



Pattern Recognizer

Varada Kolhatkar
*Msc Computer Science,
Fergusson College*



I n t r o d u c t i o n

Image Processing has turned out to be a very useful technique. At the entrance of a big institute we can see Face Recognition system that serves security. Signature matching system is another way.

One aspect of image processing is its usage in weather forecasting and scientific applications. By knowing some historical data we can have some results that help in weather forecasting.



Project Description

The assumed application of the project will be finding out velocity of wind at a particular instance of time. Some cloud images are taken through a high resolution web cam by noting the timings of clicks of the images. The user selects a cloud as an object from the second image. Provided the time interval is very small, the background of the images is almost same and only the position of the moving object is changed. The selected object is searched in the first image. The maximum correlation point is noted.



Project Description

By knowing the correlation point in the first image and the center of mass of the object in the second image we get the relative distance covered by that object. By knowing the resolution and the relative distance we can have the actual distance. We know the time period as well as actual distance covered by that object so we can compute the velocity of the moving object.



Technology Used

The Project is developed using COM specifications. It has client server architecture. Server has the responsibility of providing all functionality to the client through interfaces. Client provides all GUI and uses services provided by the server. Client is completely encapsulated from the actual logic of the functionality he is using. He just has the interfaces and the DLL provided by the server.

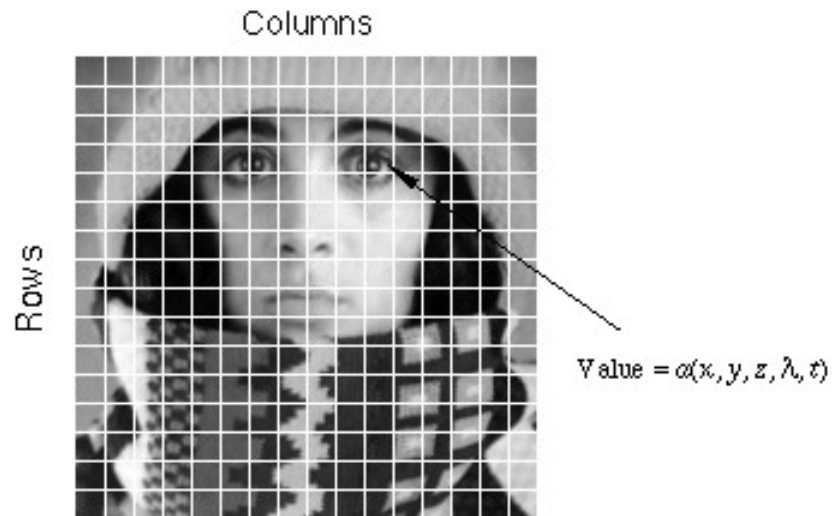


Image Processing Techniques

Definition of an Image

An image defined in the "real world" is considered to be a function of two real variables, for example, $a(x,y)$ with a as the amplitude (e.g. brightness) of the image at the *real* coordinate position (x,y) . A digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a *sampling* process that is frequently referred to as digitization. For now we will look at some basic definitions associated with the digital image.

An Image



Digitization of a continuous image. The pixel at coordinates $[m=10, n=3]$ has the integer brightness value 110.



Image Processing Techniques

Median Filter

- Replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median)
- Forces the points with distinct gray levels to more like their neighbors
- Isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $n^2/2$ (one half the filter area), are eliminated by an n^2 median filter
- Large clusters are affected considerably less

Image Processing Techniques

An example of filtering using median filter

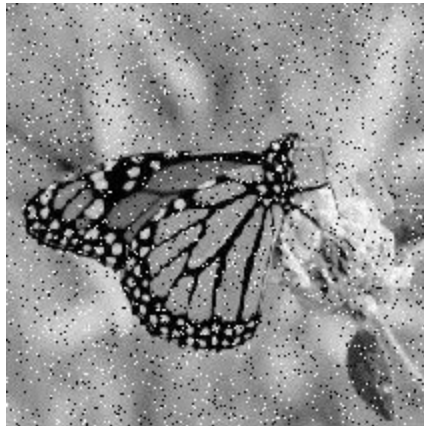





Image Processing Techniques

Edge Detection

This is the most common approach for detecting meaningful discontinuities in gray level. We discuss approaches for implementing first order derivative (Gradient operator) second order derivative (Laplacian operator).

