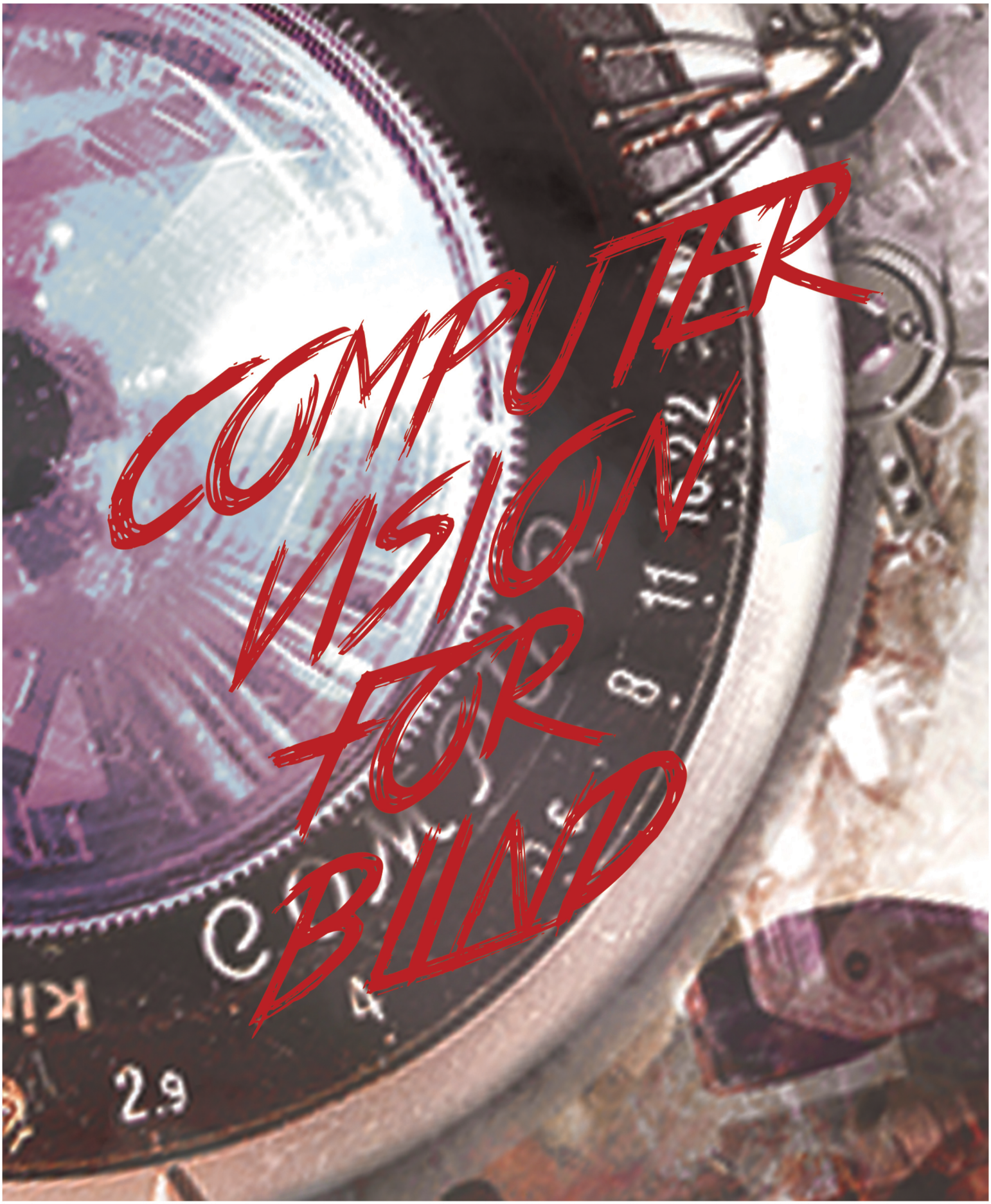
CVB | COMPUTER VISION FOR BLIND

2015-2016




AHMED MOHAMMED ABDUL-AZIM ABUAMRA
6 OCTOBER SECONDARY SCHOOL
NAIRA ATEF DRWEESH MAHDY
ANSAF SERRY SECONDARY SCHOOL

# Content

## Abstract

Our Project provide the blind with portable device that help them detect objects, people and read, using computer vision technology. We've used Opencv Cascade classifier to Detect objects and Local binary pattern to recognize people, Tesseract-OCR to recognize Text. We've used E-speak TTS to enable the blind to hear the output, we've used Odroid u3 as a hardware platform, an Logitech C270 HD webcam and 10000mA battery which enable device to run for 10: 12 hours. Our control circuit has 3 buttons.


## Introduction

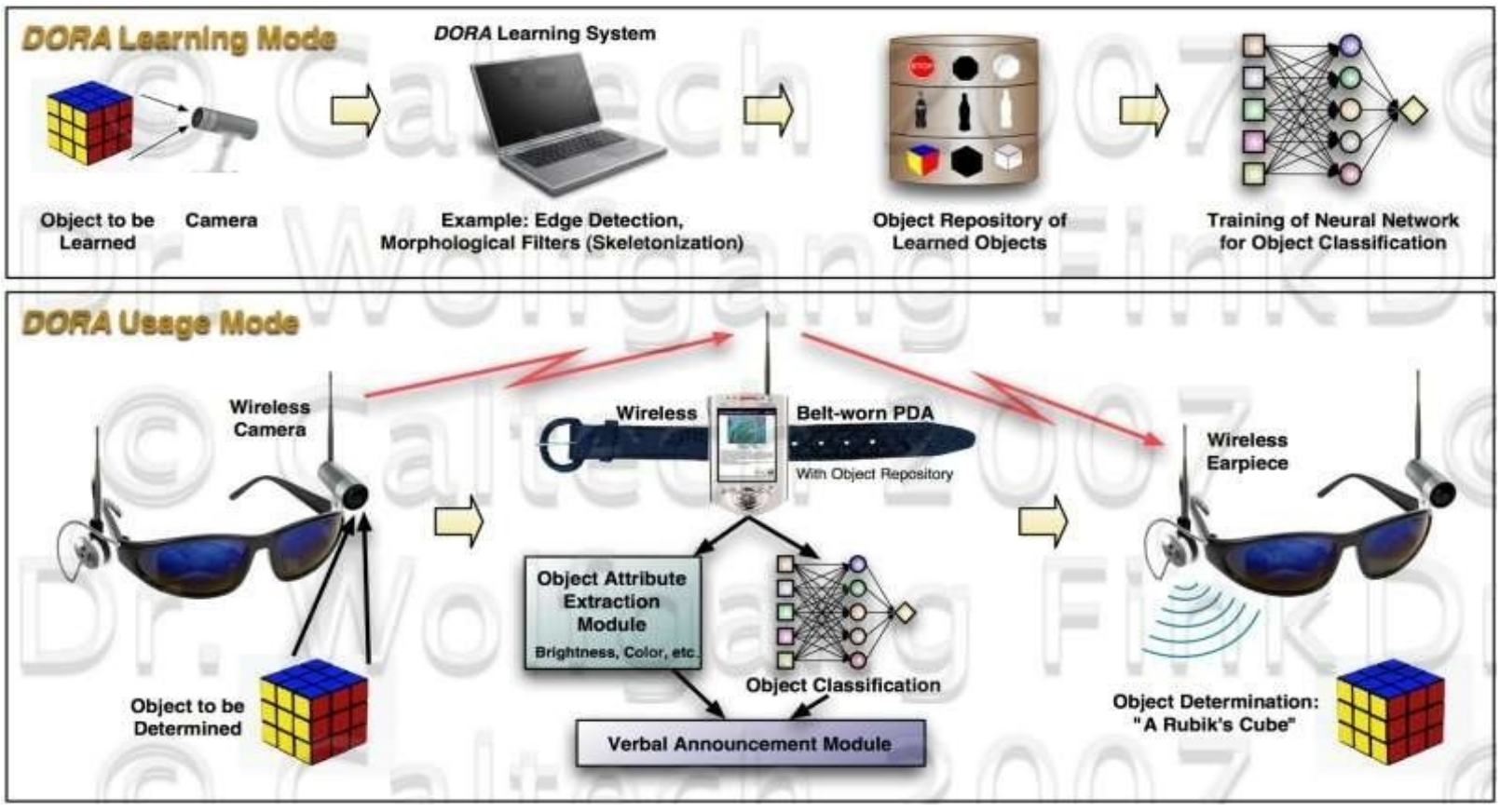
There are more than 39 million blind human around the world; Computer vision has great potential to solve their problems.
In the past engineers used desktop computer for processing these applications, it was difficult to use in real life.
The embedded computers were not fast enough to process and analyses images. But now we have a powerful processor.
So we've designed a portable system with 1.7GHz Quad core processor that can run for 10: 12 hours to recognize faces and objects and read letters.

# 3- Scientific background

## 3.1- Related works

### 3.1.1- DORA: Digital Object-Recognition Audio-Assistant for the Visually Impaired

A digital camera mounted on the patient's eyeglasses or head takes images on demand, e.g., at the push of a button or a voice command. Via near-real-time image processing algorithms, parameters such as the brightness (i.e., bright, medium, dark), color (according to a predefined color palette), and content of the captured image frame are determined. The content of an image frame is determined as follows: First, the captured image frame is processed for edge detection within a central region of the image only, to avoid disturbing effects along the borderline. The resulting edge pattern is subsequently classified by artificial neural networks that have been previously trained on a list of identifiable everyday objects such as dollar bills, credit cards, cups, plates, etc. Following the image processing, a descriptive sentence is constructed consisting of the determined/classified object and its descriptive attributes (brightness, color, etc.): "This is a dark blue cup". Via a computer-based voice synthesizer, this descriptive sentence is then announced verbally to the severely visually impaired or blind patient.

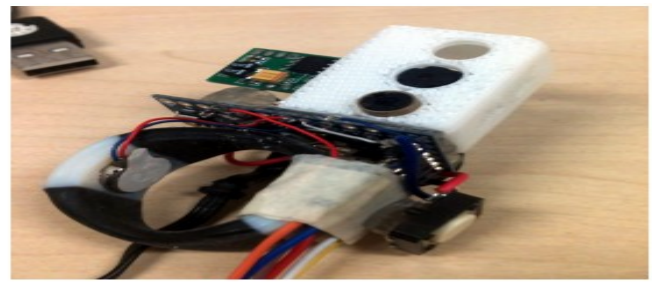


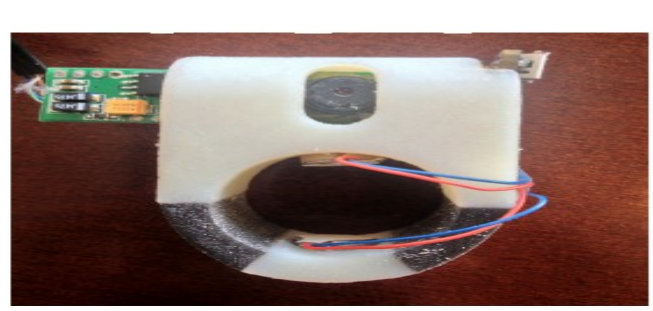
### 3.1.2- **FaceSpeaker**

FaceSpeaker uses face detection and recognition algorithms from the emguCV library, which is a C# wrapper for the openCV computer vision library. By combining accurate Fisherface and LBPH face recognition algorithms, FaceSpeaker can recognize faces in a variety of lighting conditions while limiting the number of misidentifications. This is certainly not a perfect solution but does work well enough for the purpose of building a usable prototype. This article was the main reference for learning how to perform face detection and recognition using emguCV.

### 3.1.3-**FingerReader : A wearable reading device**

FingerReader is an index-finger wearable device that supports the VI in reading printed text by scanning with the finger. The design continues the work we have done on the EyeRing; however this work features novel hardware and software that includes haptic response, video-processing algorithms and different output modalities. The finger-worn design helps focus the camera at a fixed distance and utilizes the sense of touch when scanning the surface. Additionally, the device does not have many buttons or parts in order to provide a simple interface for users and easily orient the device.





