ijecs open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 03 Issue 07 July, 2014 Page No. 7018-7031

FACIAL DETECTION SYSTEM (Using MATLAB)

Anjali Vats, R.K Bhatnagar, Shikha Vats, Antara Sarkar and Ankit

Hmr Institute Of Technology & Management, Hamidpur

CSE, KIIT College of Engineering

(Guru Gobind Singh Indraprastha University)

INRODUCTION

The face is our primary focus of attention in social life playing an important role in conveying identity and emotions. We can recognize a number of faces learned throughout our lifespan and identify faces at a glance even after years of separation. This skill is quite robust despite of large variations in visual stimulus due to changing condition, aging and distractions such as beard, glasses or changes in hairstyle.

Face detection is used in many places now a days especially the websites hosting images like picassa, photobucket and facebook. The automatically tagging feature adds a new dimension to sharing pictures among the people who are in the picture and also gives the idea to other people about who the person is in the image. In our project, we have studied and implemented a pretty simple but very effective face detection algorithm which takes human skin colour into account. Our aim, which we believe we have reached, was to develop a method of face detection that is fast, robust, reasonably simple and accurate with a relatively simple and easy to understand algorithms and techniques. The examples provided in this thesis are real-time and taken from our own surroundings.

FACE DETECTION

Face detection is a computer technology that determines the locations and sizes of human faces in digital images. It detects face and ignores anything else, such as buildings, trees and bodies. Face detection can be regarded as a more general case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). In face detection, face is processed and matched bitwise with the underlying face image in the database. Any slight change in facial expression, e.g. smile, lip movement, will not match the face.

Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars.

Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.

APPLICATIONS OF FACE DETECTION SYSTEM

Facial recognition

Face detection is used in biometrics, often as a part of (or together with) a facial recognition system. It is also used in video surveillance, human computer interface and image database management.

Photography

Some recent digital cameras use face detection for autofocus. Face detection is also useful for selecting regions of interest in photo slideshows .

Marketing

Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by. The system then calculates the race, gender, and age range of the face. Once the information is collected, a series of advertisements can be played that is specific toward the detected race/gender/age. An example of such a system is called OptimEyes and is integrated into the Amscreen digital signage system.

Criminal Investigations

Powerful 2D to 3D transformation of video or photo Facial Images.Fast, accurate and analytical comparison of multiple Facial Images for Precise Facial Recognition and Criminal Screening.

Face Detection Method

The objective was to design and implement a face detector in MATLAB that will detect human faces in an image similar to the training images.

The problem of face detection has been studied extensively. A wide spectrum of techniques have been used including color analysis, template matching, neural networks, support vector machines (SVM), maximal rejection classification and model based detection. However, it is difficult to design algorithms that work for all illuminations, face colors, sizes and geometries, and image backgrounds. As a result, face detection remains as much an art as science.

Our method uses rejection based classification. The face detector consists of a set of weak classifiers that sequentially reject non-face regions. First, the non-skin color regions are rejected using color segmentation. A set of morphological operations are then applied to filter the clutter resulting from the previous step. The remaining connected regions are then classified based on their geometry and the number of holes. Finally, template matching is used to detect zero or more faces in each connected region. A block diagram of the detector is shown in Figure 1.

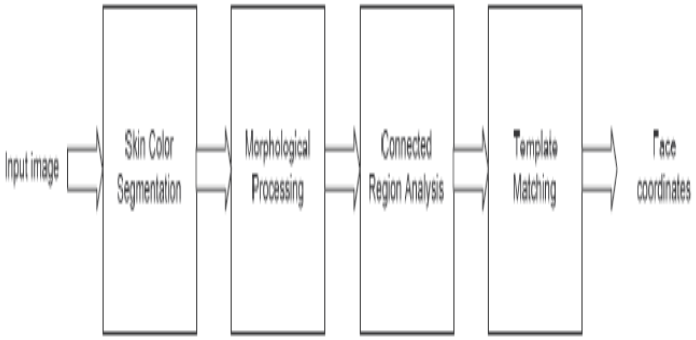


Figure 1. Block diagram of face detector

STEPS IN FACE DETECTION

a. Skin Color Segmentation

The goal of skin color segmentation is to reject non-skin color regions from the input image. It is based on the fact that the color of the human face across all races agrees closely in its chrominance value and varies mainly in its luminance value.

We chose the RGB (Red, Green ,Blue) color space for segmentation since it decouples the chrominance information from the luminance information. Thus we can only focus on the hue and the saturation component. The faces in each training image were extracted using the ground truth data and a histogram was plotted for their color component s. The histograms reveal that the

RGB color components for faces are nicely clustered. This information was used to define appropriate thresholds for RGB space that correspond to faces. The threshold values were embedded into the color segmentation routine.

