Faculty of Engineering

Computer Engineering Department

Image processing, object detection and object classification

A Graduation Project Report

Presented in Partial Fulfillment of the Requirements forGEN001

Presented by

Name	Section	BN	Grade
Abdelrahman Mohsen Abdellatif Abdelghany	16	28	
Abdelrahman Mohamed Hossni Elsayed	16	33	
Abdelrahman yousri fekry Mohamed	16	61	
Abdelrahman adel Mohamed	16	14	
Abdelrahman soliman Mohamed	16	3	
Abdelrahman medhat ahmed abdelghany	16	50	
Abdelrahman Mohamed kelany Mohamed	16	42	

In partial fulfillment of the Requirements for

GEN001 Practical and Engineering Applications (Physics part)

December2018

Contents

Abstract

This paper presents an in-depth experimental study on object classification and object detection. Feature-classifier combinations are examined with respect to their performance and efficiency. Regarding the classifiers experiment is performed on a large set of data consisting of approximant. 800 motorbikes and 800 airplanes. Meaningful results are obtained by analyzing performance variances caused by varying training and test sets. In object detection features are extracted from the object then compared with features extracted from an image which contains multiple objects. In order to get the best results the data set must be prepared first by applying image processing techniques to get stronger features.

Acknowledgement

We would like to sincerely thank Eng. Amany el Zawahry for her constant help and support.

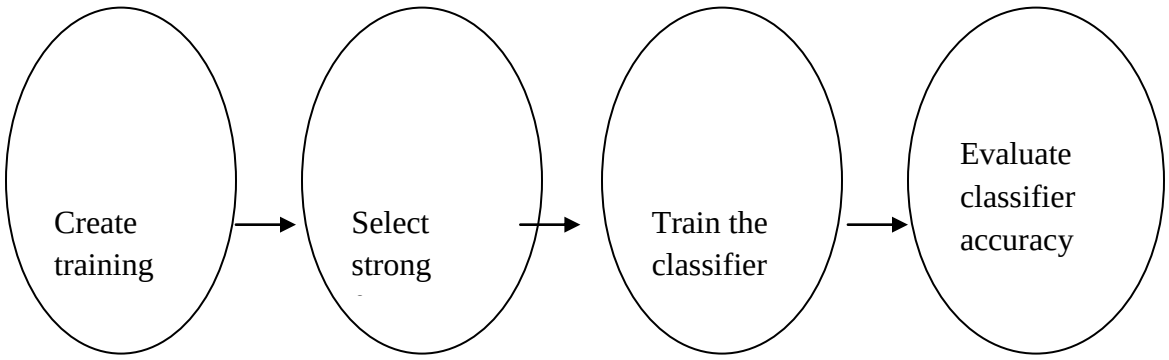
Chapter 1: Introduction

1.1. Motivation

This project was chosen for the following reasons; firstly, image processing plays an increasingly important role in a wide variety of disciplines and fields in both science and technology. For instance, the medical field uses image processing in biomedical pictures. The second reason is that this topic was previously discussed during math lab course and the proposed project intends to apply the discussed topics. The third reason is that machine learning solves a lot of our daily life problems and this small project intends to open our minds to this science.

1.2. Problem Definition

A classifier is an algorithm that takes a set of features that characterize objects and uses them to determine the class of each object. There are of two types of classification supervised and unsupervised. In supervised classification, meaning that an expert has determined into what classes an object may be categorized and also has provided a set of sample objects with known classes. This set of known objects is called the training set because it is used by the classification programs to learn how to classify objects. In unsupervised classification methods in which the induction engine works directly from the data, and there is neither training sets nor pre-determined classes. There are four steps to develop classifiers:



1.3. Summary of Approach

In order to solve this problem the data set was prepared using image processing techniques then features were extracted then different classifiers were trained and evaluated by a testing set.

1.4. Report Overview

This project aims to solve the problem of object recognition first of all we need to handle a large set of data then we apply image processing techniques then we detect this object by comparing its features with extracted features from our set this is how we managed to search for an object in an image after detecting an object we want to classify this object so we trained a classifier with features from different types of objects and the classifier tells us what this object type is.

Chapter 2: Literature Survey

Object detection is when the computer uses the available date and algorithms to detect and classify multiple objects in a matter of milliseconds in images and videos. Object detection, as an asset of artificial intelligence, rivals the human eye in detection objects. Most object detection systems applies the same basic scheme which is known as a "sliding Window".

During the last years, there has been a rapid and successful expansion on computer vision research. Parts of this success have come from adopting and adapting machine learning methods, while others from the development of new representations and models for specific computer vision problems or from the development of efficient solutions. One area that has attained great progress is object detection. The present works gives a perspective on object detection research. Given a set of object classes, object detection consists in determining the location and scale of all object instances, if any, that are present in an image. Thus, the objective of an object detector is to find all object instances of one or more given object classes regardless of scale, location, pose, view with respect to the camera, partial occlusions, and illumination conditions.

In many computer vision systems, object detection is the first task being performed as it allows to obtain further information regarding the detected object and about the scene. Object detection has been used in many applications, with the most popular ones being: (i) human-computer interaction (HCI), (ii) robotics (e.g., service robots), (iii) consumer electronics (e.g., smart-phones), (iv) security (e.g., recognition, tracking), (v) retrieval (e.g., search engines, photo management), and (vi) transportation (e.g., autonomous and assisted driving).

Each of these applications has different requirements, including: processing time (off-line, on-line, or real-time),robustness to occlusions, invariance to rotations (e.g., in-plane rotations) and detection under pose changes.

While many applications consider the detection of a single object class (e.g., faces) and from a single view (e.g., frontal faces), others require the detection of multiple object classes (humans, vehicles, etc.), or of a single class from multiple views (e.g., side and frontal view of vehicles). In general, most systems can detect only a single object class from a restricted set of views and poses.

There are also other applications like

-Security and Surveillance for detecting faces and anomalies such as bombs and explosives.