### 3.2 - Optical Character Recognition (OCR) :-

OCR Is the mechanical or electronic conversion of images of typewritten or printed text into machine-encoded text. It is widely used as a form of data entry from printed paper data records, whether passport documents, invoices, bank statement, receipts, business card, mail, or other documents. It is a common method of digitizing printed texts so that it can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as translation, text, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision. Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems that have a high degree of recognition accuracy for most fonts are now common. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

### 3.2.1- Tesseract-OCR:-

**Motivation and History**

Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994. Like a supernova, it appeared from nowhere for the 1995 UNLVAnnual Test of OCR Accuracy ,shone brightly with its results, and then vanished back under the same cloak of secrecy under which it had been developed. Now for the first time, details of the architecture and algorithms can be revealed. Tesseract began as a PhD research project in HP Labs, Bristol, and gained momentum as a possible software and/or hardware add-on for HP's line of flatbed scanners. Motivation was provided by the fact that the commercial OCR engines of the day were in their infancy, and failed miserably on anything but the best quality print. After a joint project between HP Labs Bristol, and HP's scanner division in Colorado, Tesseract had a significant lead in accuracy over the commercial engines, but did not become a product. The next stage of its development was back in HP Labs Bristol as an investigation of OCR for compression. Work concentrated more on improving rejection efficiency than on base-level accuracy. At the end of this project, at the end of 1994, development ceased entirely. The engine was sent to UNLV for the 1995 Annual Test of OCR Accuracy, where it proved its worth against the commercial engines of the time. In late 2005, HP released Tesseract for open source.

### 3.2.2- Line Finding

the line finding algorithm is one of the few parts of Tesseract that has previously been published The line finding algorithm is designed so that a skewed page can be recognized without having to de-skew, thus saving loss of image quality. The key parts of the process are blob filtering and line construction. Assuming that page layout analysis has already provided text regions of a roughly uniform text size, simple percentile height filter removes drop-caps and vertically touching characters. The median height approximates the text size in the region, so it is safe to filter out blobs that are smaller than some fraction of the median height, being most likely punctuation, diacritical marks and noise. The filtered blobs are more likely to fit a model of non-overlapping, parallel, but sloping lines. Sorting and processing the blobs by x-coordinate makes it possible to assign blobs to a unique text line, while tracking the slope across the page, with greatly reduced danger of assigning to an incorrect text line in the presence of skew. Once the filtered blobs have been assigned to lines, a least median of squares fit is used to estimate the baselines, and the filtered-out blobs are fitted back into the appropriate lines. The final step of the line creation process merges blobs that overlap by at least half horizontally, putting diacritical marks together with the correct base and correctly associating parts of some broken characters.

### 3.2.3- **Baseline fitting**

once the text lines have been found, the baselines are fitted more precisely using a quadratic spline. This was another first for an OCR system, and enabled Tesseract to handle pages with curved baselines. Which are common artifacts in scanning, and not just at book bindings.The baselines are fitted by partitioning the blobs into groups with a reasonably continuous displacement for the original straight baseline. A quadratic spline is fitted to the most populous partition, (assumed to be the baseline) by a least squares fit. The quadratic spline has the advantage that this calculation is reasonably stable, but the disadvantage that discontinuities can arise when multiple spline segments are required. A more traditional cubic spline might work better.


### 3.2.4- **Fixed Pitch Detection and Chopping**

Tesseract tests the text lines to determine whether they are fixed pitch. Where it finds fixed pitch text, Tesseract chops the words into characters using the pitch, and disables the chopper and associator on these words for the word recognition step

### 3.2.5- **Proportional Word Finding**

Non-fixed-pitch or proportional text spacing is a highly non-trivial task. Illustrates some typical problems. The gap between the tens and units of ‘11.9%’ is a similar size to the general space, and is certainly larger than the kerned space between ‘erated’ and ‘junk’. There is no horizontal gap at all between the bounding boxes of ‘of’ and ‘financial’. Tesseract solves most of these problems by measuring gaps in a limited vertical range between the baseline and mean line. Spaces that are close to the threshold at this stage are made fuzzy, so that a final decision can be made after word recognition.

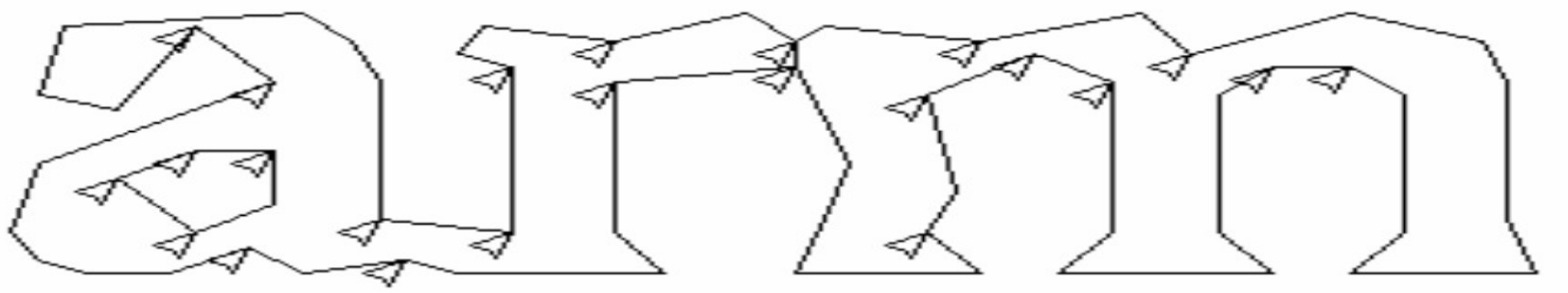
### 3.2.6- **Word Recognition**

Part of the recognition process for any character recognition engine is to identify how a word should be segmented into characters. The initial segmentation output from line finding is classified first. The rest of the word recognition step applies only to non-fixed pitch text.

### 3.2.7- Chopping Joined Characters

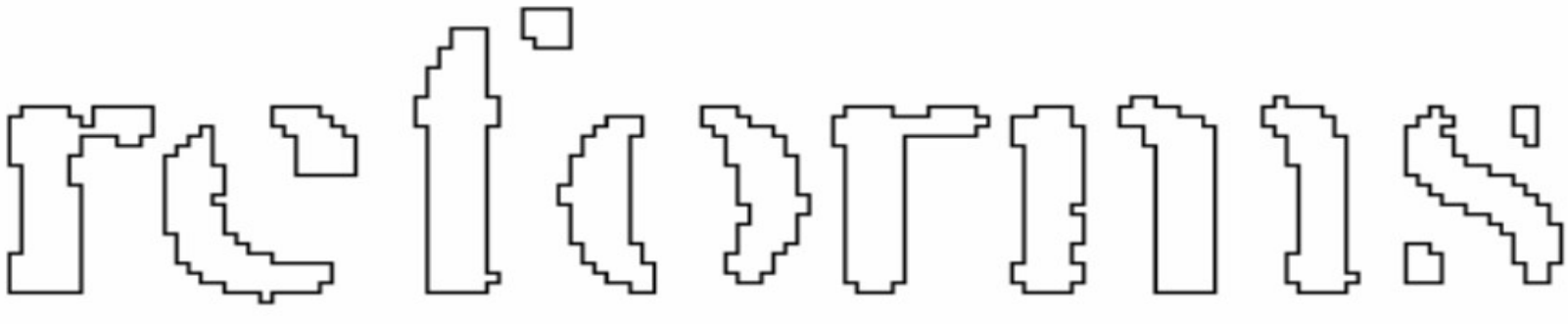
While the result from a word is unsatisfactory, Tesseract attempts to improve the result by chopping the blob with worst confidence from the character classifier. Candidate chop points are found from concave vertices of a polygonal approximation of the outline, and may have either another concave vertex opposite, or a line segment. It may take up to 3 pairs of chop points to successfully separate joined characters from the ASCII set.

Arrows and the selected chop as a line across the outline where the ‘r’ touches the ‘m’. Chops are executed in priority order. Any chop that fails to improve the confidence of the result is undone, but not completely discarded so that the chop can be re-used later by the associator if needed.



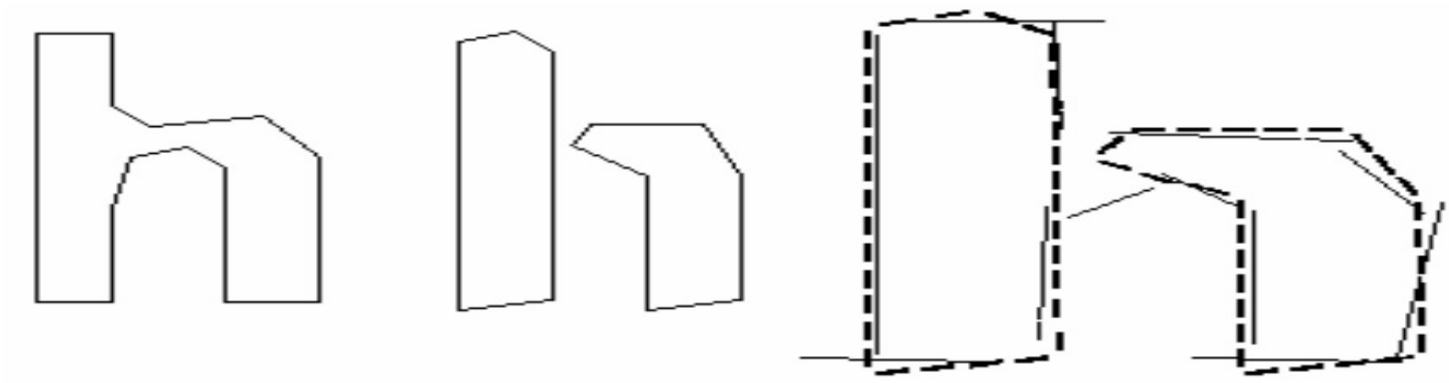
### 3.2.8- Associating Broken Characters

When the potential chops have been exhausted, if the word is still not good enough, it is given to the associator. The associator makes an A* search of the segmentation graph of possible combinations of the maximally chopped blobs into candidate characters. It does this without actually building the segmentation graph, but instead maintains a hash table of visited states. The A* search proceeds by pulling candidate new states from a priority queue and evaluating them by classifying unclassified combinations of fragments. It may be argued that this fully-chop-then-associate approach is at best inefficient, at worst liable to miss important chops, and that may well be the case.The advantage is that the chop-then-associate scheme simplifies the data structures that would be required to maintain the full segmentation graph. When the A* segmentation search was first implemented in about 1989, Tesseract’s accuracy on broken characters was well ahead of the commercial engines of the day.



### 3.2.9- **Static Character Classifier Features**

an early version of Tesseract used topological features developed from the work of Shillman et. al. to the problems found in real-life images, as Bokser describes. An intermediate idea involved the use of segments of the polygonal approximation as features, but this approach is also not robust to damaged characters. The breakthrough solution is the idea that the features in the unknown need not be the same as the features in the training data. During training, the segments of a polygonal approximation are used for features, but in recognition, features of a small, fixed length (in normalized units) are extracted from the outline and matched many-to-one against the clustered prototype features of the training data.



The short, thick lines are the features extracted from the unknown, and the thin, longer lines are the clustered segments of the polygonal approximation that are used as prototypes. One prototype bridging the two pieces is completely unmatched. Three features on one side and two on the other are unmatched, but, apart from those, every prototype and every feature is well matched.This example shows that this process of small features matching large prototypes is easily able to cope with recognition of damaged images. Its main problem is that the computational cost of computing the distance between an unknown and a prototype is very high. The features extracted from the unknown are thus 3- dimensional, (x, y position, angle), with typically 50- 100 features in a character, and the prototype features are 4-dimensional (x, y, position, angle, length), with typically 10-20 features in a prototype configuration.

### 3.2.10- **Classification**

Classification proceeds as a two-step process. In the first step, a class pruner creates a shortlist of character classes that the unknown might match. Each feature fetches, from a coarsely quantized 3-dimensional lookup table, a bit-vector of classes that it might match, and the bit-vectors are summed over all the features. The classes with the highest counts (after correcting for expected number of features) become the short-list for the next step. Each feature of the unknown looks up a bit vector of prototypes of the given class that it might match, and then the actual similarity between them is computed. Each prototype character class is represented by a logical sum-of-product expression with each term called a configuration, so the distance calculation process keeps a record of the total similarity evidence of each feature in each configuration, as well as of each prototype. The best combined distance, which is calculated from the summed feature and prototype evidences, is the best over all the stored configurations of the class.

