ABSTRACT

In this paper, we designed a web application for storing the medical information of a person, with the help of iris recognition, which helps in recognize the individual person accurately. The medical information which is given by the user will be stored in the database, when the user will come to verify then the information stored in the database will be retrieved.

This system will take all the input from the user and store the data in a database, so that when ever a user will come to verify his past medical information he can verify it my his iris which will be recognized by the system and will show the matched iris data form the database.

Iris Recognition is a biometric way of identifying persons based on distinctive patterns within the ring-shaped portion of the eye's pupil. Because each iris is unique to an individual, it is an excellent type of biometric verification.

In this project we used SQL database, A SQL database, often known as a relational database, is a collection of highly organised tables, with each row representing a data object and each column defining a specific information field. Relational databases are created, stored, updated, and retrieved using the structured query language (SQL).

In the web application, frontend part we did using HTML, CSS, JAVASCRIPT and backend part we did with the help of FLASK, Flask is a Python-based web application framework that is built on Jinja2. The Flask framework has the following advantages: a built-in development server and a quick debugger.

Introduction

One of the best methods for biometric identification is iris recognition, which is gaining more and more attention in all spheres of life for a variety of applications. However, noise interference from the eyelash, eyelid, and sampling flare makes it difficult to recognize quality and speed. In the paper, we present a method that unifies noise between login iris and sample iris photos with the goal of encouraging the usage of matching methods widely. Fingerprint recognition module has also been integrated with this prototype to avoid any discrepancy in recognition.

ICT (information and communications technology) appears to be pervasive in practically every industry.

The management of healthcare data has changed as a result of ICT. The switch from paper-based to electronic patient records has been made possible by the widespread use of electronic devices like computers, tablets, and mobile phones as well as the availability of high-speed internet.

Digital patient records that are updated in real time are known as electronic health records (EHRs). Electronic health records (EHRs) make it easier to make better healthcare decisions, track a patient's clinical development, and deliver evidence-based care.

The iris is an exposed body that can be remotely evaluated with the use of an automated iris recognition system.

- Computer vision, pattern recognition, statistical inference, and optics are all used in iris identification technologies.
- A person can be identified by their iris' apparent spatial patterns, which are very distinctive. Clinical observation in conjunction with developmental biology

Segmentation - Isolating the actual iris region in a digitized eye image is the first step in iris identification. Two circles—one for the iris/sclera boundary and another, inside to the first, for the iris/pupil boundary—can be used to represent the region of the iris in the image above. The quality of the eye pictures' imaging will determine how well segmentation works. The outer

iris pattern's radius can be determined using the pupil's center. By employing the Canny edge detector to pinpoint the edge picture, the iris' inner and outer limits can be determined.

Normalisation - The next step is to normalise the segmented iris region in order to generate the iris code and perform comparisons on it. It is necessary to normalise the iris image in order to provide a universal representation of the iris with comparable proportions because individual differences in the eye, such as the optical size of the iris, pupil position inside the iris, and iris orientation, exist.

The iris is unwrapped as part of the normalization process, then it is transformed into its polar counterpart. Using Daugman's Rubber Sheet Model, it is accomplished. The points on the Cartesian scale are converted to points on the polar scale using a Remapping formula with the pupil center as the reference point.

Feature Extraction - An iris signature, or trademark indication, including these separated features is produced as a result of the encoding, or feature extraction, which seeks to separate as many refined features as is possible from the iris template. The main goal of the matching procedure between two templates is to reduce inaccurate and invalid matches for a charlatan and increase the likelihood of an accurate match for genuine detection attempts. In other words, photos from separate irises should be labelled as being from different people, and images from the same iris taken at different periods should be recognized as being from the same person.

Encoding - A set of mathematical operations known as Gabor wavelets, which examine and extract the distinctive texture of an iris, serve as the foundation for the process of storing iris patterns. When this phase information is coarsely quantized, a random bit stream is produced that is sufficiently stable for one eye while being random and diverse for different eyes, allowing for the very quick and accurate recognition of iris patterns across large databases using just a statistical independence test. The efficiency and effectiveness of searches for matches when pattern information is represented in terms of such phase bit strings, along with certain significant aspects of the Gabor wavelets as encoders, may be partially responsible for the success of this biometric approach.

Python programming language has been used to implement the above prototype.

Python - The most recent version of the Python programming language, Python 3, is utilized for cutting-edge software development projects like web development and machine learning

applications. Python is a very good programming language for beginners, as well as for seasoned programmers with experience in other programming languages like C++ and Java.

Libraries used:

- Argparse - The Python argparse module facilitates the development of command-line programmes in a way that looks to be not only simple to design but also enhances interaction. When users supply the programme with invalid arguments, the argparse module automatically generates help and usage messages and raises errors.
- OS - Python's OS module offers tools for communicating with the operating system. OS is included in the basic utility modules for Python. This module offers a portable method of using functionality that is dependent on the operating system. There are numerous functions to deal with the file system in the *os* and *os.path* modules.
- Time - Working with time in Python is made possible by the time module. It enables features like accessing the current time and stopping a programme from running, among others.
- Savemat - With the help of the function savemat, Python data can be saved to a MAT-file. Data must be organised similarly to a loadmat, i.e., it must be made up of basic data types like dict, list, str, int, and float. A dict with the variables is required as the parameter data when saving a Python data structure to a MAT-file.
- Scipy - Python's Scipy module can be used to solve a variety of mathematical problems and methods. It is built on top of the Numpy library, which extends the possibilities for finding mathematical formulae in science, such as Matrix Rank, Inverse, polynomial equations, LU Decomposition, etc.
- Open-CV - A collection of Python bindings called OpenCV-Python was created to address issues with computer vision. An image is loaded using the cv2.imread () method from the given file. This method produces an empty matrix if the picture cannot be read (due to a missing file, poor permissions, an unsupported or invalid format, etc.).
- MySql.Connector - Python programmes can access MySQL databases with MySQL Connector/Python by using an API complying with the Python

Database API Specification v2.0 (PEP 249). With the exception of the Python Standard Library, it is entirely written in Python.

- Multiprocessing - Multiple classes are available in the Python multiprocessing module, enabling us to create parallel programmes to implement multiprocessing in Python. It provides an easy-to-use API for distributing tasks among numerous processors, fully utilising multiprocessing.
- Numpy - A general-purpose array processing package is called Numpy. It offers a multidimensional array object with outstanding speed as well as capabilities for interacting with these arrays. It is the cornerstone Python module for scientific computing. In addition to its apparent scientific applications, Numpy is a powerful multi-dimensional data container.

Problem Statement

There are many methods available for biometric identification among which the most accurate and trustworthy biometric identification framework at present is iris recognition. To avoid any discrepancy in identification this method has been used. Several times it has been observed that a person's biometric fails to let him get recognized in case of fatal injuries. Iris recognition system is able to recognize even when the subject is no more.

Previously we have seen the inability of face and fingerprint system for human recognition in some extreme cases but this system will be able to give more accurate results and adding to that it is also much faster in providing results in comparison to the above prototype. To still be more secure and avoiding discrepancy fingerprint recognition module has been integrated to give more perfect results.

These features make this system perfect for medical field where time availability is less and accuracy required is more for intensive and emergency treatment. In case of any accident if the injured person is not able to tell about itself, so medical team has to use medication on its own risk without knowing whether the subject is the allergic to particular medicine and without knowing previous medical history of subject.

LITERATURE SURVEY

S. No	Title of Paper	Description	Method	Advantage	Disadvantage	Dataset	Accuracy	Conclusion
1.	Human Iris Recognition for Clean Electoral Process in India by Creating a Fraud Free Voter Registration List.	Year of research: - 2016 To provide a clean electoral environment, this article implements the creation of a fraud-free voter ID list. The canny edge detection	The Dougman's Normalization technique is used for normalisation, the Log Gabor Filter is used for feature extraction, and the Euclidian distance approach is used for matching. The current study was created using the MATLAB 2011 edition, and	For easy and smooth iris comparisons, just the significant elements of the iris must be encoded in order to determine the uniqueness of the iris, which aids in the construction of the signature.	This area of work requires continuous development in the matching and recognition sections in order to have an efficient and higher recognition rate. The accuracy of iris recognition is mostly	CASIA	95 %	The current work is quite innovative, and it is predicted that this study will produce some high accuracy security system implementation. Also, because the electoral processing

		technique is used to localise the iris and pupils.	great attention is placed on software for efficient Irises recognition.		determined by the capture of images. If the input picture is incorrect, the accuracy will suffer.			system requires very high security measures, it will be extremely helpful to the nation to run the system smoothly by generating a completely fraud-free voter registration list.
2	IWT Based Iris Recognition for Image	Year of research: - 2020	By applying a four-level Integer Wavelet Transform to the input picture, 256	Human iris can be considered as most assuring biometric traits.	No Iris de-noising method in the proposed system,	Dataset used :- UBIRIS.v2	Accuracy: - 98.9%	An IWT-based efficient iris identification

	Authentication.	The variation (RTV) model is used in conjunction. Four levels of IWT are used to normalise and deconstruct the segmented iris area.	sub-bands are created, of which only 192 lower sub-bands are considered. High frequency	It may be quite useful in surveillance applications, such as exploiting textural changes of the iris template for this reason.	which can improve in performance of the proposed algorithm.			system was created and assessed in the article. For iris localization and segmentation, the given work employed the Circular Hough Transform and the Total Variation Model. IWT was utilised to breakdown the segmented

								picture into sub-band images from which characteristics were derived.
3	Human eye iris recognition using the mutual information.	Year of research: - 2004 Thus far, biometric traits such as fingerprints, hand geometry, face, voice, and eye iris have been extensively studied in the context	British Telecom, US Sandia Laboratories, UK National Physical Laboratory, NCR, Oki, IriScan, and other institutes have invented and commercially implemented relevant biometric technologies for public	This strategy is based on the mutual information's capacity to successfully recognise an eye iris image. Despite wearing spectacles or contact lenses, it is feasible to scan the iris visually	We definitely need to use spectacles or contact lenses.	Custom Dataset	99 %	The current study contributes by proposing and successfully verifying a new biometric measuring method for electro-optical and computer evaluati

		of personal (human) identification.	and personal usage.	without mechanical contact.				on of the human eye iris structure in relation to person identification.
4	Design and Implementation of Iris Recognition Based Attendance Management System	Year of research: - 2015 Biometric technologies are becoming increasingly common as we move towards the digital world. One of the most	The suggested outcomes will assist in choosing a low-cost attendance system that just requires a computer, a webcam, and a wireless server.	This work can initiate a new era of attendance management system. A biometric-based automated attendance management system based on fingerprints was presented.	The requirement of webcam, wireless server is compulsory	Dataset used: - CASIA	82.2%	The research developed an automatic attendance approach that makes use of an iris recognition technology. This work has the potential to usher in a new

		effective biometric-based attendance management solutions is iris recognition..						era of attendance management systems.
5	Iris Recognition; From Classic to Modern Approaches	Year of research: - 2019 Security is one of the most pressing concerns confronting the modern world.	As supervised learning, use Artificial Neural Network implementations with networks that use specified guides to update the linkages between their nodes.	The information derived from the human iris might be a reliable and difficult to replicate hotspot for biometric-based system. Of the different biometrics available,	The training network will take longer than usual because to the increase in input picture size.	CASIA	97 %	There must be tactics and procedures for accurately identifying persons for security purposes, as well as technology to support borders, airports,

				such as face, speech, fingerprint, and muscle-to-fat ratio, the iris has been determined to be the most accurate.				and ferries.
6	An efficient novel approach for iris recognition based on stylometric features and machine learning techniques	Year of research: - 2020 The inspiration for this research derives from the interdependence of biometric	To do this, we exclude Gabor wavelets and other filter banks often used in iris recognition algorithms based on John Daugman's pioneering work. We instead use machine learning	The iris of the human eye is a circular membrane that sits between the cornea and the lens. Oversampling is an element of the learning process in every fold	The existence of the FAR is one of the drawbacks of standard iris identification systems based on Daugman's pioneering work.	CASIA, MMU,IITD	Not mentioned.	We conclude that our methodology is generic with regard to the datasets used, and that ensemble learning approaches

		systems and stylometry.	approaches to identify biometric templates as numeric characteristics.	of cross-validation				outperform all other methods on all datasets evaluated.
7	Iris recognition based on elastic graph matching and Gabor wavelets	Year of research: - 2011 We describe a new approach for iris identification based on elastic graph matching and Gabor wavelets in this research.	The human iris is an annular region between the pupil and the sclera with a complicated pattern that includes arching ligaments, furrows, ridges, crypts, corona, and freckles.	Biometrics is a more dependable and competent approach for determining a person's identification based on physiological or behavioural features. When compared to other matching approaches, the	Current biometric applications employ facial traits, fingerprints, iris, palm prints, retina, handwritten signature, DNA, gait, and other characteristics to identify individuals, Which	Database used: - CASIA	Accuracy: - 95 %	The EGM approach has proven to be promising for iris recognition and is appropriate for the identification procedure.

				EGM was shown to have high correct identification rates.	consumes more time.			
8	Iris Recognition Using Multi-Algorithmic Approaches for Cognitive Internet of things (CIoT) Framework	Year of research: - 2018 The broad development of linked sensors, cloud, big data analytics, and ubiquitous sensing technologies in recent years has permitte	As innovative algorithms, the suggested DM and MAM showed great performance with feature reduction.	A massive number of gadgets and sensors have been linked in the cognitive Internet of things in today's digital era (CIoT). This effective practical paradigm enables dealing with the social and physical realities	All the gadgets and sensors should have to be connected, otherwise whole system will be inaccurate	CASIA	99.4%	n This method is appropriate for generic hardware as well as real-time applications. A general-purpose processor was used, as well as any target CPU.

		d the cognitive Internet of things.		without the assistance of others.				
9	The Biometrics System Based on Iris Image Processing: A Review	Year: 2019 Biometrics is the automatic identification and validation of a human identity based on the individual's unique traits.	DNA, face, earlobe, and iris are examples of physiological characteristics, whereas signature, voice, keystroke recognition, and gait analysis are examples of behavioural characteristics.	Biometrics is the automatic identification and validation of a human identity based on the individual's unique traits. Biometrics is made up of two components: physiological and behavioural traits.	From one scientist to the next, there is always a different technique to feature extraction. This is not the most precise.	MMU, UTIRIS	92.8%	Biometrics is the automatic identification and validation of a human identity based on the individual's unique traits.

10	Iris Recognition System by Using Support Vector Machines	Year: 2008 A dependable personal identity infrastructure is essential in the modern world to regulate access to sensitive locations or resources.	Each biometric technique has benefits and downsides in terms of usability and security.	A dependable personal identity infrastructure is essential in the modern world to regulate access to sensitive locations or resources. The purpose of picture acquisition is to create an image of the user's ocular area.	It is dependent on the personal identity infrastructure	CASIA	Not mentioned.	m. The Support Vector Machine was used as a classifier to create the user model based on his or her iris code data.
11	Iris Recognition System: A Novel	Year : 2017	The suggested method comprises extracting	There are three types of authentication	It will take more processing time	CASIA	93.7%	In biometrics, the iris is one of

	Approach For Biometric Authentication	Nowadays, the most crucial factor for systems is safety. The authentication secures the system.	biometric information using two techniques: GLCM (Gray Scale Co-occurrence Matrix) and Hausdorff Dimension (HD)	systems: password, card, and biometric. The processing system includes an Iris biometric authentication mechanism.	than other systems.			the most user-friendly biometrics since without individual interaction, the features of a person may be easily determined while simultaneously providing security and authentication in an effective manner.
12	Iris Recognition System	Year: 2014	Bitwise comparison was performed	Face, fingerprints, hand profile,	The approach's key weaknes	CASIA	80 %	In this study, we present

	for Smart Environments	Biometric approaches are used to identify persons by utilising each person's unique traits (physiological or behavioural).	to assess the trustworthiness of the bits during enrolment.	iris, retinal scanning, and DNA testing are examples of physiological traits. In the template comparison procedure, the given iris template must be compared to just 395 templates rather than 2639 base templates.	s is that if imaging circumstances alter significantly between enrollment and verification times.			and analyse a technique for smart environments based on template fusion, which requires several scans of the same eye for enrollment to acquire more trustworthy templates.
13	A Brief Review of the Iris Recognition	Year: 2017	Biometric authentication provides an individual with a	Biometrics are automated means of identifying or	A thorough search is required to recognis	Custom dataset	Not mentioned	A quick survey of the literature on iris recognit

	Systems for Developing a User-Friendly Biometric Application	Biometrics are automated means of identifying or validating a person's identification based on physiological or behavioural characteristics.	convenient, accurate, irreplaceable, and highly secure option, which gives it benefits over traditional cryptography-based authentication techniques.	validating a person's identification based on physiological or behavioural characteristics. Personal identity has become increasingly important in recent years.	e the circular parameters of both pupil and iris boundaries. Occurrence of segmentation error			ion is offered in this work. In this intriguing topic, an extensive literature review was conducted.
14.	A Survey on Iris Recognition System	Year of research: - 2016 There are many iris	Algorithm used for edge detection are canny edge detection, circuler Hough transform,	The Iris recognition technology is one of the most secure authentication techniques	The accuracy of iris recognition is mostly depends on image acquisition.	CASIA V4, MMU	Not mentioned.	Most of the iris recognition model follow the 5 basic steps-localization of

		recognition models are available by different authors. This paper is based on one of the traditional method commonly used for iris recognition.	sobel operator. To remove iris occlusion Linear Hough transform is used. For Iris normalization Faughman's rubber sheet model. For feature extraction 2D Gabor wavelets, D Gabor wavelets, 2D log Gabor wavelets are used. To match features extracted Hamming distance,	available. The uniqueness of the Iris, as well as the minimal likelihood of mistaken acceptance or denial, all contribute to the advantages of adopting Iris identification technology.	If input image is not proper then accuracy will be less.			eye, image segmentation, normalization, feature extraction and matching.

			Elastic similarity matric and Euclidean distance is used					
15	Design and Analysis of Deep-Learning Based Iris Recognition Technologies by Combination of U-Net and Efficient Net	Year of research : - 2021 In this paper, the effective deep-learning based methodology is developed for iris biometric authentication based on the U-Net model and	The proposed model used semantic segmentation technology for localization and extraction of region of interest of iris. Adaptive Histogram equalization or Gabor filtering process is used to crop eye image and strengthened optionally.	Use of deep learning model increases the accuracy.	Features enhancement may reduce the no of features to match.	Dataset used :- CASIA V1.	Accuracy: - 98 %	According to the findings of the experiments, the cropped iris picture without feature augmentation preserves more iris information to match the iris characteristics. Some of the fine iris characteristics

		semantic segmentation.						on the original eye picture may be lost if the iris image is feature-enhanced.
16.	Real-Time Iris Recognition System Using A Proposed Method	Year of research: - 2009 This research is mainly based of increasing the accuracy based of the input image and to propose best method to take	Different algorithms used are: - Thresholding to obtain pupil area. Dilation to narrow the pupil boundary. Skeleton algorithm to rarefy the edge boundary. Freeman Chain Code to trace the edge. Canny edge detection to	Accuracy will be increased when input image is taken as proposed in this paper.	Glasses and mirror makes iris localization more difficult.	Custom Dataset Taken from high resolution camera.	Not mentioned.	Resolution of camera plays an important role in image acquisition. As all other steps depends on the output of this step.

		input image.	detect the edge.					
17	A Brief Study on Evolution of Iris Recognition System	Year of research : - 2017 Study of different methods of iris recognition.	The traditional method of iris recognition uses Image Acquisition, Iris localization, iris texture analysis. Different algorithms used are: - 2D Gabor Filter , Fisher's Linear Discriminant	False match ratio and false not match ratio are being discussed efficiently in this paper.	This method is not suitable for real world application due to many error in real world implementation.	Dataset used: - CASIA	Not mentioned.	The method was designed primarily for perfect settings. Daugman's algorithm has various mistakes as a result of this goal.
18	Iris Recognition System	Year of research : - 2010 Project is a system	Use of Artificial Neural Network implementations with networks employing specific	Use of ANN makes proposed method fast for limited no of input.	With the increase in input image , the training network will take	Custom dataset	Not mentioned.	Impact on performance is the size of the input states. It is

		that can take a image(as input of human eye) and can distinguish between pupillary body and iris part of the human eye. Proposed model of this research paper is implemented using modified Canny edge detector algorithm.	guides to update the links between their nodes as supervised learning. To learn to perform a task back propagation or Propagation of error is used for ANN. Also called Backward propagation of errors. Algorithm used: - Canny edge detection, Hough transform algorithm, Daugman's algorithm.		more time.			inevitable that the more input symbol sets the network must be trained on, the more sensitive it is to mistake.