-Vehicle detection in trafficking.

-Social media and face detection

-People counting for statistics as in analysing stores performance and crowd statistics.

-Industrial process where a machine is programmed to detect certain objects.

-Online images and parental control where with a few clicks a parent can filter what photos are available to the kids.

Current research problems

Multi-view; many applications require detecting more than one object class. If a Large Number of Classes Is Being Detected, the processing Speed becomes important issue, as well as the kind of classes that the system can handle without accuracy loss.

Multi-resolution, multi-pose; Most methods used in practice have been designed detect a

Single object class under a single view, thus these methods cannot Handle Multiple Views, or large pose variations.

Deformable and interlaced objects

Some of the challenge is the appearance of new classes and classification which need to be updated regularly and some new classes need to be added to the available data.

One other challenge is whether to detect the whole object first of the parts as these two complement each other.

Depth and thermal camera are not enough for detecting an object as resolution and technology are always in progress.

3D pixel level detection

OPEN PROBLEMS AND FUTURE DIRECTIONS

Open-World Learning and Active Vision;

An important problem is to incrementally learn, to detect new classes, or to incrementally learn to distinguish among subclasses after the “main” class has been learned. If this can be done in an unsupervised way, we will be able to build new classifiers based on existing ones, without much additional effort, greatly reducing the effort required to learn new object classes.

Note that humans are continuously inventing new objects, fashion changes, etc., and therefore detection systems will need to be continuously updated, adding new classes, or updating existing ones.

There are others like -Object-Part Relation

-Multi-Modal Detection

-Pixel-Level Detection (Segmentation) and Background Objects

Chapter 3: Necessary Background

Image processing

The history of image processing

The field of image processing is continually evolving during the past few years, till it reached the level of interest of image morphology, neural works, full colour image processing, image recognition and knowledge based on image analysis systems

What is image processing?

Digital image processing seems very difficult for many people but there are only few principles you need to know to use….graphics app. Modern technology had made it possible to manipulate multi-dimensional signals. For an example image we want to scan not properly scanned because of a problem at this time image processing plays a vital role in treatment of image solving problem

Why we need digital image processing?

Image is better than any other information. For the human being to perceive, vision allows humans to understand the world surrounding us. Some operation on image to get enthused image or to extract useful information. It's a type of signal processing in which input is an image and output maybe image or features related to image. Image processing basically include three steps: importing the image via image accusation tools analysing and manipulating image.

Image processing techniques

- Edge detection

  Edge detection is a common image processing technique and can be used for a variety of applications such as image segmentation and object detection. Edge detection can be a great preprocessing step for image segmentation. Edge detection can be used effectively by using the edge function.

Common edge detection algorithms

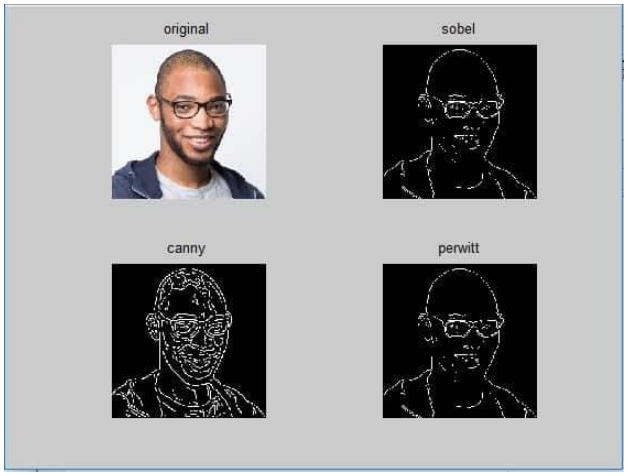
1) Sobel
   The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
2) Canny
   The Canny method finds edges by looking for local maxima of gradient of I. The gradient is calculated using the derivative Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be "fooled" by noise, and more likely to detect true weak edges.

Steps

1) Read the image
2) Convert to gray scale
3) Apply edge function

Used functions

1) imread
2) rgb2gray
3) edge



Morphological operations

Morphology is a broad set of image processing operations that process images based on shapes. In a morphological operation, each pixel in the image is adjusted based on the value of other pixels in its neighbourhood. By choosing the size and shape of the neighbourhood, you can construct a morphological operation that is sensitive to specific shapes in the input image.

The most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. Dilation and erosion are often used in combination for specific image pre-processing applications, such as filling holes or removing small objects.

The objective of using morphological operations is to remove the imperfections in the structure of image. Most of the operations used here are combination of two processes, dilation and erosion. The operation uses a small matrix structure called as structuring element. The shape and size of the structuring element has significant impact on the final result

Morphological operations are useful in many applications. To List a few they are used in hole filling, boundary extraction of Objects, extraction of connected components, Thinning and Thickening and so on.

Steps

1) Read image
2) Convert to grey scale
3) Choose a proper strel
4) Apply dilation, erosion, open and close