During the execution of the detector, segmentation is performed as follows:

1. The input image is subsampled at 2:1 to improve computational efficiency

2. The resulting image is converted to RGB color space

3. All pixels that fall outside the RGB thresholds are rejected (marked black).

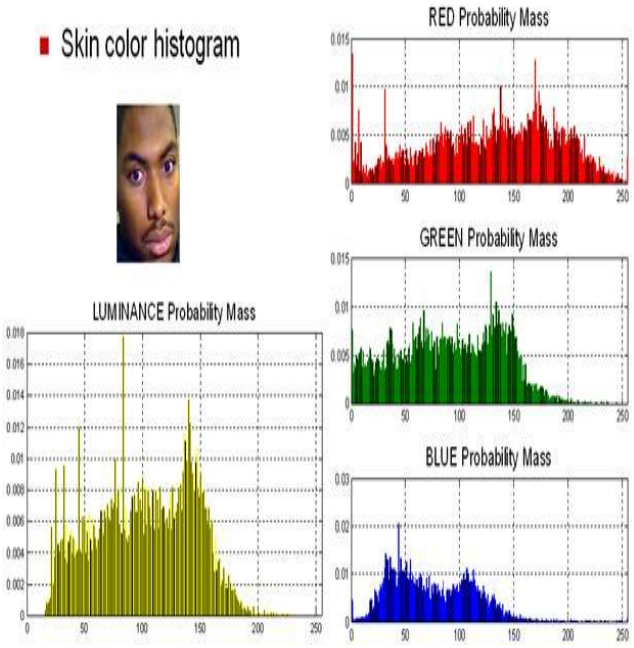


Fig 1: Histograms

b. Morphological Processing

Figure above shows that skin color segmentation did a good job of rejecting non-skin colors from the input image. However, the resulting image has quite a bit of noise and clutter. A series of morphological operations are performed to clean up the image, as shown in Figure 4. The goal is to end up with a mask image that can be applied to the input image to yield skin color regions without noise and clutter.

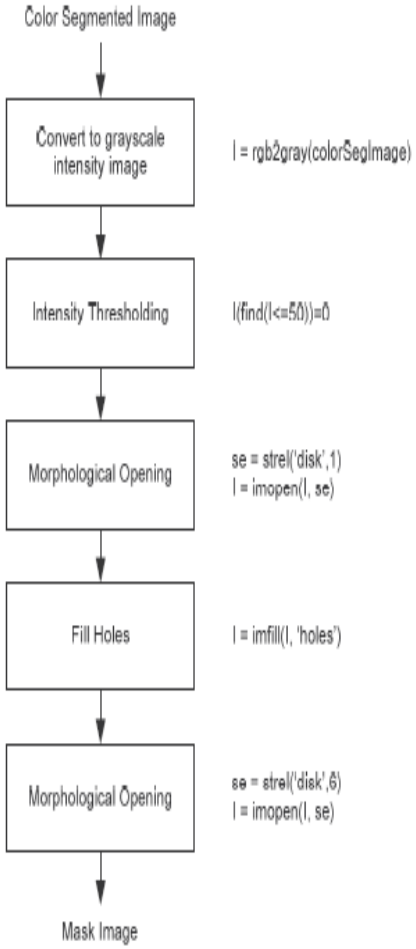


Figure 4. Morphological Processing on the color segmented image

A description of each step is as follows:

1. Since morphological operations work on intensity images, the color segmented image is converted into a gray scale image.

2. Intensity thresholding is performed to break up dark regions into many smaller regions so that they can be cleaned up by morphological opening. The threshold is set low enough so that it doesn't chip away parts of a face but only create holes in it.

3. Morphological opening is performed to remove very small objects from the image while

preserving the shape and size of larger objects in the image. The definition of a morphological opening of an image is an erosion followed by a dilation, using the same structuring element for both operations. A disk shaped structuring element of radius 1 is used.

4. Hole filling is done to keep the faces as single connected regions in anticipation of a second much larger morphological opening. Otherwise, the mask image will contain many cavities and holes in the faces.

5. Morphological opening is performed to remove small to medium objects that are safely below the size of a face.

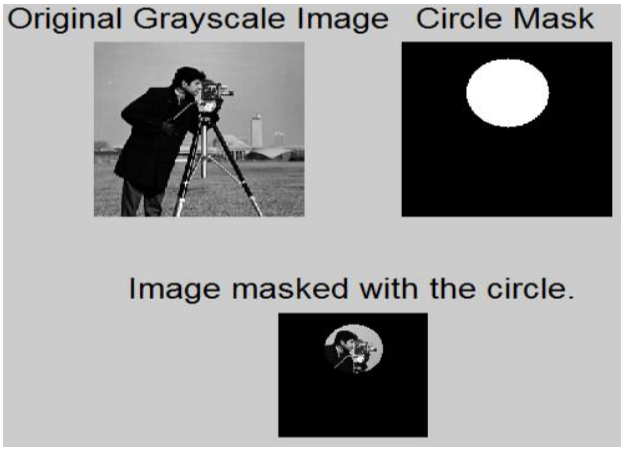


Fig 2:

Fig 3:

c. Connected Region Analysis

The image output by morphological processing still contains quite a few non-face regions. Most of these are hands, arms, regions of dress that match skin color and some portions of background. In connected region analysis, image statistics from the training set are used to classify each connected region in the image.