### 3.2.11-Training Data

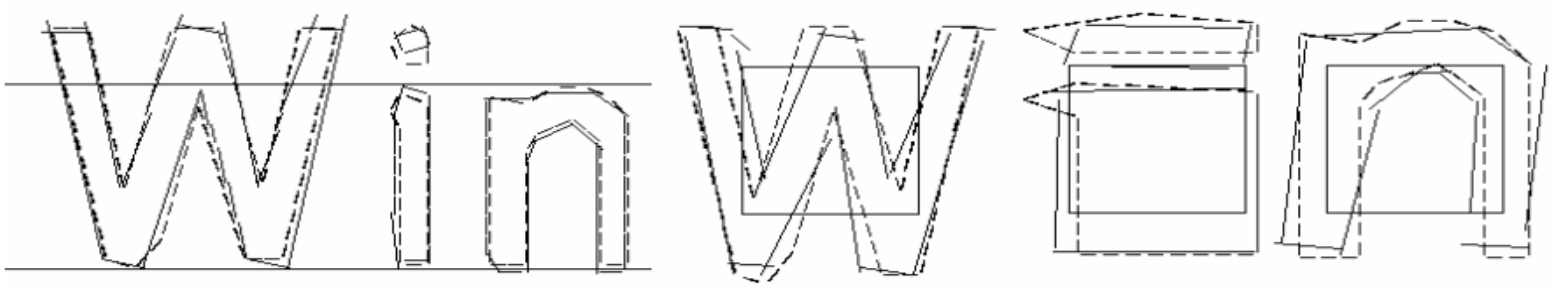
Since the classifier is able to recognize damaged characters easily, the classifier was not trained on damaged characters. In fact, the classifier was trained on a mere 20 samples of 94 characters from 8 fonts in a single size, but with 4 attributes (normal, bold, italic, bold italic), making a total of 60160 training samples. This is a significant contrast to other published classifiers, such as the Calera classifier with more than a million samples, and Baird's 100-font classifier with 1175000 training samples.

### 3.2.12- Linguistic Analysis

Tesseract contains relatively little linguistic analysis. Whenever the word recognition module is considering a new segmentation, the linguistic module (mis-named the permute) chooses the best available word string in each of the following categories: Top frequent word, Top dictionary word, Top numeric word, Top UPPER case word, Top lower case word (with optional initial upper), Top classifier choice word. The final decision for a given segmentation is simply the word with the lowest total distance rating, where each of the above categories is multiplied by a different constant. Words from different segmentations may have different numbers of characters in them. It is hard to compare these words directly, even where a classifier claims to be producing probabilities, which Tesseract does not. This problem is solved in Tesseract by generating two numbers for each character classification. The first, called the confidence, is minus the normalized distance from the prototype. This enables it to be a "confidence" in the sense that greater numbers are better, but still a distance, as, the farther from zero, the greater the distance. The second output, called the rating, multiplies the normalized distance from the prototype by the total outline length in the unknown character. Ratings for characters within a word can be summed meaningfully, since the total outline length for all characters within a word is always the same.

### 3.2.13- Adaptive Classifier

It has been suggested and demonstrated that OCR engines can benefit from the use of an adaptive classifier. Since the static classifier has to be good at generalizing to any kind of font, its ability to discriminate between different characters or between characters and non-characters is weakened. A more font-sensitive adaptive classifier that is trained by the output of the static classifier is therefore commonly used to obtain greater discrimination within each document, where the number of fonts is limited. Tesseract does not employ a template classifier, but uses the same features and classifier as the static classifier. The only significant difference between the static classifier and the adaptive classifier, apart from the training data, is that the adaptive classifier uses isotropic baseline/x-height normalization, whereas the static classifier normalizes characters by the centroid for position and second moments for anisotropic size normalization. The baseline/x-height normalization makes it easier to distinguish upper and lower case characters as well as improving immunity to noise specks. The main benefit of character moment normalization is removal of font aspect ratio and some degree of font stroke width. It also makes recognition of sub and superscripts simpler, but requires an additional classifier feature to distinguish some upper and lower case characters.



## 3.3- OpenCV Machine learning (ML)

ML is a scientific discipline that explores the construction and study of algorithms that can learn from data. Such algorithms operate by building a mode based on inputs and using that to make predictions or decisions, rather than following only explicitly programmed instructions.

Machine learning can be considered a subfield of computer science and statistics. It has strong ties to artificial intelligence and optimization, which deliver methods, theory and application domains to the field. Machine learning is employed in a range of computing tasks where designing and programming explicit, rule-based algorithms is infeasible.

### 3.3.1- K-Means

K-means is a clustering algorithm implemented in the cxcore because it was written long before the ML library. K-means attempts to find the natural clusters or “clumps” in the data. The user sets the desired number of clusters and then K-means rapidly finds a good placement for those cluster centers, where “good” means that the cluster centers tend to end up located in the middle of the natural clumps of data. It is one of the most used clustering techniques and has strong similarities to the expectation maximization algorithm for Gaussian mixture (implemented as CvEM () in the ML library) as well as some similarities to the mean-shift algorithm discussed in K-means is an iterative algorithm and, is also known as Lloyd’s algorithm* or (equivalently) “Voronoi iteration”. the algorithm runs as follows.

1. Twwake as input (a) a data set and (b) desired number of clusters K (chosen by the user).
2. Randomly assign cluster center locations.
3. Associate each data point with its nearest cluster center.
4. Move cluster centers to the centroid of their data points.
5. Return to step 3 until convergence (centroid does not move).

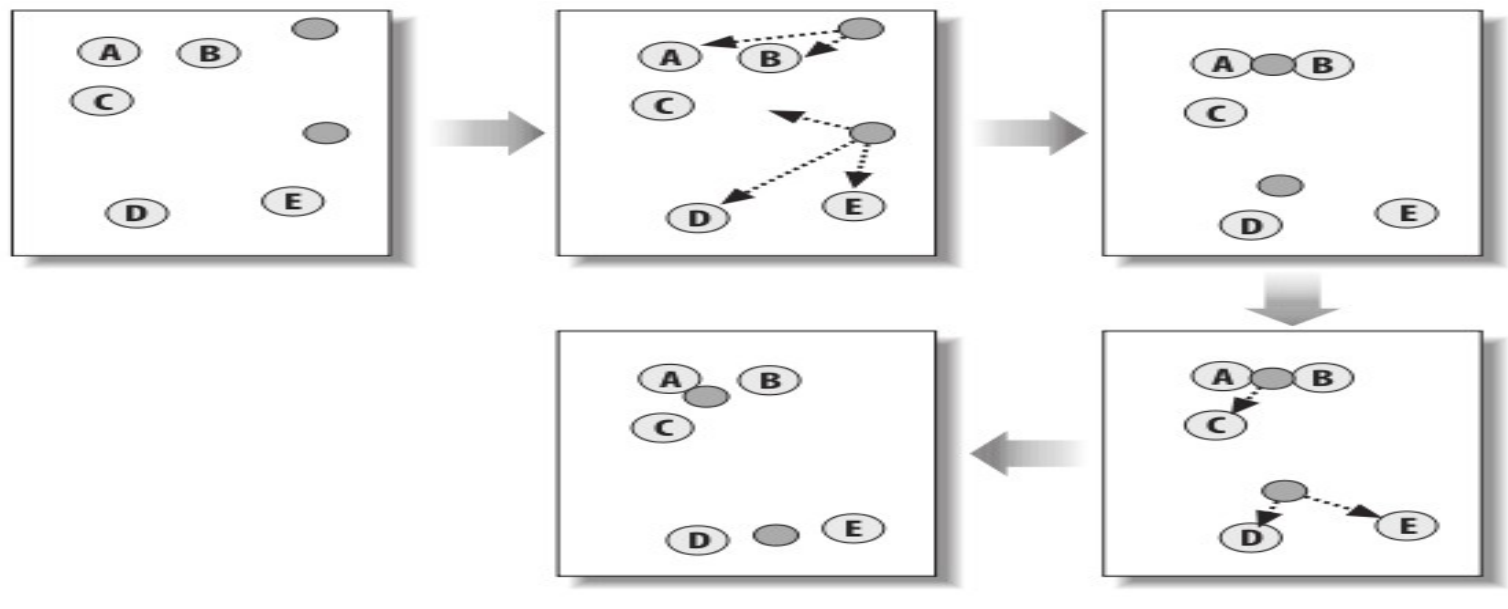
## Problems

K-means is an extremely effective clustering algorithm, but it does have three problems.

1- K-means isn’t guaranteed to find the best possible solution to locating the cluster centers.

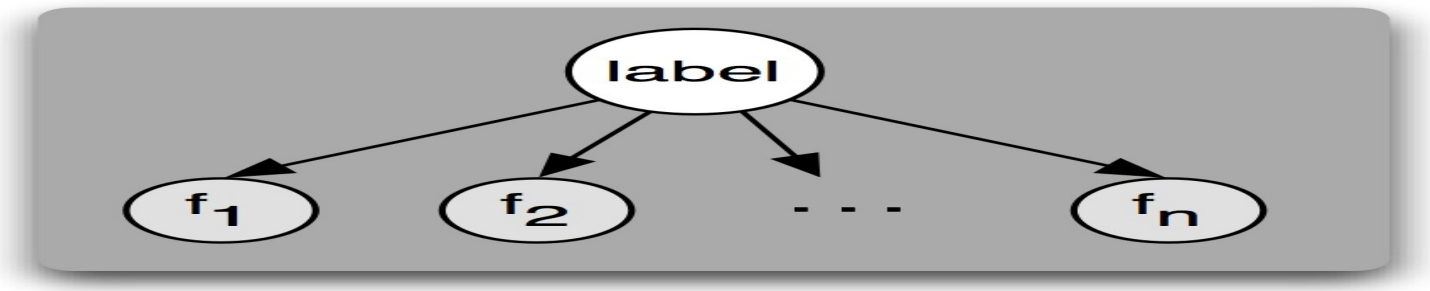
2- K-means does not tell you how many cluster centers you should use. If we had chosen two or four clusters the results would be different and perhaps non intuitive.

3- K-means presumes that the covariance in the space either does not matter or has already been normalized.



### 3.3.2- **Naïve/Normal Bayes Classifier**

the preceding routines are from cxcore. We'll now start discussing the machine learning (ML) library section of OpenCV. We'll begin with OpenCV's simplest supervised classifier, Cv Normal Bayes Classifier, which is called both a normal Bayes classifier and a naïve Bayes classifier. It's "naïve" because it assumes that all the features are independent from one another even though this is seldom the case. Zhang discusses possible reasons for the sometimes surprisingly good performance of this classifier. Naïve Bayes is not used for regression, but it's an effective classifier that can handle multiple classes, not just two. This classifier is the simplest possible case of what is now a large and growing field known as Bayesian networks, or "probabilistic graphical models". Bayesian networks are causal models; the face features in an image are caused by the existence of a face. In use, the face variable is considered a hidden variable and the face features—via image processing operations on the input image—constitute the observed evidence for the existence of a face. We call this a generative model because the face causally generates the face features. Conversely, we might start by assuming the face node is active and then randomly sample what features are probabilistically generated given that face is active. This top-down generation of data with the same statistics as the learned causal model (here, the face) is a useful ability that a purely discriminative model does not possess. For example, one might generate faces for computer graphics display, or a robot might literally "imagine" what it should do next by generating scenes, objects, and interactions.

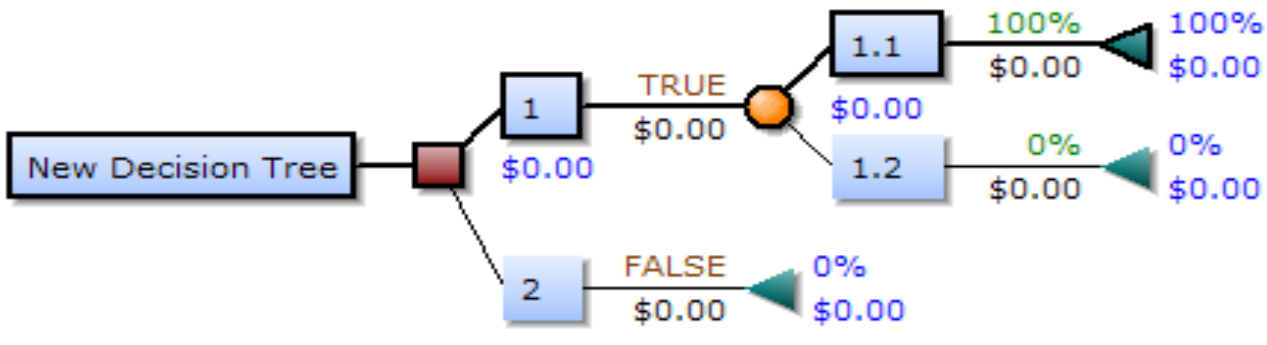



### 3.3.3- **Decision Tree Classification**

is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm.

#### 3.3.3.1- **Decision tree building blocks**

Decision tree elements Drawn from left to right, a decision tree has only burst nodes (splitting paths) but no sink nodes (converging paths). Therefore, used manually, they can grow very big and are then often hard to draw fully by hand. Traditionally, decision trees have been created manually - as the aside example shows - although increasingly, specialized software is employed.