19	A Review on Iris Recognition	Year of research: - 2014 The aim of paper is to discuss the steps involved in iris recognition on used by different researchers.	Almost all models uses same steps: - Image acquisition, Pre-processing, Feature extraction, Template matching	A brief description about all the method available for iris recognition is given.	Does not proposed its own model.	CASIA MMU	Not mentioned.	Performance of above steps depends on different algorithms used.
20	Research and Implementation of Iris Recognition Algorithm	Year of research: - 2012 Proposed method	To identify iris outer iris edge Circular detection operator is used.	Accuracy is further increased from traditional method to this method of iris	Training of neural network is done multiple time to increased	Database used: - CASIA	Accuracy: - 90.74 %	Tests suggest that the algorithm can enhance the speed and

		uses both method of wavelet analysis and equation of circle to get inner edge boundary.	Wavelet packet decomposition and RBF neural network is used.	recognition.	accuracy.			accuracy of recognition. Nevertheless, due to the restricted number of experimental samples, a significant number of samples are required to run tests to evaluate the algorithm's resilience.

21	Iris recognition in visible spectrum based on multi-layer analogous convolution and collaborative representation	Year of research: - 2019 This paper uses Collaborative representation method to extract and classify iris texture.	A visible light mobile iris recognition method based on multi-layer analogous convolutional structure and collaborative representation is used.	The algorithm proposed in this paper provides better accuracy, AUC and EER compared to the iris recognition method such as Gabor, PCA, SRC, MACS + SRC and CRC.	Proposed method is complex to implement.	Not motioned	Not mentioned	This proposed algorithm provides better accuracy, AUC and EER
22	A field study of the accuracy and reliability of a biometric iris recognition system	Year of research: - 2013 The goal of this paper is to evaluate the	Mobile-Eyes. Total of 2710 identification and verification attempts in which 5540 iris	False rate is very less.	Dataset used is small.	Custom dataset of 277 iris.	The accuracy rate is about 99.03 %	Conclusion: - There were no false verification or false identification.

		biometric iris recognition system :	comparisons.					
23	Computer Vision and Image Understanding Feature-domain super-resolution for iris recognition	Year of research : - 2013 Proposed a framework for applying feature-domain super-resolution to biometric.	Image processing is fast due to conversion of image from spatial domain to feature domain. This model is using nonlinear 2D Gabor phase-quadrant feature.	Superior performance of the proposed approach (with an EER of 0.5%) against the simple feature-level fusion scheme (with an EER of 0.8%)	Image enhancement technique can be better.	MMU data set.	Not mentioned	The use of this method has shown to improve the recognition performance with respect to linear feature in feature domain super resolution.
24	A survey of iris datasets	Year of research : - 2021	Currently at the time of this research out of 158 only 81	Gives a brief information about all the iris	It only searched the Web of Science	All dataset available present	Not mentioned	Iris databases produced by the

		Paper consist a comprehensive overview of all the existing publicly available dataset of iris.	were available. Around 158 different datasets are referenced from 689 different research articles. 57.37% of the research evaluated their method only on one database, 30.35% of publications were evaluated on custom (non-published) databases	dataset available will help new researchers to choose dataset more suitable to their need.	online library for bibliometric data. Another limitation is that we obtained the dataset through e-mail and phone contacts.	for iris recognition.		Chinese Academy of Sciences (CASIA) are the most popular and have been used 453 times in 305 articles, or in more than 44% of the studies.
25	Human Iris Recognition for	Year of research: -	To encode the distinctive pattern of	This proposed method can	Two iris templates were found to	Dataset used: - LIE & CASIA	False accept	This iris recognition system

	Biometric Identification	2007 This Iris recognition model uses an automatic segmentation system that is based on the Hough transform.	the iris into a bit-wise biometric template, phase data from 1D Log-Gabor filters was recovered and quantized to four levels. The Hamming distance was used to classify iris templates. Iris recognition system that was evaluated utilising two databases of grayscale eye photos in order to validate iris recognition technology's stated	decrease False accept rate of and false reject rate.	match if a test of statistical independence was failed.		rate of 0.005 % and false reject rate of 0.238 %	was tested using two databases of grayscale eye images in order to verify the claimed performance of iris recognition technology.

			performanc e					
2 6	Iris recogniti on using artificial neural networks	Year of research : - 2011 This research uses a methodo logy for pre-processi ng iris images and the design and training of a feedfor ward artificial neural network for iris recogniti on.	To reduce the cost of ANN iris image has been partitioned in three ways. Horizontal strip partitioning(row) Vertical strip partitioning (columns). Block partitioning	Cost of ANN is reduced.	Using different irises of different colours may enhance the ANN's accuracy more.	Custom databas e of 20 brown coloure d iris images of differen t person is used.	Acc urac y achi eve d = 93. 33 %	Using different irises of different colours may enhance the ANN's accurac y more.

27	A Brief Research and Lookback of the Development of Iris Recognition	Year of research: - 2019 The paper contains a study of both traditional method as well as the deep learning method	From this study says that traditional approach plays significant role but deep learning method is making great difference from traditional. Also tells about pros and cons of both the method.	Gives the difference between traditional approach and deep learning approach.	Does not analyse other methods available for iris recognition.	Dataset used: - CASIA. V4	Accuracy rate of 95.9% in the CASIA. V4 data set	Most challenging step of iris recognition is image acquisition as distance, lighting affect heavily on the result.
28	How Iris Recognition Works	Year: 2004 Even the	The iris starts to develop in the third month of pregnancy	The iris has the significant mathematical benefit of having	It is more time consuming and is depende	Custom dataset	Not mentioned	By comparing the HD distri

		finest current algorithms have been demonstrated to have inaccuracy rates of 43%-50% for "mug shot" photographs taken at least one year apart.	[13], and the structures that make up its pattern are mostly complete by the eighth month, but pigment accumulation can continue into the first postnatal years.	vast pattern variety among different individuals. Iris recognition methods take advantage of binomial combinatorics' exceptionally quick attenuation of the HD distribution tail to support very massive database searches with no false matches.	nt on the personal identity infrastructure.			butions for same versus different irises, the general "decidability" of the problem of distinguishing people by their iris patterns is shown.

29.	Iris Recognition Using Convolutional Neural Network	The development of artificial intelligence offers a significant chance to expand the use of iris recogniti on for protecting people's personal information. The suggested CNN-based iris identific	Convolutional neural network, PCA-CNN	Future study will centre on increasing the variety of iris samples while preserving prediction accuracy and improving computing efficiency.	The black box aspect of the model, in which the inner workings of the system are not understood, the small number of iris pairs utilised to train it, the computational complexity that results in a lengthy training period.	CASIA-irisV4)	99 %	With a very high prediction accuracy, the suggested CNN-based iris identification system can recognise the iris of up to 20 different people, making it ideally suitable for usage in future works.

		ation system is particularly well suited for usage in future works because it can accurately forecast the iris of up to 20 different people.						
30.	Iris Recognition Technology	The procedure digitises the features of the iris, creates a 256-byte code from the photogra	lmaqe acquisition, Iris definition, Zones of analysis, lmaqe analvsis, Hamminq distance calculation, Identification or	Demonstrates potential to improve upon prior deficiencies and to provide exceptionally high confidence positive	It should not be expected that the dynamic range of the system's acquisition and processing capabilit	Database with 538 unique IrisCode files.	99.99%	Using the iris of a human eye, highly accurate, positive personal identification is now

		phs, and compares it in less than two seconds to the full database. A non-duplicable organ with around 400 degrees of freedom, or quantifiable variables, is created by the numerous contraction furrows, coronas,	reiection, Calculation of confidence level	identification of an individual without contact or invasion.	ies will make up for conditions.			possible.

		vasculature, freckles, etc.						
31.	Biometric Personal Identification Based on Iris Recognition	The already existing technology used in Iris Recognition has been illustrated and their performance has been evaluated. Haar wavelet performed best among mother wavelets. Gabor filter	Haar wavelet, Gabor Filter, Hamming distance, PNN, LVQ	Haar wavelet, Gabor Filter, Hamming distance, PNN gave best results.	Certain training patterns were not retained by LVQ.	CASIA and UBIRIS	92.6%.	Some methods like Haar Wavelets, PNN etc gave accurate results while certain training patterns were not recognised by LVQ methods.

		recognised most accurately consuming more time and space. Hamming distance gave best result as per simplistic and recognition rate. PNN accurately gave best result with Haar 98.15% db10 and syml 96.3%						

		LVQ was not able to learn some patterns with recognition rate of 92.6%						
32.	Cross-spectral iris recognition using phase-based matching and homomorphic filtering	Several spectral bands are used in cross-spectral iris recognition to gather comprehensive information about the human iris. Phase-only correlation	Homomorphic filtering, segmentation, normalization, EER and ROC curve parameters	It was intended to build a more reliable approach and combine the BLPOC and POC matching scores with Gradientface-based normalisation in future work to significantly	Phase correlation between NIR and VIS iris pictures is challenging to achieve.	VIS-NIR dataset	99.61%	Phase-based matching and homomorphic filtering were proposed as a cross-spectral iris identification approach to improve recognition performance.

		(POC) and band-limited phase-only correlation (BPLOC) has been presented. The experimental findings showed that, with an EER of 0.59%, the performance of the suggested framework utilising a VIS-		enhance the performance of crossspectral iris recognition.				

		NIR dataset was the best among the baseline techniques taken into consideration in the study.						
33.	A Novel Unified-Noise Algorithm for Iris Recognition	A technique that unifies the noise between login iris and test iris images with the goal of promoting the wider use of matching method	Gabor filters, The standard deviation method, Hough transformation, normalization, feature matching	Iris matching calculation results have a better impact on direct gray-level surfaces.	Demands that computers handle information more quickly.	Manual	High Accuracy	It may instantly conduct a rapid comparison and matching between the registered images and the sampled images.

		has been used in this paper. About the methods for extracting the core features of iris images, one technique is to compare the features of an iris library, and another is to compare the distances between two iris images.						

34.	An End-to-End Deep Neural Network for Iris Recognition	In this paper, Iris segmentation, normalisation, feature extraction, and matching are all integrated into a single network in the iris identification application of the combined network model based on EfficinetNet-b0. To increase decision transpar	EfficinetNet-b0, CNN, DNN	It optimises the use of the target resources and strikes a balance between the network model's width, depth, and resolution.	Despite the possibility of high-resolution class detection, these challenges depict models rather than making predictions about a specific input image.	CASIA Thousand and Mmu2	High Accuracy	A deep network model based on CNN, Efficient Net-b0 was proposed for Iris recognition.

		ency, "visual interpretation" technology has been introduced.						
35.	Iris recognition systems – A review	The works/research papers performed by various researchers was analysed and some drawbacks were pointed out like noise, accuracy, unwanted data etc.	Various method for iris recognition such as NMF, NIR etc were reviewed	Several research articles in the relevant subject were gathered from various sources, reviewed for length and breadth, and a review paper in the form of this one is now being given.	Very expensive, a few methods fail when there is noise like reflection. Less effort is put into improving performance and accuracy, which results in numerous capture errors, etc.	NA	NA	In a summary, this research provides a brief overview of the many works produced by various authors in their research articles.

36.	Neural Network Approach to Iris Recognition in Noisy Environment	The two algorithms proposed in this research study are a unique technique for iris image noise removal and an approach that combines Local Binary Pattern (LBP) and Gray Level Co-occurrence Matrix to extract texture feature	Local Binary Pattern (LBP) and Gray Level Co-occurrence Matrix (GLCM)	Due to the efficient noise removal approach and feature extraction suggested in this research, outcomes are enhanced.	If other noisy artefacts that were prevalent across different databases had been eliminated, this effort would have been more effective.	CASIA and MMU	96.5%	A methodology for improving the performance of the iris recognition system in noisy imaging environments was proposed.

		information (GLCM). Our suggested method yields the highest recognition rate of around 96%, has a low error rate, and takes less time to execute.						
37.	A New Texture Analysis Approach for Iris Recognition	Neighbourhood-based Binary Pattern is method proposed by this paper. This is	Neighbour-based Binary Pattern	We will research combining the NBP method with other methodologies, such as the Gabor transform,	The test images and reference was less in number which may give less accuracy	CASIA	High Accuracy	It was observed that the suggested NBP approach is intriguing since it gathers

		inspired from LBP. The propose d NBP approac h is intriguin g since it gathers relative relevant data from pixel neighbo urs.		in further publicatio ns.	when worked on large-scale data.			the relative pertinen t informat ion from pixels' neighbo urs.
3 8.	An effective authentic ation scheme for a secured IRIS recogniti on system based on a novel encoding	In this work, a novel encodin g strategy is used to offer a novel model that creates a One Time	Preprocessi ng, localization, normalizati on, Gabor filter, Hamming distance and Euclidian distance	Entropy-based analysis is used to assess the proposed model's security, and BS score analysis was used to statisticall	To increase the security of other biometri c modaliti es, this work can be expande d.	CASIA	Nea rly %	In this work, an OTIC encodin g scheme-based authenti cation model for iris recognit ion systems

	techniqu e	Iris Code (OTIC) for each user authenti cation. The Iris Code will be unique for each authenti cation session under the propose d paradig m, which generate s it using a pseudo-random number generato r, timesta mp, and rotation		y measure the severity of common vulnerabil ity assaults.				is develop ed.

		parameter.						
39.	Research use of electronic health records: patients' perspectives on contact by researchers	The purpose of this study was to examine patients' attitudes and perspectives towards prospective contact with researchers with access to their electronic health records (EHRs). EHRs may also be utilised for non-	A questionnaire, Educational videos, A moderator's guide, A worksheet	Future research should look at how other people, including scientists, doctors, and ethics experts, see the same problems.	The research participants was concentrated only from southeastern US	15 focused groups from southeastern US	NA	Only self-reported race and having a regular healthcare provider showed statistically significant differences.

		research reasons, therefore stakeholder viewpoints must be empirically studied.						
40.	Research on an Electronic Medical Record System Based on the Internet	The management of medical records' information is accomplished using an electronic medical record system. Medical records and advice can be accessed	Doctor Function Module, Patient Function Module, Administrator Function Module	Sharing paper medical records with other hospitals is challenging. Other hospital doctors can see the hospital's medical records with permission from the electronic medical	Electronic data record can be leaked online but physical paper can be kept safely with concerned person.	NA	NA	The management of medical records' information is accomplished using an electronic medical record system. Medical records and advice can be accesse

		online by patients and doctors at any time and from any location. The electronic medical record system makes it simple for doctors to save, retrieve, and browse patients' medical records. They can use the electronic		record system.				d online by patients and doctors at any time and from any location.

		medical records to undertake research and statistical analysis.						
41.	Iris Recognition: An Emerging Biometric Technology	In this study, automated iris recognition is examined as a biometrically based technique for identity verification and personal identification. This paper's main body	Image Acquisition, Iris Localization, Pattern Matching, Recapitulation	Current machine vision research has produced automated systems that move closer to fulfilling this possibility.	It is difficult to have enhancements while maintaining compact, efficient, and low-cost implementations.	10^2 irises	NA	These systems are rather efficient and small, and preliminary testing has revealed promising results. Existing systems operate somewhat close to the operator

		describes problems with the construction and operation of such systems. The descriptions of existing systems are for illustration purposes only.						and require a considerable amount of operator input. Thus, controlled assessment scenarios are where they work best.
42.	Enhanced Iris Recognition Method by Generative Adversarial Network-Based Image	In this study, a technique for improving the quality of iris photographs has been put forth that	Handcrafted Feature-Based Methods, Deep Feature-Based Methods, DeblurGAN-based methodDeblurring of Iris Images	The great accuracy of iris recognition can be derived from the different iris patterns in the case of iris recognitio	There are many instances where iris patterns are not evident when iris recognition using visible light	CASIA, NICE.II training dataset and MICHE dataset	NA	Future research will examine alternative strategies for enhancing recognition perform

	Reconstruction	involves blurring the iris region and deep-learning-based deblurring.		n using a near-infrared light image.	image, as in our investigation.			ance by automatically creating the iris region obscured by reflected light, eyelash, eyelid, and glasses frame.
43.	DeepIris Net: Deep Iris Representation with applications in iris recognition and cross-sensor iris recognition	A resilient, discriminative, concise, and incredibly simple-to-implement iris representation with state-of-the-art accuracy	CNN, Deeply-learned iris pattern (DLIP)	The DeepIrisNet network is resistant to modest segmentation and transformation differences as well as sensor interoperability.	In the event of non-linear transformations, the entire process of matching irises by shifting left and right might be incredibl	ND-iris-0405 [2], ND-CrossSensor-Iris-2013	99.9%	Strong baseline performance based on descriptor is greatly outperformed by DeepIrisNet, which also generalises well to new datasets.

		is provided by DeepIrisNet, which is particularly effective in modelling the microstructures of the iris.			y time consuming and even ineffective.			The models are resistant to common segmentation and transformation changes as well as cross-sensor recognition.
44.	Review of Iris Recognition Techniques	There is still a lot of room for improvement in the low constraint IRIS recognition area and a lot of work that has	Localization, Boundary segmentations, Iris Normalization, feature extractions, training, and matching.	The most reliable biometric technology is iris recognition.	Iris Segmentation with high accuracy is still an issue.	CASIA, UPOL, Bath, MMU, and UBIRIS	High Accuracy	Each IRIS recognition system's fundamental design is the same: finding the eye, segmenting, converti

		to be done.						ng or normalising radial to rectangular data, feature extraction, and matching.
45.	A Review of Iris Recognition System ROI and Accuracy	The accuracy of their iris recognition technology is displayed in this report along with all the places that have been studied.	Pre-processing using localization, Eualization, Normalization, Feature Extraction using B/w ratio, GLCM, Gabor filter	Iris Recognition system has the highest accuracy	Unnecessary observations were performed on other organs	Manual	97.78 %	In this study, they employ feature extraction from the Gray Level Co-Occurrence Matrix (GLCM) and Bayesian regularisation (BR) classification to

								determine if the person has the disease or not.
46.	Iris Recognition After Death	This study, which involved 1,200 near-infrared and 1,787 visible-light samples taken from 37 deceased people stored in mortuary conditions, was done to identify human iris postmortem.	VeriEye, IriCore, MIRLIN, OSIRIS	The investigation of post-mortem iris recognition in this paper is the most thorough. The final remarks are offered in the following paragraphs as responses to the questions that served as the catalyst	The biggest drawback we are aware of is the very tiny dataset.	37 deceased subjects	98 %	The institutional review board approved this study, and the authors made sure to adhere strictly to the Helsinki Declaration's ethical precepts.

				for this study.				
47.	A Unifying Framework of Mining Trajectory Patterns of Various Temporal Tightness	In this research, we propose what we refer to as "unifying trajectory patterns," a framework for mining trajectory patterns with varying degrees of temporal tightness.	UT-pattern, granularity adjustment	An extensive range of temporal tightness can be covered by UT-patterns.	A data set may include groups of moving objects that arrive at various locations over the course of one minute or one hour.	Synthetic Data set 2	96.42 %	The algorithm's primary benefit is the ability to identify trajectory patterns with varying degrees of temporal tightness, including time-constrained, time-relaxed, and time-

								independent patterns.
48.	Text Detection and Recognition for Images of Medical Laboratory Reports With a Deep Learning Approach	In order to extract textual information from photos of medical laboratory reports, this research provides a deep-learning-based method that could assist doctors in resolving the data-	Text Detection, Text Recognition,	In upcoming efforts, the structured health records that are recovered from document images will be used for medical data mining to enhance health services.	Image resolution can have a significant impact on text detection investigations.	Chinese Medical Documents Dataset (CMDD)	98.6%	This article describes a deep learning method for extracting text from photos of medical laboratory reports.

		sharing issue.						
49.	Iris Detection And Recognition By Image Segmentation Using K-Means Algorithm And Artificial Neural Network	With high precision and a significant reduction in time and computational load, this technique is able to distinguish the iris from the rest of the eye's image. Our scheme can also determine	K-means algorithm, ANN Detection,	In addition to saving memory space while storing the picture segmentation, our idea reduces computational costs and processing time.	Entire contour was not used for while obtaining the results.	Manual	High Accuracy	The entire operation enhances the existing segmentation algorithms in a number of ways, making it an algorithm that can be simply extended to additional biometric methods.

		whether the extraction of characteristics after segmentation is feasible.						
50.	Iris Liveness Detection: A Survey	The methods used to detect the liveness of an iris were divided into two categories: feature-level methods and sensor-level methods. Sensor-level methods use	Sensor-Level Methods, Feature-Level Methods	Various Iris spoofing means can be avoided like photo aattacks, video attacks, Contact-Lens Attacks, Artificial-Eye Attacks	The hypothesis may no longer hold true and these types of techniques may be rendered invalid in the future with the advancement of printing and manufacturing	CASIA-Iris, LivDet-Iris, VSIA and PAVID	NA	While feature-level methods are acknowledged as being affordable and flexible, sensor-level methods are considered as having an acceptable detection rate

		additional hardware to detect the subjects' vital signs while feature-level methods use software-implemented algorithms to assess the presentation's liveness.			technologies.			but are relatively expensive and rigid.