Used functions

1) Imread
2) Strel
3) Imdiate
4) Imerode
5) Imopen
6) Imclose

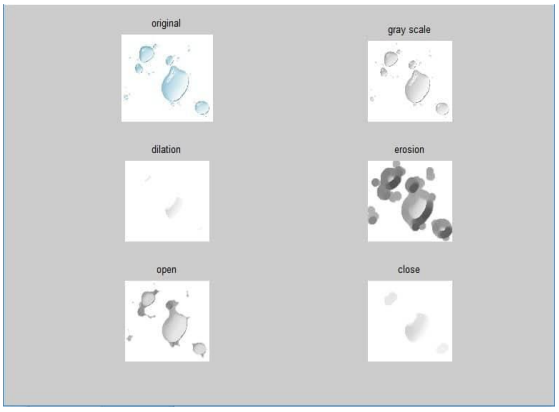
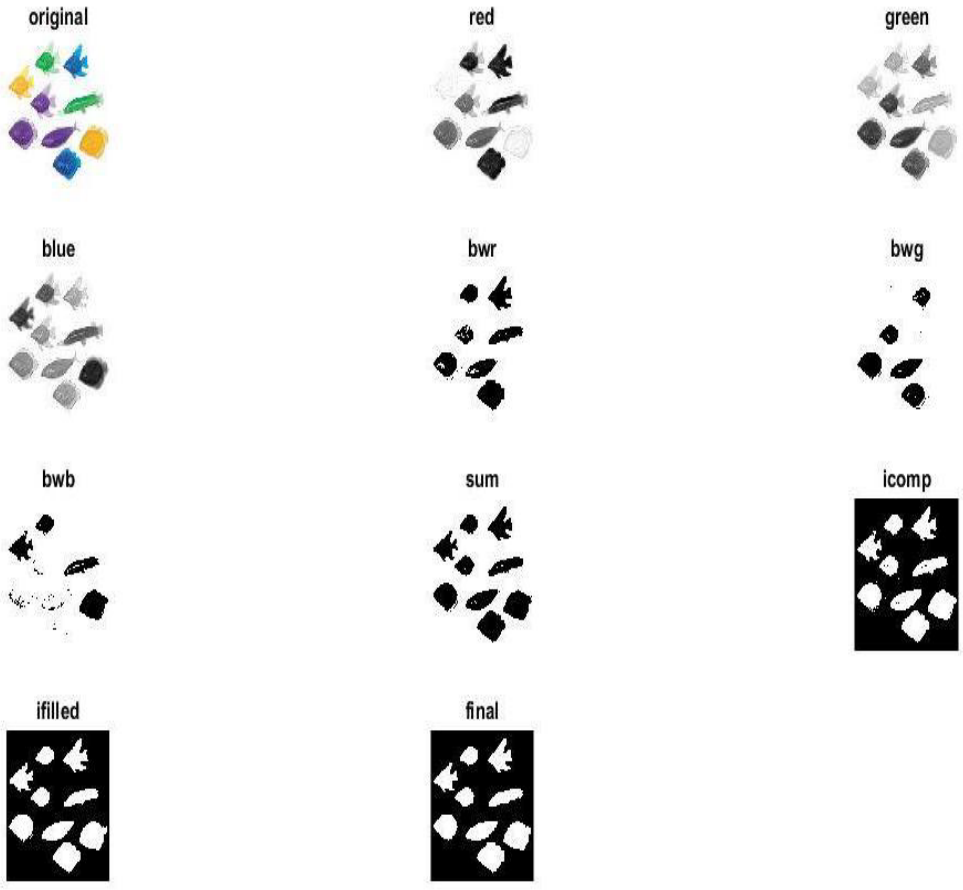


Image segmentation

Image segmentation is the process of dividing an image into multiple parts this is typically used to identify objects and other relevant information within an image.

Steps

1) Read the image.
2) Get the RGB matrices.
3) Convert the RGB matrices to binary.
4) Get the sum of the RGB binary matrices.
5) Get the complement matrix.
6) Fill the holes -empty spots-.
7) Smooth the edges.
8) Show the image.



Used functions

1) imread (‘name.jpg’)
2) im2bw(I)
3) imcomplement(I)
4) imfill(I,’holes’)
5) strel(‘disk’ r)
6) imopen(I,se)
7) imshow(I)

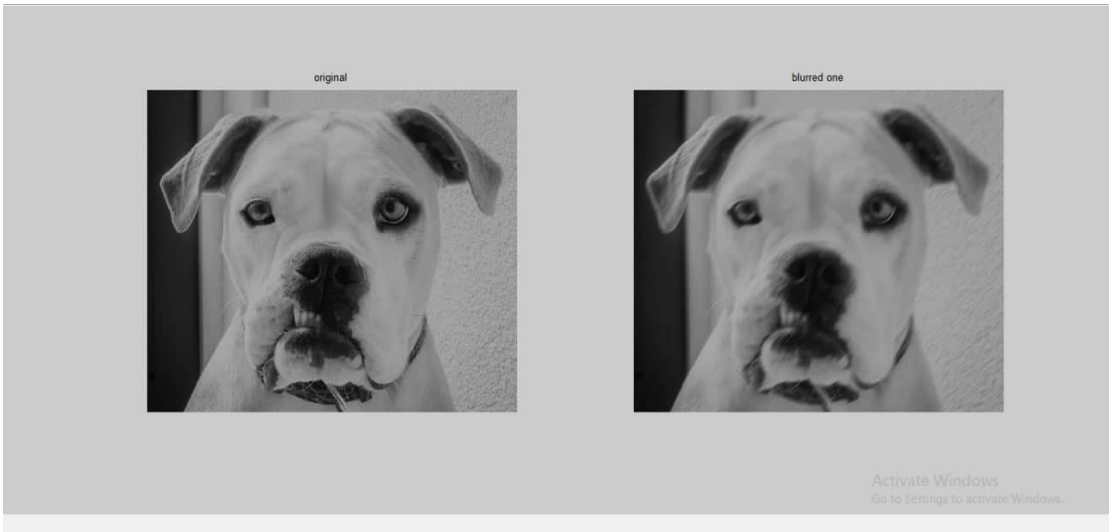
-

Blurring

It's that thing that happens when your camera is out of focus or the dog steals your glasses. What happens is that what should be seen as a sharp point gets smeared out, usually into a disc shape. In image terms this means that each pixel in the source image gets spread over and mixed into surrounding pixels. Another way to look at this is that each pixel in the destination image is made up out of a mixture of surrounding pixels from the source image. The operation we need for this is called convolution. The way it works is this: we imagine sliding a rectangular Array of numbers over our image. This array is called the convolution kernel. For every pixel in the image, we take the corresponding numbers from the kernel and the pixels they are over, multiply them together and add all the results together to make the new pixel. For example, imagine we want to do a really simple blur where we just average together each pixel and its eight immediate neighbors.