Rejection based on Geometry

We defined four classes of regions that have a very high probability of being non-faces based on their bounding box:

Narrow: Regions that have a small width

Short : Regions that have a small height

Narrow and tall : Regions that have a small width but large height

Wide and short: Regions that have a large width but small height

We did not define the wide and tall class because that interferes with large regions that contain multiple faces.

Based on the training set image statistics, thresholds were calculated for each class. The constraints were then applied in the following order:

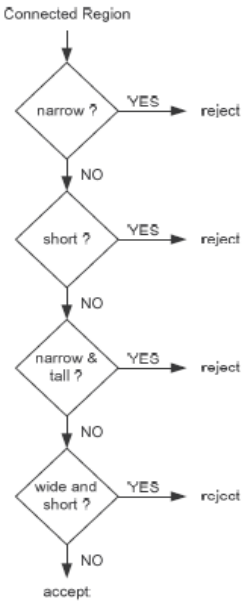


Figure 6. Flow chart for rejection based on region geometry

d. Template Matching

The basic idea of template matching is to convolve the image with another image (template) that is representative of faces. Finding an appropriate template is a challenge since ideally the template (or group of templates) should match any given face irrespective of the size and exact features.

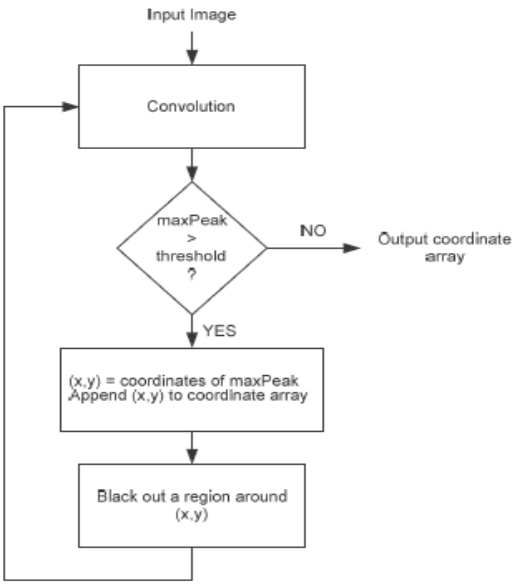


Figure 13. Flowchart of the template matching algorithm

Viola – Jones Technique

The **Viola–Jones object detection framework** is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Violaand Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection.

Components and Framework

- Feature types and evaluation[

The features employed by the detection framework universally involve the sums of image pixels within rectangular areas. As such, they bear some resemblance to Haar basis functions, which have been used previously in the realm of image-based object detection.[1]However, since the features used by Viola and Jones all rely on more than one rectangular area, they are generally more complex. The figure at right illustrates the four different types of features used in the framework. The value of any given feature is always simply the sum of the pixels within clear rectangles subtracted from the sum of the pixels within shaded rectangles. As is to be expected, rectangular features of this sort are rather primitive when compared to alternatives such as steerable filters. Although they are sensitive to vertical and horizontal features, their feedback is considerably coarser. However, with the use of an image representation called the integral image, rectangular features can be evaluated in *constant* time, which gives them a considerable speed advantage over their more sophisticated relatives. Because each rectangular area in a feature is always adjacent to at least one other rectangle, it follows that any two-rectangle feature can be computed in six array references, any three-rectangle feature in eight, and any four-rectangle feature in just nine.

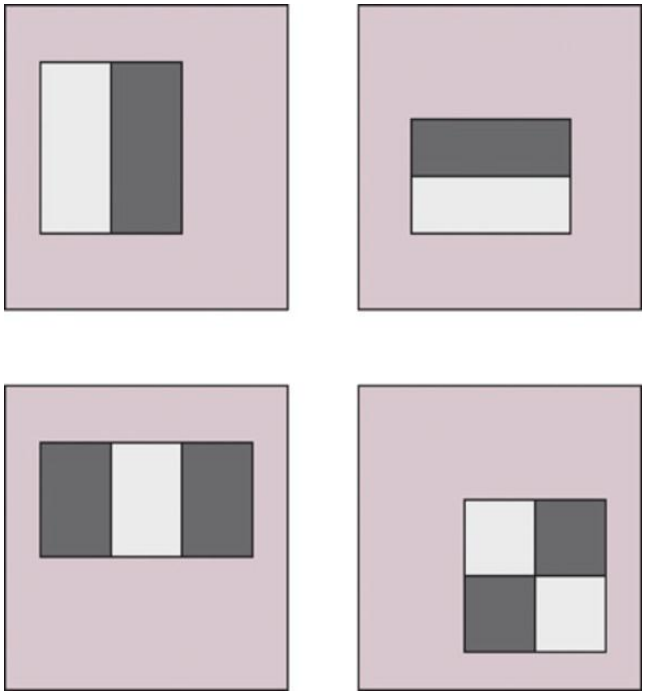


Fig 5:

Viola-Jones defines several different types of features (see Figure above)

- The first feature is a light block next to a dark one, vertically.
- The second is the same but horizontally.
- The third is a light block sandwiched between two dark blocks (or vice versa).
- The fourth is a four-rectangle feature as shown.