DIGITAL IMAGE FUNDAMENTALS

Many of the techniques used in digital image processing—or digital picture processing, as it was formerly known—were created in the 1960s at Bell Laboratories, the Jet Propulsion Laboratory, Massachusetts Institute of Technology, the University of Maryland, and a few other research institutions. These techniques have applications in satellite imagery, the conversion of wire-photo standards, medical imaging, videophone, character recognition, and picture enhancement. The goal of early image processing was to raise the image's quality. Its goal was to enhance individuals's ability to be seen by other people. A low-quality image is the input for image processing, and the result is an image of higher quality. Image enhancing, restoration, encoding, and compression are frequently used image processing techniques. US Jet Propulsion Laboratories made the initial successful application.

On the many lunar photographs returned by the Space Detector Ranger 7 in 1964, they applied image processing techniques including geometry correction, gradation transformation, noise reduction, etc. while accounting for the sun's position and the moon's surroundings.

The computer's effective mapping of the moon's surface has had a significant positive influence. Subsequently, more intricate image processing was applied to the over 100,000 photographs that the spacecraft brought back, yielding the topographic map, colour map, and panoramic mosaic of the moon, which produced astonishing results and set the stage for a successful landing of astronauts on the moon.

Yet, the computing hardware at the time made processing quite expensive. This changed in the 1970s when more affordable computers and specialised gear became accessible and digital picture processing spread. Due to this, pictures were eventually handled in real-time for some specific issues like converting television standards. General-purpose computers began to replace specific hardware for all but the most specialised and computationally expensive tasks as they got faster. Digital image processing is now the most widely used type of image processing because to the quick computers and signal processors that became accessible in the 2000s. This is because it is not only the most flexible way, but also the least expensive.

When processing numerical data, computers process it more quickly and precisely than humans. However, when it comes to recognising skills, people outperform computers. The human brain is so developed that we can easily and quickly detect items in a few of seconds.

Even when a friend's look has changed over the course of 10 years, we may still be able to recognise them because of how humans acquire information for identification.

Medical pictures are utilised by doctors to detect issues and deliver therapy, therefore images are not just employed for entertainment. Modern technology makes it possible to photograph almost every anatomical feature, which is incredibly helpful for clinicians who utilise medical pictures to identify illnesses and administer treatments. Virtually all anatomical features can now be imaged thanks to contemporary technology, which greatly aids doctors in delivering better care. To identify offenders, forensic imaging software process fingerprints, faces, and irises. Satellite photos are used by commercial applications to find the earth's mineral resources. As a result, pictures are widely used in our daily lives.

Improving the quality of graphical information for better human interpretation is one of the main goals of image processing. Facilitating automated machine interpretation of pictures is another goal.

The amplitude of F at each given pair of coordinates (x,y) is referred to as the intensity of that picture at that location. An image is defined as a two-dimensional function, F(x,y), where x and y are spatial coordinates. We refer to it as a digital image when F's x, y, and amplitude values are all finite.

In other words, a two-dimensional array specially set up in rows and columns may be used to describe a picture. A digital image is made up of an infinite number of discrete pieces, each of which has a unique value at a unique place. These components are also known as pixels, image elements, and picture elements. The most frequent usage of a pixel is to indicate a component of a digital image.

Digital image processing is the practise of manipulating digital pictures using a computer. Each element in a digital image has a specific position and value and is made up of a limited number of them. These components are referred to as pixels, pels, picture elements, and image elements. The most often used word to describe the components of a digital image is "pixel."

At Bell Laboratories in 1969, Willard S. Boyle and George E. Smith created the charge-coupled device. They discovered, while studying MOS technology, that an electric charge could be held on a small MOS capacitor and was akin to the magnetic bubble. They attached a sufficient voltage to them in order to step the charge from one to the next because it was very simple to

construct a row of MOS capacitors. The first digital video cameras for television transmission used a semiconductor circuit called a CCD.

An image is a two-dimensional function that represents a measurement of a particular characteristic, such as the brightness or colour of the scene being viewed. A three-dimensional scene is projected into a two-dimensional projection plane to create an image.

Types of an image

- Binary Image: As its name implies, a binary picture consists of just two pixel elements: 0 and 1, where 0 stands for black and 1 for white. Another name for this image is monochrome.
- Black and White Image: The term "black and white image" refers to an image that solely has the colours black and white.
- 8 bit Colour Format: The most popular image format is this one. It is frequently referred to as a grayscale image and contains 256 different colour shades. In this format, 0 denotes black, 255 denotes white, and 127 denotes grey.
- 16 bit Colour Format: It's a format for coloured images. It comes in 65,536 different colours. High Color Format is another name for it. The distribution of colour in this format differs from that in a grayscale image.

The intensity of the picture at any given position is defined as the amplitude to the two-dimensional function f(x,y), where x and y are spatial (plane) coordinates. The intensity of monochromatic pictures is frequently described using the phrase grey level. Combining several 2-D photos creates colour images. In the RGB colour scheme, for instance, a colour image is made up of three separate component images (red, green, and blue). Due to this, many methods created for monochrome photos may be applied to colour images by processing each of the three component images separately.

In terms of amplitude, x, and y coordinates, a picture can be continuous. Such a picture must have its coordinates and amplitude digitalized in order to be converted to digital form.

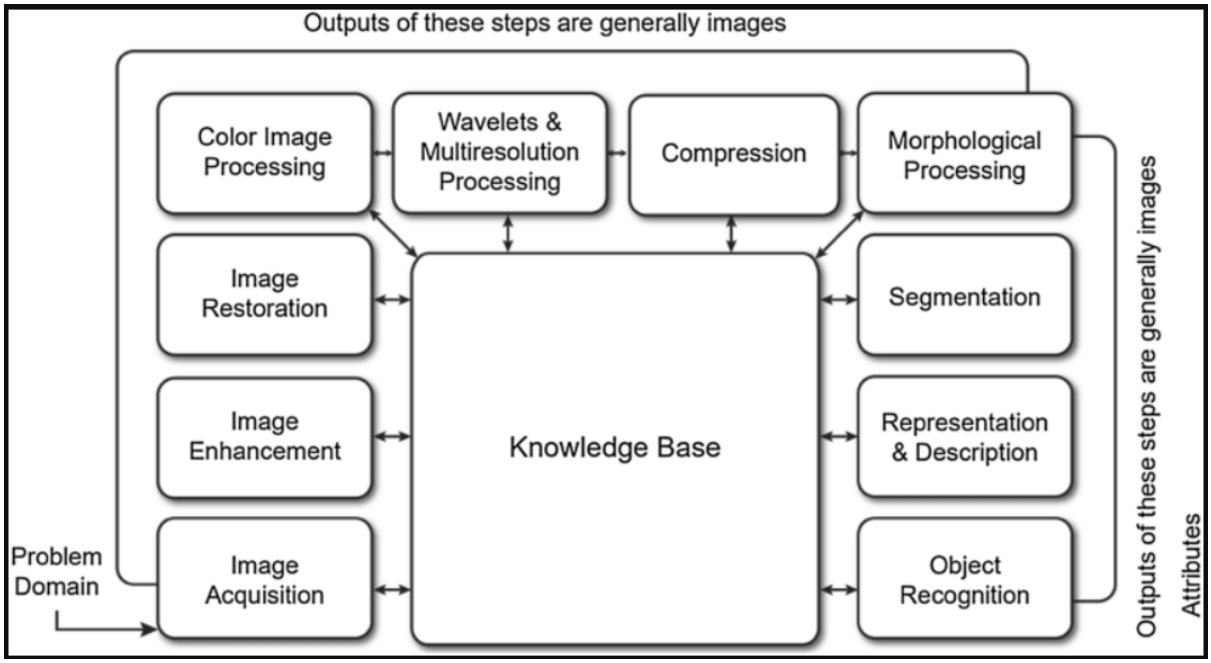
In computer science, digital image processing uses algorithms to perform image processing on digital images to extract some useful information. Digital image processing has many

advantages as compared to analog image processing. Wide range of algorithms can be applied to input data which can avoid problems such as noise and signal distortion during processing. As we know, images are defined in two dimensions, so DIP can be modeled in multidimensional systems.

The term "digital image processing" refers to the use of a digital computer to process digital images. In order to obtain an improved image or to extract some important information, we can also state that it is the usage of computer algorithms.

The employment of algorithms and mathematical models in digital image processing allows for the processing and analysis of digital pictures. Digital image processing aims to improve picture quality, extract useful information from photos, and automate operations using images.

A digital image is made up of an infinite number of discrete pieces, each of which has a unique value at a unique place. These components are also known as pixels, image elements, and picture elements. The most frequent usage of a pixel is to indicate a component of a digital image.



The kinds of pictures that interest us are produced by the interaction of a "illumination" source and the parts of the "scene" that are being photographed, which reflect or absorb energy from the illumination source.

Concept: The combination of input electrical power and a sensor material that responds to the specific sort of energy being detected causes the incoming energy to be converted into a voltage. The response of the sensor(s) is represented by the output voltage waveform, and each sensor's response is digitise to produce a digital quantity.

It is possible that differentiating the number of bits allotted to each channel will result in more optimal storage because the brain may not always distinguish differences in each channel to the same degree as in other channels. In particular, for RGB images, compressing the blue channel the most and the red channel the least may be preferable to giving each equal space.

Studies have shown that the human retina really uses the red channel to differentiate detail, along with the green channel in a lesser degree, and employs the blue channel for background or ambient information. This sort of "preferential" compression is the outcome of these findings.

Pixels, the building blocks of digital colour pictures, are formed of combinations of basic colours that are encoded as a sequence of codes. The grayscale image of the same size as a colour image,[citation needed] comprised of only one of these fundamental hues, is referred to in this context as a channel. For instance, a red, green, and blue channel will be present in a picture taken with a typical digital camera. There is just one channel in a grayscale picture.

Raster bands are a common name for channels in geographic information systems. Convolutional neural networks employ feature maps, which are a similarly related idea.

Red, green, and blue make up the three channels of an RGB picture. RGB channels are utilised in computer displays and image scanners, and they closely correspond to the colour receptors in the human eye.

The fundamentals of digital image processing are the basic procedures we use to process digital images. Images may be seen as functions with two-dimensional magnitude values, and these values are used to create the components of digital images. These components include pixels, image components, and picture components. Since the value of these components is finite, the image is referred to as a digital image. The process of altering a picture's components to improve its quality or extract information from it qualifies as digital image processing. Thus, the fundamentals of digital image processing are its basic processes.

Red, green, and blue channels each have 8 bits if the RGB picture is 24-bit (the industry standard as of 2005), which means the image is made up of three separate images, one for each

channel, each of which may hold discrete pixels with typical brightness intensities between 0 and 255. Each channel contains 16-bit per pixel colour, or 16-bits of red, green, and blue, if the RGB picture is 48-bit (a very high color-depth).

The initial duty performed by the imaging system is to gather the incoming energy and concentrate it onto an image plane. If light is the source of illumination, the imaging system's front element is a lens that focuses the seen scene onto its focal plane.

The sensor array, whose outputs are proportional to the integral of the light received at each sensor, is coincident with the focus plane. These outputs are swept by digital and analogue circuitry and converted to a (video) signal, which is subsequently processed by another component of the imaging system. A digital picture is the result.

DIGITAL IMAGE REPRESENTAION

Analog and digital pictures are the two types of images utilised in medical imaging.

The kinds of visuals we view as humans are analogue ones. Photographs, paintings, television pictures, and all of our medical images captured on film or shown on other displays, including computer monitors, are examples of what they entail. An analogue picture is made up of multiple brightness (or film density) and colour levels. Typically, it is continuous and not divided into several little bits.

Digital pictures are saved as a series of integers. A matrix or array of tiny picture components, or pixels, make up the image. A number is used to describe each pixel.

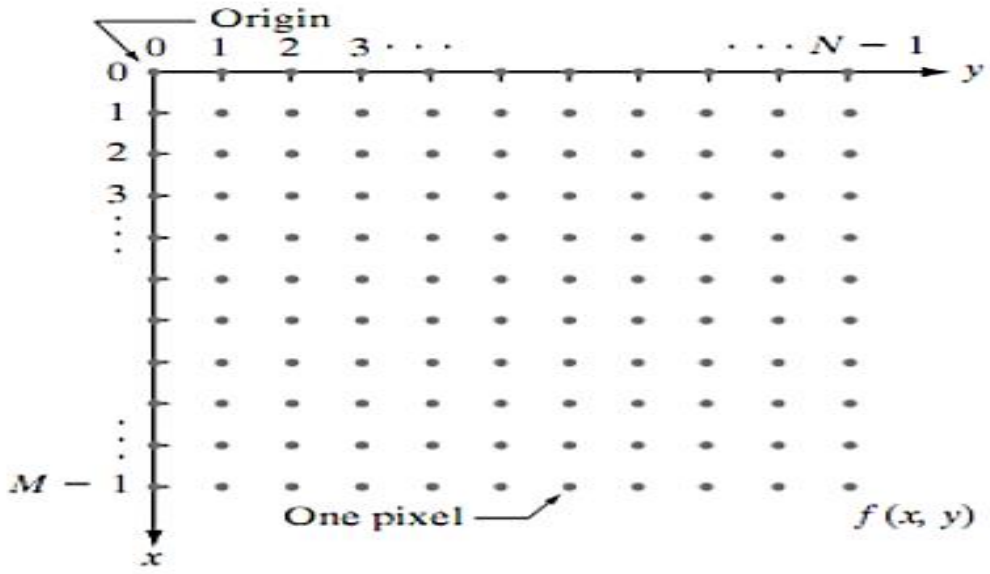


Figure 1: General representation of an image pixels

Digital pictures can be represented in two main ways. Assume that a digital picture with M rows and N columns is created by sampling the image f(x, y). X and Y's coordinate values are now discrete quantities.

When taking into account integer values for these discrete coordinates, it digital representation is created. As a result, the coordinates at the origin have the values (x, y) = (0, 0). The following coordinate values along the image's first row are shown as (x, y) = (0, 1).

In imaging and computer systems, a digital picture is represented by numbers in the form of binary digits, or bits. Here, the basic framework of a digital image is shown. It is initially separated into a matrix of pixels. Then, a set of bits is used to represent each pixel.

The picture f(x,y) is typically split into X rows and Y columns. The coordinate ranges are therefore x=0,1,...,N-1 and y=0,1,...,M-1. Pixels are found where the rows and columns meet. The term "pixel" is short for "picture element". Picture element, pixel, and pel are all interchangeable terminology. Millions of pixels make up the average digital image. As they

come together to create a digital image, pixels are regarded as the foundation of digital images. Data in pixels is discrete.

Image Representations

Image in black and white

- a two-bit single-color plane

Colour Image

- Three colour planes, each with eight bits
- YIQ, RGB, CMY, etc.

Grey Scale Image

- 8-bit single colour plane

Compressed pictures

- TIFF, JPEG, and BMP

Indexed colour image

- single plane that indexes a colour database

The context affects their meaning. A pixel may be a single sensor, a photosite (a physical component of a digital camera's sensor array), a component of a matrix, or a display element on a monitor.

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}.$$

Figure 1.2: Image pixel represented as spatial coordinates

Vertical resolution in a digital picture refers to the number of rows. Horizontal resolution is the quantity of columns. The size of the image is indicated by the number of rows and columns. The rectangular pixel dimensions of the array are a common way to indicate the size of the image. Images come in a variety of sizes.

Human seeing requires analogue visuals. In order to show and examine the pictures, all medical imaging techniques that generate and use digital images must transform the data to analogue. Humans are unable to learn much from just staring at a digital image or a display with an enormous amount of statistics.

A digital image is made up of a matrix of several tiny pieces, or pixels. A number is used to describe each pixel. The brightness or colour that we will see when the digital image is transformed into an analogue image for display and viewing is, in general, connected to the pixel value. The modifications made to the window control, which are covered in other modules, often define the real relationship between a pixel's numerical value and its displayed brightness at the time of viewing.

The range of values that can be expressed with a certain number of bits is one of the drawbacks of utilising binary numbers (binary digits). Using four bits, we study the technique in this article. As we can see, four bits may be tagged in 16 distinct ways, allowing for a total of 16 possible values.

By employing additional bits, the range of potential values that may be written is expanded. The range (number of potential values), as the equation demonstrates, is equal to two multiplied by itself or raised to the power of the number of bits. With every extra bit utilised, the range of potential values doubles, as we will shortly discover.

The amount of bits made accessible in the digital system to represent each pixel in the image is known as the pixel bit depth. This is an example of merely using four bits. As a pixel may only have 16 possible values with four bits, this is smaller than what would be utilised in any genuine medical imaging (brightness levels or shades of gray). As we will see in a moment, medical pictures call for more.

Here, three photos with varying bit depths and brightness levels are compared. One bit per pixel is used to display the first picture. There are just two potential values for a pixel: BLACK or WHITE. There are only 16 possible brightness levels available for the second image's four

bits per pixel (shades of gray). Using eight bits per pixel, the third picture may display 256 distinct brightness levels. For most purposes, this is suitable for human sight.The size of the pixel really blurs a picture when it is in digital form. This is due to the "blurring together" of all anatomical detail into a single pixel, which is then represented by a single integer. The degree of blurring introduced to the imaging process by the digitization of the picture is determined by the physical size of a pixel in relation to the anatomical objects.Here, we can observe that a picture made up of smaller pixels (less blurring) shows far more detail than a picture made out of bigger pixels.

The ratio between the size of the picture matrix and the actual image size determines the size of a pixel (and image detail). Instead of the size of a projected picture, image size refers to the parameters of the patient's body's field of vision (FOV). The number of pixels along the width and length of a picture is known as the matrix size. Although this is possible in both directions, it will often be different when producing roughly square pixels from rectangular pictures.

The sum of two components determines an image's numerical size (number of bits):

The amount of pixels is calculated by multiplying the image's pixel length and width. "Bit depth" (bits per pixel). This is typically between 8 and 16 bits per pixel, or 1-2 bytes. The relevance of numerical size is that it affects how much memory and disc storage are needed, as well as how long it takes to analyse and distribute an image.

The method of picture compression involves shrinking the numerical size of digital images. For picture compression, a variety of mathematical techniques are applied. The amount by which the numerical size is decreased is determined by the level of compression. It relies on the choice of compression level and compression technique. Lossless compression, which is frequently employed in many medical applications, preserves picture quality. Lossee compression causes some picture quality reduction and should only be used cautiously for diagnostic images.

Images have the single most significant function in human perception, which is not unexpected given that vision is the most developed of our senses. Imaging devices, on the other hand, nearly completely cover the electromagnetic (EM) spectrum, from gamma to radio waves, unlike humans, who are restricted to the visual range. They are able to work with visuals produced by sources that people aren't used to connecting with images. They consist of

computer-generated pictures, electron microscopy, and ultrasound. Digital image processing therefore has a broad and diverse range of applications. Where image processing ends and other related fields, such image analysis and computer vision, begin is not generally agreed upon by writers.

Image processing is sometimes distinguished as a discipline where pictures serve as both the input and the output of a process. This threshold, in our opinion, is somewhat arbitrary and restricting. The calculation of an image's average intensity, which produces a single value, would not be regarded as an image processing activity, for instance, under this definition. On the other side, there are disciplines like computer vision whose end objective is to have computers mimic human vision, including learning, being able to draw conclusions, and acting on visual inputs. This field is a subset of artificial intelligence (AI) with the goal of imitating human intellect.

The line connecting computer vision and image processing does not have any distinct bounds. Yet, one effective paradigm is to think of low-, mid-, and high-level computerised operations as three different sorts along this continuum. Primitive procedures like picture pre-processing to decrease noise, contrast enhancement, and image sharpening are examples of low-level processes. The fact that pictures are used as both inputs and outputs identifies a low-level operation. Images must be processed at a mid-level, which includes segmentation (the division of an image into areas or objects), description of those items to make them more computer-processible, and classification (the identification of specific objects).

One property of a mid-level process is that it typically receives pictures as inputs and produces attributes (such as edges, contours, and the like) from those images. unique object identities). Last but not least, higher-level processing entails "making sense" of a group of recognised objects, as in image analysis, and, at the extreme end of the continuum, performing the cognitive tasks typically associated with vision. It also includes procedures that extract properties from images, up to and including the identification of individual objects. Consider the field of automated text analysis as a straightforward example to help illustrate these ideas.