An Edge

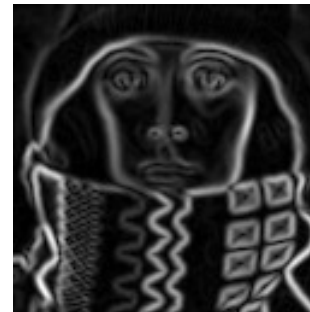
An edge is a set of connected pixel that lies on the boundary between two regions.



Just as smoothing is a fundamental operation in image processing so is the ability to take one or more spatial derivatives of the image. The fundamental problem is that, according to the mathematical definition of a derivative, this cannot be done. A digitized image is not a continuous function $a(x,y)$ of the spatial variables but rather a discrete function $a[m,n]$ of the integer spatial coordinates. As a result the algorithms we will present can only be seen as approximations to the true spatial derivatives of the original spatially-continuous image.

Image Processing Techniques

As an image is a function of two (or more) variables it is necessary to define the direction in which the derivative is taken. For the two-dimensional case we have the horizontal direction, the vertical direction, or an arbitrary direction which can be considered as a combination of the two. An example of *Edge Detection* using *Sobel* operator is as shown below.



Algorithms

Filtering

```
For i=0 to BitmapHeight - 2
  For j=0 to BitmapWidth - 2
    m=0
    n=0
    For k=i to i+3
      For l=j , n=0 to l < j+3
        Temp[m][n]=Bitmap[l][k]
      End For
    End For
    Bitmap[j+1][i+1]=CalculateMedian(Temp);
  End For
End For
```



Edge Detection

```
For i=0 to BitmapHeight - 2
  For j=0 to BitmapWidth - 2
    m=0
    n=0
    For k=i to i+3
      For l=j , n=0 to l < j+3
        Temp[m][n]=Bitmap[l][k]
      End For
    End For
    Bitmap[MiddlePosition]=CalculateGradient(Temp)
  End For
End For
```



Correlation

For i=0 to BitmapHeight – TemplateBitmapHeight

For j=0 to BitmapWidth – TemplateBitmapWidth

m=0

n=0

For k=i to i+TemplateHeight

For l=j , n=0 to l < j+TemplateWidth

Temp[m][n]=Bitmap[l][k]

End For

End For

Bitmap[MiddlePosition]=CorrelatePatterns(Temp)

End For

End For



CorrelatePattern returns Correlation Coefficient which can be calculated as:

$$C(x, y) = \frac{\sum \{ \sum [f(s, t) - \bar{f}] (w(s+x, t+y) - \bar{w}) \}}{\{ [\sum ((f(s, t) - \bar{f})^2) [\sum ((w(s+x, t+y) - \bar{w})^2)] \}^{1/2}}$$



S y s t e m R e q u i r e m e n t s

Hardware Requirements

USB port,
Web Cam,

Software Requirements

Web Cam Software,
Visual Studio 6.0



Feasibility Study

Technical

The software uses a web cam and USB port. These days web cams are readily available and are very cheap. Also no machine is without USB port.

Economical

The people who are going to use the software already have a good resolution web cam and machines with USB port so the system is definitely economically feasible.

Operational

The GUI of the system is interactive which makes general user to use it.



