VEHICLE LICENSE PLATE RECOGNITION USING LAB VIEW

ABSTRACT-

Automatic license plate recognition (ALPR) is the extraction of vehicle license plate information from an image. The system model uses already captured images for this recognition process. First the recognition system starts with character identification based on number plate extraction, Splitting characters and template matching. The proposed model has low complexity and less time consuming in terms of number plate segmentation and character recognition. This can improve the system performance and make the system more efficient by taking relevant samples. at the same time compared their advantages and disadvantages, which provide the basis for license plate recognition. Basically, for the identification of the license plate take character reorganization. And before than that localize the area of the license plate. By using this process we can identify the number plate of the vehicle. These are used for the identification of the vehicle when we lost our vehicle.

Keywords: Character recognition, Region of interest (ROI),Morphological filters

INTRODUCTION

The tasks of managing and using cars well, cracking theft and robbery of motor vehicles, as well as maintaining the normal order of urban transport have become increasingly heavy. Currently, it has become an important issue for the public security department to tom static management into dynamic change management and to tumor manual management into automation. There are urgent needs to employ Intelligent Transportation System (ITS) so as to make effective management. ITS can perform efficient and reliable management to ambient vehicles under various circumstances. As one of the core technologies of ITS, Vehicle Feature Recognition Technology is an important link to police enforcement system, automated highway toll collection system, Urban Traffic Surveillance System and Intelligent Parking Management System, etc. Thus employing image processing technology to recognize the vehicle license plate number of various kinds of vehicles is not only an important issue for information process technology, but also a research issue which is of great importance in modem transportation management.

Objective:

For the detection of license number plate detection using character reorganization. The system model uses already captured images for this recognition process. First the recognition system starts with character identification based on number plate extraction, Splitting characters and template matching. Using this process we can identify the authentication person. And we can track the vehicle also. Using this we can find out our vehicle number plate and get our vehicle. For that purpose we are using morphological filters for adding colour. Take character reorganization and segmentation. Then we can get the output.

Problem definition:

The detection of vehicle is very difficult. Because of authentication person is very intelligent. So, we want to detect that using license plate. the system of vehicle number plate detection and recognition is used to detect the plates then make the recognition of the plate that is to extract the text from an image and all that thanks to the calculation modules that use location algorithms, segmentation plate and character recognition. For this process we are using character and number plate identification.

RELATED WORK:

Security has always been a major concern for mankind. Today we have video surveillance cameras in schools, hospitals and every other public place to make us feel secured. According to a survey by HIS it is estimated that there were around 245 million security cameras installed and functioning back on days, which is like having one security camera for every 30 people on this planet. With the advancement in technology especially in Image processing and Machine Learning, it is possible to make these cameras smarter by training them to process information from the Video feed. License plate recognition systems have been widely used in parking lots. In order to identify license plates easily, traditional license plate recognition systems used in the parking lot have a fixed light source and a shooting angle. For particularly tilting angles, such as license plate images taken with super wide-angle lenses or fisheye lenses, the deformation of the license plate can be particularly severe, resulting in poor recognition of traditional license plate recognition systems. In this paper, we propose a three-stage license plate recognition system based on locallization that can be used for various shooting angles and more oblique images. Experimental results show that the proposed architecture can identify license plates with bevel angles over 0~60 degrees and achieve map rates of up to 91%. Compared with the approach using, the proposed method with character reorganization. has made significant progress in identifying characters that are inclined above 45 degrees.

However, the application of the license plate recognition system faces a completely different situation on the intersection monitor. The license plate recognition system used in the parking lot cannot meet the requirements of intersection monitors, such as various shooting angles, skewed license plates, changes in light and shadow, etc. For example, considering the high resolution wide-angle image in Fig. 1, the proportion of license plates and license plate characters in the image is very small. Furthermore, for particularly tilting angles, such as license plate images taken with super wide-angle lenses or fisheye lenses as shown in Fig. 2, the deformation of the license plate can be particularly severe, resulting in poor recognition of traditional license plate recognition systems.

Literature survey:

1.License Plate Segmentation and Recognition of Chinese Vehicle Based on BPNN

Wang Naiguo,Zhu Xiangwei\*

Suzhou Industrial Park Branch

Jiangsu Union Technical Institute

Suzhou, China

zxwook@163.com \*corresponding author

Zhang Jian

School of computer Science & Technology

Soochow University

Suzhou, China

zhangjian2012@suda.edu.cn

Abstract—In order to make the computer own the knowledge

about Chinese vehicle license plate segmentation and recognition,

the paper put forward a set of algorithms about license plate

segmentation and recognition. The algorithms are divided into

four parts: image preprocessing, license plate location, license

plate segmentation and character recognition. The aim of image

preprocessing is quickly and easily location the license plate, so

the image preprocessing algorithm is one of the important factors

that affect total system performance. Because the algorithm of

license plate location directly affects the accuracy of character

segmentation and character recognition. So, the algorithm of

license plate location is proposed according to characteristics of

Chinese vehicle license plate. The algorithm of license plate

segmentation uses the vertical projection method about license

plate in this paper. According to the license plate segmentation

character, the training model can be generated using tool of

BPNN(back propagation neural network), which is the key of the

character recognition algorithm about license plate. The results

of experiment based the algorithms in the paper illustrate that

accuracy rate of character recognition is very high, and the

algorithms can fully meet the actual demand of automatic

recognition. The algorithms can take advantage of the training

model to perfectly realize recognition the license plate, and have

application value in the real work.

Keywords-image processing; license plate segmentation;

BPNN(back propagation neural network); training model;

license plate character recognition.

2. License Plate Recognition for Moving Vehicles

Case : At Night and Under Rain Condition

Muh Ismail

Department of Informatics

Electrical Engineering Study Program

Hasanuddin University

Makassar, Indonesia

imuh77@gmail.com

Abstract — Research for the detection and identification of

existing license plate focus on static image under different weather

conditions. This research aims to detect and recognize the license

plate on moving vehicle at night and under rain condition. The

method used to detect license plate is Cascade Classifier with Local

Binary Pattern (LBP) as a descriptor, top-hat transform for image

enhancement, and recognizing character of license plate detected

is using Optical Character Recognition (OCR) and Template

Matching (TM). The testing process using 9 videos data with

duration 30 seconds. The testing result that accuracy rates LBP

OCR reach 77.42% increase 32.26% than without top-hat

transform and LBP TM 64.52% increase 45.16% than without

top-hat transform.

Keywords—Video Processing; Image Processing; License Plate

Recognition; License Plate Detection

3. This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS 1

An Iranian License Plate Recognition

System Based on Color Features

Amir Hossein Ashtari, Graduate Student Member, IEEE, Md. Jan Nordin, and Mahmood Fathy

Abstract—In this paper, an Iranian vehicle license plate recognition

system based on a new localization approach, which is

modified to reflect the local context, is proposed, along with a

hybrid classifier that recognizes license plate characters. The

method presented here is based on a modified template-matching

technique by the analysis of target color pixels to detect the

location of a vehicle’s license plate. A modified strip search enables

localization of the standard color-geometric template utilized in

Iran and several European countries. This approach uses periodic

strip search to find the hue of each pixel on demand. In addition,

when a group of target pixels is detected, it is analyzed to verify

that its shape and aspect ratio match those of the standard license

plate. In addition to being scale and rotation invariant, this method

avoids time-consuming image algorithms and transformations for

the whole image pixels, such as resizing and Hough, Fourier, and

wavelet transforms, thereby cutting down the detection response

time. License plate characters are recognized by a hybrid classifier

that comprises a decision tree and a support vector machine with

a homogeneous fifth-degree polynomial kernel. The performance

detection rate and the overall system performance achieved are

96% and 94%, respectively.

Index Terms—Color template matching, image recognition, license

plate detection, license plate localization, license plate number

identification, license plate recognition (LPR).

4. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 9, NO. 3, SEPTEMBER 2008 377

License Plate Recognition From Still Images and

Video Sequences: A Survey

Christos-Nikolaos E. Anagnostopoulos, Member, IEEE, Ioannis E. Anagnostopoulos, Member, IEEE,

Ioannis D. Psoroulas, Vassili Loumos, Member, IEEE, and Eleftherios Kayafas, Member, IEEE

Abstract—License plate recognition (LPR) algorithms in images

or videos are generally composed of the following three processing

steps: 1) extraction of a license plate region; 2) segmentation of

the plate characters; and 3) recognition of each character. This

task is quite challenging due to the diversity of plate formats

and the nonuniform outdoor illumination conditions during image

acquisition. Therefore, most approaches work only under

restricted conditions such as fixed illumination, limited vehicle

speed, designated routes, and stationary backgrounds. Numerous

techniques have been developed for LPR in still images or video

sequences, and the purpose of this paper is to categorize and assess

them. Issues such as processing time, computational power, and

recognition rate are also addressed, when available. Finally, this

paper offers to researchers a link to a public image database to

define a common reference point for LPR algorithmic assessment.

Index Terms—Image processing, license plate identification,

license plate recognition (LPR), license plate segmentation, optical

character recognition (OCR).

EXISTING METHODOLOGY:

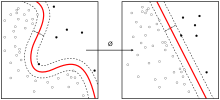
Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular 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.

The basic idea for developing the LBP operator was that two-dimensional surface textures can be described by two complementary measures: local spatial patterns and gray scale contrast. The original LBP operator (Ojala et al. 1996) forms labels for the image pixels by thresholding the 3 x 3 neighborhood of each pixel with the center value and considering the result as a binary number. The histogram of these 28 = 256 different labels can then be used as a texture descriptor. This operator used jointly with a simple local contrast measure provided very good performance in unsupervised texture segmentation (Ojala and Pietikäinen 1999). After this, many related approaches have been developed for texture and color texture segmentation.

The LBP operator was extended to use neighborhoods of different sizes (Ojala et al. 2002). Using a circular neighborhood and bilinearly interpolating values at non-integer pixel coordinates allow any radius and number of pixels in the neighborhood. The gray scale variance of the local neighborhood can be used as the complementary contrast measure. In the following, the notation (P,R) will be used for pixel neighborhoods which means P sampling points on a circle of radius of R. for an example of LBP computation.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

ore formally, a support-vector machine constructs a [hyperplane](https://en.wikipedia.org/wiki/Hyperplane) or set of hyperplanes in a [high-](https://en.wikipedia.org/wiki/High-dimensional_space) or infinite-dimensional space, which can be used for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis), or other tasks like outliers detection.[[3]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-3) Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the [generalization error](https://en.wikipedia.org/wiki/Generalization_error) of the classifier.[[4]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-4)



Kernel machine

Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not [linearly separable](https://en.wikipedia.org/wiki/Linear_separability) in that space. For this reason, it was proposed[[by whom?](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Words_to_watch#Unsupported_attributions)] that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that [dot products](https://en.wikipedia.org/wiki/Dot_product) of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a [kernel function](https://en.wikipedia.org/wiki/Positive-definite_kernel) {\displaystyle k(x,y)} selected to suit the problem.[[5]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-5) The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters  ofimagesof [featurevectors](https://en.wikipedia.org/wiki/Feature_vector) x_{i} that occur in the data base. With this choice of a hyperplane, the points x in the [feature space](https://en.wikipedia.org/wiki/Feature_space) that are mapped into the hyperplane are defined by the relation  Note that if  becomes small as grows further away from , each term in the sum measures the degree of closeness of the test point x to the corresponding data base point  In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points xmapped into any hyper plane can be quite convoluted as a result, allowing much more complex discrimination between sets that are not convex at all in the original space.