Segmentation techniques separate a picture into its individual elements or objects. Generally speaking, one of the most challenging jobs in digital image processing is autonomous segmentation. Imaging issues that need individual object identification can be successfully solved with the help of a tough segmentation approach. On the other hand, segmentation algorithms that are shaky or unpredictable virtually invariably result in failure. Generally

speaking, the likelihood of successful recognition increases with the accuracy of the segmentation.

The foundation for expressing pictures at different resolution levels is wavelets. As the name suggests, compression deals with methods for lowering the amount of storage needed to save a picture or the bandwidth needed to send it. Transmission capacity has not seen the same advancements in storage technologies during the previous ten years.

This is true in particular for Internet usage that is marked by a considerable amount of visual information. For example, the jpg file extension used in the JPEG (Joint Photographic Experts Group) image compression standard is known to most computer users (perhaps unintentionally) as a sort of picture compression.

Tools for extracting picture elements that are helpful in the representation and description of form are referred to as morphological processing.

Fields That Use Digital Image Processing

The following list includes some of the principal industries where digital image processing is employed extensively.

- Medical field
- Pattern recognition
- Transmission and encoding
- Video processing
- Machine/Robot vision
- Microscopic Imaging
- Remote sensing
- Image sharpening and restoration
- Color processing

Image sharpening and restoration

- The terms "image sharpening" and "restoration" here refer to the actions used to improve or edit photographs taken with a contemporary camera to get the desired effect. It alludes to the standard functions of Photoshop.
- Zooming, blurring, sharpening, converting from grayscale to colour, identifying edges and vice versa, image retrieval, and image recognition are included.
- The process of approximating the clean, original picture from a corrupted or noisy image is known as image restoration. Motion blur, noise, and camera misfocus are only a few examples of corruption's numerous manifestations.
- In order to restore the picture information lost to the blurring process, it is the necessary to reverse the process that caused the image to become blurry.
- This is done by imaging a point source and using the point source image, also known as the Point Spread Function (PSF), to image a point source.
- By surrendering some resolution, noise may be successfully eliminated from images via image enhancement, but this is not acceptable in many applications.
- The z-direction resolution in a fluorescent microscope is already subpar. To retrieve the item, more sophisticated image processing techniques must be used.

Medical field

The frequent uses of DIP in the realm of medicine include

- **PET scan**

  Positron emission tomography (PET) is a functional imaging method that visualises and measures changes in metabolic processes as well as in other physiological activities including blood flow, local chemical composition, and absorption. Radioactive chemicals known as radiotracers are used in PET. Depending on the target process within the body, several tracers are utilised for various imaging reasons. For instance, NaF18 F is frequently used to identify bone development, 18 F-FDG is frequently used to diagnose cancer, and oxygen-15 is occasionally used to assess blood flow.

- **X Ray Imaging**

  A penetrating kind of high-energy electromagnetic radiation is an X-ray or, much less frequently, an X-radiation. The majority of X-rays have a wavelength between 10 picometers and 10 nanometers, which translates to energies between 145 eV and 124 keV and frequencies between 30 petahertz and 30 exahertz (31016 Hz to 31019 Hz). Wavelengths of X-rays are generally longer than those of gamma rays but shorter than those of UV rays.

- **Medical CT**

  To measure the attenuations of X-rays by various tissues inside the body, CT scanners employ a revolving X-ray tube and a row of detectors arranged in a gantry. Tomographic reconstruction procedures are then used to analyse the many X-ray data acquired from various angles on a computer to create tomographic (cross-sectional) pictures (virtual "slices") of a body. Those who need a CT scan but cannot undergo an MRI do so because they have metallic implants or pacemakers.

Gamma ray imaging

The radioactive disintegration of atomic nuclei produces a penetrating kind of electromagnetic radiation known as a gamma ray, commonly known as gamma radiation (symbol or gamma). It comprises of electromagnetic waves with shorter wavelengths than X-rays, on average. It gives the maximum photon energy at frequencies exceeding 30 exahertz (3 1019 Hz). By researching radiation released by radium, French scientist and physicist Paul Villard made the discovery of gamma radiation in 1900.

UV imaging

The region of the earth is scanned by a satellite or from a very high ground in the field of remote sensing, and after that, it is processed to get data about it. The detection of earthquake-related infrastructure damage is one specific use of digital image processing in remote sensing.

A kind of electromagnetic radiation known as ultraviolet (UV) has a wavelength that is shorter than visible light but longer than X-rays. UV radiation is a component of sunlight that makes

up around 10% of the Sun's overall electromagnetic radiation output. Cherenkov radiation, electric arcs, and specialty lights like black lights, tanning lamps, and mercury vapour lamps can also create it. Since its photons lack the energy to ionise atoms, long-wavelength ultraviolet is not regarded as an ionising radiation, but it may nevertheless trigger chemical processes and make a variety of things shine or fluoresce.

Even when substantial damages are the emphasis, it takes longer for harm to be understood. Because the region affected by an earthquake can occasionally be so large, it is impossible to visually inspect it to determine the extent of the damage. Even if it is, the process is incredibly stressful and time-consuming. In light of this, digital picture processing offers a remedy. To identify the different kinds of damage caused by the earthquake, a photograph of the affected region is taken from above and examined.

UV radiation's ability to interact with organic molecules has several practical implications, including chemical and biological impacts. These interactions don't always result in heating but might entail absorption or changing the energy states of molecules.

Sunlight contains UV radiation, which accounts for around 10% of the Sun's total electromagnetic radiation output. It can also be produced by Cherenkov radiation, electric arcs, and specialised lighting such as black lights, tanning lamps, and mercury vapour lamps. Long-wavelength UV is not considered an ionising radiation because its photons lack the energy to ionise atoms, yet it may nevertheless cause chemical reactions and cause a variety of substances to glow or fluoresce.

The extraction of edges, analysis, and augmentation of distinct types of edges are the major processes in the analysis.

Transmission and encoding

A underwater cable carried the first image ever to be broadcast over the wire, travelling from London to New York. Below is a photo that was sent.

Three hours were needed for the photograph to travel from one location to another.

The World Wide Web Consortium's XML schema language is used to define the Metadata Encoding and Transmission Standard (METS), a metadata standard for encoding descriptive, administrative, and structural metadata about items in a digital library (W3C). The MARC standards of the Library of Congress are maintained, and the Digital Library Federation is working to develop the standard (DLF).

As METS is open and flexible, it can be used by a wide range of institutions and with a wide range of document formats because there is no set vocabulary. Although METS is extremely functional internally, interoperability is constrained by customisation. When the institutions exporting and importing have vocabularies, interoperability is challenging. The establishment of institutional profiles has been a common solution to this issue. These profiles provide evidence of how METS has been implemented in that institution, aiding in the mapping of content to make shared METS papers more useful across institutions.

Imagine for a moment that we can now view live video feeds or live cctv footage with only a few seconds of delay from one continent to another. It implies that significant effort has also been made in this area. This area also emphasises encoding in addition to transmission. To encode photographs for streaming over the internet or other high or low bandwidth networks, a wide variety of formats have been created.

Machine/Robot vision

One of the main issues facing robots today, besides the many others, is still to improve its eyesight. Make the robot capable of seeing things, recognising them, recognising obstacles, etc. This field has made significant contributions, and a completely new branch of computer vision has been established to focus on it.

Machine vision (MV) is a field of computer science that focuses on providing imaging-based automatic inspection and analysis for a variety of industrial applications, including process control, robot guiding, and automatic inspection. The term "machine vision" covers a wide range of technologies, software and hardware items, integrated systems, procedures, and knowledge. Computer vision is a branch of computer science, whereas machine vision is a field of systems engineering. It makes an effort to employ already existing technology in novel ways

to address issues in the real world. The word is frequently used for these activities in situations involving industrial automation, but it is also used for similar functions in environments involving vehicle guiding.

Planning the specifics of the project's needs and execution comes first in the entire machine vision process, followed by solution development. The method begins during run-time with imaging, then moves on to automatically analyse the image and extract the necessary data.

Hurdle detection

One frequent job that has been accomplished by image processing is hurdle detection. To do this, different types of objects in the image are first identified, and then the distance between the robot and the obstacles is calculated.

Obstacle avoidance in robotics is the process of achieving a control goal while adhering to non-intersection or non-collision position limitations. The rising need for the employment of unmanned aerial vehicles in urban environments, particularly for military applications where they may be extremely helpful in city conflicts, is what is crucial about the obstacle avoidance concept in this field. Obstacle avoidance is typically thought of as being separate from route planning since the former typically entails the pre-computation of an obstacle-free path that a controller will later use to direct a robot along. A good and reliable obstacle avoidance function of a driverless platform is also necessary to have a robust obstacle detection module given recent advancements in the autonomous vehicle sector.

A robot's reactive obstacle avoidance is a behavior-based control technique. It results in a motion that is devoid of collisions and is analogous to the navigation issue.

The OCAS system is intended to safeguard objects that are at risk from aeroplanes flying at low altitudes. OCAS monitors an incoming aircraft's ground speed, direction, and altitude and assesses whether it will be able to sufficiently avoid the obstruction. Depending on the predicted time to impact with the obstacle, a customizable set of criteria is used to define and decide when to alert the aircraft and by what warning device and signal.

Line follower robot

The majority of robots in use today operate by following lines, thus the term "line follower robots." This facilitates a robot's movement and some of its functions. Image processing has also been used to do this.

A robot that follows a predetermined course under the control of a feedback system is called a line follower.

Using Workspace or AutoCAD, the mechanical component or body of the robot may be designed. A base with wheels placed at each of its two ends can make up a simple line follower robot. The foundation might be a rigid plastic sheet cut into a rectangular shape. Other rigid bodies, such as cylinders, can be added, along with various body shapes that are interconnected by joints and each have a set motion in a certain direction. The Line follower robot can be a legged mobile robot with numerous rigid bodies connected by joints, a wheeled mobile robot with a permanent base, or both.

Determining the robot's Kinematics is the next phase. The description of the robot's motion in relation to a given coordinate system is a component of kinematic analysis. It is mostly focused on the robot's mobility, as well as the motion of each body when the robot has legs. Usually, it has to do with the robot's motion dynamics. The robot's whole path is predetermined using kinematic analysis. With the Workspace programme, this is possible.

Colour processing

Processing coloured pictures and the many colour spaces that are employed are both included in colour processing. For instance, the RGB colour model, YCbCr, and HSV. Additionally, transmission, storage, and encoding of these colour pictures are studied.

The CMYK colour model, commonly referred to as process colour or four colour, is a subtractive colour model that is used in colour printing and is also used to describe the printing

process itself. It is based on the CMY colour model. Cyan, magenta, yellow, and key (CMYK) are the four ink plates that are utilised (black).

White is the "additive" mixture of all main coloured lights in additive colour schemes, such as RGB, whereas black is the absence of light. Contrarily, with the CMYK paradigm, white is the actual colour of the paper or other backdrop, and black is produced by mixing all of the different coloured inks. Unsaturated and dark colours are created by utilising black ink instead of cyan, magenta, and yellow to reduce ink costs and provide deeper black tones.

The CMYK paradigm conceals some or all of the colours over a lighter backdrop, often white. The light that would ordinarily reflect is diminished by the ink. Since inks "subtract" the hues red, green, and blue from white light, a model like this is referred to as subtractive. White light without the red, green, or blue components is cyan, magenta, and yellow, respectively.

Pattern recognition

Study of image processing and numerous other subjects, such as machine learning, are necessary for pattern detection (a branch of artificial intelligence). Image processing is used in pattern recognition to identify the items in an image, and machine learning is then used to teach the system to recognise changes in patterns. Pattern recognition is utilised in computer assisted diagnosis, handwriting recognition, picture identification, etc.

The automatic detection of patterns and regularities in data is known as pattern recognition. It has uses in machine learning, bioinformatics, data compression, computer graphics, signal processing, image analysis, and statistical data analysis. Pattern recognition has its roots in engineering and statistics, and some contemporary methods for doing so make use of machine learning, thanks to the greater accessibility of massive data and the wealth of computing power. These activities, which may be seen as two sides of the same application sector, have advanced significantly in recent years.

Systems for pattern recognition are frequently trained using labelled "training" data. Other methods can be employed to find previously unidentified patterns when labelled data are not available. Unsupervised techniques are more the emphasis of KDD and data mining, and their relationship to commercial applications is stronger. In addition to considering acquisition and signal processing, pattern recognition places additional emphasis on the signal. The phrase is

well-known in the area of computer vision and has engineering roots. In fact, the premier computer vision meeting is called the Conference on Computer Vision and Pattern Recognition.

The fundamental goal of pattern recognition algorithms is to conduct "most likely" matching of the inputs, taking into account their statistical fluctuation, and to offer a fair response for each and every conceivable input. In contrast, pattern matching algorithms seek out perfect matches between the input and already-existing patterns. Regular expression matching is a typical example of a pattern-matching algorithm. It searches for patterns of a certain type in textual material and is a feature of many text editors and word processors' search functions.

Video processing

A video is nothing more than just a series of images that move really quickly. The quantity of frames or photos per minute and the calibre of each frame employed determine the video's quality. Noise reduction, detail improvement, motion detection, frame rate conversion, aspect ratio conversion, colour space conversion, and other processes are all part of video processing.

Video processing is a specific type of signal processing, specifically image processing, used in electronics engineering where the input and output signals are video files or video streams and frequently uses video filters. Televisions, VCRs, DVD players, video scalers, video players, and other devices all employ video processing techniques. As an illustration, TV sets from various manufacturers often only differ in their design and visual processing.

In the video/film industry, the phrase "post-processing" (or "postproc" for short) refers to technologies used in video playback equipment such standalone DVD-Video players, video playing software, and transcoding software for quality-improving image processing (particularly digital image processing). In order to provide more effects, it is frequently utilised in real-time 3D rendering (such as in video games).

They are available as chips or as a standalone unit that is attached to a source device (such as a DVD player or set-top box) and positioned between it and a display with less powerful processing. The market's most well-known manufacturers of video processors are:

- Evolution Microchip (with the FLI chipset – was Genesis Microchip, STMicroelectronics completes acquisition of Genesis Microchip on January 25, 2008)

- Design Sigma (with the VXP chipset – was Gennum, Sigma Designs purchased the Image Processing group from Gennum on February 8, 2008, Sigma Designs is now part of Silicon Labs)
- Device Integration Technology (with the HQV chipset and Teranex system products – was Silicon Optix, IDT purchased SO on October 21, 2008, IDT is now part of Renesas)
- Google Image (with the VRS chipset and DVDO system products - was Anchor Bay Technologies, Silicon Image purchased ABT on February 10, 2011)

These firms' chips may be found in a wide variety of products, including DVD upconverters (for Standard Definition), HD DVD/Blu-ray Disc players, set-top boxes, plasma displays, DLP (both front and rear projection), LCD (both flat-panel and projectors), and LCOS/"SXRD" displays. Also, more standalone gadgets are becoming accessible with these chips (see "External links" below for links to a few of these).

Simple Image Model

The following factors must be known in order to develop models, which require the following parameters:

1. The amount of light that the light source produces

2. Surface characteristics

3. The amount of light that the sensor is able to catch

The power of the light produced by a light source is measured in luminous flux, much as the power of a bulb, which is expressed using the unit Watt. The term "luminous flux" describes how much light energy a light source produces in a given length of time. Illuminance is the measurement of the quantity of light flux that falls on a certain region of a surface. The kind of the image—vector or raster—depends on whether the resolution is fixed. The lux unit of measurement for illumination is also known as luminous flux density.

A digital image is one that is made up of picture elements, also referred to as pixels, each of which has a finite, discrete quantity of numerical representation for its intensity or grey level

as an output from its two-dimensional functions fed as input by its spatial coordinates denoted with the letters x, y on the x-axis and the y-axis, respectively. The kind of the image—vector or raster—depends on whether the resolution is fixed. The phrase "digital picture" used by itself often refers to raster or bitmapped images (as opposed to vector images).

Only a relatively small frequency range can be perceived by humans. 400–700 nm is the range of visible light. The term "spectrum" or "spectral power distribution" refers to this band of frequencies. Therefore, L(x, y, 1, λ) may be used to describe the energy distribution of light travelling across a plane, where x and y are spatial coordinates, & is the wavelength, and is the duration. This function is represented by L(λ) if x, y, and I are taken to be constants. Objects respond to light differently based on their material characteristics.

When light strikes an item, it may reflect or refract. Reflection is the phenomenon of an incoming light beam returning after striking a surface. Refraction is the process by which a light beam is bent, altering its wavelength and speed.

The term "spectrum" or "spectral power distribution" refers to this band of frequencies. Diffraction is when light curves around a barrier.

Digital cameras and computers emerged before early digital fax devices like the Bartlane cable picture transmission system by decades. The Standards Eastern Automatic Computer (SEAC) at NIST presented the first image to be scanned, saved, and regenerated in digital pixels. Throughout the early 1960s, digital imaging technology continued to evolve along with the expansion of the space programme and medical research. Digital pictures were employed in research projects at the Jet Propulsion Laboratory, MIT, Bell Laboratories, the University of Maryland, and other institutions to expand videophone technology, character recognition, medical imaging, satellite imagery, and other fields.

The light causes the medium molecules in some things to vibrate. Since some energy is absorbed by this vibration, less energy is reflected. It is possible to write the light emitted by an item at a specific wavelength λ as follows:

$I(x, y, \lambda) = (x, y, \lambda)\ XL(A)$

For an opaque object, $o(x, y, \lambda) = 0$ and for a perfect reflector, it is 1.

Illuminance per unit solid angle, with a value of "nit," is the term used to describe the amount of light that comes from the surface and enters a sensor. These variables are known as

photometry parameters as photometric measurements take into consideration the human visual system. Literally, photometry implies the measuring of light. These measurements are known as radiometry when the complete electromagnetic (EM) spectrum is taken into consideration. Irradiance and radiance are the radiometric equivalents of luminance and illuminance, respectively. A common application of radiometry in computer graphics Both a camera and the human eye can see the reflected light.

Raster pictures are made up of a finite number of picture elements, often known as pixels. There are a predetermined number of pixels in each row and column of the digital image. The tiniest component of a picture, a pixel stores quantized values that describe the brightness of a given colour at any given location.

Normally, the pixels are kept in computer memory as a two-dimensional array of tiny numbers called a raster picture or raster map. A compressed version of these data is frequently used for transmission or storage.

Images are seen via picture viewer software. Standard internet image formats including JPEG, GIF, and PNG may be seen in web browsers. Some can display the SVG format, a W3C standard. It used to be normal practise to offer "preview" pictures that would load and show on a website before being replaced with the main image when the Internet was still sluggish (to give at preliminary impression). Nowadays days, the Internet is quick enough, thus this preview picture is rarely used.

With the debut of MOS integrated circuits in the 1960s and microprocessors in the early 1970s, together with advancements in associated computer memory storage, display technologies, and data compression methods, rapid advancements in digital imaging started.

Large scientific photos are sometimes possible (for instance, the 46 gigapixel size image of the Milky Way, about 194 Gb in size).

Such photos are typically seen online using more complicated web interfaces and are challenging to download.

Image processing is a technique for applying various procedures to a picture in order to improve it or extract some relevant information from it. It is a kind of signal processing where the input is an image and the output may either be another picture or features or characteristics related to that image.

The CCD, created in 1969 at Bell Laboratories by Willard S. Boyle and George E. Smith, was the first semiconductor image sensor. [16] They discovered, while studying MOS technology, that an electric charge could be held on a small MOS capacitor and was akin to the magnetic bubble. They attached a sufficient voltage to them in order to step the charge from one to the next because it was very simple to construct a row of MOS capacitors. Eventually utilised in the first digital video cameras for television broadcasts, the CCD is a semiconductor circuit.

Several input methods and tools, including digital cameras, scanners, coordinate measuring machines, seismographic profiling, aerial radar, and others, can produce raster pictures.

Medical diagnostics greatly benefited from the development of computerised axial tomography (CAT scanning), which uses x-rays to create a digital picture of a "slice" through a three-dimensional object. In addition to the creation of digital photographs, digitalization of analogue photos enabled the improvement and restoration of archaeological artefacts. These images were then employed in a variety of sectors, including nuclear medicine, astronomy, law enforcement, the military, and business.

The discrete cosine transform (DCT), a lossy compression algorithm initially put out by Nasir Ahmed in 1972, was a significant advancement in the field of digital image compression technology. The Joint Photographic Experts Group created JPEG in 1992, and it uses DCT compression.JPEG is the most used image file format on the Internet because it allows images to be compressed into significantly lower file sizes.

