Application Specific Image Feature Extractor

2015

An application which utilizes computer vision to automatically identify objects available at a departmental store; a more versatile replacement of the common bar-code scanner.

CSE Mini-Project 7th Semester

Table of Contents

INTRODUCTION

Overview –

This report is a documentation of the workings of the Application Specific Image Feature Extractor. This includes the black box test scenarios coupled alongwith the white box test scenarios.

Motivation and Background –

In a departmental store, a bar-code scanner is used at the billing counter to log the commodities purchased by the customers and generate bills. The process of finding the bar-code on the commodity is often cumbersome for the cashier manning the billing counter. Moreover many a times, the bar-code on the commodity is damaged or unreadable by the bar-code scanner causing the cashier to manually enter the bar-code number into the system. All these predicaments translate into loss of precious business time for the corporations or individuals owning the stores. This application seeks to speed up the billing process by replacing the bar-code scanner with a camera. The commodity to be billed may be placed in any orientation in front of the camera. The application will click a snapshot of the same, analyse the image and subsequently add the corresponding price of the commodity to the customer's bill.

ACKNOWLEDGEMENT

I would like to thank Prof. R.R. Tewari for his invaluable guidance provided throughout the project development.

Also, this project would have been impossible without the continuous support received from the Microsoft Developer Network (MSDN) and the MATLAB help forums.

Lastly, but not the least, I thank everyone else involved directly or indirectly with the development of this software, as this page is too short to list everyone.

CERTIFICATE

This is to certify that Mr. Souham Biswas has successfully prepared and completed the project under my direct and close supervision and that this is a bonafide piece of work done by him.

Class: B.Tech , VIIth semester

Branch: Computer Science Engineering

Academic Year: 2014-15

Institution Name: J.K. Institute of Applied Physics & Technology

Signature of Examiner: _____

Date: _____

SOFTWARE PRE-REQUISITES

- .NET Runtime 4.0

  The application front-end will be developed in C#, and is a Windows Presentation Foundation (WPF) Based Application. The fact that it is .NET 4.0 based allows it to run on most Windows based machines, as .NET 4.0 is shipped by default alongwith all Windows Operating Systems.

- MATLAB

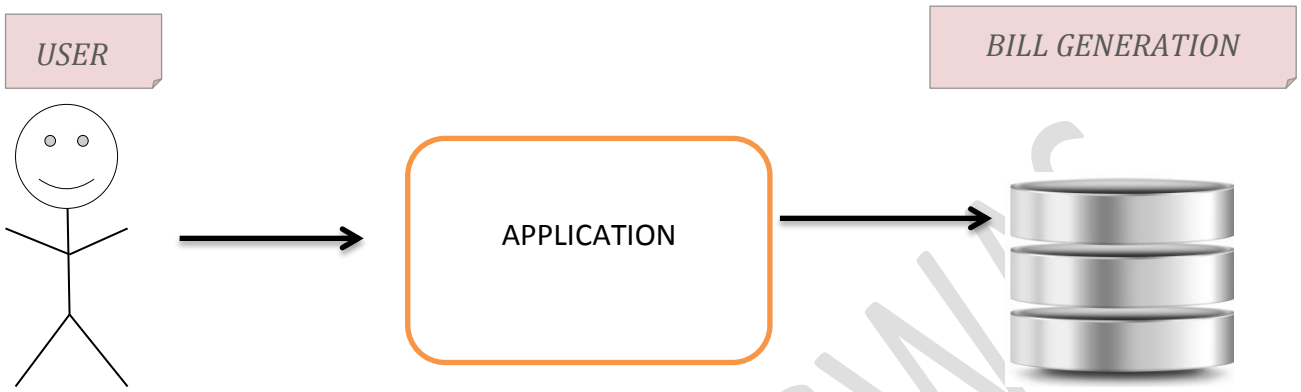
  The computer vision algorithms for procuring and enhancement of the image are implemented in MATLAB. Also, the neural network computations for image recognition are also carried out in MATLAB.

- Internet Connection

  An active network connection is required for accessing the commodity database which maintains the details of the commodities available at the dept. store.

SOFTWARE OVERVIEW

This software is basically a graphical interface to identify commdities being billed with the help of a camera



The software "Application Specific Image Feature Extractor" is an application which will take the image of a commodity in a departmental store and output the description of the object which will include the price, weight etc. The software will work in conjunction with additional camera hardware which will be used to obtain the image of the object to be classified. The user will have to place the object in front of the camera and the software will automatically obtain the image and add the corresponding price of the object as stated in the database, to the customer's bill. The system can be further upgraded to incorporate analysis of an image containing multiple objects and simultaneously processing and recognizing each object parallelly.