Designed Interfaces

IImageAcquisition

CaptureBitmap(HWND)

LoadBitmap(HWND , char * , int *)

ShowBitmap(HDC , int)

SaveBitmap(HANDLE , int)

IImageProcessing

Filtering(int)

EdgeDetection(int)



IColorProcessing

RemoveRedComponent(int , char , int)

RemoveGreenComponent(int , char , int)

RemoveBlueComponent(int , char , int)

RedScaling(int , char , int)

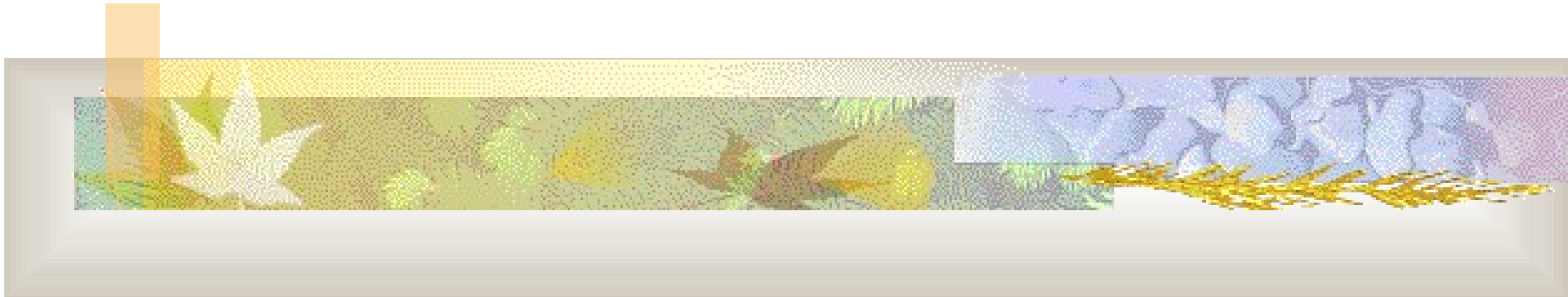
GreenScaling(int , char , int)

BlueScaling(int , char , int)

ICorrelate

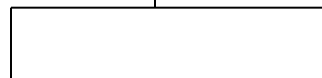
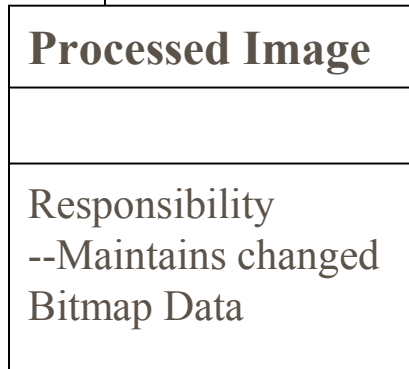
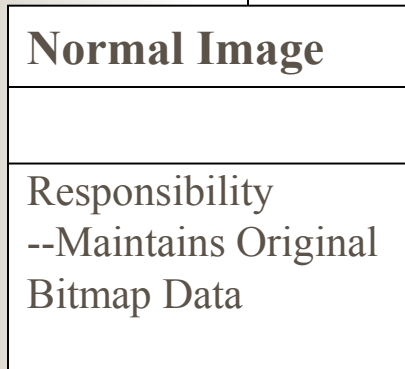
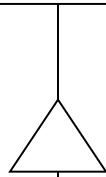
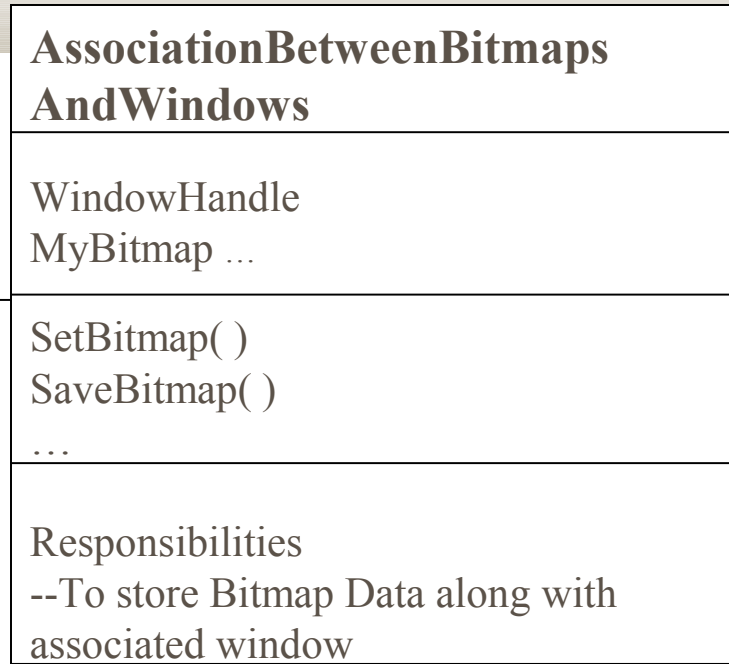
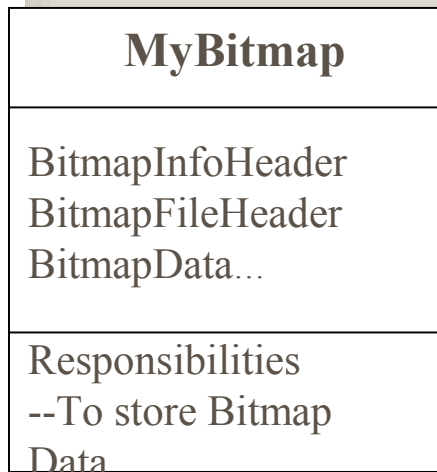
Correlate(HWND, int, int, int, int, int, int, POINT &)