In a sample way, an image looks sharper or more detailed, so we simple reduce The edge content and makes the transition from one color to the other very smooth. You might have seen a blurred image when you zoom an image. When you zoom an Image using pixel replication, and zooming factor is increased, you saw a blurred image. This image also has fewer details, but it is not true blurring. Because in zooming, you add new pixels to an image that increase the overall number of pixels in an image, whereas in blurring, the number of pixels of a normal image and a blurred image remains the same.

For example of blurred image



The best blurring I see is Threshold Blurs Something which is often wanted is a blur which blur spots of the image which are very similar but preserves sharp edges. This is digital wrinkle cream and you can see this in any movie poster ever printed - the stars' faces have all those nasty blemishes ironed out without the image appearing blurry. Often this is so overdone that the actors look like waxworks or computer-generated figures

Steps

1) Read image

2) Put title to the original one

3) Adjust the THETA and LEN

4) Use function fspecial

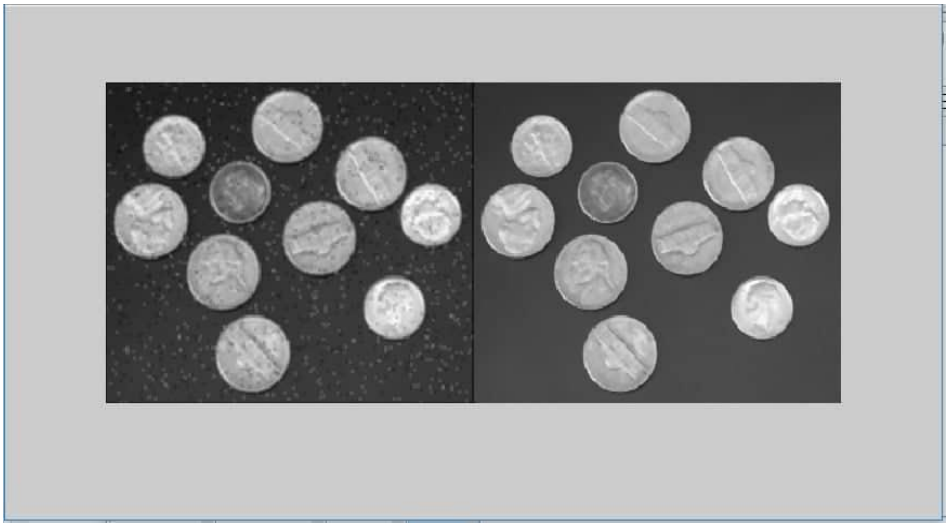
5) Use function imfilter

6)show the two images

Noise reduction

first of all why the image loses some parts? The answer is in common wireless sensors the image is transmitted over the wireless channel block by block, it divides the image into blocks each one 8*8 pixels so due to sever fading we may lose an entire block or even several blocks of an image the average packet lose rate in wireless environment is 3.6% and occurs in a bursty fashion.

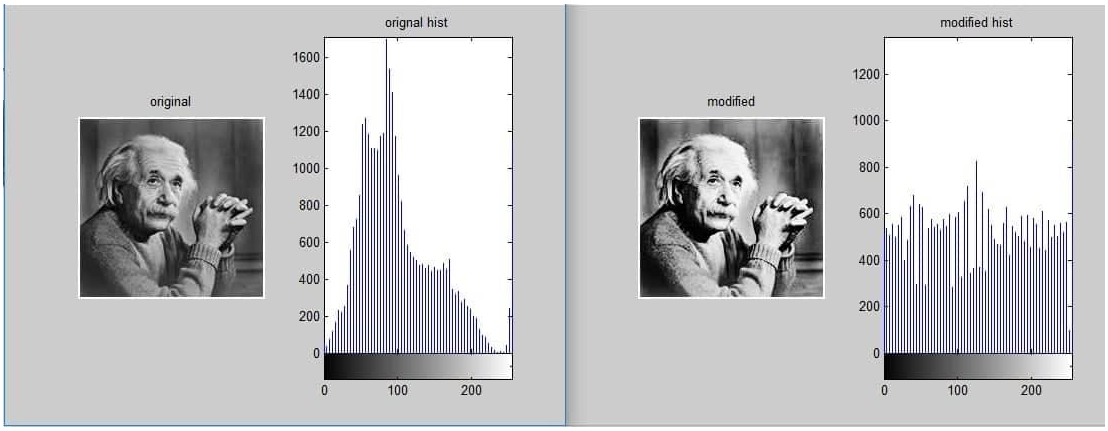
As a solution for missing parts problem there is a common technique that it's possible to satisfactorily reconstruct the lost block by using the available information surrounding them, this will result an increase in bandwidth efficiency of the transmission. The basic idea is to first automatically classify the block as textured or structured (containing edges) and then fill-in the missing block with information propagated from the surrounding pixels



Histogram equalization

Histogram equalization is used to enhance contrast. It is not necessary that contrast will always be increase in this. There may be some cases were histogram equalization can be worse. In those cases the contrast is decreased.

Let's start histogram equalization by taking this image below as a simple image.