Computers are essential for operations requiring complicated computing, massive volumes of data processing, or the extraction of quantitative data. Nonetheless, for many applications, such as those in medical, security, and remote sensing, human analysts are still indispensible since the human visual brain is a superb picture processing tool, particularly for retrieving higher-level information. As a result, several crucial image processing technologies, like edge detectors and neural networks, draw their inspiration from models of human visual perception.

The electromagnetic radiation or other waves that transmit the image's information as they travel through or reflect off objects can be used to categorise different types of digital imaging. In all types of digital imaging, data is transformed into digital signals by image sensors, which are then processed by computers and produced as visible-light images. For instance, the use of various types of digital cameras with visible light enables for digital photography and filmmaking (including digital video cameras). Digital radiography, fluoroscopy, and CT are all possible using X-rays, and gamma-ray imaging is also possible (digital scintigraphy, SPECT,

and PET). Ultrasonography (such as in medical imaging) and sonar are made possible by sound, whereas radar is made possible by radio waves.

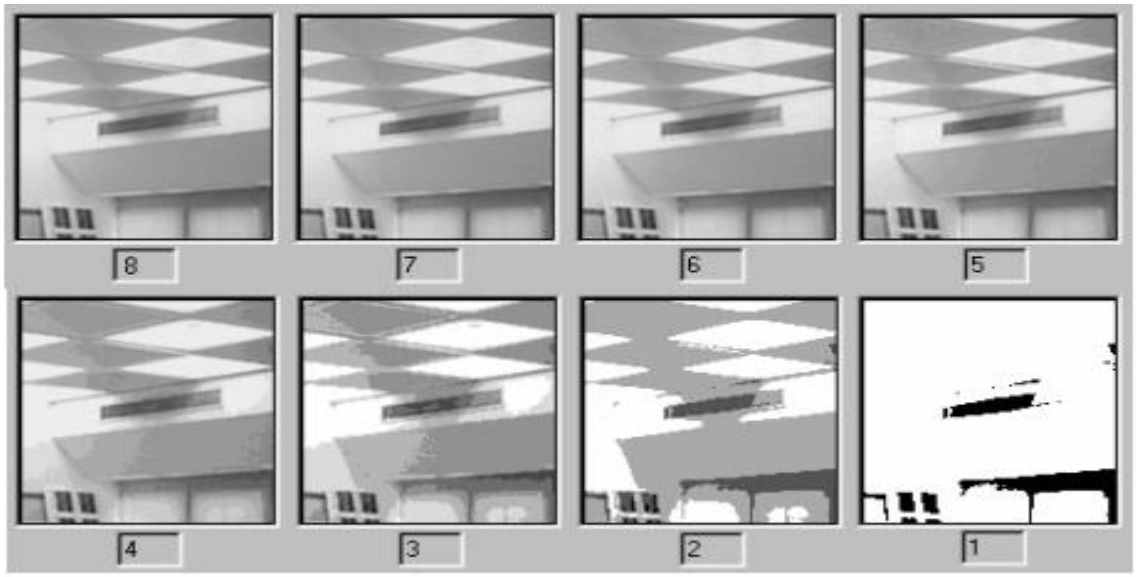


Fig 4. Images of different amplitude resolution

Digital imaging, often known as digital image capture, is the process of turning an actual physical scene or an object's internal structure into a computer representation. The phrase is frequently used to indicate or comprise such image processing, compression, storage, printing, and display. The capacity to digitally replicate copies of the original subject forever without any picture quality loss is a significant benefit of a digital image over an analogue image, such as a film photograph.

An image f(x,y) must be digitalized both spatially and in amplitude to be acceptable for computer processing.

• Image sampling is the process of digitising the spatial coordinates (x, y).

• Gray-level quantization is another name for amplitude digitalization.

A mapping of an item to an image plane occurs during the imaging process. There is a match between each point on the picture and a point on the item. An lighted item will reflect light towards a lens, which will then gather and concentrate it to form the picture. The magnification is determined by the height of the picture in relation to the height of the object. The field of vision of a lens is determined by the focal length of the lens and the size of the picture surface.

The focal length of the mirror, which determines how a mirror forms an image, is half as long as the mirror's centre of curvature.

An image's quality is determined by both geometric and physical factors. More pixel density throughout an image results in less blocky pixelation and consequently better geometric picture quality. Lens aberrations also add to image quality. Diffraction caused by the aperture stop physically limits the resolvable spatial frequencies as a function of f-number.

Luminance is the fundamental photometric quantity of perceived light. A light is referred to as monochromatic if its $L(\lambda)$ value is 0 everywhere other than at a reference frequency (λ_r), For brightness matching, the monochromatic light at λ, can be matched with another monochromatic light at the test frequency (λ_t). The experimenting can go on until the brightness matches by altering the amplitude. When there is a match, the ratio $L(\lambda_r) / L(\lambda_t)$ is referred to as the monochromatic light relative luminous efficiency at λ_t in comparison to that at λ_r. The relative luminous efficiency function, or $v(\lambda)$, is what the relative luminous function is known as when it is stated in terms of λ_r.

Basic relationships between pixels

An image is represented by the notation f(x, y) and its component pixels are indicated by p, q.

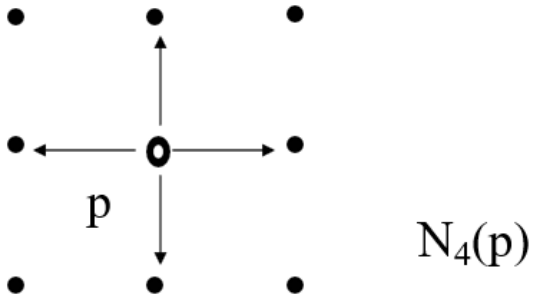
- Neighbourhood
- Adjacency
- Connectivity
- Paths
- Regions and boundaries

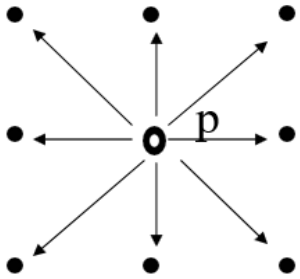
Neighbours of a pixel

The four horizontal and vertical neighbours of a pixel p at (x, y) are (x+1, y), (x-1, y), (x, y+1), and (x, y-1). The 4-neighbors of p are known as N_4 in this context (p). Each pixel is one unit away from (x, y), and if (x, y) is on the edge of the image, some of p's neighbour positions are outside the digital image.

A pixel at position (x, y) has four diagonal neighbours at positions (x+1, y+1), (x+1, y-1), (x-1, y+1), and (x-1, y-1). These are known as p's diagonal neighbours, or N_D (p).

The diagonal neighbours of p and the 4-neighbors are referred to as the 8-neighbors of p: N_8 (p). Like previously, if (x, y) is on the edge of the image, some of the neighbouring locations fall within and outside of it.





$N_8(p)$

x-1, y-1	x-1, y	x-1, y+1
x, y-1	x, y	x, y+1
x+1, y-1	x+1, y	x+1, y-1

Fig: Sub-image of size 3x3 of 8-neighbour

N_D	N_4	N_D
N_4	P	N_4
D	N_4	N_D

N_4 = 4-Neighbours

N_D = Diagonal Neighbours

N_8 = 8-Neighbours ($N_4 \cup N_D$)

Adjacency

- Specify V as the collection of intensity values that define adjacency. V = 1 is the value for binary images.
- One specific grayscale image has the formula V = 1, 3, 5, ..., 251, 253, 255.
- 4-adjacency: Two pixels with values from V are 4-adjacent if p and q are both in the set N_4.
- 8-adjacency: If q is a member of the set N_8, two pixels with values from V, p and q, are 8-adjacent.
- m-adjacency(mixed): Two pixels p and q with values from V are m-adjacent if
  - q is in $N_4(p)$ OR q is in $N_D(p)$,

    And
  - $N_4(p) \cap N_4(q)$ has no pixels with values from V.

Example:

The pixel configuration for V=1 is given in fig.(a). As seen by the lines, the three pixels in the top fig.(b) display multiple 8-adjacency. With m-adjacency, as demonstrated in fig.(c), this ambiguity is eliminated.

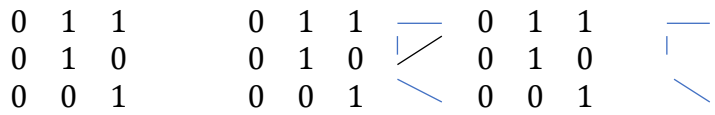


Fig.(a) Fig.(b) Fig.(c)

8-adjacency m-adjacency

From pixel p with coordinates (x, y) to pixel q with coordinates (s, t), a (digital) path (or curve) is a collection of unique pixels with the following coordinates:

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$$

Where,

$$(x_0, y_0) = (x, y), (x_n, y_n) = (s, t),$$

And pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$.

N is the path's length in this case. The path is closed if $(x_0, y_0) = (x_n, y_n)$. Depending on the type of adjacency supplied, we define 4-, 8-, or m-paths. For instance, the top right and bottom right pathways in Fig. (b) are 8-paths, whereas the path in Fig. (c) is an m-path.

Let S stand in for a portion of an image's pixels. If there is a path connecting two pixels p and q in S that is made up of of pixels in S, then the two pixels are said to be connected in S. A connected component of S is the collection of pixels in S that are connected to any pixel p in S. Set S is referred to as a connected set if there is just one connected component in it.

Let R represent a subset of an image's pixels. If R is a connected set, we refer to it as a region of the image. When two areas combine to form a connected set, those two regions R_i and R_j are said to be neighbouring. Disjoint regions are those that are not neighbouring.

Example:

Only when 8-adjacent is utilised are the two zones considered to be adjacent. There is no 4-path connecting the two areas, hence the union is not a connected set.

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix} \} R_i$$

$$\begin{matrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \} R_j$$

Suppose an image has K sections that are not connected, R_k,k=1,2, … ,K, none of which extend past the edge of the image. Let $(R_u)^c$ stand for its complement, and let R_u signify the union of all the K regions. All of the points in R_u are called the foreground and each and every point in $(R_u)^c$ are called the picture's background.

The perimeter of a region, sometimes known as its border or counter Instead, the border of a region is the collection of pixels in the area that have at least one background neighbour. R is the set of points that are adjacent to points in R's complement.

Example:

If 4-connectivity is used to connect the region and its background, the point marked in the following picture does not form part of the border of the 1-valued region.

In order to handle situations like this, adjacency between points in a region and its background is typically specified in terms of 8-connectivity.

0	0	0	0	0
0	1	1	0	0
0	1	1	0	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Example:

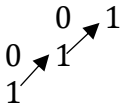
Take into account the following arrangement of pixels, assuming that each of them has a value of 1, and that the remaining ones can either be 0 or 1:

$$\begin{matrix} & p3 & p4 \\ p1 & p2 & \\ p & & \end{matrix}$$

Let's say we take adjacent pixels with a value of 1 (v=1) into consideration.

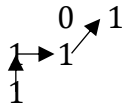
1) If p1 and p3 are taken to be 1,

   D_m (m-path) distance between p and p_4 is 2



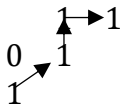
2) If p1 is assumed to be 2

   D_m (m-path) distance between p p1p2p4 is 3



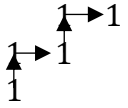
3) If p3 is taken to be 1 and p1 to be 0

Then D_m distance between p p2p3p4 is 3



4) If p1 and p3 are considered to be 1

Then the D_m distance between p p1p2p3p4 is 4



Connectivity between pixels

It is a crucial idea in the processing of digital images. It is employed for drawing the boundaries between objects and the elements that make up regions in a picture. Supposed connection between two pixels:

- If their grey levels meet a specific criterion for similarity and
- they are adjacent in some way (4/8/m-adjacency, neighbour pixels) (equal intensity level)

According to adjacency, there are three different types of connectedness. These are:

a) **4-connectivity:** A pair of pixels are considered to be 4-connected if they are 4-adjacent to one another.

b) 8-connectivity: Pixels are considered 8-connected if they are 8-adjacent to one another.

c) m-connectivity: It refers to the relationship between two or more pixels when they are m-adjacent to one another.

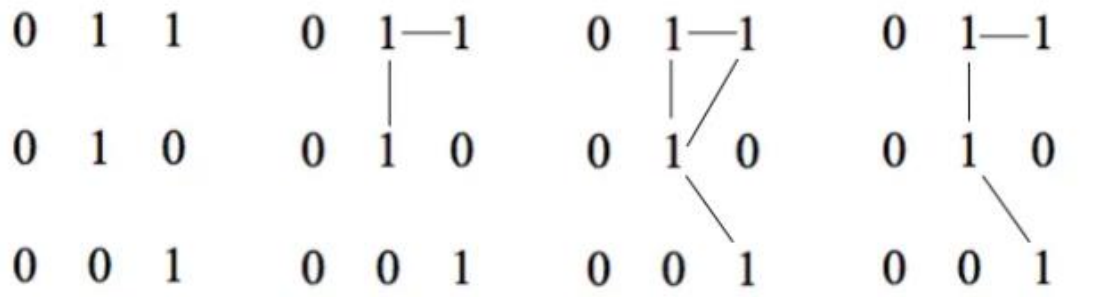


Fig: An arrangement of pixels **Fig: 4-connectivity of pixels** **Fig: 8-connectivity of pixels** **Fig: m-connectivity of pixels**

Paths

The sequence of separate pixels with the coordinates (x_0, y_0), (x_1, y_1), and (x_2, y_2) makes up the path from pixel p with coordinates (x, y) to pixel q with values (s, t). $(x_0, y_0) = (x, y)$ and $(x_n, y_n) = (s, t)$, respectively;

(x_i, y_i) is adjacent to (x_{i-1}, y_{i-1}), $1<=i<=n$

The path's length, n, is indicated here.

Depending on the kind of adjacency employed, we can define 4-, 8-, and m-paths.

Region

A region is another name for a connected set. If the union of two regions (let R_i and R_j) results in a connected set, then those regions are said to be nearby. Joint regions or adjacent regions Disjoint regions are those that do not border each other. When referring to regions, 4- and 8-adjacency is taken into account. It is important to specify the adjacency type when discussing a specific area.

Boundary

The region's perimeter (border or contour) The collection of points known as R are those that border the complement of R's points. A group of pixels in the area that share a border with the background. The collection of pixels in the region with one or more neighbours who are not in the area R constitutes the region's perimeter.

Inner Border: Foreground's inner border

Outside border: Border of the background

Digital Image Paradigm distinguishes between a border and an edge.

Distance Measures

D is a distance function or metric for pixels p, q, and z with coordinates (x, y), (s, t), and (v, w), respectively if:

1. D(p,q)≥0, D(p,q)=0 if p=q
2. D(p,q)=D(q,p), and
3. D(p,z)≤D(p,q)+D(q,z).

- Euclidean Distance:

$$D_e(p,q) = [(x-s)^2 + (y-t)^2]^{1/2}$$

With this distance measurement, the points enclosed in the disc of radius r centred at (x,y) are the pixels with a distance from (x,y) less than or equal to some value r. (x,y).

- City Block Distance:

$$D_4(p,q) = |x-s| + |y-t|$$

The pixels in this example that are closer to a diamond centred at (x,y) than or equal to a value r (x,y).

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

- Chess Board Distance:

$$D_8(p,q) = \max(|x-s|, |y-t|)$$

The pixels in this instance that are closer to a square centred at (x,y) than or equal to a value r (x,y).

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

Note:

Because these distances solely concern the coordinates of the points, the D_4 and D_8 distances between p and q are unaffected by any possible pathways between the points.

- **D_m-Distance:**

  The shortest m-path between two places is used to measure distance between them. In this instance, the separation between two pixels will be determined by the values of both the pixels in their immediate surroundings and those down the path.

Basic transformation: Scaling, Translation, Rotational

Contrary to point operations, geometric operations can alter the spatial arrangement of items (elephants) in an image. The process of transferring the pigments within the image might be viewed as this procedure. It's outlined as:

f (x,y) stands for the input image, g (x,y) for the output image, a (x,y) and b (x,y) for the spatial transformation, and if they are continuous, the connectivity in the image is preserved.

In contrast to geometric transformations, where the grayscale value of g (x,y) is typically determined by the value of f (x,y) on a non-integer coordinate, a pigment in g generally corresponds to the position between several pictograms in f, grayscale interpolation is a crucial part of geometric operations because images are typically defined by the pigment at the integer position. The opposite is also true; a pigment in f is frequently mapped to the space between a few of the pictoses in g.

There are two techniques to carry out geometric operations. The first is forward mapping, which transfers each input pigment's grayscale to the output image one by one. The second is interpolation, which interpolates each input pigment's grayscale value between the four output elephants. The output pixels are divided between them, and the second method is known as the backward mapping method (pixel fill method), in which the output pixels are sequentially mapped back to the original image. If the output pixels are mapped to a location between the

four original input pixels, the interpolation of those pixels determines the grayscale of the resulting image. In real life, reverse mapping is frequently applied.

For the purpose of using images for geometric measures, geometric transformation is frequently utilised in the geometric correction process of cameras.

Two typical geometric procedures are the affine transformation and image warping. The former is a part of the shadow geometric transformation and is primarily used for image registration (ImageRegistration) as a pre-processing process of comparison or matching, whereas the latter uses the control point and interpolation process to define the image deformation process of gradually changing one image into another and is more frequently used in the creation of film and television stunts and advertising.

Translation

An image is translated when it is moved to a different location. Consider moving the point in the matrix F's coordinate position X = (x, y) to a new position X' whose coordinate position is (x', y'). This can be expressed mathematically as "a translation of a point X to the new position X." The translated phrase appears as

$$x' = x + \delta x$$

$$y' = y + \delta y$$

This is written as F' = F+ T in vector notation, where S and Sy are translations parallel to the x and y axes. Here, the original and translated images are denoted by F and F', respectively. Scaling and rotation, on the other hand, are multiplicative transformations. The transformation process for scaling is given as F' = SF, and the transformation process for rotation is given as F' = RF, where R is the transformation matrix to accomplish rotation. The scaling transformation matrix in this case is S.

It is necessary to employ a homogeneous coordination system, where all transformations are handled as multiplications, to create uniformity and consistency. The expression for a point (x,

y) in 2D space is (wx, wy, w) for w0. The following are the characteristics of homogeneous coordinates:

1. At least one point in homogeneous coordinates must not be zero. Hence, in the homogeneous coordinate system, (0, 0, 0) does not exist.

2. When two points are multiplicative of one another, they are equal. Because the second point is 3 X, the points (1, 3, 5) and (3, 9, 15) are identical (1, 3, 5).

3. The homogeneous coordinate system's point (x, y, w) corresponds to the point $\left(\frac{x}{w}, \frac{y}{w}\right)$ in two-dimensional space.

The point (x, y) of the original image F to the new point (x', y') of the image F' is translated as follows in the homogeneous coordinate system:

$$x' = x + \delta x$$

$$y' = y + \delta y$$

And

This can be stated in matrix form as,

$$[x', y', 1] = \begin{pmatrix} 1 & 0 & \delta x \\ 0 & 1 & \delta y \\ 0 & 0 & 1 \end{pmatrix} [x, y, 1]^T$$

The image might not always be there at the start. In that situation, aligning the image with the origin can be accomplished by applying a sufficient negative translational value.

Translation operation in 3D

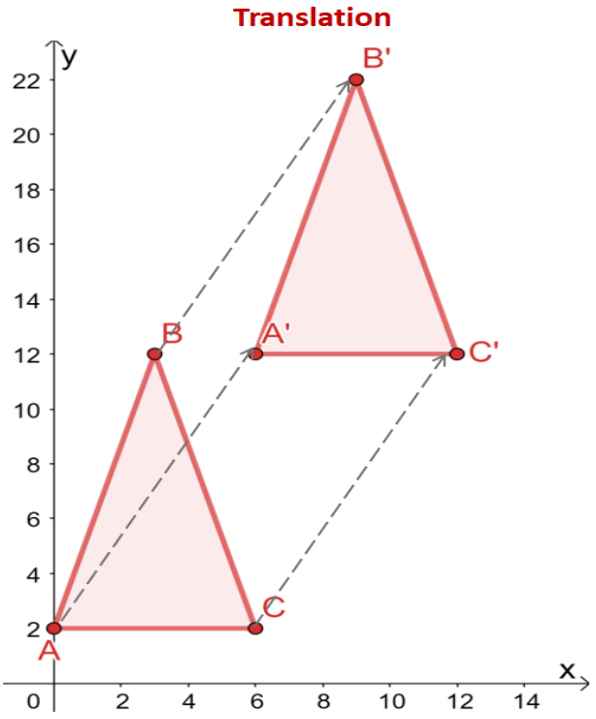
The new point that is detected when p(x, y, and z) is translated over the point (x_0, y_0, and z_0) is q(x', y', z') => x' = x+x_0, y' = y+y_0 and z' = z+z_0.