MAIN FUNCTIONAL OUTLINE

The main working moduled of the application is divided into the following parts–

- **Camera Functions**

  This module controls the working of the camera harfware and focusses mainly on procuring images properly –

  - **ClickImage**

    This function, as the name suggests, returns an image from the camera in the given format upon being called.

    Format Types (not exhaustive as it is H/W dependent):

    - RGB
    - YUV

  - **MainCameraFunction**

    This handles the entire operation. It is the starting point of the application. The app starts by initializing itself with a reference picture of the background when no commodity is placed in front of it.

➢ Image Processing Functions

This module contains routines to process and enhance the image from the camera. One of the keyfunction performed by this module is the removal of background so as to have only the commodity's image in the foreground–

- **BackgroundRemove**
  As the name suggests, this function in at the highest level of abstraction in the process of background removal.

- **ImageSubtract**
  This function is used to subtract 2 images so as to enumerate the object and separate it from the background.

- **GetImageCorners**
  Used to extract specific areas of the image. This function is used in the process of background of the image.

- **GetShadecover**
  This function returns the percentage of a grayscale image containing pixels of intensity value as specified in its arguments. This function is also used in the process of background removal.

➢ Neural Network Functions

This module conatins functions which generate the dataset to be utilized for training the neural network model to recognize

the images. Also, this contains functions which take input as an image and return the name of the object. –

- **GenerateDataset**
  This function takes a list of images as its argument and subsequently parses it to make it feedable to the neiral network for training.

- **RecognizeImage**
  This function takes an image as input and returns the name of the commodity in the image by utilizing the trained neural network.

FEATURES

I. Maintainability-

The software will be developed in a modular manner. The modules will have the capability to be deployed in other scenarios. Therefore, provision to interface new modules and functionality will be provided.

II. Portability-

Portability is limited as the application will be developed using C# and MATLAB limiting the platform of operation to mostly windows based systems and system supporting the MATLAB runtime environment.

III. Reliability-

For sufficient reliability, it is imperative that the camera hardware be of a certain quality standard required for proper and robust image recognition. For a reliable interface with the commodity database, it is also important that the network connection possess sufficient bandwidth.

TECHNOLOGIES AND SOFTWARES USED

C# Programming Language –

C# is a .NET based object oriented programming language which is analogous to Java, but is much more powerful and object oriented.

It was developed by Microsoft as a response to rising utility for internet and cloud based applications.

C# applications run in a virtual machine called the CLR or Common Language Runtime, which allows for automatic memory management and native machine code optimization.

The CLR allows any .NET based language (VB, F#, etc.) to run in a single common environment, which hence allows for code sharing.

C# code on compilation is converted to MSIL or Microsoft Intermediate Language, instead of native machine code as in the case of C or C++. Then, when the application is run, only those modules in use, are conditionally compiled to native machine code before execution. This compilation technique is popularly known as

JIT or Just In Time compilation and results in a smaller memory footprint.

MATLAB –

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

Windows Presentation Foundation (WPF) –

Earlier all desktop based GUI apps were developed using Windows Forms.

WPF is a new technology, which allows for advanced controls and graphical effects to be integrated into desktop apps.