INTRODUCTION:

DIGITAL IMAGE PROCESSING

The identification of objects in an image and this process would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures.

The clever bit is to interpret collections of these shapes as single objects, e.g. cars on a road, boxes on a conveyor belt or cancerous cells on a microscope slide. One reason this is an AI problem is that an object can appear very different when viewed from different angles or under different lighting. Another problem is deciding what features belong to what object and which are background or shadows etc. The human visual system performs these tasks mostly unconsciously but a computer requires skilful programming and lots of processing power to approach human performance. Manipulation of data in the form of an image through several possible techniques. An image is usually interpreted as a two-dimensional array of brightness values, and is most familiarly represented by such patterns as those of a photographic print, slide, television screen, or movie screen. An image can be processed optically or digitally with a computer.

1. Basics of Image Processing:-

FUNDAMENTALS OF DIGITAL IMAGE

1.1 IMAGE:

An image is a two-dimensional picture, which has a similar appearance to some subject usually a physical object or a person.

Image is a two-dimensional, such as a photograph, screen display, and as well as a three-dimensional, such as a statue. They may be captured by optical devices—such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.

The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or an abstract painting. In this wider sense, images can also be rendered manually, such as by drawing, painting, carving, rendered automatically by printing or computer graphics technology, or developed by a combination of methods, especially in a pseudo-photograph.

Fig: Colour image to Gray scale Conversion Process

An image is a rectangular grid of pixels. It has a definite height and a definite width counted in pixels. Each pixel is square and has a fixed size on a given display. However different computer monitors may use different sized pixels. The pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and color.

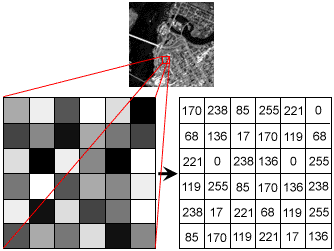


Fig: Gray Scale Image Pixel Value Analysis

Each pixel has a color. The color is a 32-bit integer. The first eight bits determine the redness of the pixel, the next eight bits the greenness, the next eight bits the blueness, and the remaining eight bits the transparency of the pixel.

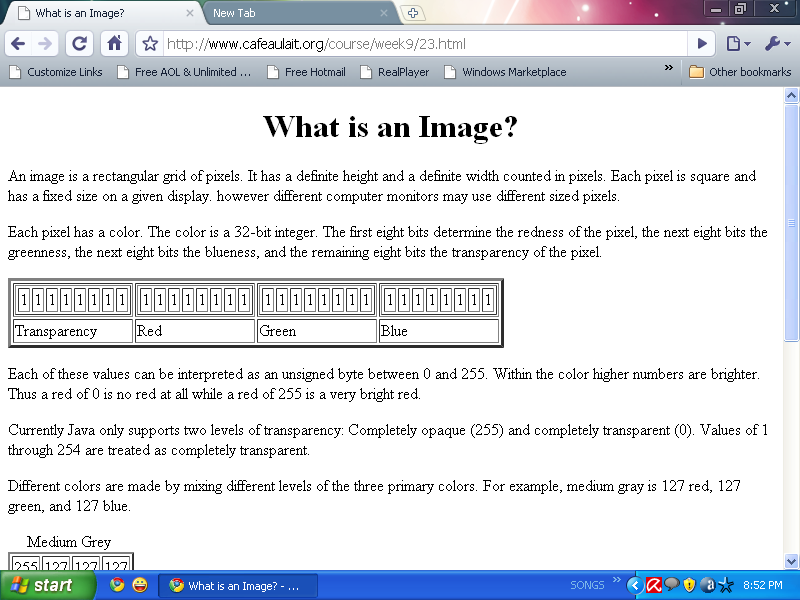


Fig: BIT Transferred for Red, Green and Blue plane (24bit=8bit red;8-bit green;8bit blue)

IMAGE FILE SIZES:

Image file size is expressed as the number of bytes that increases with the number of pixels composing an image, and the color depth of the pixels. The greater the number of rows and columns, the greater the image resolution, and the larger the file. Also, each pixel of an image increases in size when its color depth increases, an 8-bit pixel (1 byte) stores 256 colors, a 24-bit pixel (3 bytes) stores 16 million colors, the latter known as true color.Image compression uses algorithms to decrease the size of a file. High resolution cameras produce large image files, ranging from hundreds of kilobytes to megabytes, per the camera's resolution and the image-storage format capacity. High resolution digital cameras record 12 megapixel (1MP = 1,000,000 pixels / 1 million) images, or more, in true color. For example, an image recorded by a 12 MP camera; since each pixel uses 3 bytes to record true color, the uncompressed image would occupy 36,000,000 bytes of memory, a great amount of digital storage for one image, given that cameras must record and store many images to be practical. Faced with large file sizes, both within the camera and a storage disc, image file formats were developed to store such large images.

IMAGE FILE FORMATS:

Image file formats are standardized means of organizing and storing images. This entry is about digital image formats used to store photographic and other images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. Including proprietary types, there are hundreds of image file types. The PNG, JPEG, and GIF formats are most often used to display images on the Internet.

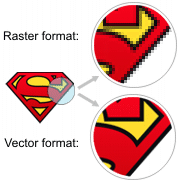


Fig: Horizontal and Vertical Process

In addition to straight image formats, Metafile formats are portable formats which can include both raster and vector information. The metafile format is an intermediate format. Most Windows applications open metafiles and then save them in their own native format.

IMAGE PROCESSING:

Digital image processing, the manipulation of images by computer, is relatively recent development in terms of man’s ancient fascination with visual stimuli. In its short history, it has been applied to practically every type of images with varying degree of success. The inherent subjective appeal of pictorial displays attracts perhaps a disproportionate amount of attention from the scientists and also from the layman. Digital image processing like other glamour fields, suffers from myths, mis-connect ions, mis-understandings and mis-information. It is vast umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology. It is truly multidisciplinary endeavor ploughed with imprecise jargon.

Several factor combine to indicate a lively future for digital image processing. A major factor is the declining cost of computer equipment. Several new technological trends promise to further promote digital image processing. These include parallel processing mode practical by low cost microprocessors, and the use of charge coupled devices (CCDs) for digitizing, storage during processing and display and large low cost of image storage arrays.

FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING:

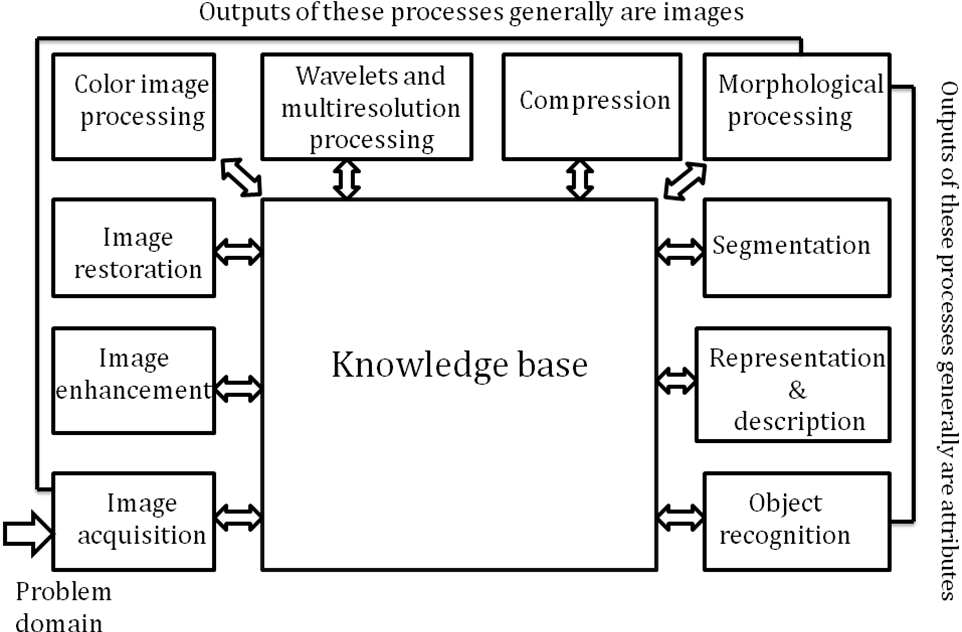
Fig: Basics steps of image Processing

Image Acquisition:

Image Acquisition is to acquire a digital image. To do so requires an image sensor and the capability to digitize the signal produced by the sensor. The sensor could be monochrome or color TV camera that produces an entire image of the problem domain every 1/30 sec. the image sensor could also be line scan camera that produces a single image line at a time. In this case, the objects motion past the line.



Fig: Digital camera

Scanner produces a two-dimensional image. If the output of the camera or other imaging sensor is not in digital form, an analog to digital converter digitizes it. The nature of the sensor and the image it produces are determined by the application.



Fig: Mobile based Camera

Image Enhancement:

Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interesting an image. A familiar example of enhancement is when we increase the contrast of an image because “it looks better.” It is important to keep in mind that enhancement is a very subjective area of image processing.

  
Fig: Image enhancement process for Gray Scale Image and Colour Image using Histogram Bits

1.5.3 Image restoration:

Image restoration is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.



Fig: Noise image🡪 Image Enhancement

Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a “good” enhancement result. For example, contrast stretching is considered an enhancement technique because it is based primarily on the pleasing aspects it might present to the viewer, where as removal of image blur by applying a deblurring function is considered a restoration technique.

Color image processing:

The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray. This second factor is particularly important in manual image analysis.



Fig: gray Scale image 🡪 Colour Image

Segmentation:

Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually.

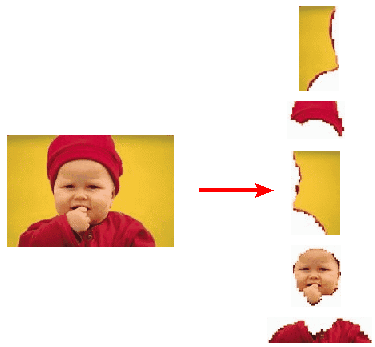


Fig: Image Segment Process

On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely recognition is to succeed.

Digital image is defined as a two dimensional function f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called intensity or grey level of the image at that point. The field of digital image processing refers to processing digital images by means of a digital computer. The digital image is composed of a finite number of elements, each of which has a particular location and value. The elements are referred to as picture elements, image elements, pels, and pixels. Pixel is the term most widely used.

Image Compression

Digital Image compression addresses the problem of reducing the amount of data required to represent a digital image. The underlying basis of the reduction process is removal of redundant data. From the mathematical viewpoint, this amounts to transforming a 2D pixel array into a statically uncorrelated data set. The data redundancy is not an abstract concept but a mathematically quantifiable entity. If n1 and n2 denote the number of information-carrying units in two data sets that represent the same information, the relative data redundancy [2] of the first data set (the one characterized by n1) can be defined as,



Where called as compression ratio [2]. It is defined as



=



In image compression, three basic data redundancies can be identified and exploited: Coding redundancy, interpixel redundancy, and phychovisal redundancy. Image compression is achieved when one or more of these redundancies are reduced or eliminated. The image compression is mainly used for image transmission and storage. Image transmission applications are in broadcast television; remote sensing via satellite, air-craft, radar, or sonar; teleconferencing; computer communications; and facsimile transmission. Image storage is required most commonly for educational and business documents, medical images that arise in computer tomography (CT), magnetic resonance imaging (MRI) and digital radiology, motion pictures, satellite images, weather maps, geological surveys, and so on.