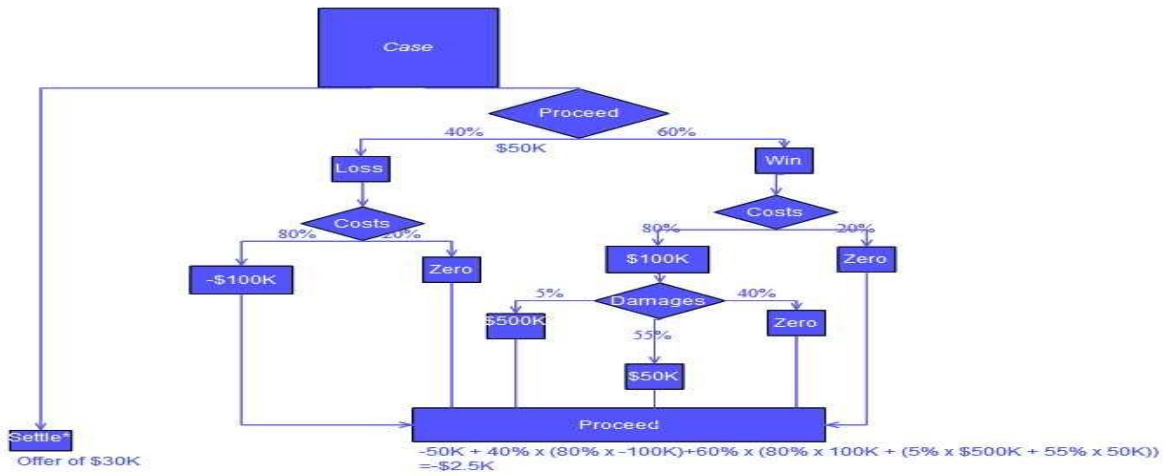


3.3.3.2- **Decision rules**

The decision tree can be linearized into decision rules, where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause. In general, the rules have the form:
If condition1 and condition2 and condition3 then outcome.
Decision rules can also be generated by constructing association rules with the target variable on the right.

**Decision tree using flowchart symbols**

Commonly a decision tree is drawn using flowchart symbols as it is easier for many to read and understand.




3.3.3.3- **Advantages and disadvantages**

among decision support tools, decision trees (and influence diagrams) have several advantages. Decision trees:
Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation. Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.
Allow the addition of new possible scenarios Help determine worst, best and expected values for different scenarios Use a white box model. If a given result is provided by a model.
Can be combined with other decision techniques.
Disadvantages of decision trees:
For data including categorical variables with different number of levels, information gain in decision trees are biased in favor of those attributes with more levels.
Calculations can get very complex particularly if many values are uncertain and/or if many outcomes are linked.

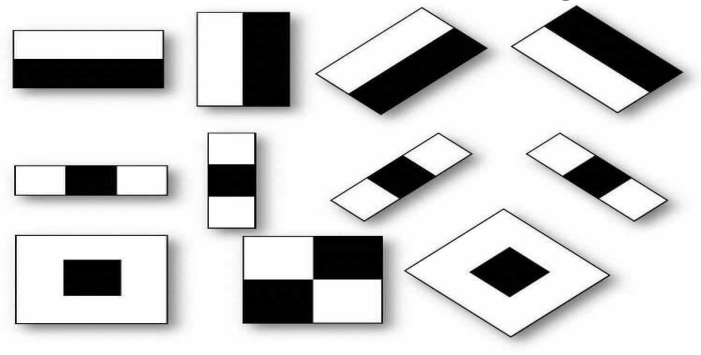
3.3.4- **Face Detection or Haar Classifier**

we now turn to the final tree-based technique in OpenCV: the Haar classifier, which builds a boosted rejection cascade. It has a different format from the rest of the ML library in OpenCV because it was developed earlier as a full-f edged face-recognition application. Thus, we cover it in detail and show how it can be trained to recognize faces and other rigid objects. Computer vision is a broad and fast-changing field, so the parts of OpenCV that implement a specific c technique—rather than a component algorithmic piece—are more at risk of becoming out of date. The face detector that comes with OpenCV is in this “risk” category. However, face detection is such a common need that it is worth having a baseline technique that works fairly well; also, the technique is built on the well-known and of en used field of statistical boosting and thus is of more general use as well. In fact, several companies have engineered the “face” detector in OpenCV to detect “mostly rigid” objects (faces, cars, bikes, human body) by training new detectors on many thousands of selected training images for each view of the object.

This technique has been used to create state-of-the-art detectors, although with a different detector trained for each view or pose of the object. Thus, the Haar classifier is a valuable tool to keep in mind for such recognition tasks.
OpenCV implements a version of the face-detection technique first developed by Paul Viola and Michael Jones—commonly known as the Viola-Jones detector—and later extended by Rainer Lien hart and Jochen Maydt to use diagonal features (more on this distinction to follow). OpenCV refers to this detector as the "Haar classifier" because it uses Haar features or, more precisely, Haar-like wavelets that consist of adding and subtracting rectangular image regions before thresholding the result. OpenCV ships with a set of pretrained object-recognition files, but the code also allows you to train and store new object models for the detector. We note once again that the training (create samples (), haartraining ()) and detecting (cvHaarDetectObjects ()) code works well on any objects (not just faces) that are consistently textured and mostly rigid. The pretrained objects that come with OpenCV for this detector are in …/openCV/data/ haarcascades, where the model that works best for frontal face detection is haarcascade_ frontalface_alt2.xml. Side face views are harder to detect accurately with this technique (as we shall describe shortly), and those shipped models work less well. If you end up training good object models, perhaps you will consider contributing them as open source back to the community

### 3.3.4.1- **Viola-Jones Classifier Theory**

The Viola-Jones classifier employs AdaBoost at each node in the cascade to learn a high detection rate at the cost of low rejection rate multitree (mostly multistump) classifier at each node of the cascade. This algorithm incorporates several innovative features.



1. It uses Haar-like input features: a threshold applied to sums and differences of rectangular image regions.
2. Its *integral image* technique enables rapid computation of the value of rectangular regions or such regions rotated 45 degrees. This data structure is used to accelerate computation of the Haar-like input features.
3. It uses statistical boosting to create binary (face–not face) classification nodes characterized by high detection and weak rejection.4. It organizes the weak classifier nodes of a rejection cascade. In other words: the first group of classifiers is selected that best detects image regions containing an object while allowing many mistaken detections; the next classifier group is the second-best at detection with weak rejection; and so forth. In test mode, an object is detected only if it makes it through the entire cascade. At all scales, these features form the "raw material" that will be used by the boosted classifiers. They are rapidly computed from the integral image representing the original gray scale image.

### 3.3.4.2- **Feature Discussion**

Rectangle features are somewhat primitive when compared with alternatives such as steerable filters, and their relatives, are excellent for the detailed analysis of boundaries, image compression, and texture analysis. In contrast rectangle features, while sensitive to the presence of edges, bars, and other simple image structure, are quite coarse. Unlike steerable filters the only orientations available are vertical, horizontal, and diagonal. The set of rectangle features do however provide a rich image representation which supports effective learning. In conjunction with the integral image, the efficiency of the rectangle feature set provides ample compensation for their limited flexibility

### 3.3.4.3- **Learning Classification Functions**

given a feature set and a training set of positive and negative images,

any number of machine learning approaches could be used to learn a classification function. In our system a variant of AdaBoost is used both to select a small set of features and train the classifier. In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple (sometimes called weak) learning algorithm. There are a number of formal guarantees provided by the AdaBoost learning procedure. Freund and Schapiro proved that the training error of the strong classifier approaches zero exponentially in the number of rounds. More importantly a number of results were later proved about generalization performance. The key insight is that generalization performance is related to the margin of the examples, and that AdaBoost achieves large margins rapidly.Recall that there are over 180,000 rectangle features associated with each image sub-window, a number far larger than the number of pixels. Even though each feature can be computed very efficiently, computing the complete set is prohibitively expensive. Our hypothesis, which is borne out by experiment, is that a very small number of these features can be combined to form an effective classifier. The main challenge is to find these features. In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples (this is similar to the approach of in the domain of image database retrieval). For each feature, the weak learner determines the optimal threshold classification function, such that the minimum numbers of examples are misclassified. A weak classifier % thus consists of a feature, a threshold - and parity!  Here is a 24x24 pixel sub-window of an image. See Table 1 for a summary of the boosting process. In practice no single feature can perform the classification task with low error. Features which are selected in early rounds of the boosting process had error rates between 0.1 and 0.3. Features selected in later rounds, as the task becomes more difficult, yield error rates between 0.4 and 0.5.

3.34.4- **Learning Discussion**

many general feature selection procedures have been proposed. Our final application demanded a very aggressive approach which would discard the vast majority of features. For a similar recognition problem Papa Georgiou et al. proposed a scheme for feature selection based on feature variance. They demonstrated good results selecting 37 features out of a total 1734 features. Roth et al. propose a feature selection process based on the Winnow exponential perceptron learning rule. The Winnow learning process converges to a solution where many of these weights are zero. Nevertheless a very large.

3.3.4.5- **Training a Cascade of Classifiers**

the cascade training process involves two types of tradeoffs. In most cases classifiers with more features will achieve higher detection rates and lower false positive rates. At the same time classifiers with more features require more time to compute. In principle one could define an optimization framework in which:

1) the number of classifier stages,
2) the number of features in each stage, and
3) The threshold of each stage,

are traded off in order to minimize the expected number of evaluated features. Unfortunately finding this optimum is a tremendously difficult problem.

In practice a very simple framework is used to produce an effective classifier which is highly efficient. Each stage in the cascade reduces the false positive rate and decreases the detection rate. A target is selected for the minimum reduction in false positives and the maximum decrease in detection. Each stage is trained by adding features until the target detection and false positives rates are met (these rates are determined by testing the detector on a validation set). Stages are added until the overall target for false positive and detection rate is met.

3.3.4.6- **K-Nearest Neighbors (K-NN)**

one of the simplest classification techniques is K-nearest neighbors (KNN), which merely stores all the training data points. When you want to classify a new point, look up its K nearest points (for K an integer number) and then label the new point according to which set contains the majority of its K neighbors.

This algorithm is implemented in the CvKNearest {} class in OpenCV.
The KNN classification technique can be very effective, but it requires that you store the entire training set; hence it can use a lot of memory and become quite slow. People of en cluster the training set to reduce its size before using this method. Readers interested in how dynamically adaptive nearest neighbor type techniques might be used in the brain (and in machine learning)

### 3.3.5- **Face recognition**

A facial recognition system is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a facial database.It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems

#### 3.3.5.1- **Face detection**

is a computer technology that determines the locations and sizes of human 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.

**To detect faces**

to detect faces in an image (Not recognize it yet) Challenges A picture has 0, 1 or many faces.

Faces are not the same: with spectacles, mustache etc.

Sizes of faces vary a lot. Available in most digital cameras nowadays the simple method

Slide a window across the window and detect faces.

Too slow, pictures have too many pixels. (1280x1024=1.3M pixels)

#### 3.3.5.2- **Viola-Jones detection approach**

Viola and Jones' face detection algorithm the first object detection framework to provide competitive object detection rates in real-time

Implemented in OpenCV

Components

Features

Haar-features

Integral image

Learning

Boosting algorithm

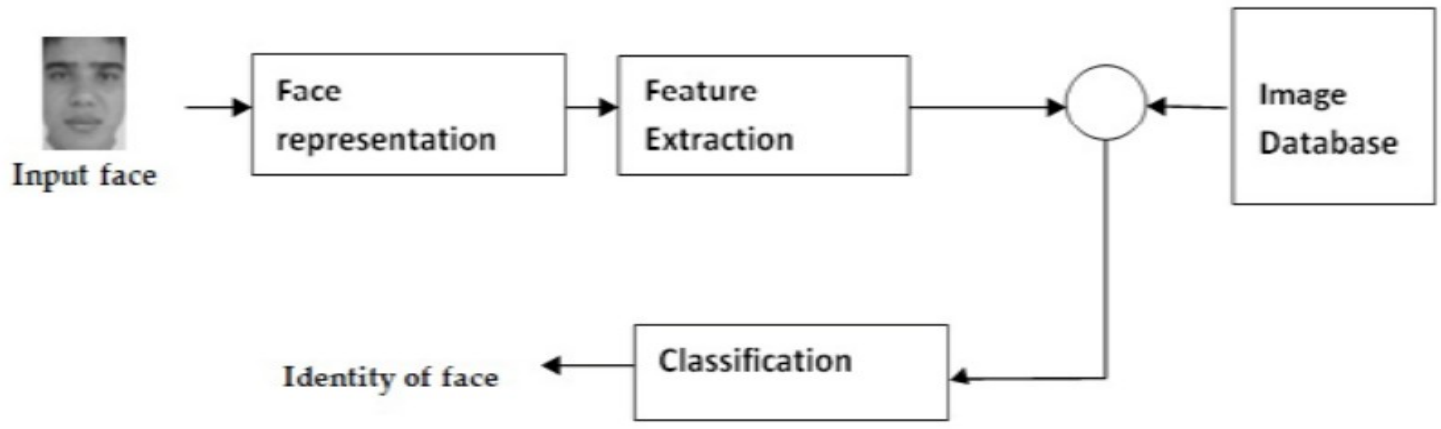
Cascade method

##### 3.3.5.2.1- **AdaBoost**

Boosting algorithms are used to train $T$ weak classifiers $h$, $t \epsilon \{1... T\}$ these classifiers are generally very simple individually. In most cases these classifiers are decision trees with only one split (called *decision stumps*) or at most a few levels of splits (perhaps up to three). Each of the classifiers is assigned a weighted vote $\alpha t$ in the final decision-making process. We use a labeled data set of input feature vectors $xi$, each with scalar label $yi$ (where $i = 1,..., M$ data points). For AdaBoost the label is binary, $yi \in \{-1, +1\}$, though it can be any floating-point number in other algorithms. We initialize a data point weighting distribution $DT(i)$ that tells the algorithm how much misclassifying a data point will "cost". The key feature of boosting is that, as the algorithm progresses, this cost will evolve so that weak classifiers trained later will focus on the data points that the earlier trained weak classifiers tended to do poorly on.