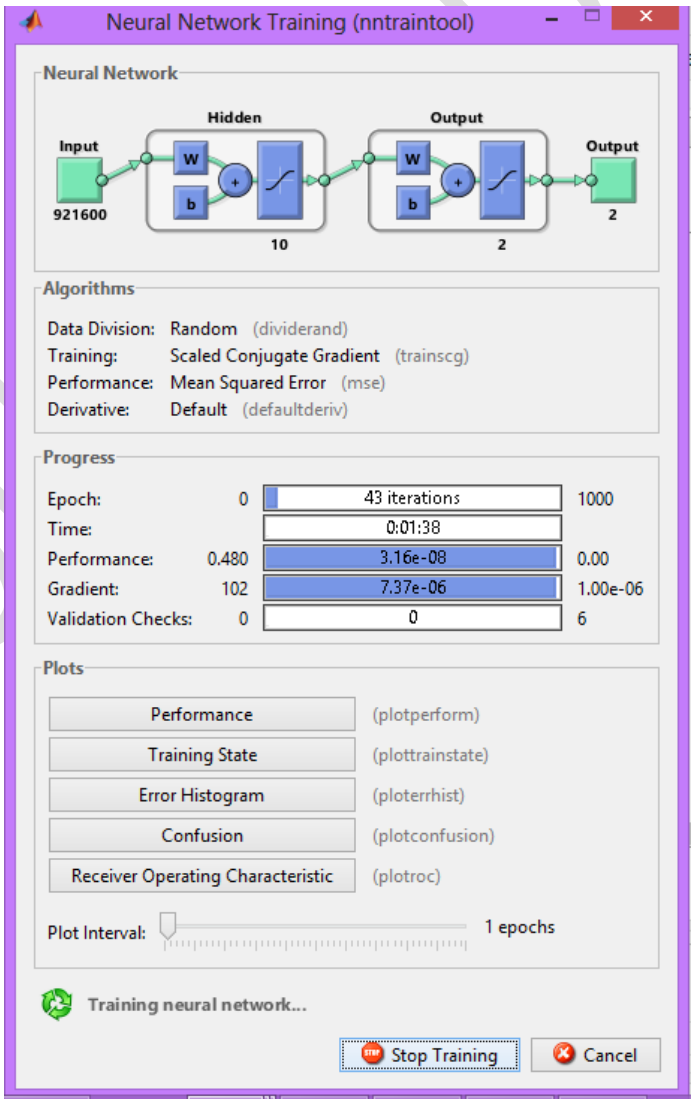
Unlike Windows forms, the frontend of the application can be designed using XAML (Xtensible Application Markup Language)

which is analogous to XML with extension ".xaml", while the backend logic part is a code-behind file with the extension ".xaml.cs".

In Windows Forms, the frontend was also programmatically designed, which made maintenance and modification of the GUI a nightmare on large applications.

SOFTWARE WORKINGS

Neural Network Training



This window is displayed when the neural network is being trained with the test images. The training progress can be monitored in real-time by the progress bars in the window. Also, to check the correctness and robustness of the neural model trained, provisions are there to plot the Confusion Matrix, Error Histogram etc.