Image Compression Model

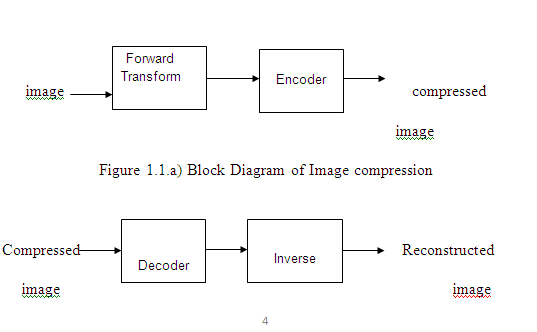


Fig:1.1b) Decompression Process for Image

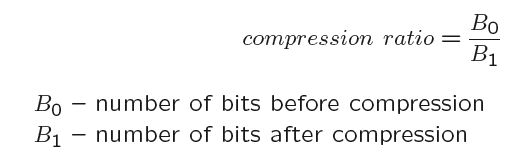
Image Compression Types

There are two types’ image compression techniques.

1. Lossy Image compression

2. Lossless Image compression

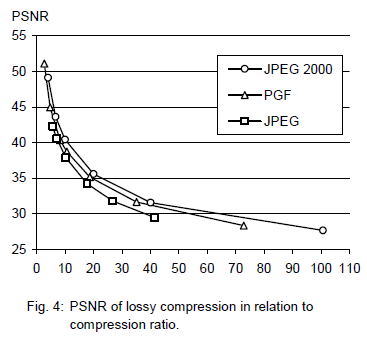
Compression ratio:



1. Lossy Image compression :

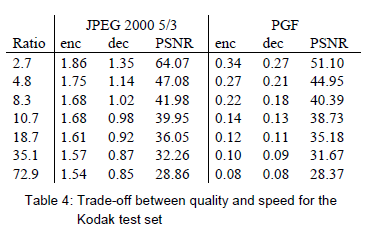
Lossy compression provides higher levels of data reduction but result in a less than perfect reproduction of the original image. It provides high compression ratio. lossy image compression is useful in applications such as broadcast television, videoconferencing, and facsimile transmission, in which a certain amount of error is an acceptable trade-off for increased compression performance. Originally, PGF has been designed to quickly and progressively decode lossy compressed aerial images. A lossy compression mode has been preferred, because in an application like a terrain explorer texture data (e.g., aerial orthophotos) is usually mid-mapped filtered and therefore lossy mapped onto the terrain surface. In addition, decoding lossy compressed images is usually faster than decoding lossless compressed images.

In the next test series we evaluate the lossy compression efficiency of PGF. One of the best competitors in this area is for sure JPEG 2000. Since JPEG 2000 has two different filters, we used the one with the better trade-off between compression efficiency and runtime. On our machine the 5/3 filter set has a better trade-off than the other. However, JPEG 2000 has in both cases a remarkable good compression efficiency for very high compression ratios but also a very poor encoding and decoding speed. The other competitor is JPEG. JPEG is one of the most popular image file formats.

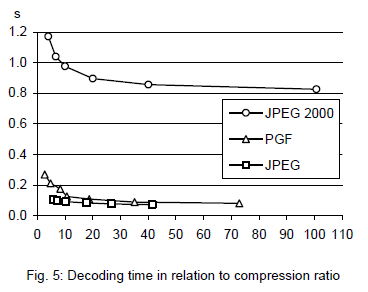


It is very fast and has a reasonably good compression efficiency for a wide range of compression ratios. The drawbacks of JPEG are the missing lossless compression and the often missing progressive decoding. Fig. 4 depicts the average rate-distortion behavior for the images in the Kodak test set when fixed (i.e., nonprogressive) lossy compression is used. The PSNR of PGF is on average 3% smaller than the PSNR of JPEG 2000, but 3% better than JPEG.

These results are also qualitative valid for our PGF test set and they are characteristic for aerial ortho-photos and natural images. Because of the design of PGF we already know that PGF does not reach the compression efficiency of JPEG 2000. However, we are interested in the trade-off between compression efficiency and runtime. To report this trade-off we show in Table 4 a comparison between JPEG 2000 and PGF and in Fig. 5 (on page 8) we show for the same test series as in Fig. 4 the corresponding average decoding times in relation to compression ratios.Table 4 contains for seven different compression ratios (mean values over the compression ratios of the eight images of the Kodak test set) the corresponding average encoding and decoding times in relation to the average PSNR values. In case of PGF the encoding time is always slightly longer than the corresponding decoding time. The reason for that is that the actual encoding phase (cf. Subsection 2.4.2) takes slightly longer than the corresponding decoding phase. For six of seven ratios the PSNR difference between JPEG 2000 and PGF is within 3% of the PSNR of JPEG 2000. Only in the first row is the difference larger (21%), but because a PSNR of 50 corresponds to an almost perfect image quality the large PSNR difference corresponds with an almost undiscoverable visual difference. The price they pay in JPEG 2000 for the 3% more PSNR is very high. The creation of a PGF is five to twenty times faster than the creation of a corresponding JPEG 2000 file, and the decoding of the created PGF is still five to ten times faster than the decoding of the JPEG 2000 file. This gain in speed is remarkable, especially in areas where time is more important than quality, maybe for instance in real-time computation.



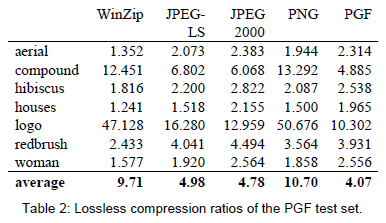
In Fig. 5 we see that the price we pay in PGF for the 3% more PSNR than JPEG is low: for small compression ratios (< 9) decoding in PGF takes two times longer than JPEG and for higher compression ratios (> 30) it takes only ten percent longer than JPEG. These test results are characteristic for both natural images and aerial ortho-photos. Again, in the third test series we only use the ‘Lena’ image. We run our lossy coder with six different quantization parameters and measure the PSNR in relation to the resulting compression ratios. The results (ratio: PSNR) are:



2.Lossless Image compression :

Lossless Image compression is the only acceptable amount of data reduction. It provides low compression ratio while compared to lossy. In Lossless Image compression techniques are composed of two relatively independent operations: (1) devising an alternative representation of the image in which its interpixel redundancies are reduced and (2) coding the representation to eliminate coding redundancies.

Lossless Image compression is useful in applications such as medical imaginary, business documents and satellite images.Table 2 summarizes the lossless compression efficiency and Table 3 the coding times of the PGF test set. For WinZip we only provide average runtime values, because of missing source code we have to use an interactive testing procedure with runtimes measured by hand. All other values are measured in batch mode.



In Table 2 it can be seen that in almost all cases the best compression ratio is obtained by JPEG 2000, followed by PGF, JPEG-LS, and PNG. This result is different to the result in [SEA+00], where the best performance for a similar test set has been reported for JPEG-LS. PGF performs between 0.5% (woman) and 21.3% (logo) worse than JPEG 2000. On average it is almost 15% worse. The two exceptions to the general trend are the ‘compound’ and the ‘logo’ images. Both images contain for the most part black text on a white background. For this type of images, JPEG-LS and in particular WinZip and PNG provide much larger compression ratios. However, in average PNG performs the best, which is also reported in [SEA+00].

These results show, that as far as lossless compression is concerned, PGF performs reasonably well on natural and aerial images. In specific types of images such as ‘compound’ and ‘logo’ PGF is outperformed by far in PNG.

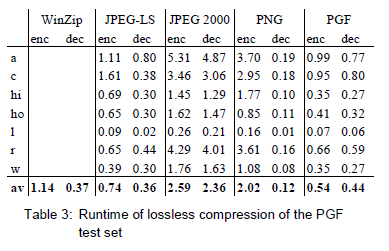


Table 3 shows the encoding (enc) and decoding (dec) times (measured in seconds) for the same algorithms and images as in Table 2. JPEG 2000 and PGF are both symmetric algorithms, while WinZip, JPEG-LS and in particular PNG are asymmetric with a clearly shorter decoding than encoding time. JPEG 2000, the slowest in encoding and decoding, takes more than four times longer than PGF. This speed gain is due to the simpler coding phase of PGF. JPEG-LS is slightly slower than PGF during encoding, but slightly faster in decoding images.

WinZip and PNG decode even more faster than JPEG-LS, but their encoding times are also worse. PGF seems to be the best compromise between encoding and decoding times.

Our PGF test set clearly shows that PGF in lossless mode is best suited for natural images and aerial ortho photos. PGF is the only algorithm that encodes the three Mega Byte large aerial ortho photo in less than second without a real loss of compression efficiency. For this particular image the efficiency loss is less than three percent compared to the best. These results should be underlined with our second test set, the Kodak test set.

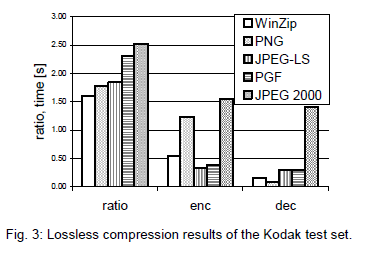


Fig. 3 shows the averages of the compression ratios (ratio), encoding (enc), and decoding (dec) times over all eight images. JPEG 2000 shows in this test set the best compression efficiency followed by PGF, JPEG-LS, PNG, and WinZip. In average PGF is eight percent worse than JPEG 2000. The fact that JPEG 2000 has a better lossless compression ratio than PGF does not surprise,

because JPEG 2000 is more quality driven than PGF.

However, it is remarkable that PGF is clearly better than JPEG-LS (+21%) and PNG (+23%) for natural images. JPEG-LS shows in the Kodak test set also a symmetric encoding and decoding time behaviour. It is encoding and decoding times are almost equal to PGF. Only PNG and WinZip can faster decode than PGF, but they also take longer than PGF to encode.

If both compression efficiency and runtime is important, then PGF is clearly the best of the tested algorithms for lossless compression of natural images and aerial ortho photos. In the third test we perform our lossless coder on the ‘Lena’ image.

To digitally process an image, it is first necessary to reduce the image to a series of numbers that can be manipulated by the computer. Each number representing the brightness value of the image at a particular location is called a picture element, or pixel. A typical digitized image may have 512 × 512 or roughly 250,000 pixels, although much larger images are becoming common. Once the image has been digitized, there are three basic operations that can be performed on it in the computer. For a point operation, a pixel value in the output image depends on a single pixel value in the input image. For local operations, several neighbouring pixels in the input image determine the value of an output image pixel. In a global operation, all of the input image pixels contribute to an output image pixel value.

Correspondingly, these combinations attempt to strike a winning tradeoff: be flexible and hence bring tolerance toward intraclass variation, while also being discriminative enough to be robust to background clutter and interclass similarity. An important feature of our contour-based recognition approach is that it affords us substantial flexibility to incorporate additional image information. Specifically, we extend the contour-based recognition method and propose a new hybrid recognition method which exploits shape tokens and SIFT features as recognition cues. Shape-tokens and SIFT features are largely orthogonal, where the former corresponds to shape boundaries and the latter to sparse salient image patches. Here, each learned combination can comprise features that are either 1) purely shape-tokens, 2) purely SIFT features, or 3) a mixture of shape-tokens and SIFT features. The number and types of features to be combined together are learned automatically from training images, and represent the more discriminative ones based on the training set. Consequently, by imparting these two degrees of variability (in both the number and the types of features) to a combination, we empower it with even greater flexibility and discriminative potential. A shorter version of this paper appeared in [9].