MATLAB:

To test the accompanying code, hist_eq.m, type

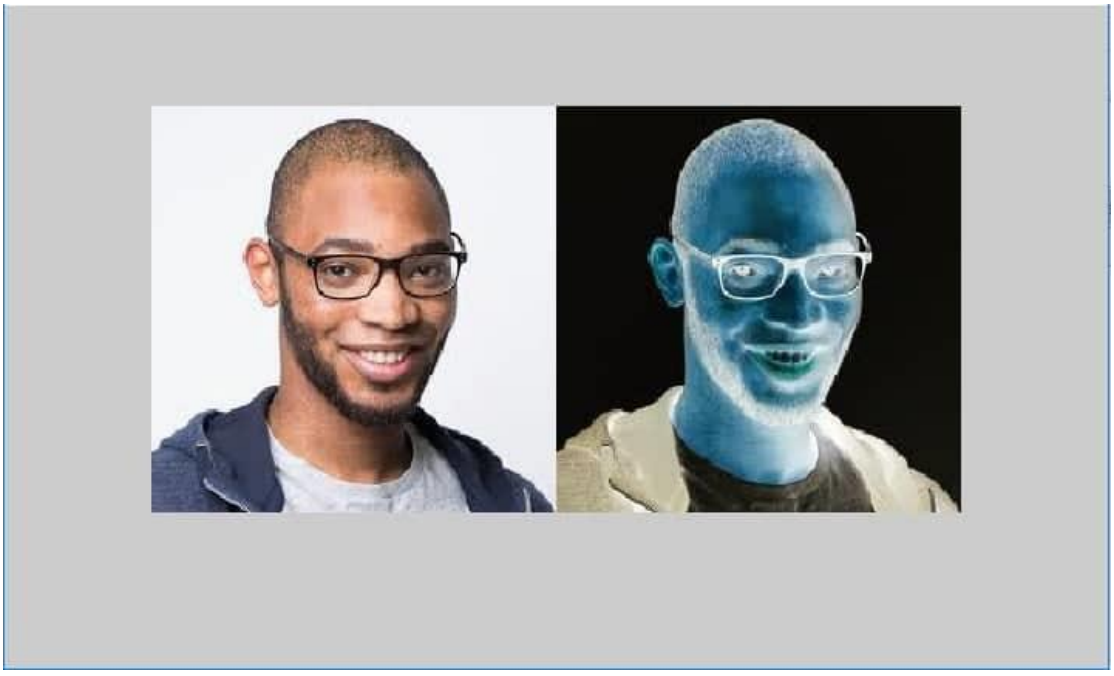
g=hist_eq('elvis_low_contrast.bmp');

Negative image

In photography, a negative is an image, usually on a strip or sheet of transparent plastic film, in which the lightest areas of the photographed subject appear darkest and the darkest areas appear lightest. This reversed order occurs because the extremely light-sensitive chemicals a camera film must use to capture an image quickly enough for ordinary picture-taking are darkened, rather than bleached, by exposure to light and subsequent photographic processing.

In the case of color negatives, the colors are also reversed into their respective complementary colors. In the complement of a binary image, zeros become ones and ones become zeros. Black and white is reversed.

In the complement of a gray scale or color image, each pixel value is subtracted from the maximum pixel value supported by the class (or 1.0 for double-precision images). The difference is used as the pixel value in the output image. In the output image, dark areas become lighter and light areas become darker. For color images, reds become cyan, greens become magneta, blues become yellow, and vice versa



Brightness and contrast

An image must have the proper brightness and contrast for easy viewing. Brightness refers to the overall lightness or darkness of the image. Contrast is the difference in brightness between objects or regions. For example, a white rabbit running across a snowy field has poor contrast, while a black dog against the same white background has good contrast. Figure 23-10 shows four possible ways that brightness and contrast can be misadjusted. When the brightness is too high, as in (a), the whitest pixels are saturated, destroying the detail in these areas. The reverse is shown in (b), where the brightness is set too low, saturating the blackest pixels. Figure (c) shows the contrast set to high, resulting in the blacks being too black, and the whites being too white. Lastly, (d) has the contrast set too low; all of the pixels are a mid-shade of gray making the objects fade into each other. Figures 23-11 and 23-12 illustrate brightness and contrast in more detail. A test image is displayed in Fig. 23-12, using six different brightness and contrast levels. Figure 23-11 shows the construction of the test image, an array of 80:32 pixels, with each pixel having a value between 0 and 255. The background of the test image is filled with random noise, uniformly distributed between 0 and 255. The three square boxes have pixel values of 75, 150 and 225, from left-to-right. Each square contains two triangles with pixel values only slightly different from their surroundings. In other words, there is a dark region in the image with faint detail, this is a medium region in the image with faint detail, and there is a bright region in the image with faint detail. Figure 23-12 shows how adjustment of the contrast and brightness allows different features in the image to be visualized. In (a), the brightness and contrast are set at the normal level, as indicated by the B and C slide bars at the left side of the image. Now turn your attention to the graph shown with each image, called an output transform, an output look-up table, or a gamma curve. This controls the hardware that displays the image. The value of each pixel in the stored image, a number between 0 and 255, is passed through this look-up table to produces another number between 0 and 255. This new digital number drives the video intensity circuit, with 0 through 255 being transformed into black through white, respectively. That is, the look-up table maps the stored numbers into the displayed brightness. Figure (a) shows how the image appears when the output transform is set to do nothing, i.e., the digital output is identical to the digital input. Each pixel in the noisy background is a random shade of gray, equally distributed between black and white. The three boxes are displayed as dark, medium and light, clearly distinct from each other. The problem is, the triangles inside each square cannot be easily seen; the contrast is too low for the eye to distinguished these regions from their surroundings. Figures (b) & (c) show the effect of changing the brightness. Increasing the brightness shifts the output transform to the left, while decreasing the brightness shifts it to the right. Increasing the brightness makes every pixel in the image appear lighter. Conversely, decreasing the brightness makes every pixel in the image appear darker. These changes can improve the view

ability of excessively dark or light areas in the image, but will saturate the image if taken too far. For example, all of the pixels in the far right square in (b) are displayed with full intensity, i.e., 255. The opposite effect is shown in (c), where all of the pixels in the far left square are displayed as blackest black or digital number 0. Since all the pixels in these regions have the same value, the triangles are completely wiped out. Also notice that none of the triangles in (b) and (c) are easier to see than in (a). Changing the brightness provides little (if any) help in distinguishing low contrast objects from their surroundings. Figure (d) shows the display optimized to view pixel values around digital number 75. This is done by turning up the contrast, resulting in the output transform increasing in slope. For example, the stored pixel values of 71 and 75 become 100 and 116 in the display, making the contrast a factor of four greater. Pixel values between 46 and 109 are displayed as the blackest black, to the whitest white. The price for this increased contrast is that pixel values 0 to 45 are saturated at black, and pixel values 110 to 255 are saturated at white. As shown in (d), the increased contrast allows the triangles in the left square to be seen, at the cost of saturating the middle and right squares. Figure (e) shows the effect of increasing the contrast even further, resulting in only 16 of the possible 256 stored levels being displayed as none saturated. The brightness has also been decreased so that the 16 usable levels are centered on digital number 175. The details