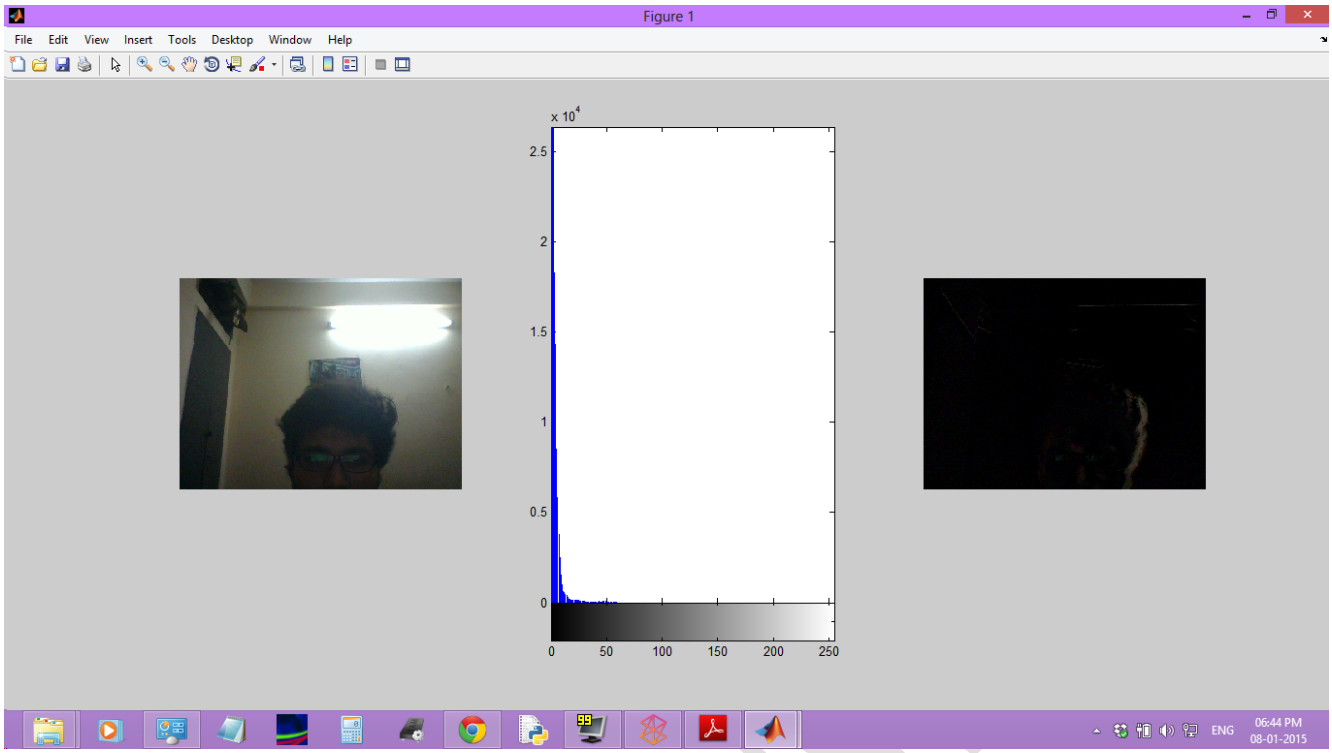
Background Removal

Illustrated previous is the process of background removal. The algorithm involves subtracting the input image from a reference image which is assumed to be the background. To account for the change in the environment and lighting conditions, the reference image is changed adaptively. For this purpose, a difference image is calculated by initially subtracting the input image from the previous reference image. This difference image is then subsequently converted into grayscale, and its histogram is analysed.

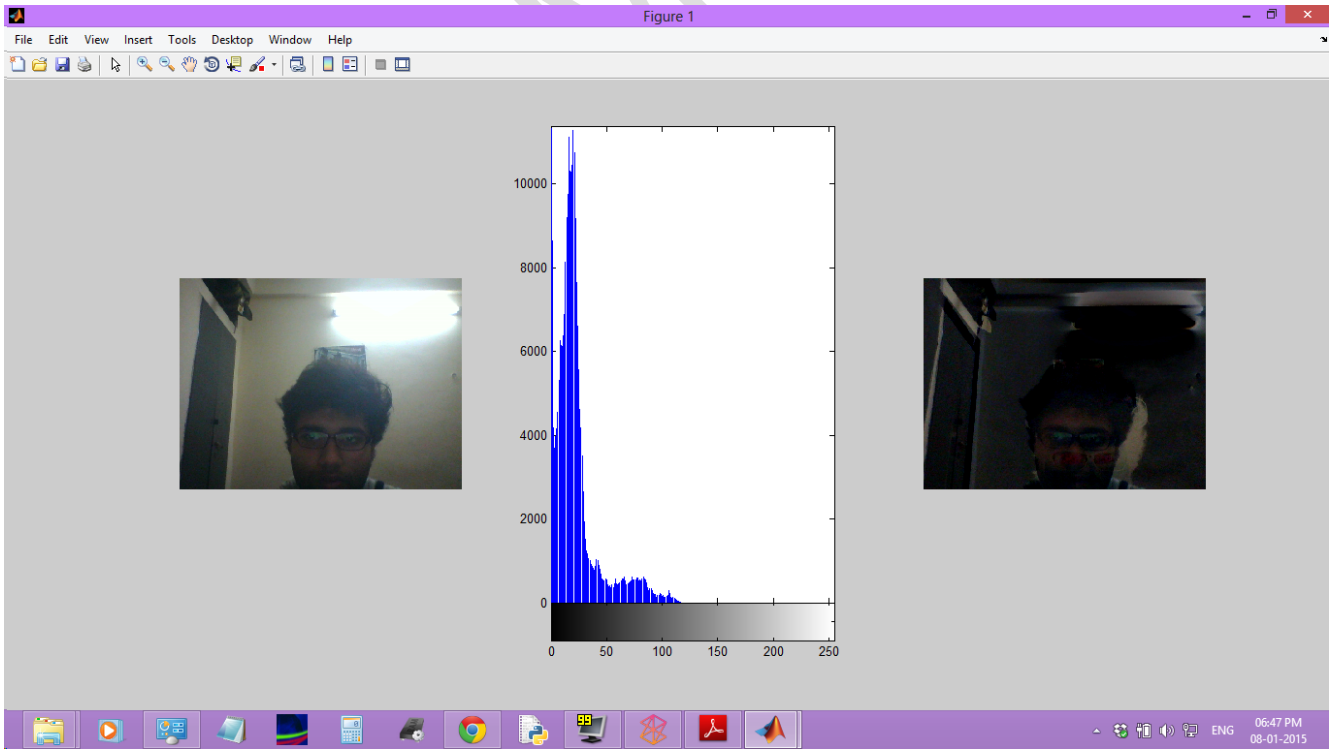
The double derivative at each point on the histogram is discretely calculated, and the point having the maximum value is found out. Following this, the histogram is integrated from the beginning to this point using simple trapezoidal integration. The value so obtained is divided by the total number of pixels to get a value denoting the "amount of darkness" in the image.