Note that in a feature the rectangular blocks are the same size. The features can be at any scale or at any position in the image, so the features shown in Figure 1 are just examples at an arbitrary size. A particular feature can be positioned and scaled onto the original image.

The feature value is calculated as the sum of the pixel intensities in the light rectangle(s) minus the sum of the pixels in the dark rectangle(s). The value of the feature is then used in a filter to determine if that feature is 'present' in the original image.

It still sounds computationally expensive (that is, slow). To improve the speed of summing the intensities of the pixels in a given rectangle, we make use of a trick. For a given image we can calculate what's known as the integral image at every point.

This is merely the sum of the pixels of the rectangle from the upper-left corner to the given point, and it can be calculated for every point in the original image in a single pass across the image. The word 'integral' comes from the same meaning as 'integrating' – finding the area under a curve by adding together small rectangular areas.

Calculations

Once we have the integral image for every point, we can calculate the sum of intensities for any arbitrary rectangle by using the identities shown in Figure 2.

We want to calculate the sum of the pixels for the rectangle abcd.

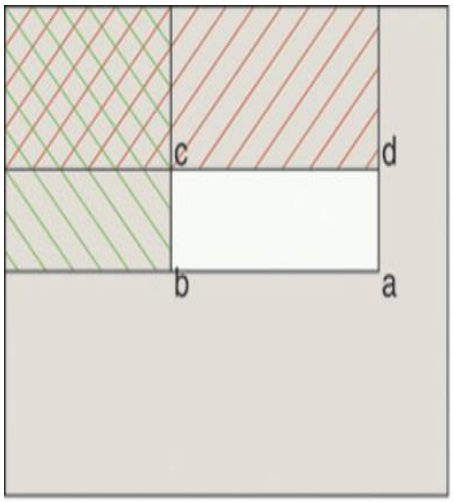


Fig 5: Calculating sums of intensities using integral images

We start off with the integral image for point a, and then subtract the integral images for points b and d. This unfortunately takes off too much (as shown by the double hatching) and so we add back in the integral image for point c. As you can see, it's a very quick calculation once we have generated all the integral images, presumably stored in an array.

We now have in place all the requisites to calculate the summed intensity values for a set of features. But what are we going to do with them? Compared with analysing the pixels directly, features provide a coarse, low-resolution view of the image. They're good at detecting edges between light and dark, bars, and other simple structures.

What we do is to implement a learning system. We give the face detection routine a set of 24 x 24 images of faces (and another set of 24 x 24 images of things that are not faces) and train the routine to recognise faces and discard non-faces. Viola and Jones used a database culled from the internet of about 4,000 faces and 10,000 non-faces to do the training. What's involved in the training?

Using 24 x 24 images, there are some 45,000 different ways to place one of the four types of feature onto the image. For example, for the first type of feature, you could have rectangles one pixel wide by two pixels deep, all the way up to 1 x 24, then 2 x 2 to 2 x 24, and so on. These different-sized features would be placed in various positions across the image to test every possible feature for every possible size at every possible position.
Note that the number of possible features, 45,000, is far greater than the number of pixels in a 24 x 24 image (a mere 576 pixels) and so you must reduce the number you use. Remember, you're calculating the difference between the pixel sum for the light and dark parts of the feature.

You could decide on a threshold for difference (which could be tweaked during training) whereby a feature is assumed to be detected or not. Using this, you would then apply every one of the 45,000 possible features to your training set.

What you'd find is that certain features are worthless at determining whether an image is a face or not – that is, there would be no correlation between the feature identifying a face and it not being one, and vice versa. These would be rejected.

- Learning algorithm

The speed with which features may be evaluated does not adequately compensate for their number, however. For example, in a standard 24x24 pixel sub-window, there are a total of 162,336 possible features, and it would be prohibitively expensive to evaluate them all. Thus, the object detection framework employs a variant of the learning algorithmAdaBoost to both select the best features and to train classifiers that use them.

Viola and Jones then experimented with the remaining features to determine the best way of using them to classify an image as 'face' or 'non-face'. After experimentation they decided to use a training system called AdaBoost to build a classifier.

AdaBoost is an artificial intelligence (AI) technique similar to a neural network, devised to combine 'weak' features into a 'stronger' classifier. Each feature within a classifier is assigned a weighting (tweaked during training) that defines how 'accurate' that classifier is. Low weighting means a weak feature, high weighting a stronger one.