DIAGRAMS

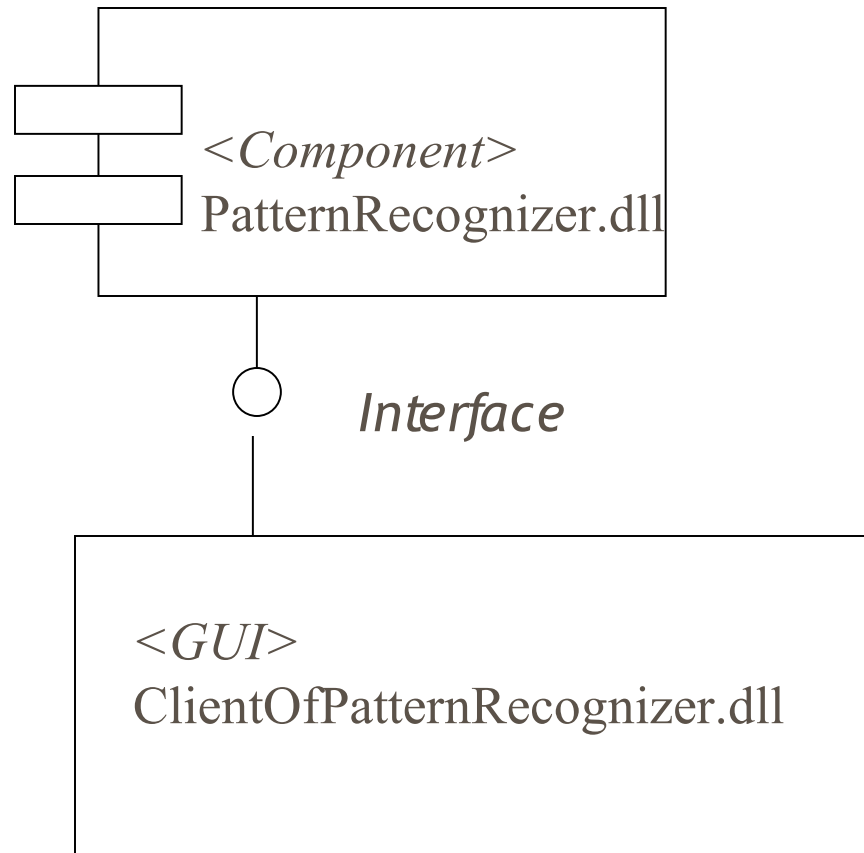




Class Diagram



Component Diagram





Acknowledgments

- Pattern Recognizer is based on the software which was developed by Sheetal Kudi and Varada Kolhatkar in 2003 under the guidance of Dr Amit Kesarkar.
- New features in this version are
 - Uses COM architecture
 - Edge Detection

Thank You