Furthermore, this quantity is also calculated at the corners and a few more specific areas of the image. All the quantities are finally then added and averaged to get a final quantity termed *bFactor*.

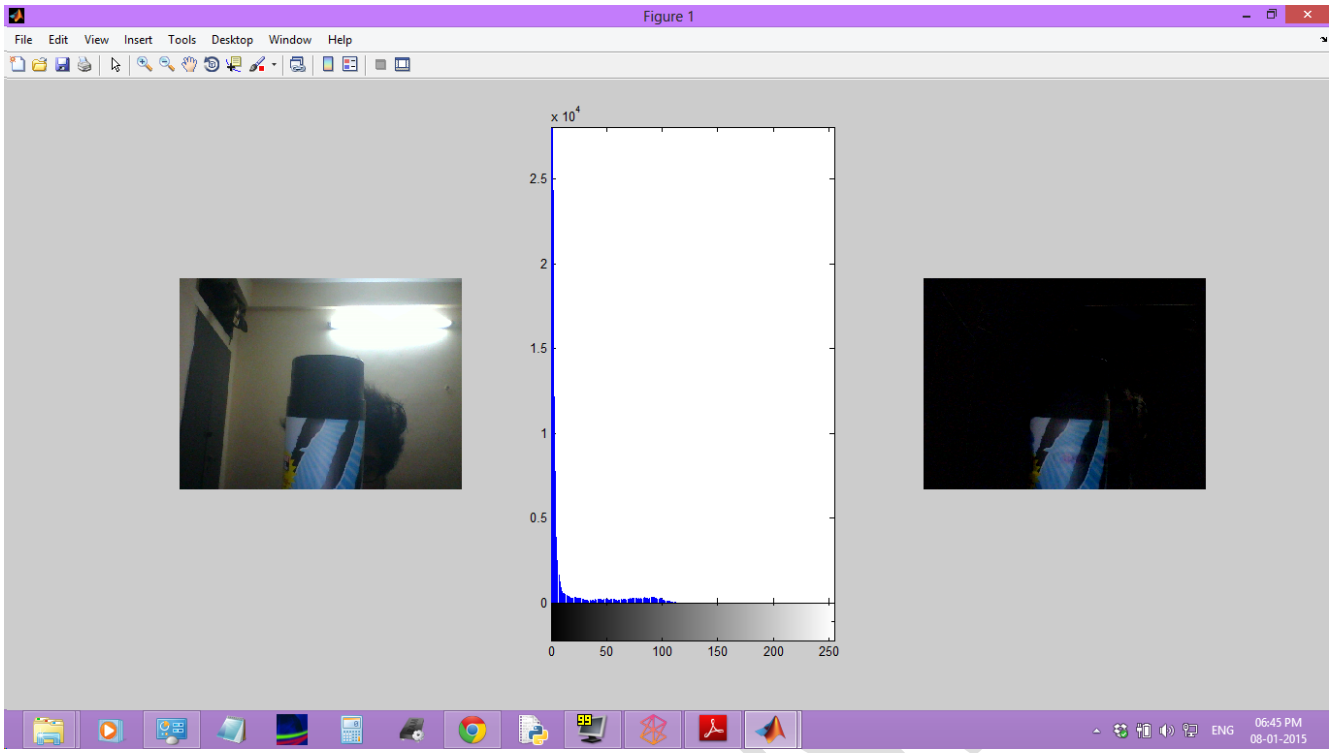
The value of *bFactor* lies in different ranges for object detection and background distortion respectively. This is used to adaptively control the reference image according to the lighting conditions. It is illustrated ahead.



