x3 (CImgData^ pI,CImgData^ pR)
void Box_Filter_5x5 (CImgData^ pI,CImgData^ pR)
void Binomial_Filter_3x3 (CImgData^ pI,CImgData^ pR)
void Binomial_Filter_7x7 (CImgData^ pI,CImgData^ pR)
void Rotational_Filter_3x3 (CImgData^ pI,CImgData^ pR)
void Gaussian_Filter_3x3 (CImgData^ pI,CImgData^ pR)
void Gaussian_Filter_5x5 (CImgData^ pI,CImgData^ pR)

TheBlur filter is designed to remove high frequency components from an image, resulting in blured image objects. The results are useful when the blurred objects are converted to B&W to calculate the centroid of objects.

void Blur_Filter (CImgData^ pI,CImgData^ pR)

Rank Order Filtering

This class of filters is most effective in processing exponential noise also referred to speckle noise. Blurring of image objects is not realized; however the filter may erode corners and thin lines.

void Median_Filter_3x3 (CImgData^ pI,CImgData^ pR)
void Median_Filter_5x5 (CImgData^ pI,CImgData^ pR)

CImgFunctions

Class CImgFunctions contains the low level image processing algorithms.

```
void CImgFunctions(void)
```

Bitmap

The LoadBitmap function loads a bitmap into memory space and populates the CImageData.

```
void LoadBitmap (Bitmap^ _Bmp,
                 CImgData^ pOrigImg,
                 CImgData^ pI,
                 CImgData^ pImgObj1)
```

Image

The set of functions provides for copying CImgData objects from and to bitmap objects. Image data can also be copied between CImgData objects. The input parameters are organized in From Source – To Destination fashion.

```
void CopyBmpToImageArray (Bitmap^  Bmp, CImgData^  pI)
void CopyImageArrayToBMP (CImgData^  pI, Bitmap^  Bmp)
void CopyImageArrayToImageArray (CImgData^  pI, CImgData^  pR)
void CropImage (CImgData^  pOrigImg, CImgData^  pI)
void ClearImage (CImgData^ pI)
```

B&W or Binary

The Convert_To_Binary function produces a binary image where all pixels are either black (0, 0 , 0) or white (255,255,255). The conversion is obtained by creating a histogram with 13 bins. Pixel values from the bin with the largest count and a tolerance of +/- 15 become white, all other pixels become black. Bin[0] and Bin [12] are not included in selecting the conversion factor.

The Band +/- Range can be adjusted to determine the conversion to the binary image.
A Threshold can also be set to perform the conversion. Refer to the Image User Manual for specific operation.

```
void Convert_To_Binary (CImgData^  pI )
void Convert_To_Binary_Range (CImgData^  pI )
void Convert_To_Binary_Threshold (CImgData^  pI )
```

```
int Band
int Range
int Threshold
```

Normalization

Normalization of pixel values may be necessary when calling the convolution functions. The convolution functions may create negative pixel values and values greater than 255.

The NormalizeTruncate function truncates pixel values to the range [0 .. 255] by setting all negative values to 0 and pixel values greater than 255 to value 255.

The NormalizeCompress function takes the actual range of pixel values and sets it to range [0 .. 255].

```
void NormalizationI (CImgData^  pI)
void NormalizationII (CImgData^  pI)
void NormalizeTruncate (CImgData^  pI)
void NormalizeCompress (CImgData^  pI)
void AverageLuminance (CImgData^  pI)
```

Histograms

```
void Histogram (Bitmap^  Bmp)
void Histogram (CImgData^  pI)
array<int>^RedHistogramArray
array<int>^GreenHistogramArray
array<int>^BlueHistogramArray

array<int>^GrayHistogramArray
array<int>^CDFHistogramArray
array<int>^CDFNormalizedArray
```

Arithmetic Operations

TheSubtractImage function provides the difference between two CImgData objects. The results are normalized to illiminate negative pixel values.

p1 = p1 – p2.

```
void SuperImpose (CImgData^ pImg1, CImgData^ pImg2, int I1, int I2);
void AvgImg (CImgData^ pImg1, CImgData^ pImg2);
```

```
void SubtractImage (CImgData^ pImg1, CImgData^ pImg2);
```

Logical Functions and Operators

```
void Invert (CImgData^  pI)
void AND (CImgData^ p1, CImgData^  p2)
void OR (CImgData^  p1, CImgData^  p2)
```

Convolution

Two convolution masks are provided, which are of size 3x3 and 5x5.
Call functions Set_Mask_3x3() or Set_Mask_5x5() before executing Convolve3x3() or Convolve5x5().
The function RotateConvolve3x3() should not be called directly.

```
Set_Mask_3x3 (int _W11, int _W12, int _W13,
              int _W21, int _W22, int _W23,
              int _W31, int _W32, int _W33,
              int _SumOfWeight)

Set_Mask_5x5 (int _W11, int _W12, int _W13, int _W14, int _W15,
              int _W21, int _W22, int _W23, int _W24, int _W25,
              int _W31, int _W32, int _W33, int _W34, int _W35,
              int _W41, int _W42, int _W43, int _W44, int _W45,
              int _W51, int _W52, int _W53, int _W54, int _W55,
              int _SumOfWeight)

void Set_Mask_7x7 (
        int _W11, int _W12,  int _W13, int _W14, int _W15,  int _W16, int _W17,
        int _W21, int _W22, int _W23, int _W24, int _W25, int _W26, int _W27,
        int _W31, int _W32, int _W33, int _W34, int _W35, int _W36, int _W37,
        int _W41, int _W42, int _W43, int _W44, int _W45, int _W46, int _W47,
        int _W51, int _W52, int _W53, int _W54, int _W55, int _W56, int _W57,
        int _W61, int _W62, int _W63, int _W64, int _W65, int _W66, int _W67,
        int _W71, int _W72, int _W73, int _W74,  int _W75, int _W76,  int _W77,
        int _SumOfWeight);

void Convolve3x3 (CImgData^  pI, CImgData^  pR)
void Convolve5x5 (CImgData^  pI, CImgData^  pR)
void Convolve7x7 (CImgData^  pI, CImgData^  pR)

void RotateConvolve3x3 (CImgData^  pI, CImgData^  pR)
```

ColorSpaceConversion

Class ColorSpaceConversion contains functions to convert between color spaces RGB and HSV.

void ColorSpaceConversion(void)

Grayscale Conversions

The set of functions provide for converting a color image to grayscale. Operations can be performed between the HSL and RGB data.

Function ConvertToGrayScale converts the image to a grayscale image. It calls functions CalculateGrayscale() and Grayscale() to perform the conversion. CalculateGrayscale() calculates the grayscale values and stores them in the image array *Gray[]* . Function Grayscale() copies the grayscale value into the arrays pRed, pGreen, and pBlue.

void ConvertToGrayScale (CImgData^ pI)
void Grayscale(CImgData^ pI)
void CalculateGrayscale (CImgData^ pI)

RGB and HSL Data

void ConvertHSLtoRGB (CImgData^ pI);
void ConvertRGBtoHSL (CImgData^ pI);

CImgData

CImgData contains the RGB and HSL image components. The image algorithms in this library are designed to operate on the CImgData objects.

```
CImgData (void)
CImgData (Bitmap^ Bmp)
CImgData (CImgData^  pI)
CImgData (int width, int height)

int Size;
int Height;
int Width;

array<int>^ pRed;
array<int>^ pGreen;
array<int>^ pBlue;

array<int>^ pHue;
array<int>^ pSat;
array<int>^ pGray;
```