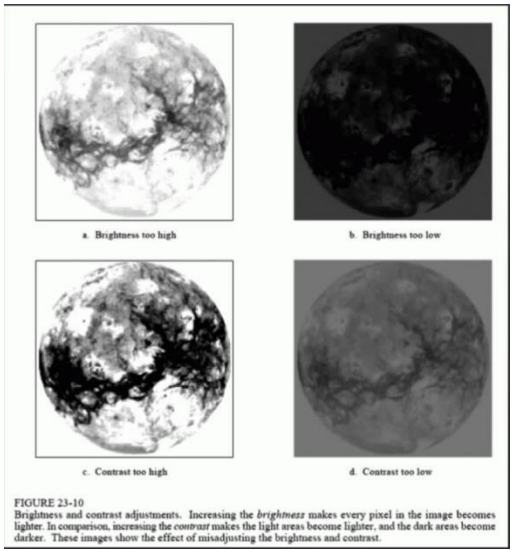


FIGURE 23-10
Brightness and contrast adjustments. Increasing the *brightness* makes every pixel in the image becomes lighter. In comparison, increasing the *contrast* makes the light areas become lighter, and the dark areas become darker. These images show the effect of misadjusting the brightness and contrast.

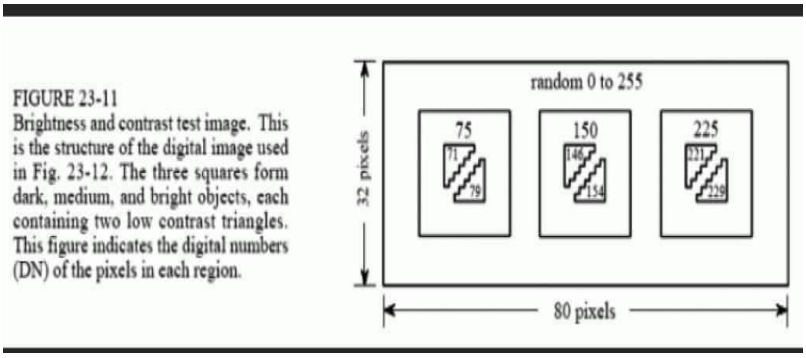


FIGURE 23-11
Brightness and contrast test image. This is the structure of the digital image used in Fig. 23-12. The three squares form dark, medium, and bright objects, each containing two low contrast triangles. This figure indicates the digital numbers (DN) of the pixels in each region.

Machine learning

Machine learning is the science of getting computers to act without being explicitly programmed.

Object classification

Classification is the categorization of the object based on a previously defined classes or types. Object classification could take arbitrary boundaries based on the problem domain and independent of detection. Object classification is an important task in many computer vision applications, including surveillance, automotive safety, and image retrieval. For example, in an automotive safety application, you may need to classify nearby objects as pedestrians or vehicles.

Object detection

Effective object detection must be able to handle cluttered scenes: changes to the object size, location, orientation, and other challenges

Machine learning work flow

1) Train : iterate till you find the best model
2) Predict: integrate trained models into apps

Challenges

1) Handling large sets of images
2) How to extract discriminative information from images
3) How to model data using machine learning

Feature extraction has two steps

1-Feature detection

. Where to extract features

. Not all locations are good for features extraction

2-Feature extraction

.orientation of extraction window

. What to encode in the feature

Application of feature matching

-image alignment and stitching

-3d structure from motion

- Motion tracking

-robot navigation

-object and face recognition

-Human action recognition

Feature detectors - classic and state of art

features	Detection	Extraction	Opencv	published
SIFT	Yes	Yes	Yes	IJCV 2004
SURF	Yes	Yes	Yes	CVIU 2008
BRISK	Yes	Yes	Yes	ICCV 2011
FREAK	Yes	Yes	Yes	CVPR 2012
Harris	Yes	No	Yes	1988

Examples of feature extraction

- Haar features

-HOG: histogram of oriented gradients

- lbp : local binary patterns

-harris corners

-Corner detection: basic idea

Feature extraction technique

Bag of visual words

Bag of words model in computer vision in computer vision [BOW] used in image classification; this by treating image features as some words. In document classification, bag of words is a spars vector of occurrence counts of words. (1) IMAGE REPRESENTATION BASED BOW MODEL; to represent image using bag of words image can be treated as a document. (Words) in image needed to be defined too. To achieve this it usually includes following steps. (1)FEATURE DETECTION (2) FEATURE DESCRIPTION (3) CODE BOOK GENERATION definition of bow mode can be histogram representation based on independent features and after feature detection image is abstracted by several local patches. Features representation deal with how to represent patches as numerical vectors .these features are called by feature descriptors. Good descriptors should have the ability to handle the intensity, rotation, scale, and affine variations to some extents (scale invariant feature transform) [SIFT] converts each patch to 128 dimensional vectors. CODE BOOK GENERATION; final step of bow to convert these patches to code words

Classifiers

- SVM
- Decision trees
- Ada boost
- Bagged trees
- KNN
- Discriminant analysis
- Bayes classifiers