3.3.5.2- **Face recognition**

a facial recognition system is a computer application for automatically identifying and verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a facial database. It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems




**The Paradigm of the Face Recognition**

Despite of the fact that at this moment already numerous of commercial face recognition systems are in use, this way of identification continues to be an interesting topic for researchers. This is due to the fact that the current systems perform well under relatively simple and controlled environments, but perform much worse when variations in different factors are present, such as pose, viewpoint, facial expressions, time (when the pictures are made) and illumination (lightening changes). The goal in this research area is to minimize the influence of these factors and create robust face recognition system. The process of person identification by using face recognition can be split into three main phases. These are face representation, feature extraction and classification Face representation is the first task, that is, how to model a face. The way to represent a face determines the successive algorithms of detection and identification. For the entry-level recognition (that is, to determine whether or not the given image represents a face), the image is transformed (scaled and rotated) till it has the same ‘position’ as the images from the database. In the feature extraction phase, the most useful and unique features (properties) of the face image are extracted. With these obtained features, the face image is compared with the images from the database. This is done in the classification phase. The output of the classification part is the identity of a face image from the database with the highest matching score, thus with the smallest differences compared to the input face image. Also a threshold value can be used to determine if the differences are small enough. After all, it could be that a certain face is not in the database at all.

3.3.6- **Eigen faces (PCA – Principal Component Analysis)**

Eigenfaces refers to an appearance-based approach to face recognition that seeks to capture the variation in a collection of face images and use this information to encode and compare images of individual faces in a holistic (as opposed to a parts-based or feature-based) manner. Specifically, the eigenfaces are the principal components of a distribution of faces, or equivalently, the eigenvectors of the covariance matrix of the set of face images, where an image with *N* pixels is considered a point (or vector) in *N*-dimensional space. The idea of using principal components to represent human faces was developed by Sirovich and Kirby (Sirovich and Kirby 1987) and used by Turk and Pentland (Turk and Pentland 1991) for face detection and recognition. The Eigenface approach is considered by many to be the first working facial recognition technology, and it served as the basis for one of the top commercial face recognition technology products. Since its initial development and publication, there have been many extensions to the original method and many new developments in automatic face recognition systems. Eigenfaces is still often considered as a baseline comparison method to demonstrate the minimum expected performance of such a system.

The motivation of Eigenfaces is twofold:

Extract the relevant facial information, which may or may not be directly related to human intuition of face features such as the eyes, nose, and lips. One way to do so is to capture the statistical variation between face images. Represent face images efficiently. To reduce the computation and space complexity, each face image can be represented using a small number of parameters. The eigenfaces may be considered as a set of features which characterize the global variation among face images. Then each face image is approximated using a subset of the eigenfaces, those associated with the largest eigenvalues. These features account for the most variance in the training set. Pattern recognition in high-dimensional spaces Problems arise when performing recognition in a high-dimensional space (curse of dimensionality).Significant improvements can be achieved by first mapping the data into a lower-dimensional sub-space. The goal of PCA is to reduce the dimensionality of the data while retaining as much information as possible in the original dataset.

**Geometric interpretation**

PCA projects the data along the directions where the data varies the most

These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues

The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions

How to choose K (i.e., number of principal components)?

To choose K, use the following criterion:

In this case, we say that we “preserve” 90% or 95% of the

information in our data.

If K=N, then we “preserve” 100% of the information in our data.

What is the error due to dimensionality reduction?

The original vector x can be reconstructed using its principal components:

PCA minimizes the reconstruction error:

It can be shown that the error is equal to:

original variables as well as on Standardization

The principal components are dependent on the units used to measure the the range of values they assume.

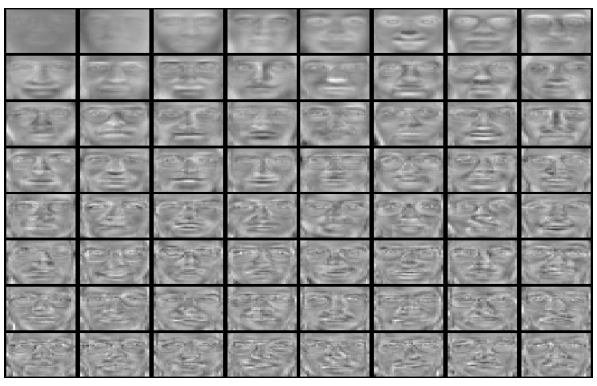
You should always standardize the data prior to using PCA.

A common standardization method is to transform all the data to have zero mean and unit standard

**Training images**

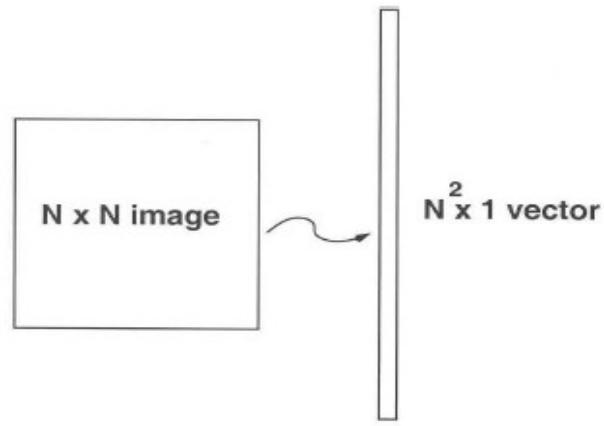
Eigenfaces example

Top eigenvectors: $\mathbf{u}_1 \ldots \mathbf{u}_k$



Mean: $\boldsymbol{\mu}$

Representing faces onto this basis
Face Recognition
The simplest approach is to think of it as a template matching problem
Problems arise when performing recognition in a high-dimensional space.
Significant improvements can be achieved by first mapping the data into a lower dimensionality space.




**Face Recognition Using Eigenfaces**

The distance $e_r$ is called *distance within face space* (*difs*)

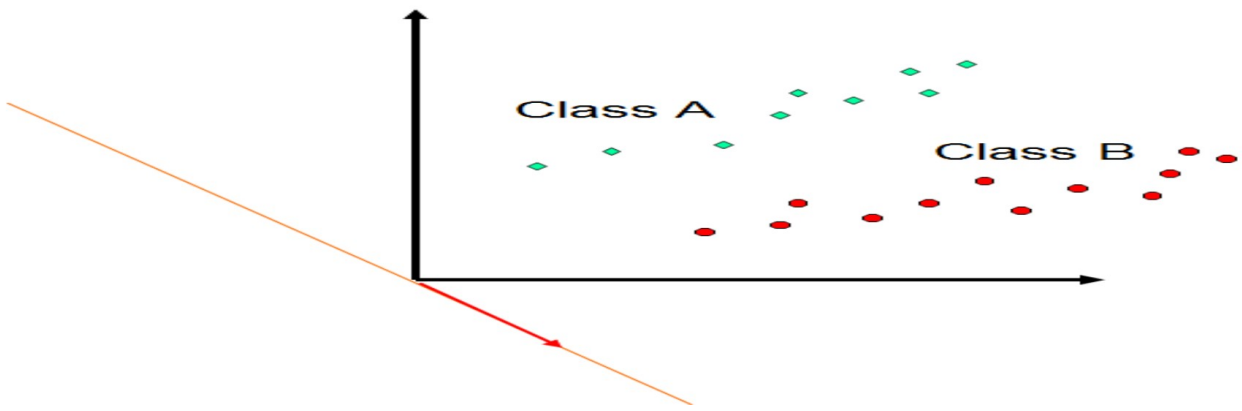
The *Euclidean distance* can be used to compute $e_r$, however, the *Mahalanobis distance* has shown *to* work better:

**Problems with eigenfaces**

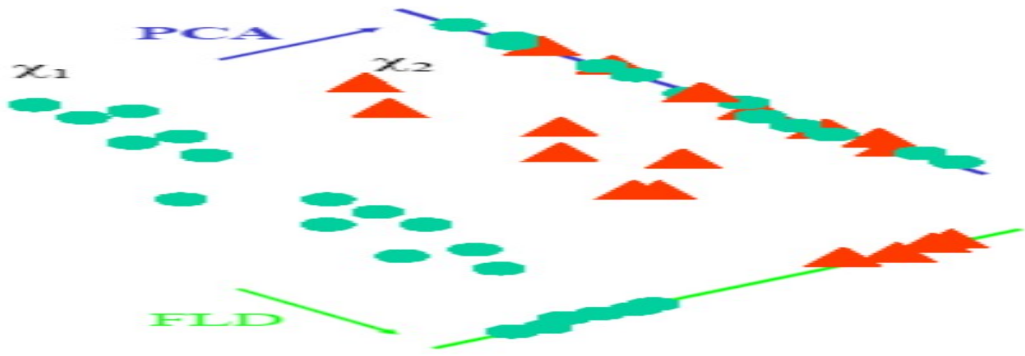
Different head pose
Different alignment
Different facial expression
Background dependent (changes cause problems).
Light changes
Faces must be centered

3.3.7- **Fisherfaces (LDA – Linear Discriminant Analysis)**

Developed in 1997 by P.Belhumeur et al.
Based on Fisher's Linear Discriminant Analysis (LDA)
Faster than eigenfaces, in some cases
Has lower error rates Works well even if different illumination Works well even if different facial express. Constructs a subspace that:
Minimizes the scatter between data points of the same class
Maximizes the scatter between data points of different classes

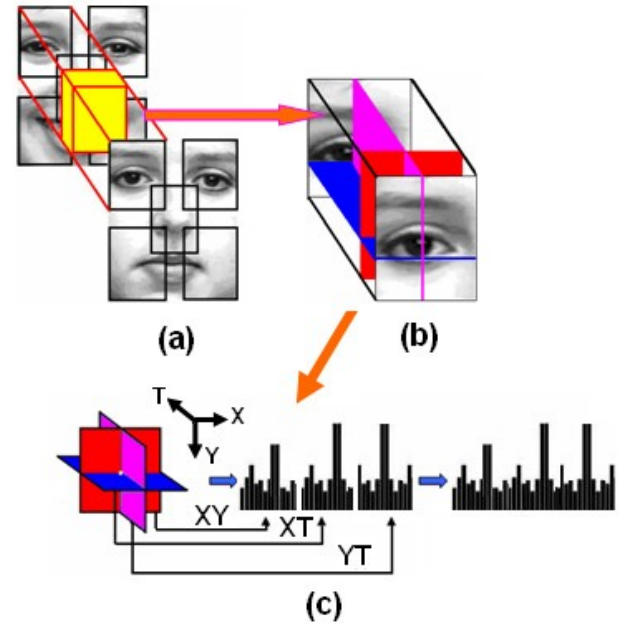


LDA seeks directions that are efficient for discrimination between the data
LDA maximizes the between-class scatter
LDA minimizes the within-class scatter




3.3.8- **Local Binary Patterns (LBP)**

Is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. Perhaps the most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings.