Matrix representation:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Unified expression:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Scaling

The item can be scaled depending on the need. Scaling involves expanding and contracting. The scaling of the point (x, y) of the image F to the new point (r, y) of the image F' in the homogeneous coordinate system is defined as

$$x' = x \times S_x$$

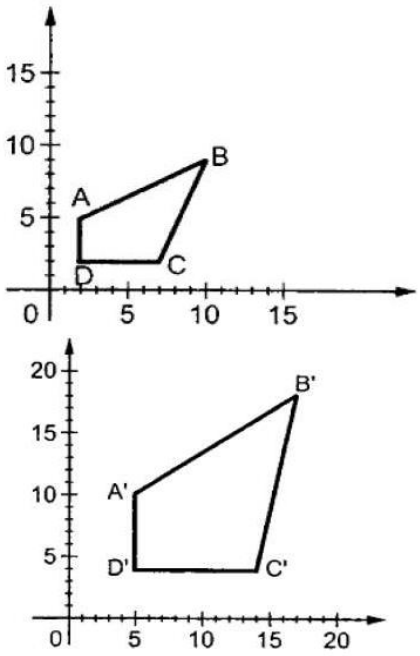
$$y' = y \times S_y$$

$$[x', y'] = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} [x, y]$$

Scaling factors for the r and y axes are S and S, respectively. The object would appear larger if the scale factor was 1. The object would shrink if there were fractional scaling factors. In a similar way, scaling is uniform, if S_x and S_y are equal. This process is called isotropic scaling. Instead, differential scaling is used. It is modelled as follows in the homogeneous coordinates system:

$$[x', y', 1] = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} [x, y, 1]^T$$

Where, $\begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$ is called scaling matrix.



We can shrink the size of the object by giving values to the scaling factor S that are less than 1. The object's size can be increased if we supply values larger than 1.

In case of 3D operations:

$$[x', y', z', 1] = \begin{pmatrix} S_x & 0 & 0 & 1 \\ 0 & S_y & 0 & 1 \\ 0 & 0 & S_z & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} [x, y, 1]^T$$

Rotational

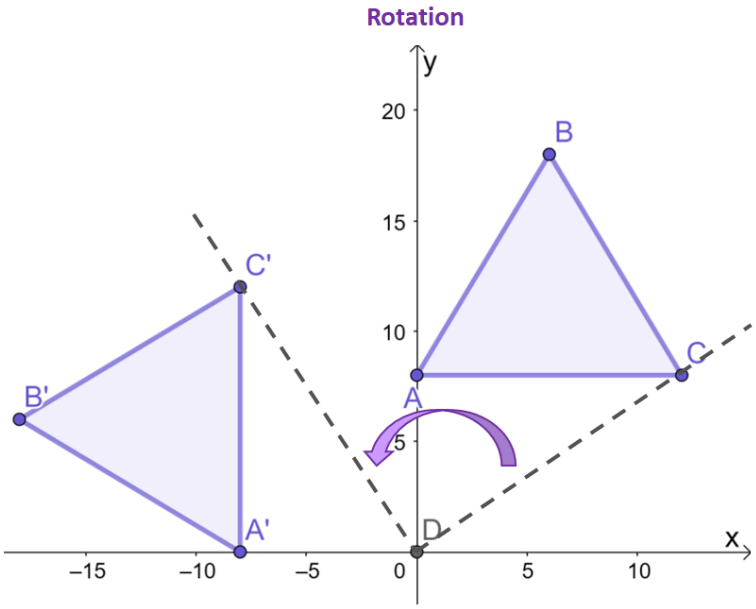
A picture can be rotated by different angles, such 90°, 180°, or 270°. The matrix for provides it as

$$[x', y'] = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} [x, y]^T$$

F' = RA can be used to symbolise this. The rotational angle with respect to the x-axis is the parameter θ. The rotation of the object is thought to be about its origin. θ can have a positive or negative value. A positive angle denotes rotation in the anticlockwise direction, whereas a negative angle denotes rotation in the clockwise direction. Rotation in the homogeneous coordinate system is denoted by

$$[x', y', 1] = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} [x, y, 1]^T$$

The image is rotated by matrix in clockwise direction if θ is substituted by $-\theta$.

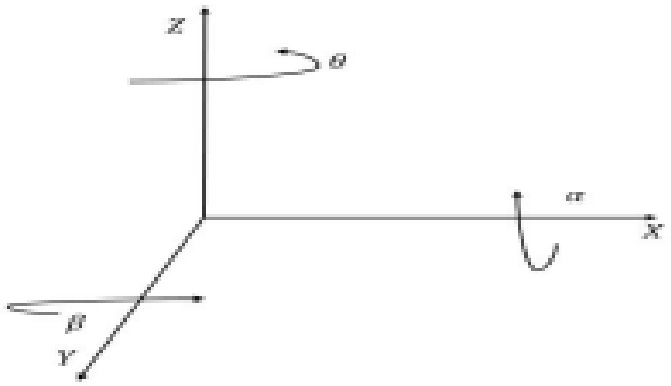


In case of 3D operations:

α→along x-axis,

β→along y-axis,

θ→along z-axis



$$R_{\theta} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_{\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_{\beta} = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pseudocolor Image Processing

Color Fundamentals

Sir Isaac Newton discovered in 1666 that a glass prism splits a sunlight beam into a rainbow of colours when it passes through it.

The type of light reflected from an object determines the colours that humans and the majority of animals see in it.

For instance, green objects absorb the majority of energy at other wavelengths while reflecting light with wave lengths mostly between 500 and 570 nm.

The characteristics of a chromatic light source are quantified by three fundamental variables:

- Radiance: the total energy emitted by the source of light (measured in watts)

- Luminance: the quantity of energy a light source emits that is seen by the viewer (measured in lumens)

• Take note that there can be high radiance and low brightness.

Brightness is a personal (practically immeasurable) concept that encompasses the achromatic idea of light intensity.

Chromatic light lies in the region of the electromagnetic spectrum between 400 and 700 nm.

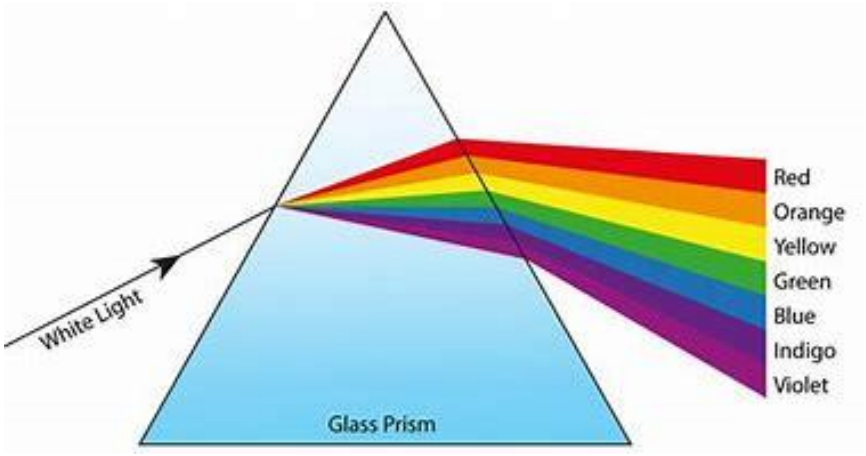
Six to seven million cones are responsible for human colour vision in each eye.

There are three main sensing groups:

These cones are 2% sensitive to blue light, 33% sensitive to green light, and 66% sensitive to red light.

Experimentally, the absorption curves for the various cones have been identified.

Interestingly, since the standards were created before to the studies, these do not match the CIE standards for red (700nm), green (546.1nm), and blue (435.8nm) light.



The secondary colours can be created by mixing the primary colours.

White is the result of combining the three primaries.

When a secondary and the opposing primary are combined, white results (for instance, red+cyan).

Light's primary colours (red, green, blue)

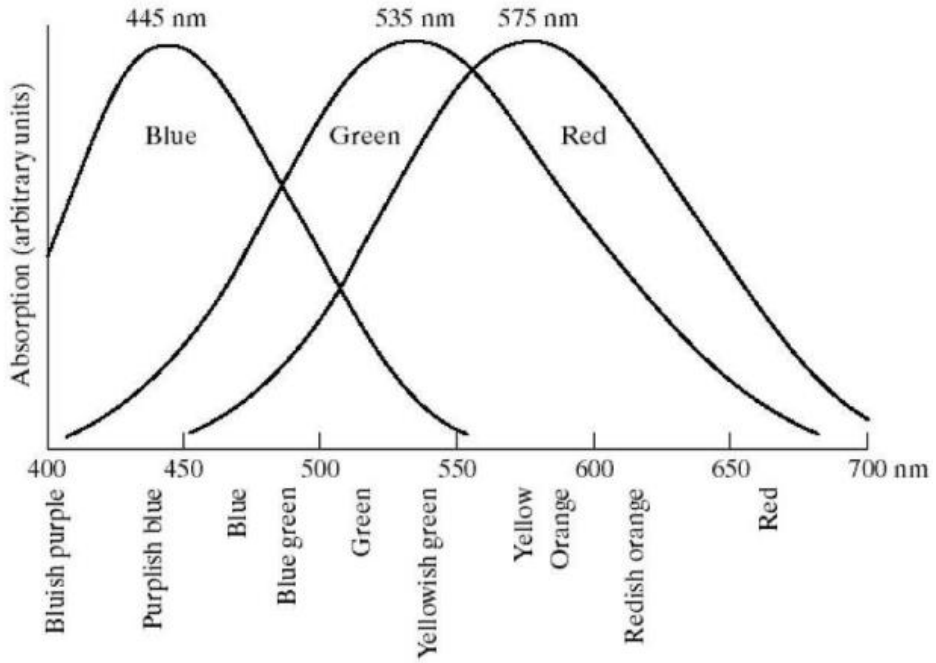
Primary colours of pigments (colourants) are hues that reflect the other two primary colours of light while detracting from one of them. Cyan, Magenta, and Yellow are these (CMY).

- Black is created by properly combining the pigment primaries.

-Overall energy emitted from a light source is known as its radiation (W)

-Luminance: The amount of energy that a light source appears to have to an observer. (In the IR example, R might be high when L is 0)

-Brightness is difficult to quantify and is subject to subjective description.



How can one hue be distinguished from another?

The achromatic concept of intensity is brightness.

The dominating wavelength in a combination of light waves is called hue.

Note: Hue is the predominant colour that an observer perceives; for example, when we say that something is red or orange, we are referring to its hue. Saturation is the quantity of white light combined with a hue. Colors that are pure are totally saturated. Less saturated is pink (red+white).

The terms "hue" and "saturation" refer to chromaticity.

Hence, the brightness and chromaticity of any colour define it.

The tristimulus values, which are represented by the letters X, Y, and Z, are the proportions of red, green, and blue required to create a specific colour.

Following that, a colour is described by its trichromatic coefficients:

$$x = \frac{X}{X+Y+Z}, y = \frac{Y}{X+Y+Z}, z = \frac{Z}{X+Y+Z}$$

$$X + Y + Z = 1$$

The curves developed by much experimentation provide the tristimulus needed to create any visible wavelength.

Color Models

There are various approaches to modelling colour.

There are two very well-liked models for colour picture processing:

- RGB (Red, Green, Blue)
- HSI (Hue Saturation Intensity)

-The **RGB model** divides each colour into its fundamental spectral components of red, green, and blue. This is known as the RGB model.

The model uses a Cartesian coordinate system as its foundation.

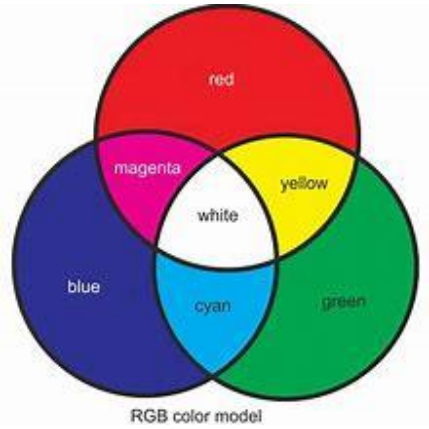
Three corners have RGB values.

Yellow, magenta, and cyan are at three additional corners.

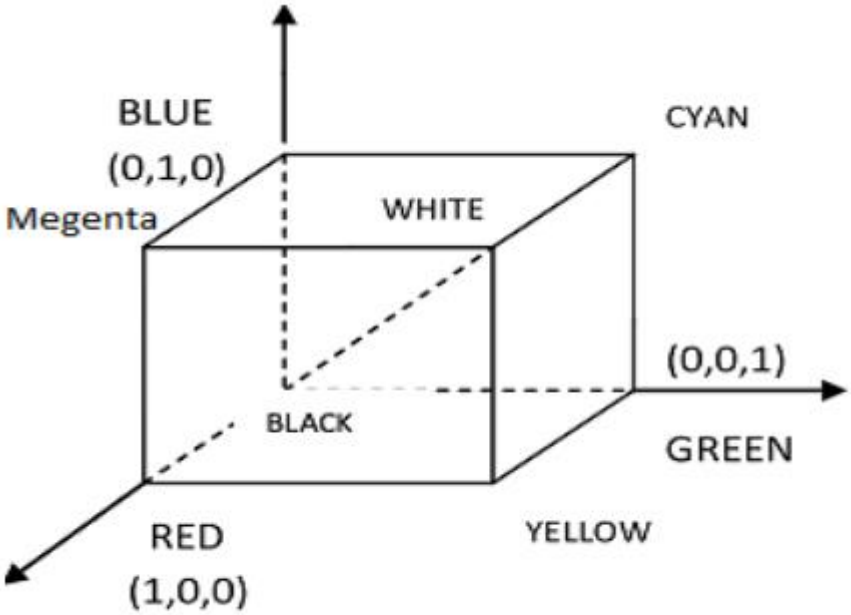
Black is the starting point.

White is the end that is farthest from the starting point.

RGB vectors represent different colours as points on or inside the cube.



RGB color model



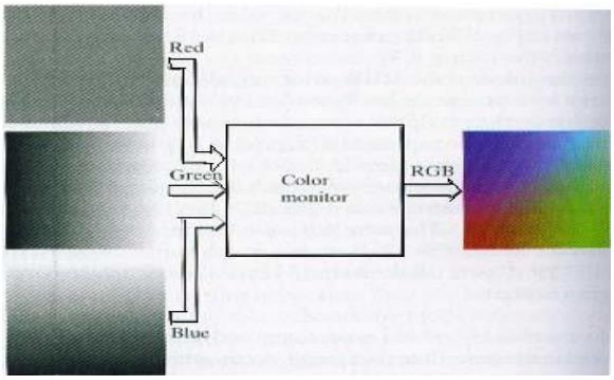
Three component pictures, one for each primary colour, are used to represent images in the RGB colour paradigm.

These photos are merged and transmitted into a monitor to produce a composite colour image.

The colour depth is the quantity of bits utilised to represent each pixel.

A 24-bit image, which supports 16,777,216 colours, is frequently referred to as a full-color image.

RGB image generation:



HSI COLOR MODEL:

RGB is advantageous for hardware implementations and coincidentally relates to how the human visual system functions.

Unfortunately, using RGB to describe colours is not especially logical.

Instead, individuals tend to utilise hue, saturation, and brightness when describing colours.

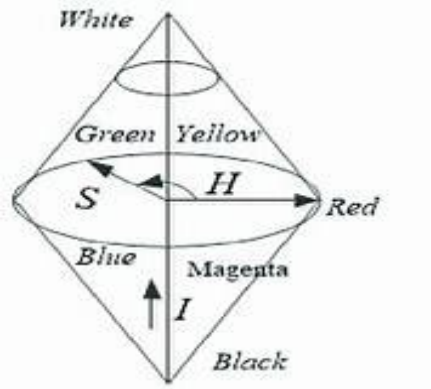
RGB is excellent for creating colours, whereas HSI is excellent for describing colours.

Hue is a colour characteristic that describes a pure colour, such as a pure red, orange, or yellow.

Saturation: Indicates how much white light has diluted a pure hue.

Brightness is extremely subjective, making it difficult to quantify. We use intensity instead.

The concept of intensity is the same achromatic one as in images with a grey level.



Pseudocolor

In pseudocolor (also known as false colour) image processing, colours are assigned to grey values according to a predetermined criterion.

Pseudocolor image processing is mostly used for human

Humans are able to distinguish between hundreds of different colour hues and intensities, as opposed to simply a few dozen or so shades of grey.

Slices of intensities:

One of the most straightforward methods of processing pseudocolor images is intensity slicing and colour coding.

As a 3D function that maps spatial coordinates to intensities, we first consider a picture (that we can consider heights).

Think about putting planes parallel to the coordinate plane at various levels.

A value is represented by one colour on one side of such a plane and a different colour on the other.

The general definition of intensity slicing is: Let [0, L-1] denote the grey scale.

Let I_0 stand for the colour black (f (x, y) = 0) and

Let I_{L-1} for the colour white (f (x, y) = L-1).

Assume that levels I_1, I_2, ..., I_p are used to define P planes that are perpendicular to the intensity axis.

The P planes divide the grey scale into P + 1 intervals, V_1, V_2, ..., V_{P+1}, assuming that 0 < P < L-1.

Then, it is possible to assign colours to grey levels using the relation:

f (x, y) = ck, if f(x, y) ∈ V, where ck denotes the colour of the kth intensity level V_k, which is determined by the dividing planes at I = k - 1 and I = k.

$$g(x,y) = \begin{cases} C_1 & if\ f(x,y) \le T \\ C_2 & if\ f(x,y) > T \end{cases}$$

C_1 = Color No. 1

C_2 = Color No. 2

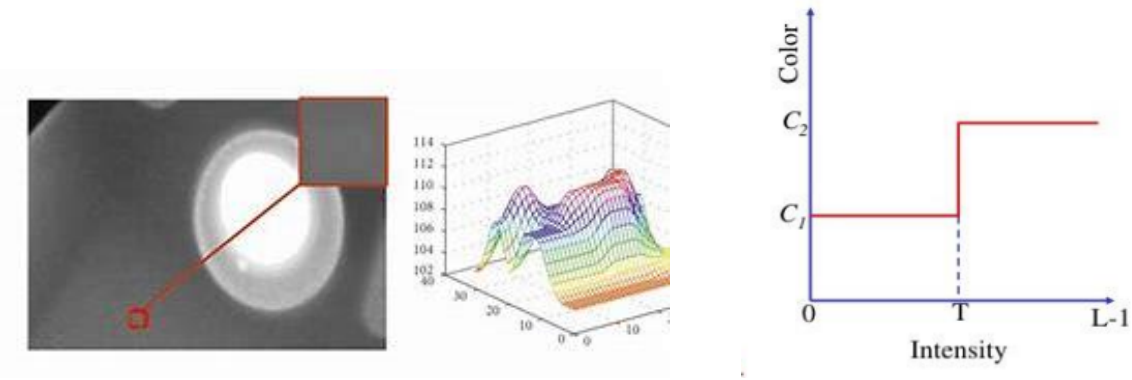


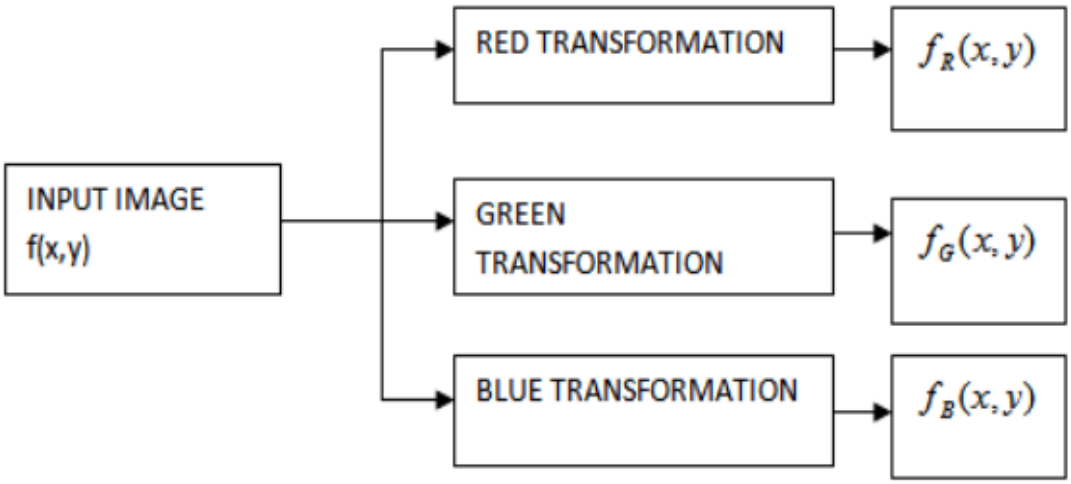
Fig: A grayscale image viewed as 3D surface

Intensity to Color Transformation

The intensity changed three times independently.