KNN

Making Predictions with KNN KNN makes predictions using the training dataset directly. Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances. For regression this might be the mean output variable, in classification this might be the mode (or most common) class value. To determine which of the K instances in the training data-set are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidean distance. Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (xi) across all input attributes j. Euclidean Distance(x, xi) = sqrt(sum((xj – xij)^2)) Other popular distance measures include: Hamming Distance: Calculate the distance between binary vectors (more). Manhattan Distance: Calculate the distance between real vectors using the sum of their absolute difference. Also called City Block Distance (more). Minkowski Distance: Generalization of Euclidean and Manhattan distance (more). There are many other distance measures that can be used, such as Tanimoto, Jaccard, Mahalanobis and cosine distance. You can choose the best distance metric based on the properties of your data. If you are unsure, you can experiment with different distance metrics and different values of K together and see which mix results in the most accurate models. Euclidean is a good distance measure to use if the input variables are similar in type (e.g. all measured widths and heights). Manhattan distance is a good measure to use if the input variables are not similar in type (such as age, gender, height, etc.). The value for K can be found by algorithm tuning. It is a good idea to try many different values for K (e.g. values from 1 to 21) and see what works best for your problem. The computational complexity of KNN increases with the size of the training dataset. For very large training sets, KNN can be made stochastic by taking a sample from the training dataset from which to calculate the K-most similar instances. KNN has been around for a long time and has been very well studied. As such, different disciplines have different names for it, for example: Instance-Based Learning: The raw training instances are used to make predictions. As such KNN is often referred to as instance-based learning or a case-based learning (where each training instance is a case from the problem domain). Lazy Learning: No learning of the model is required and all of the work happens at the time a prediction is requested. As such, KNN is often referred to as a lazy learning algorithm. Non-Parametric: KNN makes no assumptions about the functional form of the problem being solved. As such KNN is referred to as a non-parametric machine learning algorithm. KNN can be used for regression and classification problems. KNN for Regression When KNN is used for regression problems the prediction is based on the mean or the median of the K-most similar instances. KNN for Classification When KNN is used for classification; the output can be calculated as the class with the highest frequency from the K-most similar instances. Each instance in essence votes for their class and the class with the most votes

is taken as the prediction. Class probabilities can be calculated as the normalized frequency of samples that belong to each class in the set of K most similar instances for a new data instance. For example, in a binary classification problem (class is 0 or 1): p (class=0) = count (class=0) / (count (class=0) +count (class=1)) If you are using K and you have an even number of classes (e.g. 2) it is a good idea to choose a K value with an odd number to avoid a tie. And the inverse, use an even number for K when you have an odd number of classes. Ties can be broken consistently by expanding K by 1 and looking at the class of the next most similar instance in the training dataset.

Naive bayes

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it's known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods. Bayes theorem provides a way of calculating posterior probability P (c|x) from P(c), P(x) andP (x|c).

Some applications on naive bayes

● **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure

Fast. Thus, it could be used for making predictions in real time.

● **Multi class Prediction:** This algorithm is also well known for multi class

Prediction feature. Here we can predict the probability of multiple classes of

Target variable.

● **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes Classifiers mostly used in text classification (due to better result in multi class Problems and independence rule) have higher success rate as compared to Other algorithms. As a result, it is widely used in Spam filtering (identify spam E-mail) and Sentiment Analysis (in social media analysis, to identify positive and Negative customer sentiments)

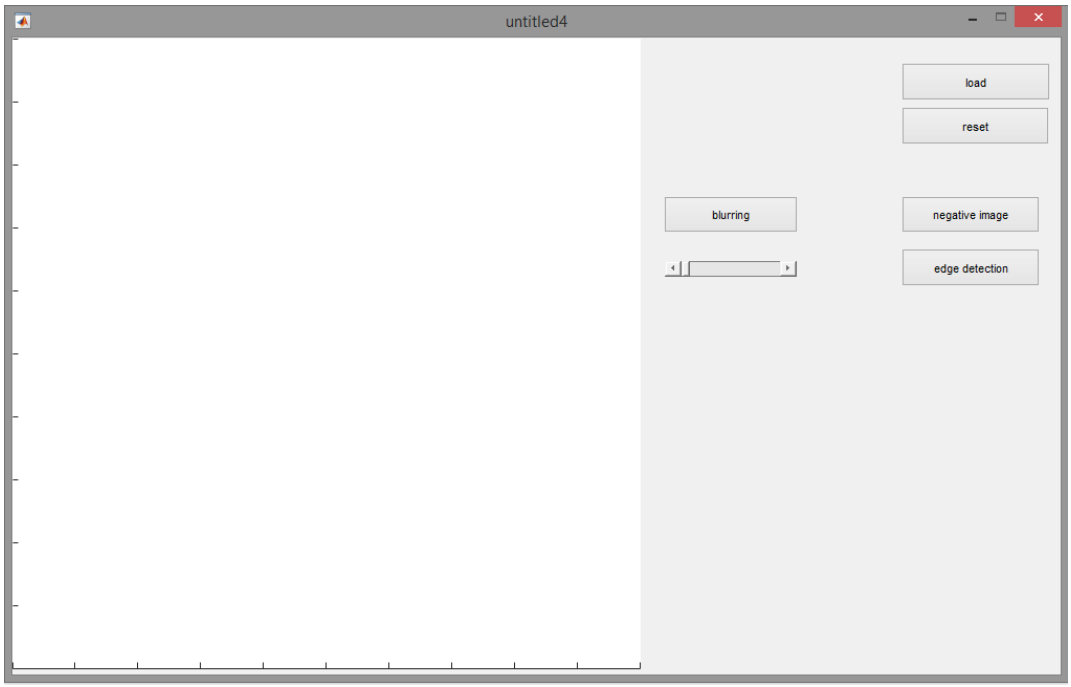
● **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering Together builds a Recommendation System that uses machine learning and data Mining techniques to filter unseen information and predict whether a user would Like a given resource or not

Note: many permutations and combinations to fit the needs of your problem.