Background Normal

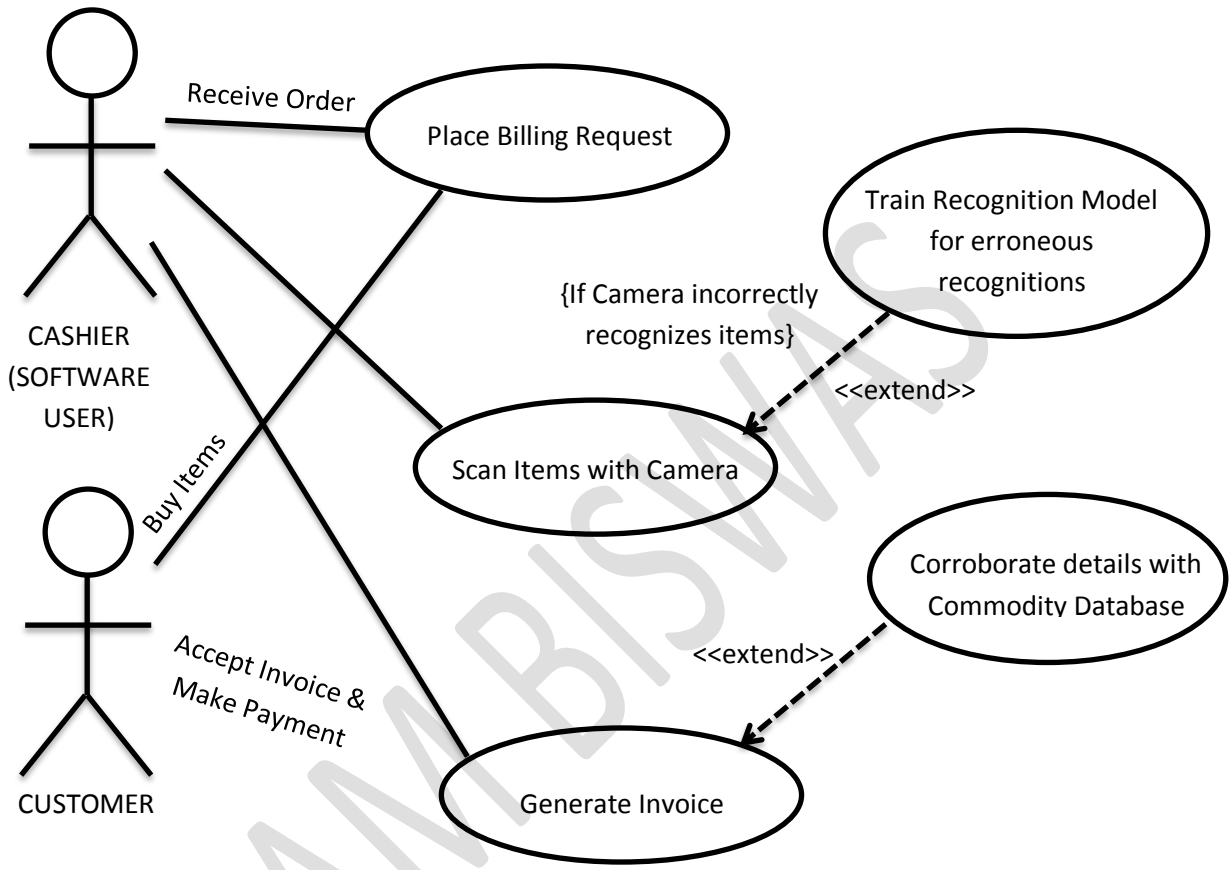


Distortion in background



Object in front of Camera

Use Case Diagram –



The main functions performed by the system are listed below –

1. Billing- The user places the commodity in front of the camera, whose feed is shown in real-time in the Video Feed control/window. The object upon being recognized is shown in the Identified Object window/control, waiting for the user to press the Confirmation Button. Provided the object is correctly identified, the user presses the Confirmation Button following which, the details pertaining to the object is pulled from the commodity database and is added to the customer invoice. If the object is incorrectly identified, the user will press the Incorrect Identification Button following which, the

user will be taken to the New Commodity Window for the addition of the new item in question.

2. Addition of New Item- The process of training the object recognition model involves placing the object in front of the camera and taking multiple images of it. The New Commodity Window will have the provisions to carry out all such functions at the press of a button. The multiple images recorded will be used to train the model which will be followed by validation and testing of the model using more images captured by the camera. It is imperative that the object be kept in front of the camera for the entire process following which, a confirmation message will be displayed. The newly added item will be recognized in the future by the system.