Add up the weightings of the features that test positive for a particular image and if that sum is above a threshold (again, tweakable during training) then that image is determined to be a face.

As it happens, during this training they found that there were two features that, when combined and properly tuned by AdaBoost into a single

classifier, would pass though 100 per cent of the faces with a 40 per cent false positive rate (60 per cent of the non-faces would be rejected by this classifier).

Figure 6 shows this simple classifier in action. It uses two features to test the image: a horizontal feature that measures the difference between the darker eyes and the lighter cheekbones, and the three-rectangle feature that tests for the darker eyes against the lighter bridge of the nose.

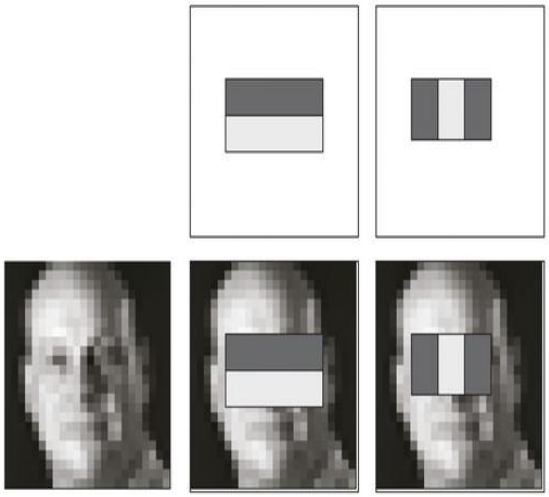


Fig 6: The first classifier on a 24 x 24 image of the author's face showing the two features in use

Although they had been trying to implement a strong classifi er from a combination of 200 or so weak classifi ers, this early success prompted them to build a cascade of classifiers instead of a single large one (see Figure 4).

Each subwindow of the original image is tested against the first classifier. If it passes that classifier, it's tested against the second. If it passes that one, it's then tested against the third, and so on. If it fails at any stage of the testing, the subwindow is rejected as a possible face. If it passes through all the classifiers then the subwindow is classified as a face.

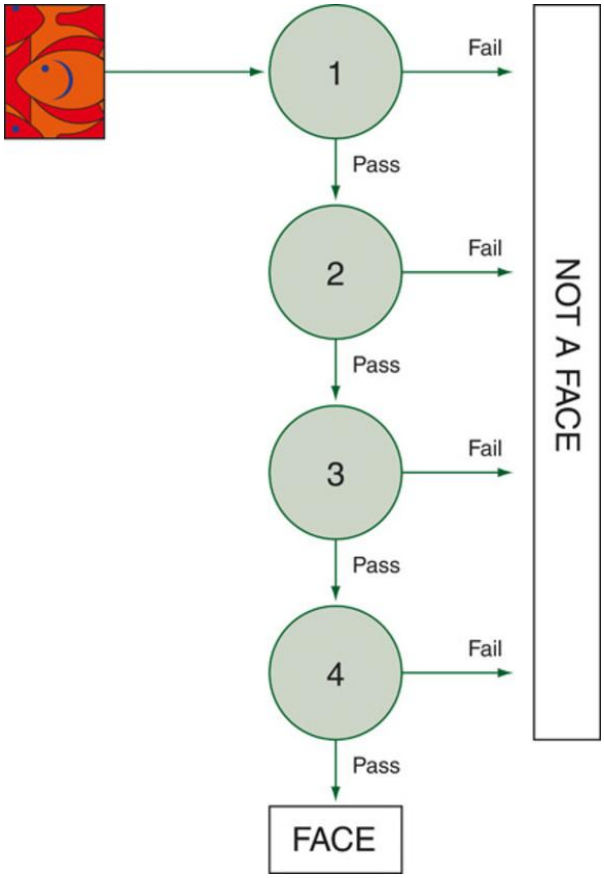


Fig 7: The Viola-Jones cascade of classifiers

The interesting thing is that the second and subsequent classifiers are not trained with the full training set. Instead they are trained on the images that pass the prior classifiers in the chain.

The second and subsequent classifiers are more complex and have more features than the first one, so they require more computational time. It seems remarkable then that there is such a simple classifier that will reject so many sub-windows without having to go through the calculations required for the more complex classifiers.

Remember that it's far more likely that a random image contains no face at all, and so being able to reject the majority of sub-windows with as small a set of calculations as possible is a good thing.

- **Final reckoning**

The eventual cascade developed by Viola and Jones had 32 stages and used a total of 4,297 features (out of the original total of 45,000). The first classifier uses the two features described

above, the next uses five further features and rejects 80 per cent of the non-faces. The next three use 20 features and subsequent ones even more. The whole training took several weeks.

Finally, when presented with an actual image, the face detector scans the complete image at multiple scales and with multiple locations per scale. The researchers found that scaling the image by a factor of 1.25 each time produced the most accurate results. They also discovered that they didn't need to test each individual location: they could skip a couple or so pixels each time and still get good results.