In this work we will focus on descriptors based on Local Binary Patterns (LBP), as they are simple, computationally efficient and have proved to be highly effective features for face recognition. Nonetheless, the methods described in this paper can be readily adapted to operate with alternative local descriptors. Within LBP-based algorithms, most of the face cognition algorithms using LBP follow the approach proposed by Ahonen et al in. In this approach the face image is divided into a grid of small of non-overlapping regions, where a histogram of the LBP for each region is constructed. The similarity of two images is then computed by summing the similarity of histograms from corresponding regions. One drawback of the previous method is that it assumes that a given image region corresponds to the same part of the face in all the faces in the dataset. This is only possible if the face images are fully frontal, scaled, and aligned properly. In addition, while LBP are invariant against monotonic gray scale transformations, they are still affected by illumination Changes that induce non monotonic gray-scale changes such as self-shadowing. In this paper, we propose and compare two algorithms for face recognition that are specially designed to deal with moderate pose variations and misaligned faces. These algorithms are based on previous techniques from the object recognition literature: spatial pyramid matching and Naive Bayes Nearest Neighbors (NBNN). Our main contribution is the inclusion of flexible spatial matching schemes based on an “image-to-class” relation which
Provides an improved robustness with respect to intra-class Variations. These matching schemes use spatially dependent variations of the “bag of words” models with LBP histogram descriptors. As a further refinement,we also incorporate a state of the art illumination compensation algorithm toimprove robustness against illumination changes.

**Face Descriptions with Local Binary Patterns**

The original LBP operator, introduced by Ojala et al., is a powerful means of texture description. The operator labels the pixels of an image by thresholding the 3x3-neighbourhood of each pixel with the center value and considering the Result as a binary number. Then the histogram of the labels can be used as a Texture descriptor. Later the operator was extended to use neighbourhoods of different sizes. Using circular

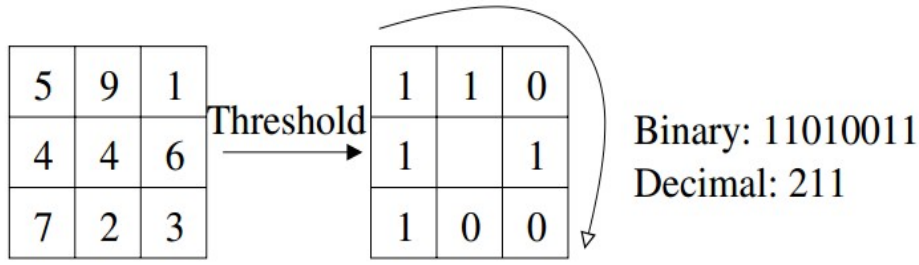



**Fig. 1.** The basic LBP operator.

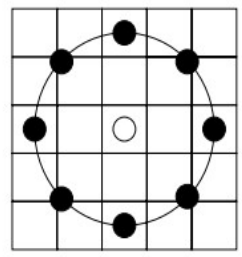


**Fig. 2.** The circular (8,2) neigbourhood. The pixel values are bilinearly interpolated whenever the sampling point is not in the center of a pixel.

neighbourhoods and linearly interpolating the pixel values allow any radius and number of pixels in the neighbourhood. For neighbourhoods

we will use the notation (P, R) which means P sampling points on a circle of radius of R. neighbourhood.

Another extension to the original operator uses so called uniform pattern. A Local Binary Pattern is called uniform if it contains at most two bitwise

Transitions from 0 to 1 or vice versa when the binary string is considered circular. For example, 00000000, 00011110 and 10000011 are uniform patterns. Ojala et al. noticed that in their experiments with texture images, uniform patterns account for a bit less than 90 % of all patterns when using the (8,1) neighbourhood and for around 70 % in the (16,2) neighbourhood. We use the following notation for the LBP operator: $LBP^{u2}_{P,R}$ The subscript represents using the operator in a (P, R) neighbourhood.

Superscript u2 stands for using only uniform patterns and labelling all remaining patterns with a single label.

This histogram contains information about the distribution of the local micro

A histogram of the labeled image $f_l(x, y)$ can be defined as

$$H_i = \sum_{x,y} I\{f_l(x, y) = i\}, i = 0, \ldots, n-1, \quad (1)$$

in which $n$ is the number of different labels produced by the LBP operator and

$$I\{A\} = \begin{cases} 1, & \text{A is true} \\ 0, & \text{A is false.} \end{cases}$$

terns, such as edges, spots and flat areas, over the whole image. For efficient face representation, one should retain also spatial information. For this purpose, the image is divided into regions R0, R1...Rm−1 and the spatially enhanced histogram is defined as In this histogram, we effectively have a description of the face on three different levels of locality: the labels for the histogram contain information about the patterns on a pixel-level, the labels are summed over a small region to produce information on a regional level and the regional histograms are concatenated to build a global description of the face.
From the pattern classification point of view, a usual problem in face recognition is having a plethora of classes and only a few, possibly only one, training sample(s) per class.
For this reason, more sophisticated classifiers are not needed but a nearest-neighbour classifier is used. Several possible dissimilarity measures have been proposed for histograms:

$$H_{i,j} = \sum_{x,y} I\{f_l(x,y) = i\} I\{(x,y) \in R_j\}, i = 0,\ldots,n-1, j = 0,\ldots,m-1. \quad (2)$$

All of these measures can be extended to the spatially enhanced histogram by simply

- Histogram intersection:

$$D(\mathbf{S},\mathbf{M}) = \sum_i \min(S_i, M_i) \quad (3)$$

- Log-likelihood statistic:

$$L(\mathbf{S},\mathbf{M}) = -\sum_i S_i \log M_i \quad (4)$$

- Chi square statistic ($\chi^2$):

$$\chi^2(\mathbf{S},\mathbf{M}) = \sum_i \frac{(S_i - M_i)^2}{S_i + M_i} \quad (5)$$

summing over i and j. When the image has been divided into regions, it can be expected that some of the regions contain more useful information than others in terms of distinguishing between people. For example, eyes seem to be an important cue in human face recognition To take advantage of this, a weight can be set for each region based on the importance of the information it contains. For example, the weighted χ2 statistic becomes

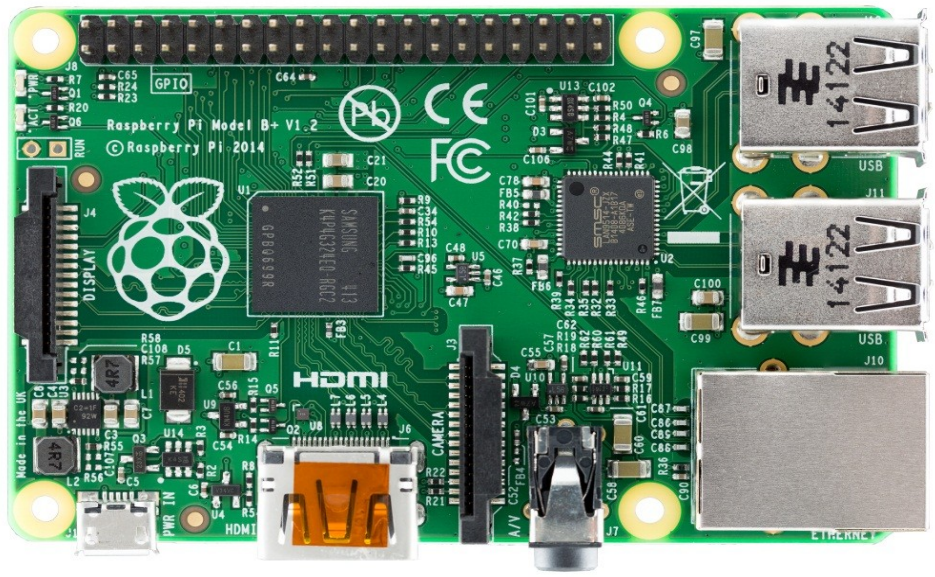
$$\chi^2_w(\mathbf{S},\mathbf{M}) = \sum_{i,j} w_j \frac{(S_{i,j} - M_{i,j})^2}{S_{i,j} + M_{i,j}}, \quad (6)$$

in which $w_j$ is the weight for region $j$.

## 3.4- **Credit-card sized computers**

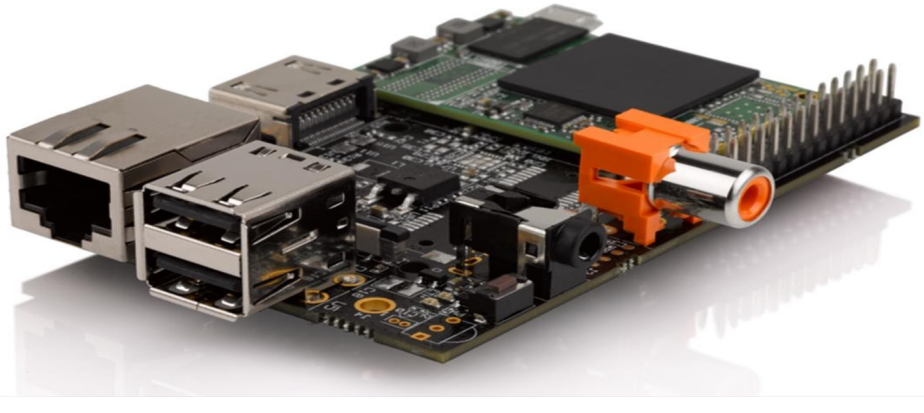
### 3.4.1- **Raspberry pi B+:-**

RAM: 512 MB
Processor: ARM11-76JZF-S
Clock speed: 700MHz
GPU: Broadcam dual core video core IV
@ 250 MHz openGLES 2.0
Price: 35$



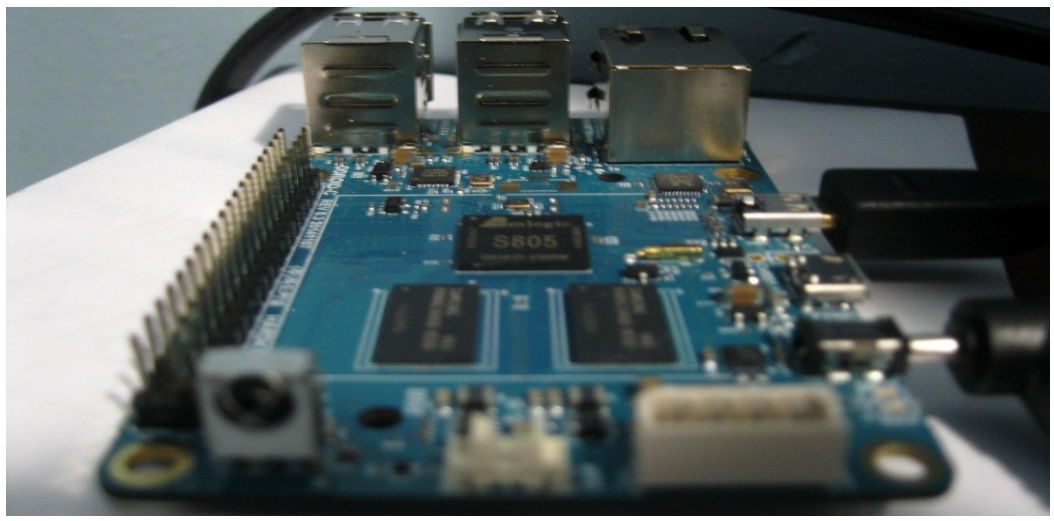

### 3.4.2- **Humming-board-i2ex:-**

RAM: 1GB
Processor: ARM cortex-A9
Clock speed: 1GHz
GPU: Vivante GC 2000 Quad shader
Price: 110$