CLASSIFICATION OF IMAGES:

There are 3 types of images used in Digital Image Processing. They are

1. Binary Image
2. Gray Scale Image
3. Colour Image

BINARY IMAGE:

A binary image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that has only two possible values for each [pixel](http://en.wikipedia.org/wiki/Pixel).  Typically the two colors used for a binary image are black and white though any two colors can be used.  The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1).This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as [grayscale images](http://en.wikipedia.org/wiki/Grayscale)

Binary images often arise in [digital image processing](http://en.wikipedia.org/wiki/Digital_image_processing) as [masks](http://en.wikipedia.org/w/index.php?title=Mask_(image_processing)&action=edit&redlink=1) or as the result of certain operations such as [segmentation](http://en.wikipedia.org/wiki/Segmentation_(image_processing)), [thresholding](http://en.wikipedia.org/wiki/Thresholding_(image_processing)), and [dithering](http://en.wikipedia.org/wiki/Dither). Some input/output devices, such as [laser printers](http://en.wikipedia.org/wiki/Laser_printer), [fax machines](http://en.wikipedia.org/wiki/Fax), and bi-level [computer displays](http://en.wikipedia.org/wiki/Visual_display_unit), can only handle bi-level images

GRAY SCALE IMAGE

A grayscale Image is [digital image](http://en.wikipedia.org/wiki/Digital_image) is an image in which the value of each [pixel](http://en.wikipedia.org/wiki/Pixel) is a single [sample](http://en.wikipedia.org/wiki/Sample_(signal)), that is, it carries only [intensity](http://en.wikipedia.org/wiki/Luminous_intensity) information. Images of this sort, also known as [black-and-white](http://en.wikipedia.org/wiki/Black-and-white), are composed exclusively of shades of [gray](http://en.wikipedia.org/wiki/Gray)(0-255), varying from black(0) at the weakest intensity to white(255) at the strongest.

Grayscale images are distinct from one-bit [black-and-white](http://en.wikipedia.org/wiki/Black-and-white) images, which in the context of computer imaging are images with only the two [colors](http://en.wikipedia.org/wiki/Color), [black](http://en.wikipedia.org/wiki/Black), and [white](http://en.wikipedia.org/wiki/White) (also called bi-level or [binary images](http://en.wikipedia.org/wiki/Binary_image)). Grayscale images have many shades of gray in between. Grayscale images are also called [monochromatic](http://en.wikipedia.org/wiki/Monochromatic), denoting the absence of any [chromatic](http://en.wikipedia.org/wiki/Chromaticity) variation.

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the [electromagnetic spectrum](http://en.wikipedia.org/wiki/Electromagnetic_spectrum) (e.g. [infrared](http://en.wikipedia.org/wiki/Infrared), [visible light](http://en.wikipedia.org/wiki/Visible_spectrum), [ultraviolet](http://en.wikipedia.org/wiki/Ultraviolet), etc.), and in such cases they are monochromatic proper when only a given [frequency](http://en.wikipedia.org/wiki/Frequency) is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale.

COLOUR IMAGE:

A (digital) color image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that includes [color](http://en.wikipedia.org/wiki/Color) information for each [pixel](http://en.wikipedia.org/wiki/Pixel). Each pixel has a particular value which determines its appearing color. This value is qualified by three numbers giving the decomposition of the color in the three primary colors Red, Green and Blue. Any color visible to human eye can be represented this way. The decomposition of a color in the three primary colors is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0); and say, bright pink will be : (255,0,255).

In other words, an image is an enormous two-dimensional array of color values, pixels, each of them coded on 3 bytes, representing the three primary colors. This allows the image to contain a total of 256x256x256 = 16.8 million different colors. This technique is also known as RGB encoding, and is specifically adapted to human vision

|  |
| --- |
| http://images.gamedev.net/features/programming/imageproc/image004.gif |

Fig.1 Hue Saturation Process of RGB SCALE Image

From the above figure, colors are coded on three bytes representing their decomposition on the three primary colors. It sounds obvious to a mathematician to immediately interpret colors as vectors in a three dimension space where each axis stands for one of the primary colors. Therefore we will benefit of most of the geometric mathematical concepts to deal with our colors, such as norms, scalar product, projection, rotation or distance.

Literature survy:

1. 2016 12th International Conference on Computational Intelligence and Security

License Plate Segmentation and Recognition of Chinese Vehicle Based on BPNN

Wang Naiguo,Zhu Xiangwei\*

Suzhou Industrial Park Branch

Jiangsu Union Technical Institute

Suzhou, China

zxwook@163.com \*corresponding author

Zhang Jian

School of computer Science & Technology

Soochow University

Suzhou, China

zhangjian2012@suda.edu.cn

Abstract—In order to make the computer own the knowledgeabout Chinese vehicle license plate segmentation and recognition,the paper put forward a set of algorithms about license platesegmentation and recognition. The algorithms are divided intofour parts: image preprocessing, license plate location, licenseplate segmentation and character recognition. The aim of image preprocessing is quickly and easily location the license plate, sothe image preprocessing algorithm is one of the important factorsthat affect total system performance. Because the algorithm oflicense plate location directly affects the accuracy of charactersegmentation and character recognition. So, the algorithm oflicense plate location is proposed according to characteristics ofChinese vehicle license plate. The algorithm of license platesegmentation uses the vertical projection method about licenseplate in this paper. According to the license plate segmentationcharacter, the training model can be generated using tool ofBPNN(back propagation neural network), which is the key of the character recognitionalgorithm about license plate. The resultsof experiment based the algorithms in the paper illustrate thataccuracy rate of character recognition is very high, and thealgorithms can fully meet the actual demand of automaticrecognition. The algorithms can take advantage of the trainingmodel to perfectly realize recognition the license plate, and haveapplication value in the real

work.Keywords-image processing; license plate segmentation;BPNN(back propagation neural network); training model;license plate character recognition

2. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 9, NO. 3, SEPTEMBER 2008 377

License Plate Recognition From Still Images and

Video Sequences: A Survey

Christos-Nikolaos E. Anagnostopoulos, Member, IEEE, Ioannis E. Anagnostopoulos, Member, IEEE,

Ioannis D. Psoroulas, Vassili Loumos, Member, IEEE, and Eleftherios Kayafas, Member, IEEE

Abstract—License plate recognition (LPR) algorithms in imagesor videos are generally composed of the following three processingsteps: 1) extraction of a license plate region; 2) segmentation ofthe plate characters; and 3) recognition of each character. Thistask is quite challenging due to the diversity of plate formatsand the nonuniform outdoor illumination conditions during imageacquisition. Therefore, most approaches work only underrestricted conditions such as fixed illumination, limited vehiclespeed, designated routes, and stationary backgrounds. Numeroustechniques have been developed for LPR in still images or videosequences, and the purpose of this paper is to categorize and assesthem. Issues such as processing time, computational power, andrecognition rate are also addressed, when available. Finally, thispaper offers to researchers a link to a public image database todefine a common reference point for LPR algorithmic assessment.

Index Terms—Image processing, license plate identification,license plate recognition (LPR), license plate segmentation, opticalcharacter recognition (OCR).

3. This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS 1

An Iranian License Plate Recognition

System Based on Color Features

Amir Hossein Ashtari, Graduate Student Member, IEEE, Md. Jan Nordin, and Mahmood Fathy

Abstract—In this paper, an Iranian vehicle license plate recognitionsystem based on a new localization approach, which is

modified to reflect the local context, is proposed, along with a hybrid classifier that recognizes license plate characters. The

method presented here is based on a modified template-matchingtechnique by the analysis of target color pixels to detect the

location of a vehicle’s license plate. A modified strip search enableslocalization of the standard color-geometric template utilized inIran and several European countries. This approach uses periodicstrip search to find the hue of each pixel on demand. In addition,when a group of target pixels is detected, it is analyzed to verifythat its shape and aspect ratio match those of the standard licenseplate. In addition to being scale and rotation invariant, this methodavoids time-consuming image algorithms and transformations forthe whole image pixels, such as resizing and Hough, Fourier, andwavelet transforms, thereby cutting down the detection responsetime. License plate characters are recognized by a hybrid classifierthat comprises a decision tree and a support vector machine witha homogeneous fifth-degree polynomial kernel. The performance detection rate and the overall system performance achieved are96% and 94%, respectively.

Index Terms—Color template matching, image recognition, license

plate detection, license plate localization, license plate numberidentification, license plate recognition (LPR)

4. License Plate Recognition for Moving Vehicles

Case : At Night and Under Rain Condition

Muh Ismail

Department of Informatics

Electrical Engineering Study Program

Hasanuddin University

Makassar, Indonesia

imuh77@gmail.com

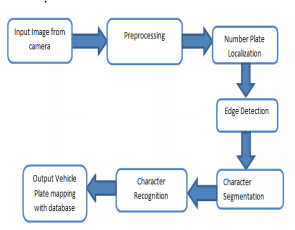
Abstract — Research for the detection and identification ofexisting license plate focus on static image under different weatherconditions. This research aims to detect and recognize the licenseplate on moving vehicle at night and under rain condition. Themethod used to detect license plate is Cascade Classifier with LocalBinary Pattern (LBP) as a descriptor, top-hat transform for imageenhancement, and recognizing character of license plate detected s using Optical Character Recognition (OCR) and TemplateMatching (TM). The testing process using 9 videos data withduration 30 seconds. The testing result that accuracy rates LBPOCR reach 77.42% increase 32.26% than without top-hat transform and LBP TM 64.52% increase 45.16% than without

top-hat transform.

Keywords—Video Processing; Image Processing; License Plate

Recognition; License Plate Detection

PROPOSED METHOD:



Pre-processing is a common name for operations with images at the lowest level of abstraction -- both input and output are intensity images.ο The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

CONVERSION:

RGB TO GRAY:

COLOR TO GRAY CONVERSION:

Color Image to Grayscale Image Conversion C. Saravanan, Image processing is a vital research area and the utilization of images increases in various applications. On different research areas, scientists are working on such as image compression, image restoration, image segmentation etc. to enhance the existing image processing techniques and invent new method of solving image processing problems. The latest image processing applications such as medical image processing, satellite image processing, and molecular image processing uses various image processing techniques. Conversion of color image to grayscale image is one of the image processing applications used in different fields effectively. In publication organizations’ printing, a color image is expensive compared to a grayscale image. Thus, color images have converted to grayscale image to reduce the printing cost for low priced edition books. Similarly, color deficient viewer requires good quality of grayscale image to perceive the information, as the normal people perceive the color picture. Likewise, various image processing applications require conversion of color image to grayscale image for different purpose. Conversion of a color image to a grayscale image requires more knowledge about the color image. A pixel color in an image is a combination of three colors Red, Green, and Blue (RGB). The RGB color values are represented in three dimensions XYZ, illustrated by the attributes of lightness, chroma, and hue. Quality of a color image depends on the color represented by the number of bits the digital device could support. The basic color image represented by 8 bit, the high color image represented using 16 bits, the true color image represented by 24 bit, and the deep color image is represented by 32 bit. The number of bits decides the maximum number of different colors supported by the digital device. If each Red, Green, and Blue occupies 8 bit then the combination of RGB occupies 24 bit and supports 16,777,216 different colors. The 24 bit represents the color of a pixel in the color image. The grayscale image has represented by luminance using 8 bits value. The luminance of a pixel value of a grayscale image ranges from 0 to 255. The conversion of a color image into a grayscale image is converting the RGB values (24 bit) into grayscale value (8 bit). Various image processing techniques and software applications converts color image to grayscale image. However, the image processing techniques or applications are unable to handle the disparity in the chromaticity and the luminance. In the literature, several linear and non-linear techniques had discussed for converting color image to grayscale image. The recent techniques handle these disparities much better than the earlier techniques. Nevertheless, the techniques involve several computational procedures such as conversion of RGB space to XYZ space then approximations then mapping or other related techniques. Grayscale mappings of color images that are constructed by approximating spectral uniformity are often inadequate [1]. The recent technique used to convert from color image to gray image highly consumes computational time and memory. Thus, a new algorithm proposed to convert color image to grayscale image in a minimum amount of time. There are several issues related to conversion of color image to grayscale image and different solutions to address these issues have addressed in the literature. The software such as Adobe Photoshop devised custom non-linear projections and required users to set image dependent parameters by trial and error [2]. The following writings discuss recent six prime research works focusing on the conversion of color image to grayscale image. A technique proposed has utilized the L\*a\*b luminance chrominance representation [3]. The proposed technique introduces an additive correction term for spatial chrominance variations. The first step of this algorithm computes high pass filtered versions of all three channels, and the high-pass content from the two chrominance channels combined into a single signal that represents high frequency chrominance information. Another alternative, used in the implementation, is the slightly computationally simpler 1-norm metric. The main feature of this technique is 2010 Second International Conference on Computer Engineering and Applications 978-0-7695-3982-9/10 $26.00 © 2010 IEEE DOI 10.1109/ICCEA.2010.192 196 the same color in the input image can map to different grayscale values depends on the spatial surround. Another novel technique proposed to handle fluorescent colors effectively [4]. Source color image converted to uniform color space, then target differences were calculated, and finally least squares optimization technique has applied. The experiment shows that the isoluminant colors handled perfectly. The cost of setting up and solving the optimization problem is proportional to the size of the image. The proposed technique is highly resource (time and memory) consumable. In addition, the technique has not provided large improvements for scenes with high dynamic luminance range like natural scenes. A technique proposed for re-coloring of images for color-deficient viewers without introducing visual artifacts [5]. The mapping of color to grayscale preserves contrasts and maintains luminance consistency. The quadratic objective function has defined for contrast preservation. Further, constraints added to enforce luminance consistency within narrow chrominance bands. The technique performed well for certain images and as standard for other images. Another technique proposed enhances the contrast and converts color to grayscale [6]. The proposed technique used Gaussian pairing technique for image sampling, dimensionality reduction, and sampling color differences. The predominant component analysis used for analyzing color differences. The technique has satisfied Continuous mapping, Global consistency, Grayscale preservation, Luminance ordering, Saturation ordering, and Hue ordering. The process controlled by three parameters: the degree of image enhancement; the typical size of relevant image features in pixels; and the proportion of image pixels assumed to be outliers. First, the algorithm converts the RGB values into YPQ color space. Further, to analyze the distribution of color contrasts between image features, color differences between pixels considered using Gaussian pairing. Dimensionality reduction by predominant component analysis performed to find the color axis that best represents the chromatic contrasts lost when the luminance channel supplies the color to grayscale mapping. Next, has combined luminance and chrominance information. The final step used saturation to calibrate luminance while adjusting its dynamic range and compensating for image noise. The decolorize algorithm is effective at enhancing contrast. The algorithm avoids the noise, contouring, and halo artifacts. However, tuning on parameters required individually to suit each image. A recent technique demonstrated color to grayscale conversion based on the experimental background of the Coloroid system observations [7]. A survey of the coloroid system to and from CIE XYZ system formulas completed. Observations based on the Coloroid system discussed. The seven basic Coloroid hues fixed. Relative gray-equivalent differences of the basic hue pairs calculated. Proposed two formulas based on the CIELab color space and the Coloroid color space for building the gradient field. Further, the inconsistency of gradient field corrected. Finally, 2D integration applied to get the grayscale image. From the demonstration noted that the isoluminant colors and bluish colors transformed to grayscale more realistic. The technique preserves overall appearance of the color image. A most recent work converted the color image and video to grayscale [8]. The proposed technique converted the image and video perceptually accurate. First, H-K (Helmoltz-Kohlrausch) phenomenon predicted by a chromatic lightness term that corrects perceived lightness based on the color’s chromatic component. The color image converted to linear RGB by inverse gamma mapping, then transformed to CIELUV color space. Its apparent chromatic object lightness channel calculated. Lightness channel to grayscale values mapped using reference white chromatic values. Gamma mapping applied to move from a linear space to a gamma-corrected space. Local contrast increased in the grayscale image to represent better the local contrast of original image. The work carried out using CIELab and CIELuv color spaces. This two step approach a good compromise between a fully automatic technique (first step) and user control (second step) making this approach well suited for natural images, photographs, artistic reproductions as well as business graphics. The main limitation of the approach is the locality of the second step. It cannot restore chromatic contrast between non-adjacent regions. END The steps 1 to 3 calculate the luminance and chrominance values of the source color image. In the step 4 sum of chrominance value calculated. Steps 5 to 16 the RGB values are approximated using RGB components. Step 17 calculates the average of the four values R4, G4, B4 and UV. The I1 represents the resulted gray color image. The color image, 3D plot of the RGB values of the color image, and 3D plot of the reduced RGB values of the color image have shown and compared. 197 Table 1. Comparison of source image RGB and reduced RGB values. Source Color Image 3D plot of the source color image RGB values 3D plot of the reduced RGB values The 3D plot of the source image and reduced RGB values compared. The RGB values of the source image ranging from 0 to 255 were reduced to a range of 0 to 85. The reduction enhances the color range and helps to calculate the grayscale in a better way. In the above picture observed that the color ranges are in the 3D plot of the source image RGB values are enhanced many colors are highly visible in the 3D plot of the approximated and reduced RGB values without any major changes in the colors and structure. The reduction process of RGB values retained the major values of the RGB at most of the points observed. The reductions made at the RGB color level so that the resultant grayscale image to retain the luminance and chrominance property at the maximum.The proposed algorithm tested on thirty four number of different eight bit color images published in the recent research publications. The color values of the color images are in the range of 0 to 255 for each Red, Green, and Blue. In the images, seven are jpg in format and twenty seven are png in type. The Mat Lab software has used for testing the algorithm. The results of the experiment carefully examined. The difference between the results of the proposed technique and recent techniques identified and discussed. The result revealed that the proposed algorithm yields grayscale image with better luminance and chrominance property for most of the cases and as standard for other few cases. However, there is a minimum amount of loss in the grayscale image due to reduction the algorithm preserved contrasts, sharpness, shadow, and structure of the color image in the reproduced grayscale image. The following table shows four of the different color images used in the experiment and respective reproduced grayscale images using the proposed algorithm. Table 2. Experiment Results Color Image Grayscale Image using luminance components In the first grayscale image the letters A, B, and C are sharp as in the color image whereas it is not so in other earlier techniques. In the second image, the shades of the different color reproduced as sharp as in the color image in the grayscale image than the recent algorithms. The conversion of color image to grayscale image using the proposed algorithm uses approximation of RGB values using luminance RGB components approximated RGB reduced by three, added with chrominance value and average of these four value results perceptually and structurally good quality of grayscale images. First, the luminance and chrominance values are calculated. Further, the RGB values of the source color image reduced. Finally, the reduced RGB values have added chrominance values. The resulted grayscale images confirm that the luminance and chrominance properties and structure of the color images 198 retained well in the grayscale image. The results confirm that the isoluminant images have handled as handled by other recent techniques. The proposed algorithm is helpful for different applications where good quality of grayscale image is highly required. The proposed algorithm results best quality of grayscale images using RGB reduction and chrominance addition in a short amount of time.

GRAY TO BLACK AND WHITE:

A binary image is the type of image where each pixel is black or white. You can also say the pixels as 0 or 1 value. Here 0 represents black and 1 represents a white pixel.

I often find peoples calling grayscale image as black and white. Now let’s understand what is grayscale and what is a black and white or binary image with the example of my own picture.

Many peoples think the grayscale image as black and white

Many of us will think the below image as black and white:



But this is a wrong concept. Real binary image or pure black and white image look like you can see below:



Each of the pixels of the above binary image is either black or white. Or we cay each pixel value is 0 or 1.

So you have seen the same image as grayscale and binary and got the idea of a binary image.

CONVERT RGB TO BINARY:

Now I am going to show you how you can convert RGB to Binary Image or convert a colored image to black and white.

Here we are just going to write a few lines of Python code and it will convert our RGB image into a binary image.

To convert an RGB image into a binary type image, we need OpenCV. So first of all, if we don’t have OpenCV installed, then we can install it via pip:

pip install opencv-python

Now we can continue writing Python code.

At the top, we have to import the OpenCV Python library:

import cv2

After that, read our image as grayscale. Grayscale is a simplified image and it makes the process simple. Below is the code to get grayscale data of the image:

img = cv2.imread('imgs/mypic.jpg',2)

Then set the threshold value for our grayscale image:

ret, bw\_img = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

Now show the image:

cv2.imshow("Binary Image",bw\_img)

In the end, we have to use waitKey and destroyAllWindows method to keep our window always open until we press any key or close our window and also destroy all windows. Below is the Python code that will do that:

cv2.waitKey(0)

cv2.destroyAllWindows()

Also, read:

* [How to create a database in MySQL using Python](https://www.codespeedy.com/create-database-in-mysql-using-python/)
* [Graph Plot of X and Y-Axis for given values as input in Python 3](https://www.codespeedy.com/graph-plot-in-python/)

The complete and final Python code to convert an RGB or colored image into the binary is given below:

1. import cv2
2. img = cv2.imread('imgs/mypic.jpg',2)
3. ret, bw\_img = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)
4. cv2.imshow("Binary Image",bw\_img)
5. cv2.waitKey(0)
6. cv2.destroyAllWindows()

now you can run and test the above code on your system. You have to pass your own image to the imread() method. After you run the code, you will able to see the binary image of the given image which path you have given to the imread() method.

MORPHOLOGICAL FILTER:

The idea of the morphological filter are shrink and let grow process. The word “shrink” means using median filter to round off the large structures and to remove the small structures and in grow process, remaining structures are grow back by the same amount.

The morphological operation of the binary image is described first and will talk in the following outline.

Outlines are The structuring element of a binary filter, Dilation and Erosion and Composite Operation

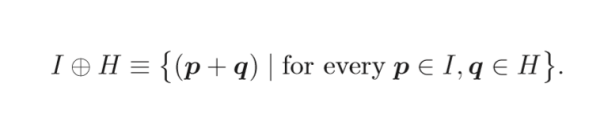
The structuring element:

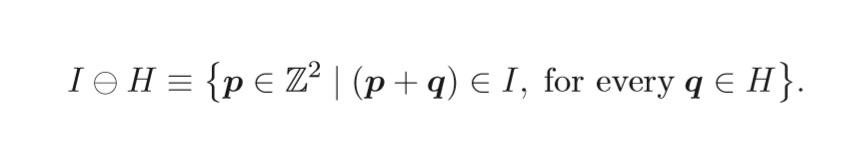
In morphological filter, each element in the matrix is called “structuring element” instead of coefficient matrix in the linear filter. The structuring elements contain only value 0 and 1. And the hot spot of the filter is the dark shade element.