CODE

Camera Functions

cmfn2obj.m

```
vid=videoinput('winvideo',1, 'YUY2_640x480');
figure(1);
pause(1);
imgbg = clkpic(vid);

while(1)
    imgobj = clkpic(vid);
    imgrev = bgrv2(imgobj, imgbg);
    imgrevgray = rgb2gray(imgrev);

    subplot(1, 3, 1)
    imshow(imgobj);

    subplot(1, 3, 2);
    imhist(imgrevgray, 10);

    subplot(1, 3, 3);
    imshow(imgrev);

%     subplot(3, 3, 4);
%     stem(x, count);

    [count, x] = imhist(imgrevgray);
    ydiff = diff(count, 2);
%     subplot(3, 3, 5);
%     xdiff = x(1:254);
%     stem(xdiff, ydiff);

    [m, i] = max(ydiff);
    iReal = i + 3

    blackCoverage1 = shadeCover([1:iReal], imgrevgray)

    [i1, i2, i3, i4, i5, i6, i7, i8] = getImgCorners(imgrevgray);
    bcvr = zeros(1,8);
    bcvr(1) = shadeCover([1:iReal], i1);
    bcvr(2) = shadeCover([1:iReal], i2);
    bcvr(3) = shadeCover([1:iReal], i3);
    bcvr(4) = shadeCover([1:iReal], i4);

    bcvr(5) = shadeCover([1:iReal], i5);
    bcvr(6) = shadeCover([1:iReal], i6);
    bcvr(7) = shadeCover([1:iReal], i7);
    bcvr(8) = shadeCover([1:iReal], i8);

    blackCoverage2 = sum(bcvr)/8
```

```
    bFactor = (blackCoverage1 + blackCoverage2)

%      subplot(1, 4, 4);
%      stem([1:4], bcvr);subplot(1, 4, 3);

%      if (bFactor > -0.6)
    if (blackCoverage2 < 0.4)
        imgbg = clkpic(vid);
    end

%      k = waitforbuttonpress;
%
%      while (k ~= 1)
%          k = waitforbuttonpress;
%      end

    % Do all image processing and analysis here
end
```

getcmfnd.m

```
% dev_info = imaqhwinfo('winvideo',1);
% sFormats = dev_info.SupportedFormats;
% sz = size(sFormats);
% len = sz(2);
% vFormat = sFormats(len);
vid = videoinput('winvideo',1, 'YUY2_640x480');
% preview(vid);
```

cmfn2.m

```
img=getsnapshot(vid);
img=ycbcr2rgb(img);
imwrite(img, 'lol2.jpg','jpg')
```

cmfn1txt.m

```
vid=videoinput('winvideo',1, 'YUY2_640x480');
%preview(vid);
%pause(3);
while(1)
    img=getsnapshot(vid);
    %img=ycbcr2rgb(img);
    bgrem;
    imshow(imgrev);
    % Do all image processing and analysis here
end
```

cmfn1obj.m

```
vid=videoinput('winvideo',1, 'YUY2_640x480');
%preview(vid);
```

```
%pause(3);
while(1)
    img=getsnapshot(vid);
    img=ycbcr2rgb(img);
    imgrev = bgrv(img); % function version not working due to mysterious reasons
:O
    imshow(imgrev);
    % Do all image processing and analysis here
end
```

clkpic.m

```
function [img] = clkpic(vid)
    img = getsnapshot(vid);
    img = ycbcr2rgb(img);
end
```

Image Processing Functions

subimgs.m

```
function [subimg] = subimgs(imgobj, imgbg)
    sz = size(imgbg);
    r = sz(1);
    c = sz(2);

    diffThreshold = 8;

    subimg = zeros(r, c, 3);

    imgobj = im2double(imgobj);
    imgbg = im2double(imgbg);

    for i = 1:r
        for j = 1:c
            diffR = imgobj(i, j, 1) - imgbg(i, j, 1);
            diffG = imgobj(i, j, 2) - imgbg(i, j, 2);
            diffB = imgobj(i, j, 3) - imgbg(i, j, 3);
%             [i j]
%             if (i == 185 && j==256)
%                 lol=0;
%             end
            diffRGBnet = ((abs(diffR) + abs(diffG) + abs(diffB))/3)
* 100;
%             diffRGBnet = (diffRGB / 256) * 100;
            if (diffRGBnet < diffThreshold)
                for k = 1:3
```

```
                    subimg(i, j, k) = 0;
                end
            else
                for k = 1:3
                    subimg(i, j, k) = imgobj(i, j, k);
                end
            end
        end
    end

end
```

shadeCover.m

```
function [cvr] = shadeCover(range, img)
    sz = size(img);
    pixelCount = sz(1) * sz(2);
    [count, x] = imhist(img);
    cvr = trapz(range, count(range)) / pixelCount;
end
```

getImgCorners.m

```
function [i1, i2, i3, i4, i5, i6, i7, i8] = getImgCorners(img)
    sz = size(img);
    h = sz(1);
    w = sz(2);
    div = h/6;
    i1 = img(1:div, 1: div);
    i2 = img(1:div, (w-div):w);
    i3 = img((h-div):h, 1:div);
    i4 = img((h-div):h, (w-div):w);

    midw = w/2;
    midh = h/2;
    hdiv = div/2;

    i5 = img(1:div, (midw-hdiv):(midw+hdiv));
    i6 = img((midh - hdiv):(midh + hdiv), 1:div);
    i7 = img((midh - hdiv):(midh + hdiv), (w-div):w);
    i8 = img((h-div):h, (midw-hdiv):(midw+hdiv));
end
```