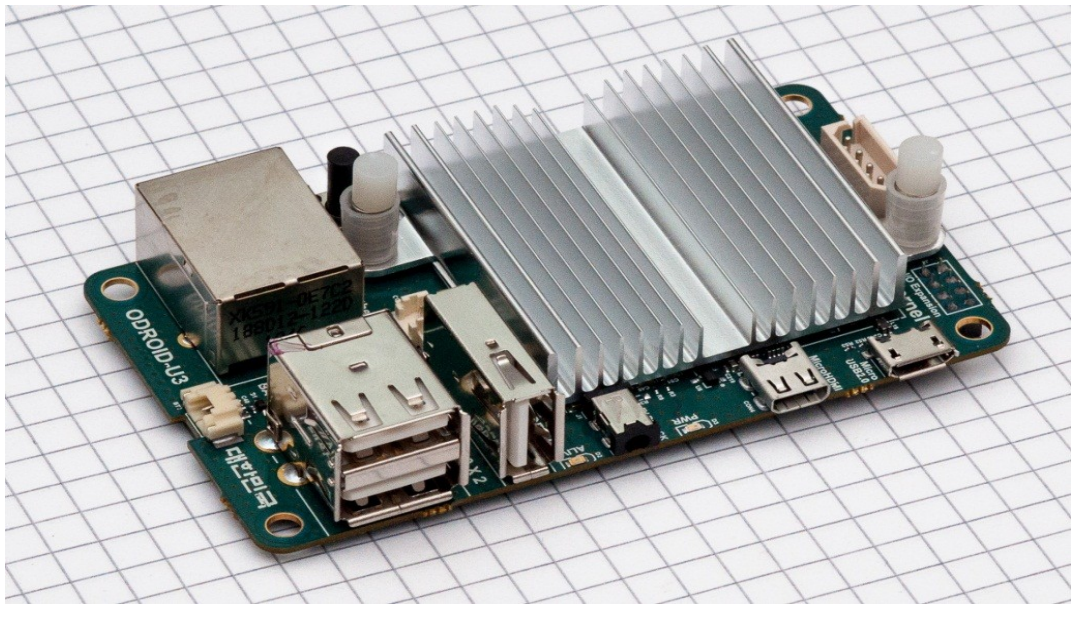
### 3.4.3- **Odroid-C1:-**

RAM: 1GB
Processor: ARM cortex-A5
Clock speed: 1.5GHz
GPU: ARM Mali - 450mp2 GPU@600MHz
OpenGL SE 2.0
Price: 35$



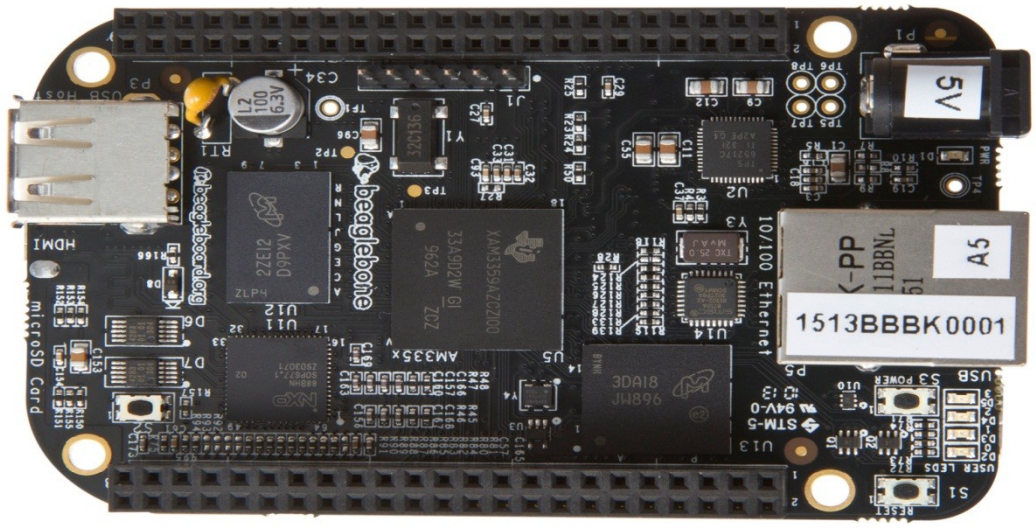

### 3.4.4- **Odroid-U3:-**

RAM: 2GB
Processor: 1.7GHz Quad core
GPU: 4 × ARM Mali 400 @ 400MHz
Price: 60$



3.4.5- **Beagle board black:-**

RAM: 512Mb
Processor: 1GHz TI Sitara AM3359 RM
Cortex-A8
GPU: PowerVR SGX530
Price: 45$




4- **Methodology**

4.1- **Hardware**

4.1.1- **Camera and microphone**

We used C270 Logitech camera board and its built-in microphone

*4.1.2*- **Headphone**

*4.1.3*- **Power bank:-**

PINENG pn 988 10000mAh Power Bank make device working for 10: 12 Hours

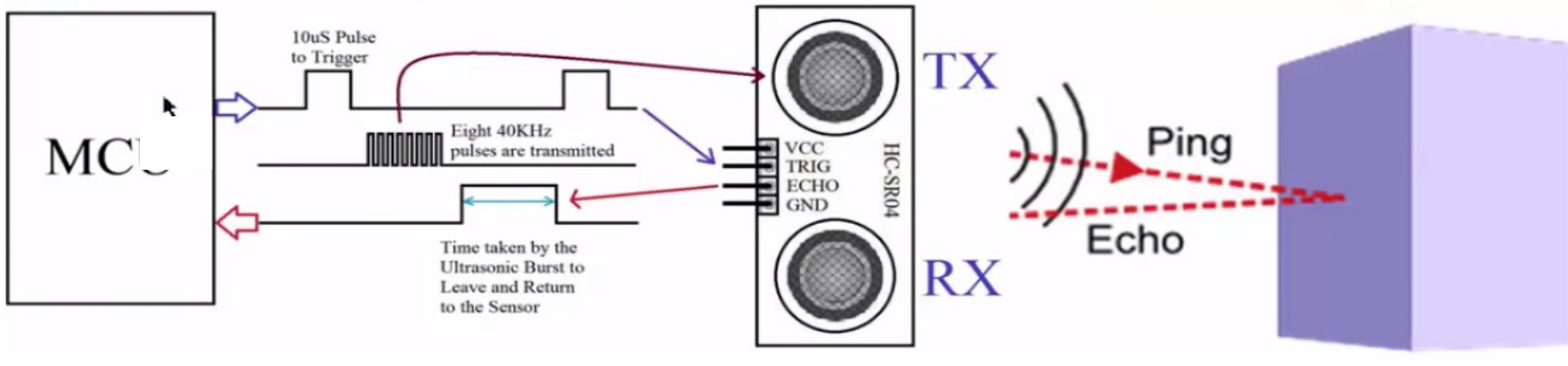
4.1.4- **Odroid-U3**

### 4.1.5- **Arduino:-**

Its open hardware for the development of ideas and projects and is controlled by a program called :- Arduino IDE ( Integrated Development Environment )
Its easy to use and can be used for any person, whether a professional or a beginner
the first arduino was introduced in 2005, aming to provide an inexpensive and easy way for novice and professionals to create devices that interact with their environment using sensors and actuators. The arduino software is easy to use for beginners, yet flexible enough for advanced users. It runs on mac, windows, and linux.

### 4.1.6- **Ultrasonic**

The Ultrasonic Sensor sends out a high-frequency sound pulse and then times how long it takes for the echo of the sound to reflect back.
The sensor has 2 openings on its front. One opening transmits ultrasonic waves the other receives them .It uses the following mathematical equation to calculate the distance:- Distance = Time x Speed of Sound divided by 2
Time = the time between when an ultrasonic wave is transmitted and when it is received , You divide this number by 2 because the sound wave has to travel to the object and back.
The speed of sound is approximately 343 meters per second in air.
Ultrasonic waves are sounds with can not be heard by humans and normally, frequencies of above 20KHZ.
The basic characteristics of ultrasonic waves are:-
1- Reflection
In order to detect the presence of an object, ultrasonic waves are reflected on objects because meral, wood, concrete, glass, rubber and paper, etc. reflect approximately 100% of ultrasonic waves. This objects can be easily detected.
Cloth, cotton, wool, etc. are difficult to detected because thay absorb ultrasonic waves.
It may often be difficult. Also, to detected objects having large surface andulation, because of irregular reflection.
2- Effects of Temperature
varies according to temperature, it is necessary to verify the temperature at all times to measures the distance to the object accurately



### 4.2- Software:-

#### 4.2.1- Text-to-speech (TTS) using eSpeak:

ESpeak is a compact open source software speech synthesizer for English and other languages. Uses a "formant synthesis" method. This allows many languages to be provided in a small size. The speech is clear, and can be used at high speeds, but is not as natural or smooth as larger synthesizers which are based on human speech recordings.

#### 4.2.2-Fswebcam

Fswebcam is a neat and simple webcam app. It captures images from a V4L1/V4L2 compatible device or file, averages them to reduce noise and draws a caption using the GD Graphics Library which also handles compressing the image to PNG or JPEG. The resulting image is saved to a file or sent to stdio where it can be piped to something like ncftpput or SCP.

#### 4.2.3- Arecord

Arecord is a command-line sound file recorder for the ALSA soundcard driver. It supports several file formats and multiple soundcards with multiple devices. If recording with interleaved mode samples the file is automatically split before the 2GB file size.

Aplay is much the same, only it plays instead of recording. For supported sound file formats, the sampling rate, bit depth, and so forth can be automatically determined from the sound file header.

#### 4.2.4- Tesseract-OCR

Tesseract-OCR interest in Latin - based OCR faded away more than a decade ago, in favor of Chinese, Japanese, and Korean (CJK) followed more recently by Arabic, and then Hindi. These languages provide greater challenges specifically to classifiers, and also to the other components of OCR systems. Chinese and Japanese share the Han script, which contains thousands of different character shapes. Korean uses the Hangul script, which has several thousand more of its own, as well as using Han characters. The number of characters is one or two orders of magnitude greater than Latin. Arabic is mostly written with connected characters, and its characters change shape according to the position in a word. Hindi combines a small number of alphabetic letters into thousands of shapes that represent syllables. As the letters combine, they form ligatures whose shape only vaguely resembles the original letters. Hindi Then combines the problems of CJK and Arabic, by joining all the symbols in a word with a line called the shiro-reka. We just used Latin Tesseract-ocr in our device.

### 4.3- OpenCV

It’s a computer vision library which we used to recognize objects and faces.

Camera running using while looping [while (true) {} ] with CvCapture class

Objects Recognition

In objects recognition we used Cascade Classification to detect specific objects then recognize it. We used open source trained data to learn the classifier (wall clocks and cars).

**C++:** CascadeClassifier::CascadeClassifier (const string& **filename**)

### 4.4- Face Recognition

First we should detect face from frame, like objects recognition, we used CascadeClassifier to detect it. But to recognize faces we used FaceRecognizer.

All face recognition models in OpenCV are derived from the abstract base class FaceRecognizer, which provides a unified access to all face recognition algorithms in OpenCV.

## FaceRecognizer

FaceRecognizer::train
Trains a FaceRecognizer with given data and associated labels.
**C++:** void FaceRecognizer::train(InputArrayOfArrays **src**, InputArray **labels**)

Parameters:

. **src** – The training images, that means the faces you want to learn. The data has to be given as a vector<Mat>.

. **labels** – The labels corresponding to the images have to be given either as a vector<int> or a

## FaceRecognizer::update

Updates a FaceRecognizer with given data and associated labels.
**C++:** void FaceRecognizer::update(InputArrayOfArrays **src**, InputArray **labels**)
Parameters:
**. src** – The training images, that means the faces you want to learn. The data has to be given as a vector<Mat>.
**. labels** – The labels corresponding to the images have to be given either as a vector<int> or a.
This method updates a (probably trained) FaceRecognizer, but only if the algorithm supports it. The Local Binary Patterns Histograms (LBPH) recognizer (see createLBPHFaceRecognizer()) can be updated. For the Eigenfaces and Fisherfaces method, this is algorithmically not possible and you have to re-estimate the model with FaceRecognizer::train(). In any case, a call to train empties the existing model and learns a new model, while update does not delete any model data.

## FaceRecognizer::predict

**C++:** void FaceRecognizer::predict(InputArray **src**, int& **label**, double& **confidence**) const = 0
Predicts a label and associated confidence (e.g. distance) for a given input image.
Parameters:
**. src** – Sample image to get a prediction from.
**. label** – The predicted label for the given image.
**. confidence** – Associated confidence (e.g. distance) for the predicted label.
suffix const means that prediction does not affect the internal model state, so the method can be safely called from within different threads.

## FaceRecognizer::save

Saves a FaceRecognizer and its model state.
**C++:** void FaceRecognizer::save(const string& **filename**) const
Saves this model to a given filename, either as XML or YAML.
Parameters:
**filename** – The filename to store this FaceRecognizer to (either XML/YAML).

## FaceRecognizer::load

Loads a FaceRecognizer and its model state.
**C++:** void FaceRecognizer::load(const string& **filename**)
**C++:** void FaceRecognizer::load(const FileStorage& **fs**) = 0
Loads a persisted model and state from a given XML or YAML file. Every FaceRecognizer has to overwrite FaceRecognizer::load(FileStorage& fs) to enable loading the model state. FaceRecognizer::load(FileStorage& fs) in turn gets called by FaceRecognizer::load(const string& filename), to ease saving a model

### 4.5- FaceRecognizer algorithms:-

#### 4.5.1- EigenFace (LDA)

Eigenfaces refers to an appearance-based approach to face recognition that seeks to capture the variation in a collection of face images and use this information to encode and compare images of individual faces in a holistic (as opposed to a parts-based or feature-based) manner. Specifically, the eigenfaces are the principal components of a distribution of faces, or equivalently, the eigenvectors of the covariance matrix of the set of face images, where an image with N pixels is considered a point (or vector) in N-dimensional space. The Eigenface approach is considered by many to be the first working facial recognition technology, and it served as the basis for one of the top commercial face recognition technology products. Since its initial development and publication, there have been many extensions to the original method and many new developments in automatic face recognition systems. Eigenfaces is still often considered as a baseline comparison method to demonstrate the minimum expected performance of such a system.