The binary image is described as sets of two-dimensional coordinate point. This is called “Point Set” Q and point set consist of the coordinate pair p = (u,v) of all foreground pixels. Some operations of point set are similar to the operation in others image. For inverting binary image is complement operation and combining two binary image use union operator. Shifting binary image I by some coordinate vector d by adding vector d to point p. Or reflection of binary image I by multiply -1 to point p.

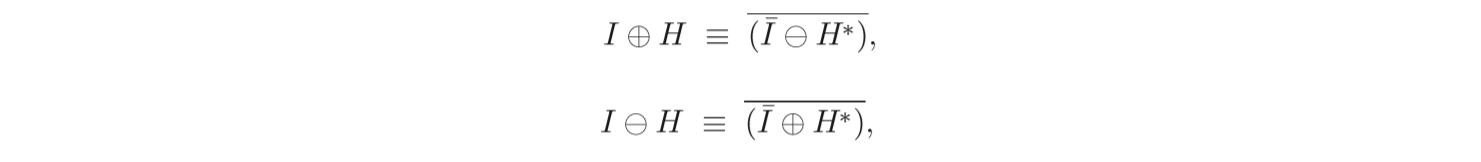
Dilation and Erosion:

Dilation is a morphological operator which works for the grow process as I mentioned before. The equation of this operator is defined as

Erosion is a morphological operator which works for the shrink process as I mentioned before as well and the equation is defined as

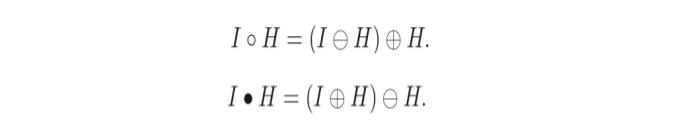


Properties of dilation and erosion are Commutative: only in dilation and Associative: only in dilation Note that: in erosion is in contrast to dilation, not have commutative property. In addition, erosion and dilation are duels, for a dilation of the foreground can be accomplished by an erosion of background and subsequent of the result in two different properties but work similarity



## Composite Operation:

In morphological process, dilation and erosion work together in composite operation. There are common way to represent the order of these two operations, opening and closing. Opening denotes an erosion followed by dilation and closing work in opposite way.



The opening and closing also are dual in sense that opening the foreground is equal to closing the background. Morphological Filter can also apply to gray-scale image, but in the different definition. It is a generalization with MIN and MAX operators. I will describe in following outline. Outlines are Structuring Elements, Dilation and Erosion, Opening and Closing

ADVANTAGES are Low complexity and High accuracy. Applications are Tracking analysis, Security analysis

EDGE DETECTION:

Edge detection includes a variety of mathematical methods that aim at identifying points in a [digital image](https://en.wikipedia.org/wiki/Digital_image) at which the [image brightness](https://en.wikipedia.org/wiki/Luminous_intensity) changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges. The same problem of finding discontinuities in one-dimensional signals is known as [step detection](https://en.wikipedia.org/wiki/Step_detection) and the problem of finding signal discontinuities over time is known as [change detection](https://en.wikipedia.org/wiki/Change_detection). Edge detection is a fundamental tool in [image processing](https://en.wikipedia.org/wiki/Image_processing), [machine vision](https://en.wikipedia.org/wiki/Machine_vision) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), particularly in the areas of [feature detection](https://en.wikipedia.org/wiki/Feature_detection_(computer_vision)) and [feature extraction](https://en.wikipedia.org/wiki/Feature_extraction).



Then detect the vehicle using the data base.

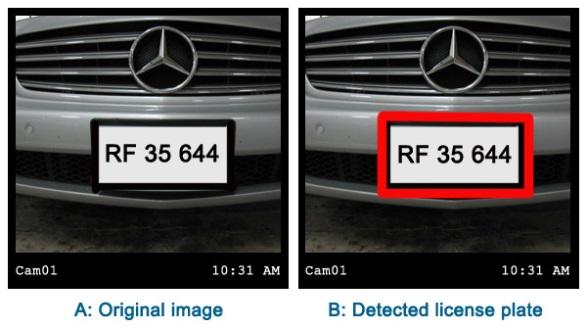
CONCLUSION:

Four algorithms of image preprocessing, license plate location, license plate segmentation and character recognition are introduced in this paper. License plate location is the basis of image preprocessing. The location of license plate has a direct impact on the accuracy of character segmentation. In our project vehicle plate detection is obtained by using character recognizatio.and region of interest.

FUTUTRE SCOPE:

According to the licence plate decetion feature extraction process is important.during this detection we can detect the thief when he will taken the vehicle.here we are using morphological filters for the addition of color to detect the features.maximum we are using on roads and car parking areas.for the future purpose we can use deep neural network algorithm.using this process we can detect more vehicles at a time.these can be use at traffic,parking ares,airport etc..for the future purpose we can avoid roberries.

EXPERIMENTAL METHOD:



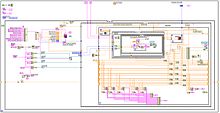
SOFTWARE USED:

* **Laboratory Virtual Instrument Engineering Workbench** (**LabVIEW**)[[1]](https://en.wikipedia.org/wiki/LabVIEW#cite_note-kring2006-1):3 is a system-design platform and development environment for a [visual programming language](https://en.wikipedia.org/wiki/Visual_programming_language) from [National Instruments](https://en.wikipedia.org/wiki/National_Instruments).
* The graphical language is named "G"; not to be confused with [G-code](https://en.wikipedia.org/wiki/G-code). The G dataflow language was originally developed by Labview,[[2]](https://en.wikipedia.org/wiki/LabVIEW#cite_note-2) LabVIEW is commonly used for [data acquisition](https://en.wikipedia.org/wiki/Data_acquisition), [instrument control](https://en.wikipedia.org/wiki/Instrument_control), and industrial [automation](https://en.wikipedia.org/wiki/Automation) on a variety of [operating systems](https://en.wikipedia.org/wiki/Operating_system) (OSs), including [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) as well as various versions of [Unix](https://en.wikipedia.org/wiki/Unix), [Linux](https://en.wikipedia.org/wiki/Linux), and [macOS](https://en.wikipedia.org/wiki/MacOS).

## Dataflow programming:

* The programming paradigm used in LabVIEW, sometimes called G, is based on data availability. If there is enough data available to a subVI or function, that subVI or function will execute. Execution flow is determined by the structure of a graphical block diagram (the LabVIEW-source code) on which the programmer connects different function-nodes by drawing wires.

## Graphical programming[[edit](https://en.wikipedia.org/w/index.php?title=LabVIEW&action=edit&section=2)]

* 
* LabVIEW code example
* LabVIEW integrates the creation of user interfaces (termed front panels) into the development cycle. LabVIEW programs-subroutines are termed virtual instruments (VIs). Each VI has three components: a block diagram, a front panel, and a connector pane. The last is used to represent the VI in the block diagrams of other, calling VIs. The front panel is built using controls and indicators. Controls are inputs: they allow a user to supply information to the VI. Indicators are outputs: they indicate, or display, the results based on the inputs given to the VI. The back panel, which is a block diagram, contains the graphical source code. All of the objects placed on the front panel will appear on the back panel as terminals. The back panel also contains structures and functions which perform operations on controls and supply data to indicators. The structures and functions are found on the Functions palette and can be placed on the back panel. Collectively controls, indicators, structures, and functions are referred to as nodes. Nodes are connected to one another using wires, e.g., two controls and an indicator can be wired to the addition function so that the indicator displays the sum of the two controls. Thus a virtual instrument can be run as either a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

## Benefits:

### Interfacing to devices:

* LabVIEW includes extensive support for interfacing to devices such as instruments, cameras, and other devices. Users interface to hardware by either writing direct bus commands (USB, GPIB, Serial) or using high-level, device-specific drivers that provide native LabVIEW function nodes for controlling the device.
* LabVIEW includes built-in support for NI hardware platforms such as [CompactDAQ](https://en.wikipedia.org/wiki/CompactDAQ) and [CompactRIO](https://en.wikipedia.org/wiki/CompactRIO), with a large number of device-specific blocks for such hardware, the *Measurement and Automation eXplorer* (MAX) and *Virtual Instrument Software Architecture* (VISA) toolsets.
* National Instruments makes thousands of device drivers available for download on the NI Instrument Driver Network (IDNet).

### Code compiling:

* LabVIEW includes a [compiler](https://en.wikipedia.org/wiki/Compiler) that produces native code for the CPU platform. The graphical code is converted into Dataflow Intermediate Representation, and then translated into chunks of executable [machine code](https://en.wikipedia.org/wiki/Machine_code) by a compiler based on [LLVM](https://en.wikipedia.org/wiki/LLVM). Run-time engine calls these chunks, allowing better performance. The LabVIEW syntax is strictly enforced during the editing process and compiled into the executable machine code when requested to run or upon saving. In the latter case, the executable and the source code are merged into a single binary file. The execution is controlled by LabVIEW [run-time](https://en.wikipedia.org/wiki/Run-time_system) engine, which contains some pre-compiled code to perform common tasks that are defined by the G language. The run-time engine governs execution flow, and provides a consistent interface to various operating systems, graphic systems and hardware components. The use of run-time environment makes the source code files portable across supported platforms. LabVIEW programs are slower than equivalent compiled C code, though like in other languages, program optimization often allows to mitigate issues with execution speed.

### Large libraries:

* Many [libraries](https://en.wikipedia.org/wiki/Library_(computing)) with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, etc., along with numerous for functions such as integration, filters, and other specialized abilities usually associated with data capture from hardware sensors is enormous. In addition, LabVIEW includes a text-based programming component named MathScript with added functions for signal processing, analysis, and mathematics. MathScript can be integrated with graphical programming using *script nodes* and uses a syntax that is compatible generally with [MATLAB](https://en.wikipedia.org/wiki/MATLAB).[[11]](https://en.wikipedia.org/wiki/LabVIEW#cite_note-11)

### Parallel programming:

### LabVIEW is an inherently [concurrent language](https://en.wikipedia.org/wiki/Concurrent_language), so it is very easy to program multiple tasks that are performed in parallel via multithreading. For example, this is done easily by drawing two or more parallel while loops and connecting them to two separate nodes. This is a great benefit for test system automation, where it is common practice to run processes like test sequencing, data recording, and hardware interfacing in parallel.

### Ecosystem:

* Due to the longevity and popularity of the LabVIEW language, and the ability for users to extend its functions, a large ecosystem of third party add-ons has developed via contributions from the community. This ecosystem is available on the LabVIEW Tools Network, which is a marketplace for both free and paid LabVIEW add-ons.

### User community:

* There is a low-cost LabVIEW Student Edition aimed at educational institutions for learning purposes. There is also an active community of LabVIEW users who communicate through several [electronic mailing lists](https://en.wikipedia.org/wiki/Electronic_mailing_list) (email groups) and [Internet forums](https://en.wikipedia.org/wiki/Internet_forum).

### Home Bundle Edition

* [National Instruments](https://en.wikipedia.org/wiki/National_Instruments) provides a low cost LabVIEW Home Bundle Edition.