bgrv2.m

```
function [imrev] = bgrv2(imgobj, imgbg)
%     imrev = imgobj - imgbg;
    imrev = subimgs(imgobj, imgbg);
```

```
end
```

bgrv.m

```
function [imgrev] = bgrv(img)
    for k = 1:2
        hsv = rgb2hsv(img);
        s = hsv(:, :, 2);
        fg1 = s > 0.25;
        fg2 = bwareaopen(fg1, 20000);

        for i = 1:3
            fg3(:, :, i) = fg2;
        end

        fg3 = im2uint8(fg3);
        imgrev = img .* fg3;
    end
end
```

Neural Network Processing

genImgDataset.m

```
function [imgdataset] = genImgDataset(imglist)
    imgsz = size(imglist{1});
    imgr = imgsz(1);
    imgc = imgsz(2);
    len = imgr * imgc * 3;

    imgcnt = size(imglist);
    imgcnt = imgcnt(2); % No. of images

    imgdataset = zeros(imgcnt, len);

    for i = 1:imgcnt
        imgdataset(i, 1:len) = reshape(imglist{i}, [1 len]);
    end
end
```

testnet1.m

```
% Solve an Input-Output Fitting problem with a Neural Network
```

```
%
% This script assumes these variables are defined:
%
%   x - input data.
%   y - target data.

inputs = x;
targets = lin;

% Create a Fitting Network
hiddenLayerSize = [100];
net = fitnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};



% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 100/100;
net.divideParam.valRatio = 0/100;
net.divideParam.testRatio = 0/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm';  % Levenberg-Marquardt

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
% net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
%  'plotregression', 'plotfit'};



% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
%outputs = net(inputs);
%errors = gsubtract(targets,outputs);
%performance = perform(net,targets,outputs)
```

```
% Recalculate Training, Validation and Test Performance
% trainTargets = targets .* tr.trainMask{1};
% valTargets = targets  .* tr.valMask{1};
% testTargets = targets  .* tr.testMask{1};
% trainPerformance = perform(net,trainTargets,outputs)
% valPerformance = perform(net,valTargets,outputs)
% testPerformance = perform(net,testTargets,outputs)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotfit(net,inputs,targets)
%figure, plotregression(targets,outputs)
%figure, ploterrhist(errors)
```

caldiff.m

```
function [opvar, netvals, flcood] = caldiff(l, h, inp, net)
%l = 1;
%h = 20;
flcood = inp(l : h, :);
sz = h - l + 1;
opvar = zeros(sz, 1);
netvals = zeros(sz, 2);
for i = l : h
    tmp = net(inp(i, :)')';
    netvals(i, :) = tmp;
    opvar(i) = tmp(2) - tmp(1);
end
%end
```

generranttbl.m

```
function [tbl] = generrantbl(l, h, inp, net, dbop)
[lol, lol2, fcood] = caldiff(l, h, inp, net);
i = l : h;
err = dbop(l : h, 1);
tbl(:, 1) = i';
tbl(:, 2) = lol;
tbl(:, 3) = err;
tbl(:, 4 : 5) = lol2;
tbl(:, 6 : 7) = fcood;
end
```

Development Softwares Required

- VISUAL STUDIO 2012

  Visual Studio Software is a comprehensive software development IDE from Microsoft which allows for software development for any windows based platform in various languages.
  Since this application is .NET based, hence this IDE was utilised.

- MATLAB

  This software is used to write the scripts handled the various computationally intensive tasks of background removal from images and the implementation of Neural Networks. MATLAB greatly eases the work of a developer especially if it involves hard to implement computationally intensive algorithms.

CONCLUSION

The objective was to develop a software solution to replace the bar-code scanning mechanism commonly employed at departmental stores.

Judging by the different functionalities and utilities of the software as discussed above, it is safe to say that all the requirements have been fulfilled.

Furthermore, the software maybe enhanced to incorporate automatic invoice generation and billing.

Also, a cross-platform solution may also be developed with not much effort in the Java language due to similarities in the semantic and syntactical structure of C# and Java.

BIBLIOGRAPHY

- www.stackoverflow.com
- www.codeproject.com
- www.dreamincode.com
- www.msdn.microsoft.com
- www.google.com
- www.wikipedia.org
- www.mathworks.in

EXAMINER'S REMARKS