**Problems with eigenfaces**

Different head pose
Different alignment
Different facial expression
Background dependent (changes cause problems).
Light changes
Faces must be centered

#### 4.5.2- FisherFace (LDA)

The previous algorithm takes advantage of the fact that, under admittedly idealized conditions, the variation within class lies in a linear subspace of the image space. Hence, the classes are convex, and, therefore, linearly separable. One can perform dimensionality reduction using linear projection and still preserve linear separability. This is a strong argument in favor of using linear methods for dimensionality reduction in the face recognition problem, at least when one seeks insensitivity to lighting conditions. Since the learning set is labeled, it makes sense to use this information to build a more reliable method for reducing the dimensionality of the feature space. Here we argue that using class specific linear methods for dimensionality reduction and simple classifiers in the reduced feature space, one may get better recognition rates than with either the Linear Subspace method or the Eigenface method. Fisher’s Linear Discriminant (FLD) is an example of a class specific method, in the sense that it tries to “shape” the scatter in order to make it more reliable for classification.

#### 4.5.3- Local Binary Patterns (LBP)

is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. Perhaps the most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings

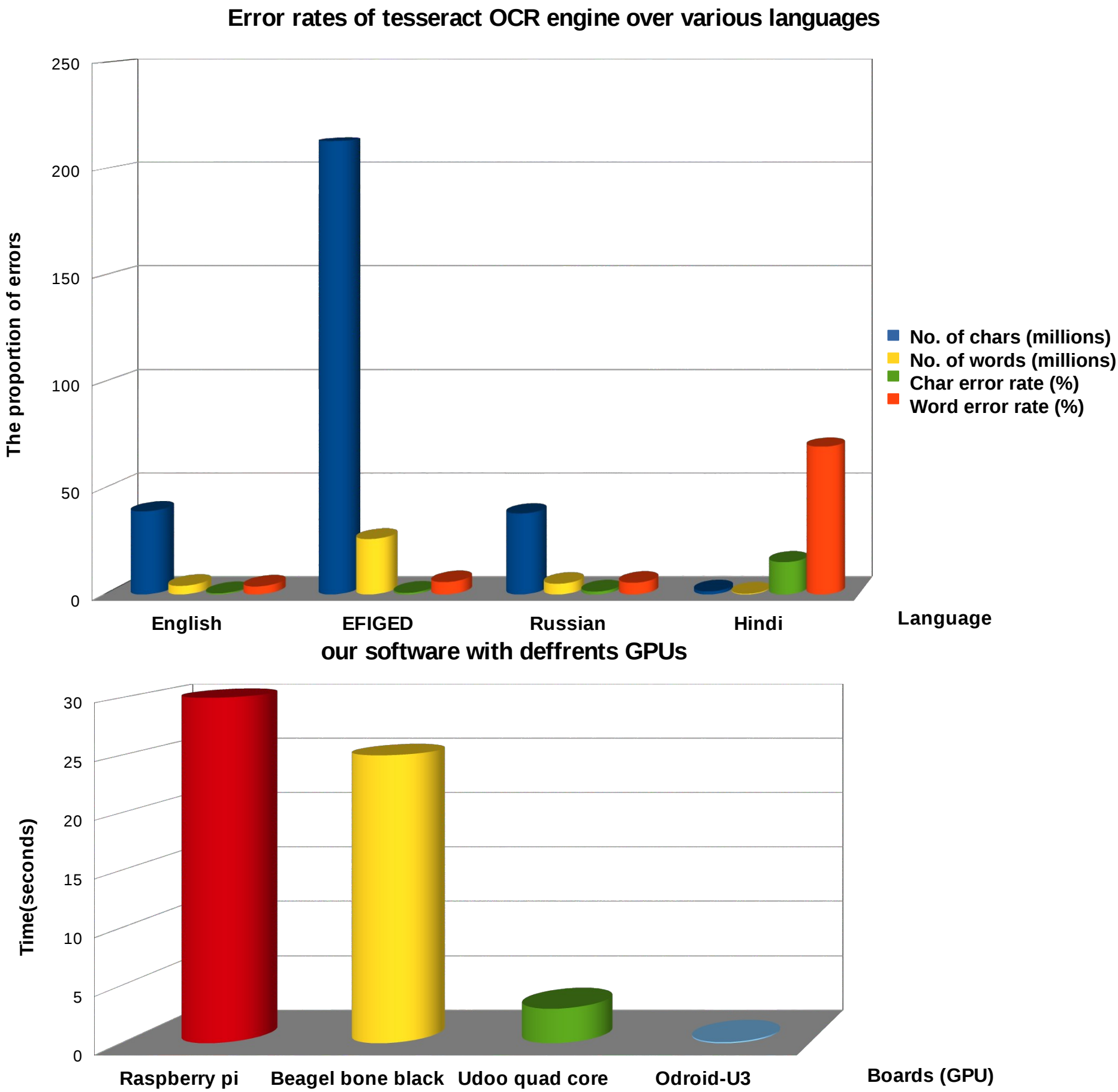
## 5- Conclusion

Our device uses computer vision to enable blind people to recognize faces, some objects and read text. It communicates with the user using voice as TTS and voice recording. The device works for 10: 12 hours constantly to ensure working all the day as minimum. It's small enough to hold or pocket it. It's have a simple controller, consisting of 3 buttons only, to ensure easy using for blind people.

## 6- Future Work

- Train more objects in objects recognition system.
- Use other languages with OCR and Text-To-Speech systems.
- Raise the accuracy of OCR.
- Edit the design.
- Release version of the software which compatible with Google glasses.

## 7- Additional data



8- **The Acknowledgments**

1- Ahmed Sobhy Abdel Fatah
2- Dr.  Saied Mohammed
3- Abdullah Mohammed El hadry
4- Abduelrahman Barghout
5- Abduelrahman sherif

9- **References**

1-Ray Smith, Google Inc, an Overview of the Tesseract OCR Engine
2- Paul Viola, Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, 2001
3- Daniel Maturana, Domingo Mery and Alvaro Soto, SCCC '09 Proceedings of the 2009 International Conference of the Chilean Computer Science Society, Face Recognition with Local Binary Patterns, Spatial Pyramid Histograms and Naive Bayes Nearest Neighbor classification
4-Rainer Lienhart and Jochen Maydt, An Extended Set of Haar-like Features for Rapid Object Detection, Intel Labs, Intel Corporation, 5- Santa Clara, CA 95052, USA VINAY HIREMATH, ASHWINI MAYAKAR, FACE RECOGNITION USING EIGENFACE APPROACH,
6- (M. Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, 1991)
7- SUMAN KUMAR BHATTACHARYYA, KUMAR RAHUL, FACE RECOGNITION BY LINEAR DISCRIMINANT ANALYSIS, Computer Science and Engineering Department, Indian School of Mines, Dhanbad, Jharkhand-826004, India
8- T i m o A h o n e n, Matti Pietika¨ inen, Face Descrip tion with Local Binary Patterns: Applic ation to Face Recog nition, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 28, NO. 12, DECEMBER 2006
9- J. Ashbourn, Avanti, V. Bruce, A. Young, ”Face Recognition Based on Symmetrization and Eigenfaces”
10- P. Belhumeur, J. Hespanha, D. Kriegman, “Eigenfaces vs Fisherfaces: Recognition using Class Specific Linear Projection”
11-Navarrete P. and Ruiz-del-Solar J. (2002), “Interactive Face Retrieval using Self-Organizing Maps”, 2002 Int. Joint Conf. on Neural Networks – IJCNN 2002, May 12-17, Honolulu, USA
“12- A tutorial on Principal Components Analysis”, By Lindsay I Smith
“13-Eigenfaces for Recognition”, Turk, M. and Pentland A., (1991)
Journal of Cognitive Neuroscience, Vol. 3, No. 1, pp. 71-86.
14-Ruiz-del-Solar, J., and Navarrete, P. (2002). “Towards a Generalized Eigenspace-based Face Recognition Framework”, 4th Int. Workshop on Statistical Techniques in Pattern
15- T.Ahonen, A.Hadid & M.Pietikäinen, Face Description with Local Binary Patterns: Application to Face Recognition. Draft, 5th June 2006.
16- T.Ahonen, M.Pietikäinen, A.Hadid & T.Mäenpaä. Face Recognition Based on the Appearance of Local Regions. Machine Vision Group, InfoTech. University of Oulu, Finland. 2004 IEEE.
17- Lindsay I Smith. A tutorial on Principal Components Analysis. February 26, 2002.
18- P.Viola & M.Jones. Rapid Object Detection using a Boosted Cascade of Simple Features, Computer Vision and Pattern Recognition, 2001.
19- M.S.Keil. “I Look in Your Eyes, Honey”: Internal Face Features Induce Spatial Frequency Preference for Human Face Processing. PLoS Computational Biology 5(3): e1000329. doi:10.1371/journal.pcbi.1000329. 2009

20- M.S.Keil. “I Look in Your Eyes, Honey”: Internal Face Features Induce Spatial Frequency Preference for Human Face Processing. PLoS Computational Biology 5(3): e1000329. doi:10.1371/journal.pcbi.1000329. 2009
21- http://opencvlibrary.sourceforge.net: OpenCV Wiki-pages
22- Adrian Kaehler , Gary Bradski, O'Reilly Media, Learning OpenCV
23- H.S. Baird, R. Fossey, “A 100-Font Classifier”, Proc. Of the 1 st Int. Conf. on Document Analysis and Recognition, IEEE, 1991, pp 332-340.
24- Brunelli, R. and Poggio, T. (1993). Face recognition: Features versus templates. IEEE Transactions on Pattern Analysis and Machine Intel ligence, 15(10):1042{1052
25- P.S. Penev and J.J. Atick, “Local Feature Analysis: A General Statistical Theory for Object Representation,” Network-Computation in Neural Systems, vol. 7, no. 3, pp. 477-500, Aug. 1996.
26- T. Ojala, M. Pietika¨inen, and D. Harwood, “A Comparative Study of Texture Measures with Classification Based on Feature Distributions,” Pattern Recognition, vol. 29, no. 1, pp. 51-59, 1996.
27- C. Shan, S. Gong, and P.W. McOwan, “Robust Facial Expression Recognition Using Local Binary Patterns,” Proc. IEEE Int’l Conf. Image Processing, pp. II: 914-917, 2005.
28- Y. Rodriguez and S. Marcel, “Face Authentication Using Adapted Local Binary Pattern Histograms,” Proc. Ninth European Conf. Computer Vision, pp. IV: 321-332, 2006.
29- Lienhart R, Kuranov A, Pisarevsky V (2002) Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. Technical report, Microprocessor Research Lab, Intel Labs
30- R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2003.
31- P.J. Rousseeuw, A.M. Leroy, *Robust Regression and Outlier Detection*, Wiley-IEEE, 2003.
32- Schapire, R.E., “The Strength of Weak Learnability” Machine Learning, 5, 1990, pp 197-227.
33- Kanungo, T., Marton, G.A., Bulbul, O., “Omnipage vs. Sakhr: paired Model Evaluation of Two Arabic OCR Products” Proc. SPIE 3651, 7 Jan 1999, pp109-120.
34- **J.** H. Munson, “The recognition of hand-printed text,” in Proc. IEEE Pattern Recognition Workrhop (Puerto Rico), Oct. 1966.
35- Official Google Blog: http://googleblog.blogspot.com/2008/07/hitting-40- languages.html.

37- Tesseract Open-Source OCR: http://code.google.com/p/tesseract-ocr.
38- P. Belhumeur, J. Hespanha, and D. Kriegman, Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):711--720, 1997.
39- P. Belhumeur, J. Hespanha, and D. Kriegman, Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):711--720, 1997.
40- OpenCV's website – http://www.opencv.org.
41- Espeak -- espeak.sourceforge.net/
42- Stack Overflow -- stackoverflow.com/
43- ODROID | Hardkernel's website -- www.hardkernel.com/
44- ODROID Forum -- forum.odroid.com/


CV
COMPUTER
VISION FOR
BLIND