### Community Edition Edition:

* National Instruments provides a free-for-non-commercial use version called LabVIEW Community Edition.[[13]](https://en.wikipedia.org/wiki/LabVIEW#cite_note-13) This version includes everything in the Professional Editions of LabVIEW, has no watermarks, and includes the LabVIEW NXG Web Module for non-commercial use. These editions may also be used by K-12 schools.[[14]](https://en.wikipedia.org/wiki/LabVIEW#cite_note-14)

## Criticism:

* LabVIEW is a [proprietary](https://en.wikipedia.org/wiki/Proprietary_software) product of [National Instruments](https://en.wikipedia.org/wiki/National_Instruments). Unlike common programming languages such as [C](https://en.wikipedia.org/wiki/C_Programming_Language) or [Fortran](https://en.wikipedia.org/wiki/Fortran), LabVIEW is not managed or specified by a third party standards committee such as the [American National Standards Institute](https://en.wikipedia.org/wiki/American_National_Standards_Institute) (ANSI), the [Institute of Electrical and Electronics Engineers](https://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers) (IEEE), the [International Organization for Standardization](https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO), and others.

### Non-textual:

* Since G language is non-textual, software tools such as versioning, side-by-side (or diff) comparison, and version code change tracking cannot be applied in the same manner as for textual programming languages. There are some additional tools to make comparison and merging of code with source code control (versioning) tools such as subversion, CVS and Perforce. [[15]](https://en.wikipedia.org/wiki/LabVIEW#cite_note-15)[[16]](https://en.wikipedia.org/wiki/LabVIEW#cite_note-16)[[17]](https://en.wikipedia.org/wiki/LabVIEW#cite_note-17)

### No zoom function:

* There was no ability to zoom in to (or enlarge) a virtual instrument (VI) which will be hard to see on a large high-resolution monitor. However, the ability to zoom has been added into LabVIEW NXG.

l

INTRODUCTION TO COMPUTER VISION

Using software to parse the world’s visual content is as big of a revolution in computing as mobile was 10 years ago, and will provide a major edge for developers and businesses to build amazing products.

Computer Vision is the process of using machines to understand and analyze imagery (both photos and videos). While these types of algorithms have been around in various forms since the 1960’s, recent advances in [Machine Learning](https://blog.algorithmia.com/introduction-to-machine-learning/), as well as leaps forward in data storage, computing capabilities, and cheap high-quality input devices, have driven major improvements in how well our software can explore this kind of content.

What is Computer Vision?

Computer Vision is the broad parent name for any computations involving visual content – that means images, videos, icons, and anything else with pixels involved. But within this parent idea, there are a few specific tasks that are core building blocks:

* In object classification, you train a model on a dataset of specific objects, and the model classifies new objects as belonging to one or more of your training categories.
* For object identification, your model will recognize a specific instance of an object – for example, parsing two faces in an image and tagging one as Tom Cruise and one as Katie Holmes.

A classical application of computer vision is handwriting recognition for digitizing handwritten content (we’ll explore more use cases below). Outside of just recognition, other methods of analysis include:

* Video motion analysis uses computer vision to estimate the velocity of objects in a video, or the camera itself.
* In image segmentation, algorithms partition images into multiple sets of views.
* Scene reconstruction creates a 3D model of a scene inputted through images or video (check out [Selva](https://www.selva3d.com/)).
* In image restoration, noise such as blurring is removed from photos using Machine Learning based filters.

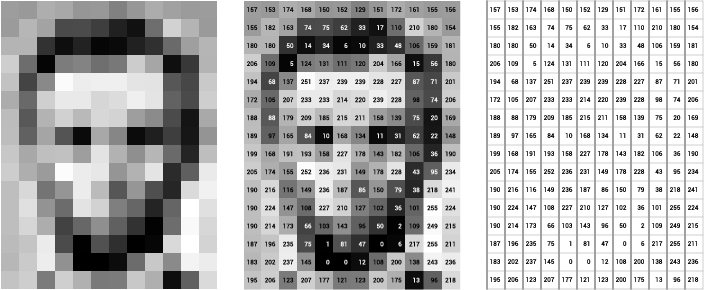
Any other application that involves understanding pixels through software can safely be labeled as computer vision.

How Computer Vision Works

One of the major open questions in both Neuroscience and Machine Learning is: how exactly do our brains work, and how can we approximate that with our own algorithms? The reality is that there are very few working and comprehensive theories of brain computation; so despite the fact that Neural Nets are supposed to “mimic the way the brain works,” nobody is quite sure if that’s actually true. Jeff Hawkins has an [entire book on this topic called On Intelligence](https://www.amazon.com/Intelligence-Understanding-Creation-Intelligent-Machines/dp/0805078533).

The same paradox holds true for computer vision – since we’re not decided on how the brain and eyes process images, it’s difficult to say how well the algorithms used in production approximate our own internal mental processes. For example, [studies have shown](https://www.technologyreview.com/s/508376/in-a-frogs-eye/) that some functions that we thought happen in the brain of frogs actually take place in the eyes. We’re a far cry from amphibians, but similar uncertainty exists in human cognition.

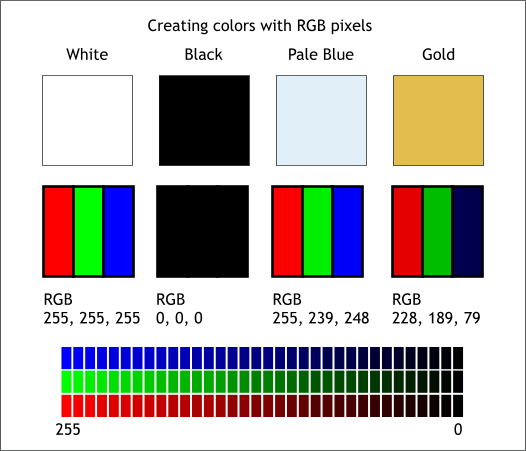
Machines interpret images very simply: as a series of pixels, each with their own set of color values. Consider the simplified image below, and how grayscale values are converted into a simple array of numbers:



Source: [Openframeworks](http://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)

Think of an image as a giant grid of different squares, or pixels (this image is a very simplified version of what looks like either Abraham Lincoln or a Dementor). Each pixel in an image can be represented by a number, usually from 0 – 255. The series of numbers on the right is what software sees when you input an image. For our image, there are 12 columns and 16 rows, which means there are 192 input values for this image.

When we start to add in color, things get more complicated. Computers usually read color as a series of 3 values – red, green, and blue (RGB) – on that same 0 – 255 scale. Now, each pixel actually has 3 values for the computer to store in addition to its position. If we were to colorize President Lincoln (or Harry Potter’s worst fear), that would lead to 12 x 16 x 3 values, or 576 numbers.



Source: [Xaraxone](http://archive.xaraxone.com/webxealot/workbook35/page_5.htm)

For some perspective on how computationally expensive this is, consider this tree:

* Each color value is stored in 8 bits.
* 8 bits x 3 colors per pixel = 24 bits per pixel.
* A normal sized 1024 x 768 image x 24 bits per pixel = almost 19M bits, or about 2.36 megabytes.

That’s a lot of memory to require for one image, and a lot of pixels for an algorithm to iterate over. But to train a model with meaningful accuracy – especially when you’re talking about [Deep Learning](https://blog.algorithmia.com/introduction-to-deep-learning/) – you’d usually need tens of thousands of images, and the more the merrier. Even if you were to use [Transfer Learning](https://en.wikipedia.org/wiki/Transfer_learning) to use the insights of an already trained model, you’d still need a few thousand images to train yours on.

With the sheer amount of computing power and storage required just to train deep learning models for computer vision, it’s not hard to understand why advances in those two fields have driven Machine Learning forward to such a degree.

Business Use Cases for Computer Vision

Computer vision is one of the areas in Machine Learning where core concepts are already being integrated into major products that we use every day. [Google is using maps](https://research.googleblog.com/2017/05/updating-google-maps-with-deep-learning.html) to leverage their image data and identify street names, businesses, and office buildings. Facebook is using computer vision to identify people in photos, and do a number of things with that information.

But it’s not just tech companies that are leverage Machine Learning for image applications. Ford, the American car manufacturer that has been around [literally since the early 1900’s](https://en.wikipedia.org/wiki/Ford_Motor_Company), is [investing heavily in autonomous vehicles (AVs)](https://media.ford.com/content/fordmedia/fna/us/en/news/2016/08/16/ford-targets-fully-autonomous-vehicle-for-ride-sharing-in-2021.html). Much of the underlying technology in AVs relies on analyzing the multiple video feeds coming into the car and using computer vision to analyze and pick a path of action.

Another major area where computer vision can help is in the medical field. Much of diagnosis is image processing, like reading x-rays, MRI scans, and other types of diagnostics. [Google has been working with medical research teams](https://research.google.com/teams/brain/healthcare/) to explore how deep learning can help medical workflows, and have made significant progress in terms of accuracy. To paraphrase from their research page:

“Collaborating closely with doctors and international healthcare systems, we developed a state-of-the-art computer vision system for reading retinal fundus images for diabetic retinopathy and determined our algorithm’s performance is on par with U.S. board-certified ophthalmologists. We’ve recently published some of our research in the [Journal of the American Medical Association](https://research.google.com/pubs/archive/45732.pdf) and summarized the highlights in a [blog post](https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html).”

But aside from the groundbreaking stuff, it’s getting much easier to integrate computer vision into your own applications. A number of high-quality third party providers like Clarifai offer [a simple API for tagging and understanding images](https://www.clarifai.com/), while Kairos [provides functionality around facial recognition](https://www.kairos.com/). We’ll dive into the open-source packages available for use below.

Computer Vision on Algorithmia

Algorithmia makes it easy to deploy computer vision applications as scalable microservices. Our marketplace has a few algorithms to help get the job done:

* [SalNet](https://algorithmia.com/algorithms/deeplearning/SalNet) automatically identifies the most important parts of an image
* [Nudity Detection](https://algorithmia.com/algorithms/sfw/NudityDetectioni2v) detects nudity in pictures
* [Emotion Recognition](https://algorithmia.com/algorithms/deeplearning/EmotionRecognitionCNNMBP) parses emotions exhibited in images
* [DeepStyle](https://demos.algorithmia.com/deep-style/) transfers next-level filters onto your image
* [Face Recognition](https://algorithmia.com/algorithms/cv/FaceRecognition)…recognizes faces.
* [Image Memorability](https://algorithmia.com/algorithms/deeplearning/LargescaleImageMemorability) judges how memorable an image is.

A typical workflow for your product might involve passing images from a security camera into Emotion Recognition and raising a flag if any aggressive emotions are exhibited, or using Nudity Detection to block inappropriate profile pictures on your web application.

For a more detailed exploration of how you can use the Algorithmia platform to implement complex and useful computer vision tasks,

### Computer Vision Resources

##### Packages and Frameworks

[OpenCV](https://opencv.org/) – “OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.”

[SimpleCV](http://simplecv.org/) – “SimpleCV is an open source framework for building computer vision applications. With it, you get access to several high-powered computer vision libraries such as OpenCV – without having to first learn about bit depths, file formats, color spaces, buffer management, eigenvalues, or matrix versus bitmap storage.”

[Mahotas](http://mahotas.readthedocs.io/en/latest/) – “Mahotas is a computer vision and image processing library for Python. It includes many algorithms implemented in C++ for speed while operating in numpy arrays and with a very clean Python interface. Mahotas currently has over 100 functions for image processing and computer vision and it keeps growing.