The overall result is that the Viola-Jones method produces accurate results very quickly, and certainly fast enough for the limited processing power of the average point-and-shoot camera to cope with.

Softwares used in Face Detection:

Notable software with face detection ability includes:

- digiKam
- iPhoto
- OpenCv
- Photoshop Elements
- Picasa
- MATLAB

digiKam

digiKam is a free and open-source image organizer and tag editor written in C++ utilizing the KDE Platform.

digiKam is an advanced digital **photo management application** for KDE, which makes importing and organizing digital photos a "snap". The photos can be organized in albums which can be sorted chronologically, by directory layout or by custom collections. digiKam also provides **tagging functionality**. You tag your images which can be spread out across multiple folders and digiKam provides fast and intuitive ways to browse these tagged images. You can also add **comments and rating** to your images. digiKam makes use of a fast and robust database to store these meta-informations which makes adding and editing of comments and tags very reliable.

digiKam runs on most known desktop environments and window managers, as long as the required libraries are installed. It supports all major image file formats, and can organize collections of photographs in directory-based albums, or dynamic albums by date, timeline, or by tags. Users can also add captions and ratings to their images, search through them and save searches for later use..

- Organization of photos in albums and sub-albums (with tags and comments support)
- Support for Exif, Iptc, Xmp, Makernotes
- SQLite powered storage for the albums and its metadata
- Support for filtering and sorting albums
- Import from more than 1100 digital camera devices
- Support for 300 RAW format pictures
- Light Table feature
- Plugins architectures: using Kipi-plugins
- Share your photos using HTML or publishing them to social web sites
- Image editor included with advanced features.

- **i Photo**

iPhoto is a digital photograph manipulation software application developed by Apple Inc. It

has been included with every Macintosh personal computer since 2002, originally as part of the iLife suite of digital media management applications. iPhoto can import, organize, edit, print and share digital photos.

iPhoto is designed to allow the importing of pictures from digital cameras, local storage devices such as USB flash drive, CDs, DVSs and hardrives to a user's iPhoto Library. Almost all digital cameras are recognized without additional software. iPhoto supports most common image file formats, including several Raw image formats. iPhoto also supports videos from cameras, but editing is limited to trimming clips.

After photos are imported, they can be titled, labeled, sorted and organized into groups (known as "events"). Individual photos can be edited with basic image manipulation tools, such as a red-eye filter, contrast and brightness adjustments, cropping and resizing tools, and other basic functions. iPhoto does not, however, provide the comprehensive editing functionality of programs such as Apple's own Aperture, or Adobe's Photoshop (not to be confused with Photoshop Elements or Album), or GIMP.

iPhoto offers numerous options for sharing photos. Photo albums can be made into dynamic slideshows and optionally set to music imported from iTunes. Photos can be shared via iMessage, Mail, Facebook, Flickr and Twitter. Creating and sharing iCloud Photostreams are possible as well,[4] both public and invitation based ones. iPhoto can also sync photo albums to any iPod with a color display. These iPods may also have an audio/video output to allow photos to be played back, along with music, on any modern television. Additionally, photos can be printed to a local printer, or, in certain markets, be sent over the internet to Kodak for professional printing. iPhoto users can order a range of products, including standard prints, posters, cards, calendars, and 100-page hardcover or softcover volumes — again, such services are available only to users in certain markets.

- **OpenCV**

  OpenCV (*Open Source Computer Vision*) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel Russia research center in Nizhny Novgorod, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself.

  OpenCV is free open-source library intended for use in image processing, computer vision and machine learning areas. It have a huge amount of different algorithms, but in this topic i will compare their existing feature detectors. At this moment OpenCV has stable 2.2 version and following types of descriptors: Fast**,** GoodFeaturesToTrack**,** Mser**,** Star**,** Sift**,** Surf**.** And few Adapters over detectors: Grid Adapted, Pyramid Adapted, Dynamic Adapted. In this article noticed above detectors will be compared by speed, quality and repeatability in different lighting and scale.

  Image-manipulations – this example demonstrates how OpenCV can be used as an image processing and manipulation library. It supports several filters, demonstrates color space conversions and working with histograms. It doesn't have special relation to computer vision, but OpenCV has powerful core and imgproc modules, which may be useful in a wide range of applications, especially in the field of Computational Photography.

Photoshop elements

Adobe Photoshop Elements is a raster graphics editor for hobbyists and consumers. It contains most of the features of the professional version but with fewer and simpler options. The program allows users to create, edit, organize and share images. It is a successor of Adobe Photoshop LE.