The R, G, and B channels receive the results.

The final composite image draws attention to specific image elements.



Grayscale ranges can be highlighted by altering the phase or frequency of the transformation functions, which are sinusoidal.

The pixels with intensities in the valleys are given a strong colour by a modest modification in the phase between the transformations.

Increasing or decreasing the phase or frequency of the transformation algorithms can highlight particular grayscale ranges.

Pixels with intensities in the valley are given a strong colour by a minor adjustment to the phase between the transformations.

Each colour can also be transformed into a grayscale image separately. Intensity adjustment, colour complement, colour slicing, tone correction, and colour imbalance correction are some examples of colour transformation processes.

Image Compression

What is image compression?

The technique of encoding or transforming an image file so that it uses less space than the original file is known as image compression.

It is a form of compression method that shrinks the size of a picture file without significantly impacting or lowering its quality.

An image/data compression technique or codec is used to achieve image compression. Such codecs/algorithms use a variety of methods to minimize the size of the image, including

- Identifying all pixels with the same color using their name, code, and number of pixels. One pixel can thus represent hundreds or thousands of other pixels in this manner.
- Wavelets in mathematics are used to construct and represent the image.

Why do we need image compression?

It saves storage, makes computation faster on an image and also decreases the transmission time of over a channel.

A compressed picture uses less memory. Your memory card may store 100 compressed photos or 10 raw (uncompressed) photographs.

While browsing we might have to wait 5 seconds to get a raw image from the Internet or 1 second to download a compressed image on a mobile phone. This improves user experience while lowering the cost of your data plan.

Some important terms: -

Uncompressed image: - Images in its raw form or original form without compression is called uncompressed image.

Compressed image: - Images which are compressed using some algorithm or codec to reduce its size is called compressed image.

Decompressed image: - Images which are converted to its raw form after inverse operation are called decompressed images.

In general a decompressed image is not exactly same as its original form due to some loss of information due to compression and decompression.

Compression Ratio: - It is the ratio of number of bits used to store original image N_1 to the number of bits used to store compressed image N_2.

$$C = \frac{N_1}{N_2}$$

Where C is compression ratio.

Image compression model: -

Image compression model include two primary elements. An Encoder (compressor) and Decoder (Decompressor). The encoder receives the picture f(x,y), which it then modifies to make it appropriate for transmission. This is the place where image compression takes place. This sent signal is received by the decoder, which then reconstructs the output image f(x,y).

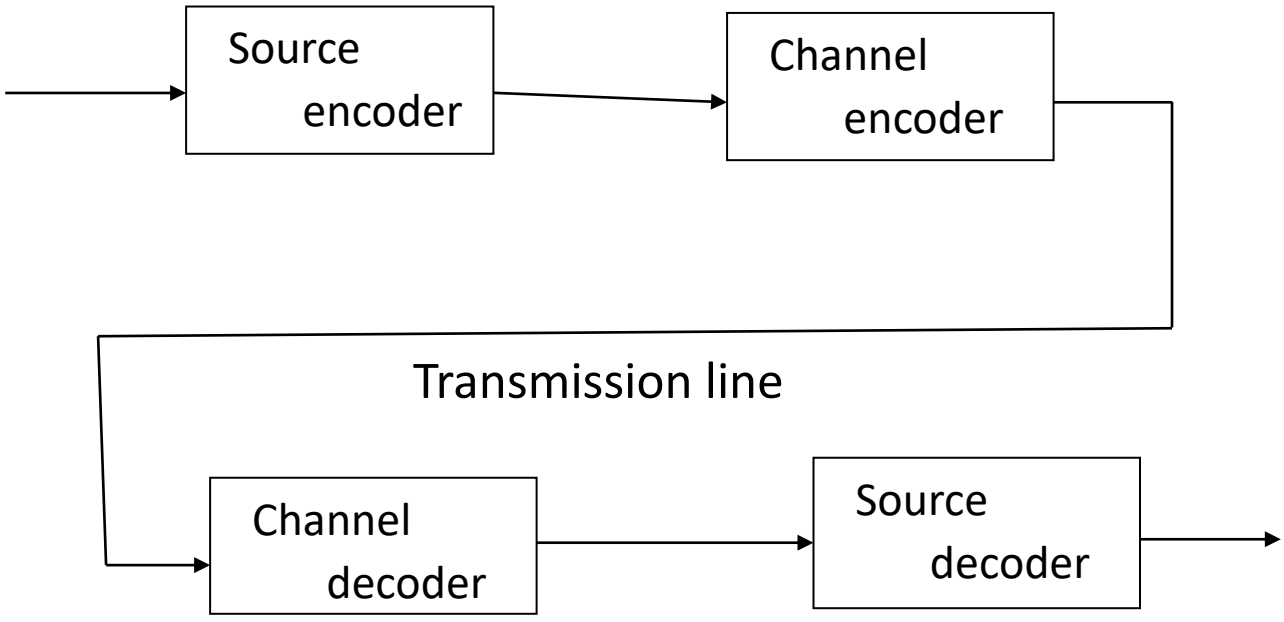
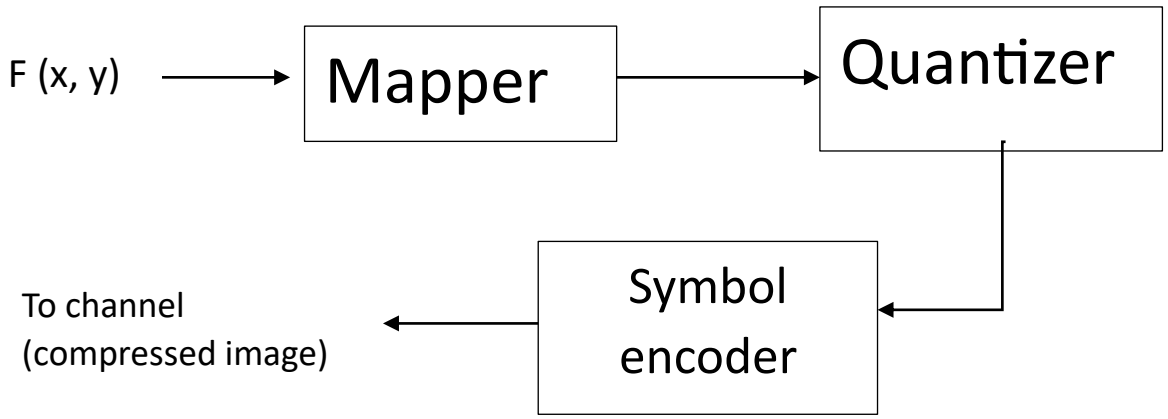


Image Compression model

Each encoder and decoder consists of two blocks. A Source encoder and a Channel encoder make up the encoder. While the channel encoder improves the source encoders' noise immunity, the source encoder eliminates input data redundancy. A source decoder and a channel decoder make up the decoder. The channel decoder's job is to make sure the system is noise-resistant. Therefore, the channel encoder and channel decoder are eliminated if the channel between the encoder and the decoder is noise-free.

Source encoder and source decoder:-

Source encoder is responsible for reducing or eliminating any coding, ingerpixel, or psychovisual redundancy.

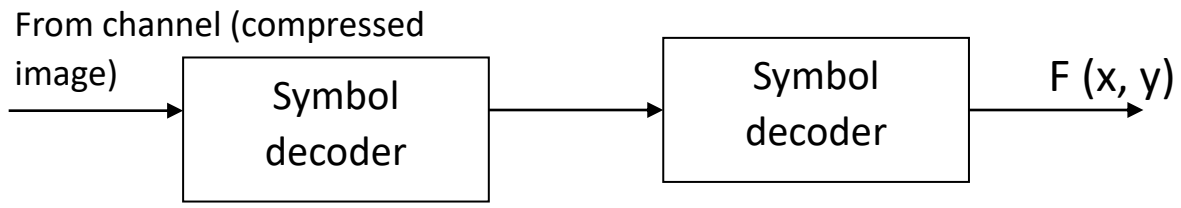


Souce encoder

The first "Mapper" block converts the input data into a (typically nonvisual) format that is intended to eliminate interpixel redundancy. This block is reversible and may or may not reduce data size. Examples include run-length encoding and picture transformation.

The Quantizer decreases the accuracy of the mapper output based on a fidelity threshold. This block decreases psychovisual redundancy and is not normally invertible.

The Symbol Encoder generates a fixed or variable length codeword to represent the quantizer output and translates it to the output. This block can be reversed, which lowers coding redundancy.



Source decoder

Except for the quantizer block, which is not invertible, the decoder blocks are inverse operations of the equivalent encoder blocks.

Compression Algorithm:- The compression algorithm's function is to compress the source data and then decompress it to obtain the original data. einsteineruploading up to get together with. Every compression method consists of two parts:

1. Modeller:- The modeller's objective is to use data knowledge to condition picture data for compression. The modeller can be found on both the transmitter and recipient sides. The models might be static (i.e., the models at the transmitter and receiver do not change) or dynamic (that is, the models change depending on the change of data during the compression or decompression process). The technique may alternatively be characterised as static or dynamic compression algorithm based on the models utilised.
2. Coder: - The coder is the second component. The encoder is the sender-side coder. This codes the symbols either alone or in conjunction with the model. The decoder is the receiver-side coder that decodes the message from the compressed data.

If the models at the sender and receiver side are same, the compression scheme is symmetric and if they are different compression algorithm is asymmetric.

Different ways of classifying image compression algorithm are as follows:-

1. Entropy encoding
2. Predictive coding
3. Transform coding
4. Layered coding

1. **Entropy encoding: -** The idea behind this category is that if pixels are not evenly distributed, then a suitable coding method may be chosen to store the information in such a way that the average number of bits is smaller than the entropy. The minimal number of bits utilised to encode information is specified by entropy (explained in Chapter 3). As a result, the entropy of the source and the chance of occurrence of the symbols are used to determine the coding. This gives rise to the concept of variable length coding. Huffman coding, arithmetic coding, and dictionary-based coding are some examples of this sort of coding.
2. **Predictive coding: -** The goal of predictive coding is to reduce the mutual reliance between subsequent pixels before encoding. The samples would normally be relatively big, but the variations would be minor.

   Lets take a example of the following poxels need to be transmitted:

Pixel	400	405	420	425
Differences	5	15	5	

   It can be seen that the pixel difference is always less than the original and takes fewer bits to represent. As a result, encoding the differences rather than the original makes sense. This method, however, may not be useful for fast changing data such as {300, 4096, 128, 4096, 151}. As a result, for slowly varying data, the number of bits necessary to code the difference is quite minimal. Differential pulse code modulation (DPCM) and delta modulation methods are examples of this group.

3. **Transform Coding:-** The goal of transform coding is to take use of the transform's information-packing capacity. The energy is compressed into fewer components, which are then encoded and sent. Lower spatial frequencies are more sensitive to the human eye than higher spatial frequencies. To generate compression, the redundant high frequency components are removed. The elimination of these frequency components results in information loss. Nonetheless, if the data loss is bearable, it can be employed for image and video applications. Hence, frequency selection, information packing, and the idea of basis pictures constitute the foundation of transform coding.

4. **Layered Coding: -** Layered code comes in handy when dealing with layered pictures. Layers are sometimes used to depict images. Pyramid data structures are excellent for representing a picture in this multiresolution manner. Depending on the application, the tiers of a pyramid would be delivered. These pictures are sometimes split as foreground

and background, and encoding is done dependent on the demands of the application. This can also take the form of selected frequency coefficients or selected bits of an image's pixels.

Data Redundancy:-

Repetitive data is known as redundancy. This might be overlapping information or data with some shared properties. The presence of this redundancy might be either implicit or apparent.

Types of Data Redundancy:-

1. Coding redundancy
2. Inter-pixel redundancy
3. Psycho-visual redundancy
4. Chromatic redundancy

1. Coding redundancy: -

Coding redundancy is produced by inadequate coding method selection. As previously explained, a coding system assigns a unique code to each symbol of the message. The message '4', for example, can be represented as 100. The bit stream '100' is then transferred through the transmission channel and decoded as message '4' at the reception end. The message is obviously encoded using binary coding. Binary coding is an example of a coding technique that uses only two code symbols (0, 1). As a result, the message is mapped to a codeword containing the letters Os and Is. There are several coding systems to choose from. ASCII (American standard code for interchange of information) is a 7-bit coding. Another common coding is grey code. A block code is one that has a fixed mapping between the source symbols and the code. A code is said to be uniquely decodable if it can be decoded just once. A non-singular code is one in which the codeword of a block code is separate. Instantaneous code is used when decoding is achievable without knowledge of the following code phrases.

It has been discovered that when using binary coding to encode six symbols, at least three binary bits are necessary. It may be demonstrated that variable length codes, such as binary codes, can be employed in the majority of circumstances. The objective behind variable length coding is to utilise shorter codes for often occurring symbols and longer codes for

seldom occurring characters. The Huffman code, a variable coding scheme, uses fewer bits to convey the same information. As a result, variable length coding is superior to fixed binary coding. As a result, a poor choice of code generates unneeded extra bits. These additional bits are referred to as redundancy.

[Coding redundancy = Average bits used to code – entropy]

Let us assume, once again, that a discrete random variable r_k in the interval [0, 1] represents the gray levels of an image and that each r_k occurs with probability pr (r_k).

$$P_r(r_k) = \frac{n_k}{n}$$

Where k = 1, 2, 3, 4, ... , L-1

where L is the number of gray levels, nk is the number of times that the kth gray level appears in the image, and n is the total number of pixels in the image. If the number of bits used to represent each value of rk is l (rk), then the average number of bits required to represent each pixel is

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

The average length of the code words allocated to the various gray-level values is determined by adding the product of the number of bits used to represent each grey level and the chance that the grey level occurs. As a result, the total number of bits needed to code a M X N picture is MNLavg.

2**. Inter-Pixel Redundancy:-** All pixel in an image is not independent instead they are interrelated with its neighbouring pixels. Due to which we can predict the value of a pixel using its neighbouring pixel. Several pixels that are not actually essential contribute to the visual aspect of the image backdrop. This is known as spatial redundancy (or geometrical redundancy). Spatial redundancy can exist inside a single frame (intra-frame) or between many frames (inter-frame or temporal redundancy). The autocorrelation function can be used to investigate spatial redundancy in images.

Large sections of the image may have the same qualities, such as colour and intensity, in intra-frame redundancy. This type of redundancy is known as spatial redundancy. Subsampling, in which alternate pixels are utilised to lower the bits, is one method for reducing duplication. The average of the collection of pixels may also be used to calculate subsampling. Another method for reducing inter-pixel dependence is to employ quantization, which uses a predetermined amount of bits to minimise the bits. Algorithms such as predictability coding approaches, bit-plane algorithms, run length coding, and dictionary-based algorithms are used to alleviate inter-pixel dependence.

3. Psychovisual Redundancy:- The majority of imaging applications generate pictures that are seen by people. Psychovisually redundant pictures are those that provide little or no information to the human viewer. For example, the human visual system is extremely sensitive to information like edges and textures. In most situations, the contour of the item reveals the thing's structure. As a result, the human visual system prioritises this type of information over all other picture information. This type of redundant information is known as psychovisual redundancy. Uniform quantization, which reduces the amount of bits, is one method for resolving this duplication. The image's least significant bits (1.SBS) do not carry much information and may thus be eliminated. This can sometimes result in edge effects that can be eliminated using a technique known as enhanced grey scale (IGS). This is a coding technique in which the least important bits of each pixel are eliminated. Then, depending on the four LSBs of a pixel and its previous pixel, a random number is produced and appended to the current pixel value. This is done to avoid edge effects, and the changed pixel values are subsequently shown. But, if the pixel is of the type 1111 xxxx, 0000 is appended to avoid overflow.

4.Chromatic redundancy :-

The presence of superfluous colours in an image is referred to as chromatic redundancy. Color channels in pictures are closely linked, and the human visual system is incapable of seeing millions of colours. As a result, colours that are not recognised by the human visual system may be deleted without impacting image quality.

The condition of exact data replication is referred to as fidelity. Distortion is the difference between the original and reconstructed pictures. The degree of distortion should be

determined. Subjective fidelity metrics are based on the belief that picture quality evaluations are frequently subjective. Subjective evaluation may be compared to the process of purchasing a television set in a store with a large selection of television sets.

Types of data compression:-

There are mainly two types of data compression techniques –

Lossless Data Compression: -

The files are compressed using lossless data compression, which preserves the information and quality of the original files. i.e. lossless data compression reduces file size while maintaining the integrity of the contents. The primary benefit of lossless data compression is the ability to recover the original data in its original format following decompression.

The PNG, RAW, GIF, and BMP file formats as well as sensitive papers and secret information typically uses lossless data compression.

Some of the Lossless data compression techniques are -

1. Huffman Coding
2. Lempel Ziv - Welch (LZW)
3. Run Length Encoding (RLE)
4. Arithmetic Coding

Huffman coding: -

Huffman coding is a type of variable length coding. Coding duplication in Huffman coding can be minimised by selecting a better method of assigning the codes. The following is the Huffman coding algorithm:

1. Make a list of the symbols and arrange them.

2. Choose the two symbols with the lowest probability.

3. Add a new node. Add the probabilities of the symbols chosen in step 2 and label the new node accordingly.

4. Repeat steps 2 and 3 until there is only one node left.

5. Begin by allocating code 0 to the left tree and code 1 to the opposite branch.

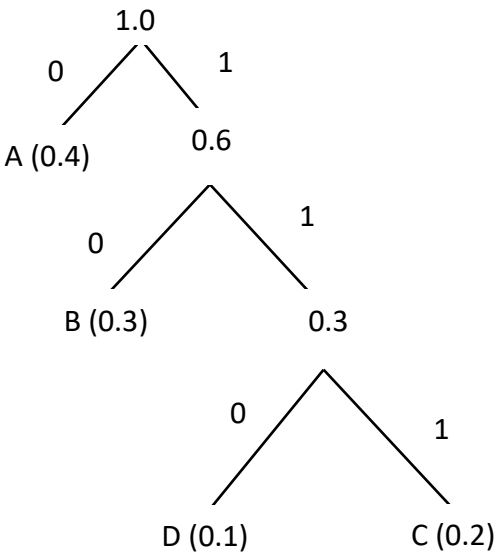
6. Follow the code from the root to the leaf for each label.

Example:-

Calculate the Huffman coding for the given set of symbols.

Symbols	Probability
A	0.4
B	0.3
C	0.2
D	0.1

The Huffman tree constructed for the given input is as follows:-



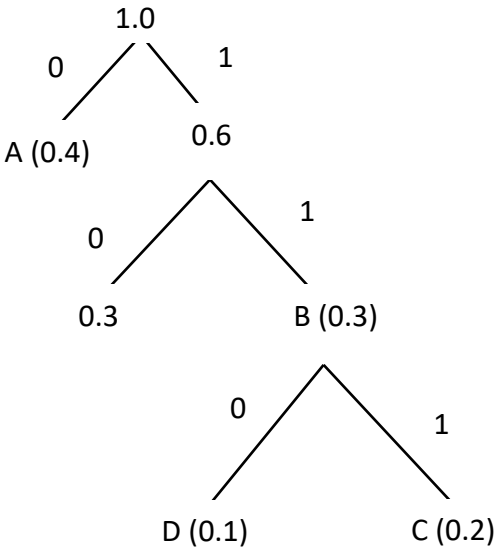
Huffman coding tree

The symbols with their Huffman codes are as follows

Symbols	Huffman code
A	0
B	10
C	111
D	110

There is problem with the Huffman code is that u may come across two different Huffman code tree for same input and for both the Huffman code will be different for the symbols. For example in the above Huffman tree B (0.3) and 0.3 can interchange their position to form another Huffman code tree. And for that tree Huffman code will be different form the previous one.

Another Huffman code tree for the same input is



Huffman coding tree

The symbols with their Huffman codes for this alternate Huffman code tree are as follows

Symbols	Huffman code
A	0
B	10
C	101
D	100

Huffman decoder: -

The decoding process used by the Huffman decoder is as follows:

1. Locate the coded message. Begin at the root.

2. Go to the left if the read bit is 0. Otherwise, walk to the tree's right.

3. Continue until you reach the leaf. Then regenerate the code and restart from the beginning.

4. Repeat steps 1-3 until the message is finished.

$$Huffman\ code\ efficiency = \frac{Entropy}{Average\ length\ of\ huffman\ code\ used}$$

Lossy Data Compression: -

Larger files can be split up into smaller files via lossy data compression. With this compression method, the original file has a certain amount of data and quality lost. Due to the loss of original

data and quality, it uses less memory than the original file. When the quality of the data is not our top priority, this strategy is typically helpful for us.