* NUM PY:
* NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.
* Audience
* This tutorial has been prepared for those who want to learn about the basics and various functions of NumPy. It is specifically useful for algorithm developers. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.
* Prerequisites
* You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.
* NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

## Operations using NumPy

Using NumPy, a developer can perform the following operations −

* Mathematical and logical operations on arrays.
* Fourier transforms and routines for shape manipulation.
* Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

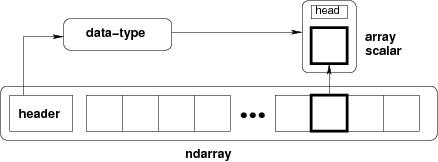
## NumPy – A Replacement for MatLab

NumPy is often used along with packages like SciPy (Scientific Python) and Mat−plotlib (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type −



An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows −

numpy.array

It creates an ndarray from any object exposing array interface, or from any method that returns an array.

Imutils:

A series of convenience functions to make basic image processing operations such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python.

Transalation

Translation is the shifting of an image in either the x or y direction. To translate an image in OpenCV you need to supply the (x, y)-shift, denoted as (tx, ty) to construct the translation matrix M:

translation_eq

And from there, you would need to apply the cv2.warpAffine  function.

Instead of manually constructing the translation matrix M and calling cv2.warpAffine , you can simply make a call to the translate  function of imutils

Rotation

Rotating an image in OpenCV is accomplished by making a call tocv2.getRotationMatrix2D  and cv2.warpAffine . Further care has to be taken to supply the (x, y)-coordinate of the point the image is to be rotated about. These calculation calls can quickly add up and make your code bulky and less readable. The rotate  function inimutils  helps resolve this problem.

resizing

Resizing an image in OpenCV is accomplished by calling the cv2.resize  function. However, special care needs to be taken to ensure that the aspect ratio is maintained. Thisresize  function of imutils  maintains the aspect ratio and provides the keyword arguments width  and height  so the image can be resized to the intended width/height while (1) maintaining aspect ratio and (2) ensuring the dimensions of the image do not have to be explicitly computed by the developer.

Another optional keyword argument, inter , can be used to specify interpolation method as well.

Skeletonization is the process of constructing the “topological skeleton” of an object in an image, where the object is presumed to be white on a black background. OpenCV does not provide a function to explicity construct the skeleton, but does provide the morphological and binary functions to do so.

For convenience, the skeletonize  function of imutils  can be used to construct the topological skeleton of the image.

The first argument, size  is the size of the structuring element kernel. An optional argument,structuring , can be used to control the structuring element — it defaults tocv2.MORPH\_RECT  , but can be any valid structuring element.

DISPLAYING WITH MATPLOTLIB

In the Python bindings of OpenCV, images are represented as NumPy arrays in BGR order. This works fine when using the cv2.imshow  function. However, if you intend on using Matplotlib, the plt.imshow  function assumes the image is in RGB order. A simple call tocv2.cvtColor  will resolve this problem, or you can use the opencv2matplotlib  convenience function.

REFERENCES:

[1] A. H. Ashtari, “License plate localization and recognition in color images

based on new feature of Iranian license plate and support vector machine,”

M.S. thesis, Dept. Eng., Islamic Azad Univ., Dezful, Iran, 2009.

[2] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas,

V. Loumos, and E. Kayafas, “License plate recognition from still images

and video sequences: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 9,

no. 3, pp. 377–391, Sep. 2008.

[3] W. Jia, H. Zhang, X. He, and M. Piccardi, “Mean shift for accurate license

plate localization,” in *Proc. IEEE Intell. Transp. Syst.*, 2005, pp. 566–571.

[4] S. L. Chang, L. S. Chen, Y. C. Chung, and S.W. Chen, “Automatic license

plate recognition,” *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1, pp. 42–

53, Mar. 2004.

[5] X. Shi,W. Zhao, and Y. Shen, “Automatic license plate recognition system

based on color image processing,” in *Proc. ICCSA*, 2005, pp. 307–314.

[6] V. Abolghasemi and A. Ahmadyfard, “An edge-based color-aided method

for license plate detection,” *Image Vis. Comput.*, vol. 27, no. 8, pp. 1134–

1142, Jul. 2009.

[7] R. W. Rodieck, *The First Steps in Seeing*. Sunderland, MA, USA:

Sinauer Associates, 1998.

[8] E. R. Lee, P. K. Kim, and H. J. Kim, “Automatic recognition of a car

license plate using color image processing,” in *Proc. IEEE ICIP*, 1994,

pp. 301–305.

[9] C.-C. Lin and W.-H. Huang, “Locating license plate based on edge features

of intensity and saturation subimages,” in *Proc. 2nd ICICIC*, 2007,

p. 227.

[10] F. Wang, L. Man, B. Wang, Y. Xiao, W. Pan, and X. Lu, “Fuzzy-based

algorithm for color recognition of license plates,” *Pattern Recognit. Lett.*,

vol. 29, no. 7, pp. 1007–1020, May 2008.

[11] Y. Yanamura, M. Goto, D. Nishiyama, M. Soga, H. Nakatani, and H. Saji,

“Extraction and tracking of the license plate using Hough transform

and voted block matching,” in *Proc. IEEE Intell. Veh. Symp.*, 2003,

pp. 243–246.

[12] T. D. Duan, T. L. H. Du, T. V. Phuoc, and N. V. Hoang, “Building an

automatic vehicle license plate recognition system,” in *Proc. Int. Conf.*

*Comput. Sci. RIVF*, 2005, pp. 59–63.

[13] N. Zimic, J. Ficzko, M. Mraz, and J. Virant, “The fuzzy logic approach to

the car number plate locating problem,” in *Proc. IIS*, 1997, pp. 227–230.

[14] S. M. Youssef and S. B. AbdelRahman, “RETRACTED: A smart access

control using an efficient license plate location and recognition approach,”

*Exp. Syst. Appl.*, vol. 34, no. 1, pp. 256–265, Jan. 2008.

[15] H. Kwanicka and B. Wawrzyniak, “License plate localization and recognition

in camera pictures,” in *Proc. 3rd Symp. Methods Artif. Intell.*, 2002,

pp. 243–246.

[16] H. Bai, J. Zhu, and C. Liu, “A fast license plate extraction method

on complex background,” in *Proc. IEEE Intell. Transp. Syst.*, 2003,

pp. 985–987.

[17] D. Zheng, Y. Zhao, and J. Wang, “An efficient method of license

plate location,” *Pattern Recognit. Lett.*, vol. 26, no. 15, pp. 2431–2438,

Nov. 2005.

[18] H. Zhao, C. Song, and S. Zhang, “License plate recognition system based

on morphology and LS-SVM,” in *Proc. IEEE GrC*, 2008, pp. 826–829.

[19] J. Jiao, Q. Ye, and Q. Huang, “A configurable method for multi-style

license plate recognition,” *Pattern Recognit.*, vol. 42, no. 3, pp. 358–369,

Mar. 2009.

[20] S. H. Kasaei, S. M. Kasaei, and S. A. Kasaei, “New morphology based

method for robust Iranian car plate detection and recognition,” *Int. J.*

*Comput. Theory Eng.*, vol. 2, no. 2, pp. 264–268, Apr. 2010.

[21] F. M. Kazemi, S. Samadi, H. R. Poorreza, and M. R. Akbarzadeh-T,

“Vehicle recognition based on Fourier, wavelet and curvelet transforms—

A comparative study,” in *Proc. ITNG*, 2007, pp. 939–940.

[22] F. M. Kazemi, S. Samadi, H. R. Poorreza, and M. R. Akbarzadeh-T,

“Vehicle recognition using curvelet transform and SVM,” in *Proc. ITNG*,

2007, pp. 516–521.

[23] V. Beëanoviéa, M. Kermitb, and A. J. Eidec, “Feature extraction

from photographical images using a hybrid neural network,” in *Proc.*

*SPIE 9th Workshop Virtual Intell./Dyn. Neural Netw.*, 1999, vol. 3728,

pp. 351–361.

[24] K. Jung, “Neural network-based text location in color images,” *Pattern*

*Recognit. Lett.*, vol. 22, no. 14, pp. 1503–1515, Dec. 2001.

[25] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory*

*and Practice*. Hoboken, NJ, USA: Wiley, 2009.

[26] M. H. Dashtban, Z. Dashtban, and H. Bevrani, “A novel approach for

vehicle license plate localization and recognition,” *Int. J. Comput. Appl.*,

vol. 26, no. 11, pp. 22–30, Jul. 2011.

[27] M. J. Ghasemi, H. R. Tajozzakerin, and A. R. Omidian, “An Iranian

national number plate localization and recognition system for private

vehicles,” *Int. J. Acad. Res.*, vol. 2, no. 6, pp. 13–19, Nov. 2010.

[28] A. Broumandnia and M. Fathy, “Application of pattern recognition for

Farsi license plate recognition,” *ICGST Int. J. Graph., Vis. Image Process.*,

vol. 5, no. 2, pp. 25–31, Jan. 2005.

[29] M. Rasooli, T. Branch, S. Ghofrani, and E. Fatemizadeh, “Farsi license

plate detection based on element analysis and characters recognition,” *Int.*

*J. Signal Process., Image Process. Pattern Recognit.*, vol. 4, no. 1, pp. 65–

80, Mar. 2011.

[30] F. PirahanSiah, S. N. H. S. Abdullah, and S. Sahran, “Adaptive image

segmentation based on peak signal-to-noise ratio for a license plate recognition

system,” in *Proc. ICCAIE*, 2010, pp. 468–472.

[31] W. Burger and M. J. Burge, *Digital Image Processing: An Algorithmic*

*Introduction Using Java*. New York, NY, USA: Springer-Verlag, 2007.

[32] M. Sezgin, “Survey over image thresholding techniques and quantitative

performance evaluation,” *J. Electron. Imaging*, vol. 13, no. 1, pp. 146–

168, Jan. 2004.

[33] K. Jung, K. In Kim, and A. K. Jain, “Text information extraction in images

and video: A survey,” *Pattern Recognit.*, vol. 37, no. 5, pp. 977–997,

May 2004.

[34] J. Kittler and J. Illingworth, “Minimum error thresholding,” *Pattern*

*Recognit.*, vol. 19, no. 1, pp. 41–47, 1986.

[35] Y. Zhang and C. Zhang, “A new algorithm for character segmentation of

license plate,” in *Proc. IEEE Intell. Veh. Symp.*, 2003, pp. 106–109.

[36] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, and T. Shiose,

“A novel adaptive morphological approach for degraded character image

segmentation,” *Pattern Recognit.*, vol. 38, no. 11, pp. 1961–1975,

Nov. 2005.

[37] C. A. Rahman, W. Badawy, and A. Radmanesh, “A real time vehicle’s

license plate recognition system,” in *Proc. IEEE Conf. Adv. Video Signal*

*Based Surveillance*, 2003, pp. 163–166.

[38] Y. P. Huang, S. Y. Lai, and W. P. Chuang, “A template-based model for

license plate recognition,” in *Proc. IEEE Int. Conf. Netw., Sens. Control*,

2004, pp. 737–742.

[39] A. Delforouzi and M. Pooyan, “Efficient Farsi license plate recognition,”

in *Proc. 7th ICICS*, 2009, pp. 1–5.

[40] M. M. Dehshibi and R. Allahverdi, “Persian vehicle license plate recognition

using multiclass Adaboost,” *Int. J. Comput. Elect. Eng.*, vol. 4, no. 3,

pp. 355–358, Jun. 2012.

[41] H. Mahini, S. Kasaei, F. Dorri, and F. Dorri, “An efficient features-based

license plate localizationmethod,” in *Proc. 18th ICPR*, 2006, pp. 841–844.