- CMYK and LAB color modes
- More tools and features that work with high-bit (16-bit and 32-bit) images
- Channels Palette
- Recording custom Actions (for batch processing)
- Adjustments: Color Balance, Match Color
- Layer Comps, and Quick Mask mode
- Smart Objects, Smart Guides
- Lens Blur Filter
- Vanishing Point Tool
- Puppet Warp
- Pen tool and paths palette
- Mixer brush and bristle tips painting tools
- Some adjustment layers (curves, color balance, selective color, channel mixer, vibrance)
- Editing History Log
- Text on a path, advanced text formatting
- Advanced Layer Style manipulation
- Advanced Color Management
- Advanced Web features (rollovers, slicing)
- Customizable tool presets, keyboard shortcuts, and menus
- In the features and tools that are shared, the Photoshop version usually offers more advanced options for fine tuning and control.

Picasa

Picasa is an image organizer and image viewer for organizing and editing digital photos, plus an integrated photo-sharing website, originally created by a company named Lifescape (which at that time may have resided at Idealab) in 2002 and owned by Google since 2004. "Picasa" is a blend of the name of Spanish painter Pablo Picasso, the phrase *mi casa* (Spanish for "my house") and "pic" for pictures (personalized art). In July 2004, Google acquired Picasa from its original author and began offering it as freeware.

We have noticed that due to the recent colossal influx of cloud based apps in software arena, many image editors and viewers are either totally drifting away from the core image organizing and editing features which a common user likes to see, or have started capitalizing on everything, from online image sharing to social media components integration with long winded configuration process to go around with. That's the reason why so many eminent image viewers and editors are being cascaded down the popularity graph.

So, what makes an image editing software stand out from the rest? I am not an advocate of outright simplicity but when it comes to choosing a good software which while offering all important features to catch up with fast track online image sharing blaze, also provides features to enhance image viewing and editing experience, only one springs readily to my mind, and that is, **Picasa**. Whether you want to brush up on your image editing skills, share photos with your friends and family, or simply like to enjoy an awesome image viewer on your desktop with web sync, Picasa is arguably the most viable option you can opt in for. In this post, we will try to delve deep into its most underrated features.

SOFTWARE USED – MATLAB

MATLAB (**mat**rix **lab**oratory) 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, and Fortran.

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.

In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

TOOLBOX USED

Computer Vision System Toolbox - provides algorithms, functions, and apps for the design and simulation of computer vision and video processing systems. You can perform object detection and tracking, feature detection and extraction, feature matching, stereo vision, camera calibration, and motion detection tasks. The system toolbox also provides tools for video processing, including video file I/O, video display, object annotation, drawing graphics, and compositing. Algorithms are available as MATLAB functions, System objects and Simulink blocks.
Key Features

- High-level language for numerical computation, visualization, and application development
- Interactive environment for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
- Built-in graphics for visualizing data and tools for creating custom plots
- Development tools for improving code quality and maintainability and maximizing performance
- Tools for building applications with custom graphical interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft Excel.

Application of MATLAB

- A very large (and growing) database of built-in algorithms for image processing and computer vision applications.
- MATLAB allows you to test algorithms immediately without recompilation. You can type something at the command line or execute a section in the editor and immediately see the results, greatly facilitating algorithm development.
- The MATLAB Desktop environment, which allows you to work interactively with your data, helps you to keep track of files and variables, and simplifies common programming/debugging tasks
- The ability to read in a wide variety of both common and domain-specific image formats.
- The ability to call external libraries, such as OpenCV
- Clearly written documentation with many examples, as well as online resources such as web seminars ("webinars").
- Bi-annual updates with new algorithms, features, and performance enhancements
- If you are already using MATLAB for other purposes, such as simulation, optimation,

statistics, or data analysis, then there is a very quick learning curve for using it in image processing.

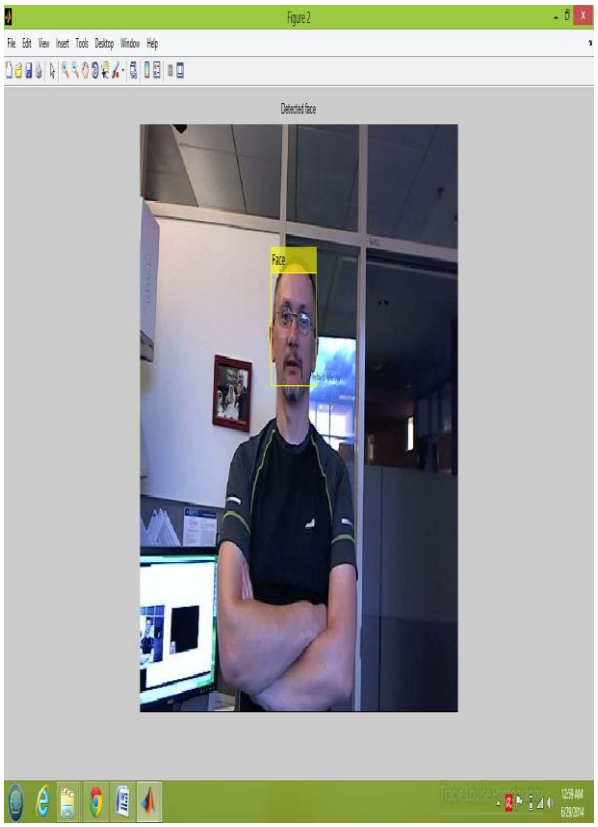
- The ability to process both still images and video.
- Technical support from a well-staffed, professional organization (assuming your maintenance is up-to-date)
- A large user community with lots of free code and knowledge sharing
- The ability to auto-generate C code, using MATLAB Coder, for a large (and growing) subset of image processing and mathematical functions, which you could then use in other environments, such as embedded systems or as a component in other software.