Chapter 4: System Description

Image processing GUI

[ESSENTIAL CHARACHTERISTICS]
(1)GRAPHICAL COMPONENTS [push buttons , edit boxes ,sliders ,labels ,menus , etc.....]
(2)STATIC COMPONENTS [frames , text strings , ...]
(3)BOTH ARE CREATED USING THE FUNCTION UICONTROL [GUIDE - BUILDING USER INTERFACES INTERACTIVELY] (GUIDE) is an interactive tool for designing and building graphical user interfaces (GUI) for the matlab applications. - GUI building process involves;
(a) Designing of the user interface and layout (The looks?)
(b) Programming of the (GUI) and its components (The works?)
(c) Testing , debugging and finallay running it. [Panels and Button Groups];
The items labelled panels and button groups in the components pallets are not in themselves control elements but usually used for grouping together of control elements. - This is desirable when you design GUI as made up of plots and graphs [BUTTON GROUPS]; -there are 2 groups of buttons (Radio Buttons) and (Toggle Buttons)



Object detection

The model presents an algorithm for detecting a specific object based on finding point correspondences between the reference and the target image. It can detect objects despite a scale change or plane rotation. This method of object detection works best for objects that exhibit non-repeating texture patterns, which give rise to unique feature matches

Step 1: Reading the Images

Suppose you have an image of a cluttered scene named (clutteredDesk.jpg), and you want to detect a particular object of which you have a separate image, stapleRemover.jpg. You can start by reading the reference image containing the object of interest into MATLAB®.

Next, you can read the target image containing a cluttered scene

Step 2: Detecting Feature Points

You can detect feature points in both images

Then, you can visualize the strongest feature points found in the reference image.

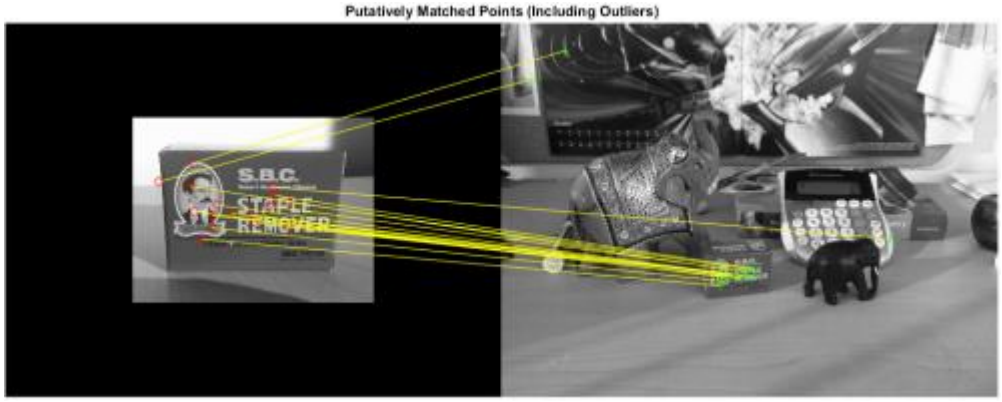
For comparison, visualize the strongest feature points found in the target image

Step 3: Extracting Feature Descriptors

You can extract feature descriptors at the interest points in both images.

Step 4: Finding Putative Point Matches

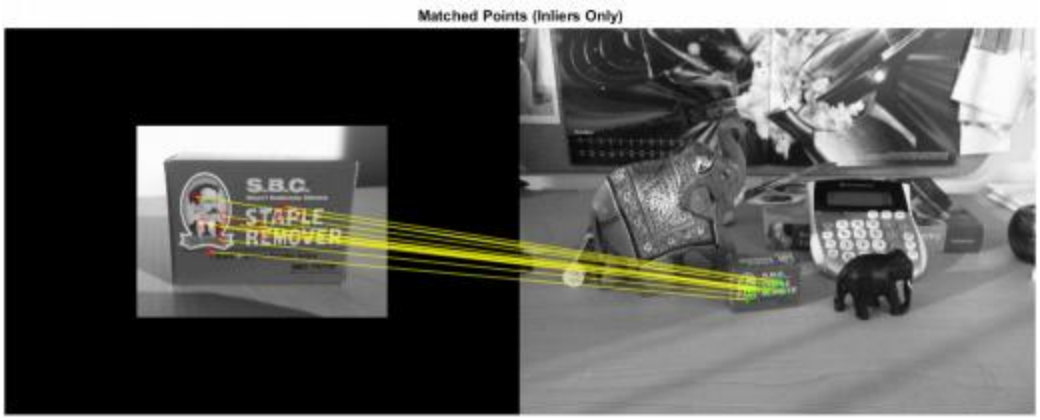
Then match the features using their descriptors.



Step 5: Locating the Object in the Scene Using Putative Matches

The estimateGeometricTransform function calculates the transformation relating the matched points, while eliminating outliers. This transformation allows you to localize the object in the scene

You can then display the matching point pairs with the outliers removed.



Next, get the bounding polygon of the reference image

To indicate the location of the object in the scene, you can transform the polygon into the coordinate system of the target image.

Then, you can display the detected object

Object classification

The model presents an algorithm for classifying objects based on extracting features from training set then we work on a classifier that deals with this feature to learn how to classify new data.

Step 1

In this step you read your image data set and store it in an image set format.

Step 2

In this step you extract features from your training set using any method you choose like ‘bag of visual words’ and this is what we chose for our model. Then you encode these features

Step 3

In this step you open classification learner app which you can train a classifier in. this app allows you to know how good is your model.

Step 4

In this step you finally test your classifier with a new data called testing set.

Chapter 5: Results

We managed to prepare a set of images using Image processing techniques (histogram equalization, contrast, morphological operations and some effects) in order to get strong features then using object detection techniques we managed to detect an object in a cluttered scene and we also managed to classify objects using classification techniques.

Chapter 6: Conclusion and Future Work

Conclusion

To train a classifier well you need a large data set but not any data set you need to pre-process this set with image processing techniques then you will get Satisfying results.

Future work

Object tracking in vehicles to avoid any accidents.

References

www.mathworks.com

R. C. Gonzalez and R. E. Woods, Digital Image Processing, Third Edition, 2008.