MATLAB CODING

Step 1: Detect a Face

Before you begin tracking a face, you need to first detect it. Use the vision.CascadeObjectDetector to detect the location of a face in a video frame. The cascade object detector uses the Viola-Jones detection algorithm and a trained classification model for detection. By default, the detector is configured to detect faces, but it can be configured for other object types.

```
>> % Create a cascade detector object.

faceDetector = vision.CascadeObjectDetector();



% Read a video frame and run the detector.

videoFileReader = vision.VideoFileReader('visionface.avi');

videoFrame = step(videoFileReader);

bbox = step(faceDetector, videoFrame);

% Draw the returned bounding box around the detected face.

videoOut = insertObjectAnnotation(videoFrame,'rectangle',bbox,'Face');

figure, imshow(videoOut), title('Detected face');
```



Step 2: Identify Facial Features

Once the face is located in the video, the next step is to identify a feature that will help you track the face. For example, you can use the shape, texture, or color. Choose a feature that is unique to the object and remains invariant even when the object moves.

In this example, you use skin tone as the feature to track. The skin tone provides a good deal of contrast between the face and the background and does not change as the face rotates or moves.

>> % Get the skin tone information by extracting the Hue from the video frame

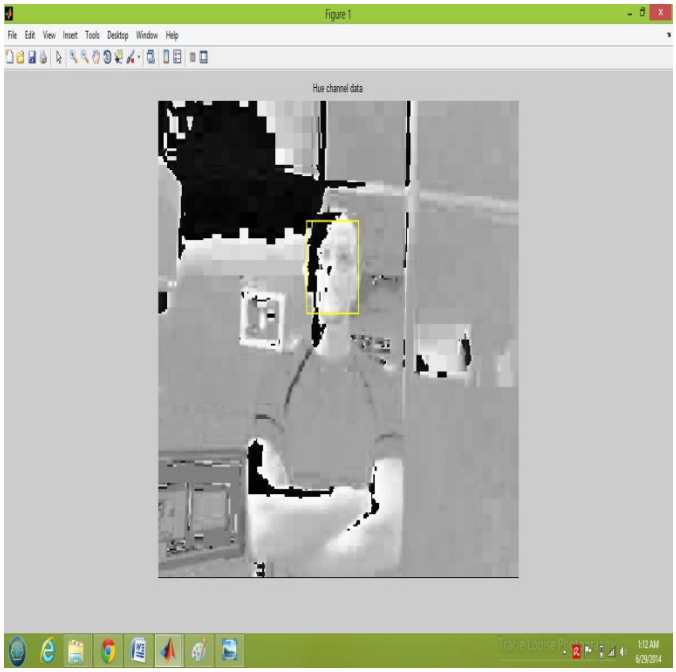
% converted to the HSV color space.

[hueChannel,~,~] = rgb2hsv(videoFrame);

% Display the Hue Channel data and draw the bounding box around the face.

figure, imshow(hueChannel), title('Hue channel data');

rectangle('Position',bbox(1,:),'LineWidth',2,'Edge Color',[1 1 0])



CONCLUSION

We have presented a face detector with a reasonably good accuracy and running time. However, many aspects of the design are tuned for the constrained scene conditions of the training images provided, hurting its robustness. This is not unfair given the scope and requirements of the project. Our algorithm is sensitive to the color information in the image and will not work for a gray scale image.

We feel that detecting connected faces was the hardest part of the project. A great deal of time was spent coming up with a template matching scheme that adapts well to connected faces, including those that are partly visible.

REFERENCES

- http://en.wikipedia.org/wiki/Face_detection
- Search the URL : http://www.mathworks.in/help/vision/examples/face-detection
- C. Garcia and G. Tziritas, "Face detection using quantized skin color region merging and wavelet packet analysis," IEEE Transactions on Multimedia Vol.1, No. 3, pp. 264--277, September 1999.
- H. Rowley, S. Baluja, and T. Kanade, “Neural Network-Based Face Detection,” IEEE
- Transactions on Pattern Analysis and Machine Intelligence, volume 20, number 1, pages 23-38, January 1998.
- Classifier,” Pattern Recognition Letters, Vol. 23, Issue 12, pp. 1459-1471, October 2002
- The Face Detection Homepage, http://home.Online.de/home/Robert.Frischholz/index.html