Lossy data compression is most widely used in JPEG images, MPEG video, and MP3 audio formats.

Note: - Human eye cannot measure the loss of data in the image.

Some important Lossy data compression techniques are -

1. Lossy predictive coding
2. Vector quantization
3. Block transform coding

Lossy Predictive coding: -

Predictive coding can also be used as a lossy compression method. Instead of taking precautions, take the greatest value for 5 bits, which is 31. The number of bits is dramatically reduced as a result. Unfortunately, information loss is increasing.

Value	Lossy predictive coding
23	23
64	64 – 23 = 41(crosses the threshold of 5 bits). However, store only 31 supported by 5 bits + one sign bot = 6 bits.
39	39 – 64 = -25
47	47 – 39 = 8
55	55 – 47 = 8
63	63 – 55 = 8

The amount of bits utilised to transmit is the same as in the original system, but the value 31 is sent instead of 41, resulting in an error. This information loss causes an error, which results in a lossy compression technique. This technique just requires 6 x6-36 bits.

Delta modulation: -Delta modulation goes one step farther by encoding the quantized error value with only one bit. Something may be both beneficial and terrible. In this case, the predictor is specified as

$$f' = \alpha * f_{n-1}$$

Where α is called the prediction coefficient. More generally, for the first digit,

$$f' = f'_{n-1}$$

Then the error is computed as follows:

$$e_n = f_n - f'_n = f_n - f'_{n-1}$$

The error is quantized as follows:

$$e_n = \begin{cases} +\zeta & for\ e_n > 0 \\ -\zeta & otherwise \end{cases}$$

This ζ is the positive number. If major data transitions occur often, this approach causes issues. It is known as an adaptive delta modulation method when ζ is modified.

Difference between lossless and lossy image: -

Lossless Data compression	Lossy Data compression
In lossless data compression, there is no loss of any data and quality.	In lossy data compression, there is a loss of quality and data, which id not measurable.
In lossless, the file is restored in its original form.	In lossy, the file does not restore in its original form.
Lossless data compression algorithm are Run Length Encoding, Huffman encoding, Arithmatic encoding, Lempel Ziv Welch encoding, etc.	Lossy data compression algorithms are: Transform coding, Discrete Wavelet Transform, fractal compression, etc.
Lossless compression is mainly used to compress text-sound and images.	Lissy compression is mainly used to compress audio, video and images.
As compare to lossy data compression, lossless data compression holds more data.	As compare to lossless data compression, lossy data compression holds less data.
File quality is high in the lossless data compression.	File quality is low in the lossy data compresssion.
Lossless data compression mainly supports RAW, BMP, PNG, WAV, FLAC, and ALAC file types	Lossy data compression mainly supports JPEG, GIF, MP3, MP4, MKV, and OGG file types.

Image Compression Standard: -

The Joint Photographic Experts Group (JPEG) was established in 1986 to define techniques for still image compression, including both colour and monochrome versions. In 1992, JPEG became a recognised worldwide standard. A Lossy picture compression technique is JPEG.

JPEG:-

The JPEG format offers the following four modes of operation:

1. Sequential DCT- based mode(Baseline algorithm)
2. Lossless mode
3. Pregressive DCT- based mode
4. Hierarchical mode

1. Sequential DCT- based mide(baseline algorithm):

The baseline algorithm comsists of the following steps:

Image preparation :- The input image may be in RGB true colour image.Hence for image processing it is necessary to comvert it into YCbCr image. Now the image is being divided into subblocks of 4x4, 8x8 or 16x16. For example, if image is 1024x1024 and the sub block size chosen is16x16 then there will be 1024/16 x 1024/16 = 64 x 64 blocks.

Image Transform:- The subblocks are transform from spetial domain to frequency domain using image transforms like DCT or wavelets.

Quantization:- After conversion to frequency domain, all the frequency component are not required as our huma eye can not identify all the frequency of the image, it sensitive to only low frequency components. For removing unidentifiable frequency , a threshold value is applied. All the frequency component whose value is lesser than threshold are removed from image.

Entropy Encoding :- Entropy coding methods are used to encode the quantized data. The initial value (0, 0) is referred to as the DC coefficient and the following 63 components are

referred to as AC coefficients if an 8x8 block is specified. The average information of the picture is captured by the DC coefficient, which has a greater amplitude. The DC coefficient is encoded via differential encoding. This fast-varying data make up the remaining 63 AC coefficients. They are set up in a zigzag pattern.

Frame building :- The transmission needs are taken into consideration in this last step. Several guidelines and requirements must be followed during transmission. As a result, extra details including the start bit, end bit, data type, and type of the picture are added to the frame header. The frames are packed with the compressed data and transmitted over the channel.

2. **Lossless mode:-** JPEG also offers lossless compression. Using this mode we can create the exact duplicate of the original image. In this mode the value of lost pixel is predicted using the neighbouring pixels.

		p	**q**	**r**				
		s	**x**	**t**				
		u	**v**	**w**				

The value of x is predicted using the value of p q r s t u v w.

3. **Pregressive Encoding :-** Progressive encoding uses the priority of the pixels to gradually compress data. The receiver receives the coarsest possible rendition of the image first. Then the new data is sent, gradually improving the image quality. The benefit of this system is that the user may regulate the encoder's stop procedure, which allows them to manage the amount of loss or image quality. This is helpful for online applications where the consumer is not willing to wait around for a picture to download for a longer period of time. So, it is preferable to obtain the coarse version of the image, and if the user is willing to wait longer, better-quality information may be supplied. This mode is comparable to the sequential DCT mode. The DCT coefficients are kept in a buffer after the subblocks have been DCT coded. After that, several spectral bands are created using the frequency coefficients. The zigzag scanning order is used to determine the spectral bands. Then, in this mode, low frequency coefficients are supplied first, followed by the remaining

coefficients, in a progressive pattern known as spectral selection. Another method is the consecutive method, in which the quantized coefficients' most important bits are delivered first. The extra, less important pixels' bits are then sent.

4. **Hierarchical mode :-** The pyramidal data structure, which is used in this technique, stores the picture at various resolutions. The benefit of this is that they provide the user the option to bargain with the application to reach the desired outcome. Thus, this mode is helpful for imaging applications that require several resolutions. In a pyramid data structure, the bottom layer represents the original picture, while the successive layers represent a subsampled version of the original image since every layer differs from the subsequent layer by a sampling factor of 2. Consequently, the differential frames are encoded using predictive coding. As a result, both progressive and lossless coding are supported by hierarchical coding.

Hardware & Software Requirements:

Hardware:

Computer system.

Configuration: intel i5 processor (8 gen or above), 8gb RAM, 20gb free disk space

Cameras- 2

Flash Lights

Camera stand

Software:

Windows 10/11

Visual studio code.

Python and its IDE

HTML, CSS, JavaScript, Flask development Kit and their extensions.

Libraries & Modules: OpenCV, Argument parser, NumPy, math, TensorFlow, OS, Multiprocessing, warning, iterator, SciPy, Skimage, Flask, Html, time etc .

Algorithm: -

Step 1: Run code, It will create Flask server for webpage to run.

Step 2: Click on the link and go to homepage.

- In the homepage there are two buttons.

For Registration: --

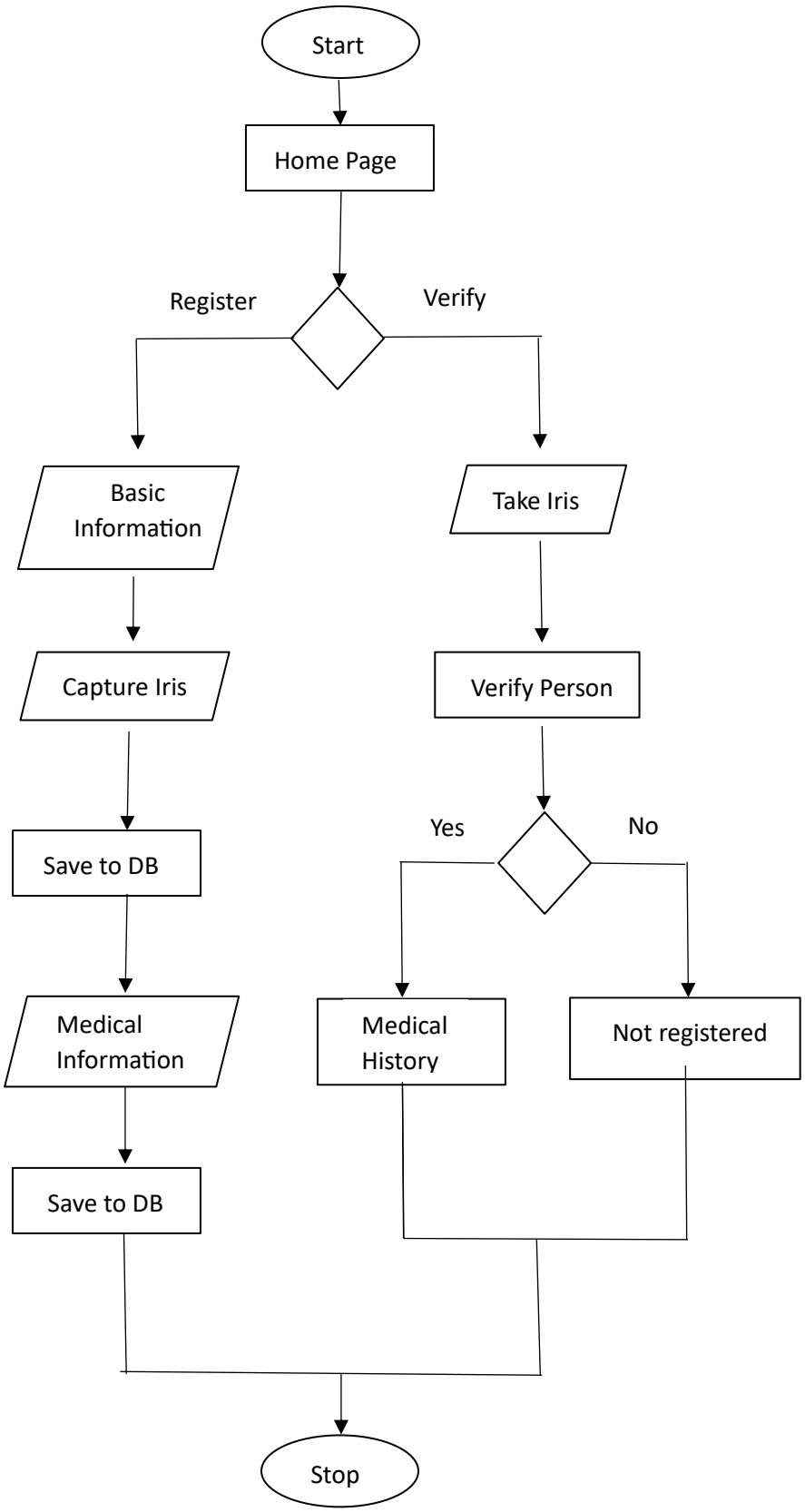
- Click on registration button.
- New page will open for collecting basic information, fill required fields and click next.
- All information will be stored in variable in html using
- Click on submit to store all these information in database.
  - For connecting database with python script file we used library named mysql_connector.
  - Use hostname, User and Password to establish connection with mysql.
- After clicking next button camera will open using opencv to capture iris.
- Press enter to take iris image and store iris in database with name as Primary key of registering person in database.
- Extract iris features from image and save its encoded pattern in .mat format.
  - To extract features we performed number of process on iris image.
  - Passing image in segment function which will find the inner boundary, outer boundary coordinates and radius of iris using canny edge detection and Hysteresis thresholding (used to binarize the image).
  - Find the mask for the top eyelid region and bottom eyelid region. This mask is used to eliminate the noise of eyelashes from the iris.
  - These inner and outer boundary coordinates are used to perform contour circular integral using Discrete Rie-mann approach.
  - Then outputs of segment is passed in normalize function which will convert the circular iris into a rectangular shape.
  - It will change iris into polar array convert noise into polar noise.
  - These polar forms are passed in encode function that will make a template of iris. It conclave each row of an image with 1d log-Gabor filter.
  - The result of encode is binary iris biometric template and binary iris noise mask.

- Open another camera for taking fingerprint using the earlier used python cv2 function videocapture().
  - An already size-defined rectangular box is formed.
  - The concerned candidate need to place his finger within the rectangular box for better fingerprint lines extraction.
  - The prototype will ignore any object outside the rectangular box for evaluation and matching of points in the further steps.
  - Press enter to take image then this image will be cropped for fingerprint using OpenCV.
  - The model with wait till the user presses any key.
  - It accepts and saves the input as soon as enter key is pressed.
  - The image is read and it's colour is converted from BGR to Gray.
  - Fingerprint is extracted from image using "Gaussian Adaptive Threshold Algorithm".
  - To apply basic thresholding, a manual threshold value T is supplied.
  - Otsu's method can automatically determine the value of T.
  - Just having one value of T may not be sufficient. To overcome this problem, adaptive thresholding is used.
  - This method considers small neighbors of pixel and find the optimal threshold value.
  - Each pixel in neighbourhood contributes equally for calculating the value of T. The Gaussian mean of pixel is used in each region.
  - The general formula for calculating the value of T is:
    - $T = \text{mean}(I_L) - C$
- After successful iris store new page will open that contain medical form to collect medical information.
- Fill the required fields as per the knowledge and click on submit.
- All the medical information we be pushed into the database.
- Registration successful message will be displayed.

For Verification: -

- Click on verify button
- Camera will open in new window to take iris. Press enter to take iris.
- Extract features of iris from image using extract_features function.
- Matching will be done on comparison of the current iris features with all the other iris template present on the database and calculate the hamming distance.
    - To calculate hamming distance we utilizes shift method in which we shift the template and compare with other template
    - This shift method is called in range of numbers and hamming distance ratio is calculated.
- Threshold value for hamming distance is set to 0.38. Iris which is bast match to database iris will be returned.
- This returned value will used to fetch data of verifying person from the database.
- After successful verification a new page will open that will display all the basic information and its medical history in the form of table .

FLOW CHART


Start
Home Page
Register
Verify
Basic Information
Take Iris
Capture Iris
Verify Person
Save to DB
Yes
No
Medical Information
Medical History
Not registered
Save to DB
Stop

COCOMO Model

- In 1981 Dr. Berry Boehm proposed Cocomo model (constructive cost model).
- It is the most generally used software estimation model in the world.
- It is used to predict the effort for project and time required for software product development based on the size of software.

Steps for software evaluation in this model are as follows:

1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).
2. Determine a set of 15 multiplying factors from various attributes of the project.
3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

The basic estimation E_i is only determined on the basis of kilo lines of code in unit of person-month by the equation

$$E_i = a \times (kloc)^b$$

Where, a, b are the constants that depends on the project type.

Software projects under COCOMO model are classified into 3 categories:

A. Organic :- All the projects that comes under this if,
   - Project is simple and small.
   - Project team should be small ad must have some prior experience of developing similar type of projects.
   - The problem on which they are working must be well understood and have been solved in the past by the team.
   - Projects should have flexible requirement.

B. Semidetached :- All the projects that comes under this if,

- Project are complex than organic.
- Project team should have more experience, more creativity, and better guidance while preparing the software.
- The size of project should neither be so small nor too large, it should have sufficient lines of code.

C. Embedded model :- All the projects that comes under this if,
- Projects should have fixed requirement of resources.
- The development of project should be done with very high constraints.
- Project team should have highest level of complexity, creativity, and experience requirement.
- Requirement of number of person in team is higher in this model.

Types of COCOMO model:

1. Basic COCOMO Model:

   It is used for quick and slightly rough calculation of software costs as it only considers based on lines of source code with same constant variables.

   $E = a \times (kloc)^b$

   $D = c \times (effort)^d$

   $P = \frac{effort}{time}$

   Where , E = effort in person month

   D = development time in months

   P = person required

   Constants a, b, c, d for the basic cocomo model are

	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

This model does not consider different factors such as reliability, expertise hence estimation is rough.

$$E = 2.4 \times (2.224)^{1.05}$$

$$= 5.5552$$

$$D = 2.5 (5.5552)^{0.35}$$

$$= 4.556$$

$$P = \frac{0.83294}{2.3450}$$

$$= 1.2193$$

So the output will be

Effort = 5.5552

Development time = 5 month

Person required = 3

Salary of average junior developer for a month is 40,000

Price of three laptop is 60,000 + 60,000 + 1,30,000 = 2,50,000

Other expenses of camera stand, connecting cables, IR camera = 2,50,000

So, total cost of project is 40,000 x 3 x 5 = 6,00,000

(6 + 2.5 + 2.5) lakh = 11 lakh

2. Intermediate COCOMO model:

The Basic COCOMO model is expanded into the Intermediate COCOMO model, which takes a number of cost factors into consideration to improve the cost estimating model's accuracy. In order to determine the price of the programme, the estimating model uses a collection of "cost driver attributes." The connection indicates the anticipated work and time.

$$E = a \times (\text{kloc})^{b} \times EAF$$

$$D = c \times (\text{effort})^d$$

Where , E = effort in person month

D = development time in months

EAF = It is effort Adjustment factor, which is calculated by multiplying parameter values if different cost drivers parameter.

	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Classification of Cost Drivers and their attributes:

- Product attributes
- Hardware attributes
- Personnel attributes
- Project attributes

Advantages of COCOMO Model

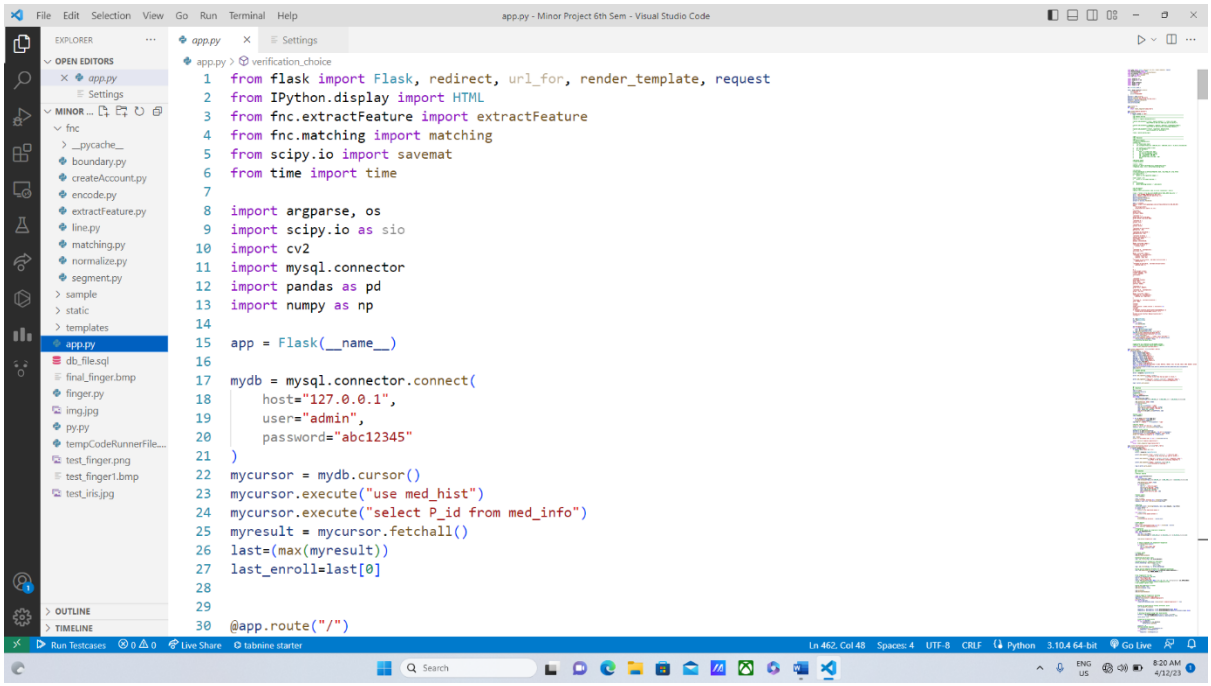
- In contrast to previous models like SLIM, COCOMO is transparent, allowing users to understand how it functions.
- The estimator may better comprehend the effects of many elements that affect project costs by using drivers.
- COCOMO offers suggestions for old projects.
- The COCOMO model makes it simple to calculate the project's overall cost.
- Understanding the effects of the many project crisis-affecting aspects is made much easier with the aid of the drivers.

Limitations of COCOMO Model

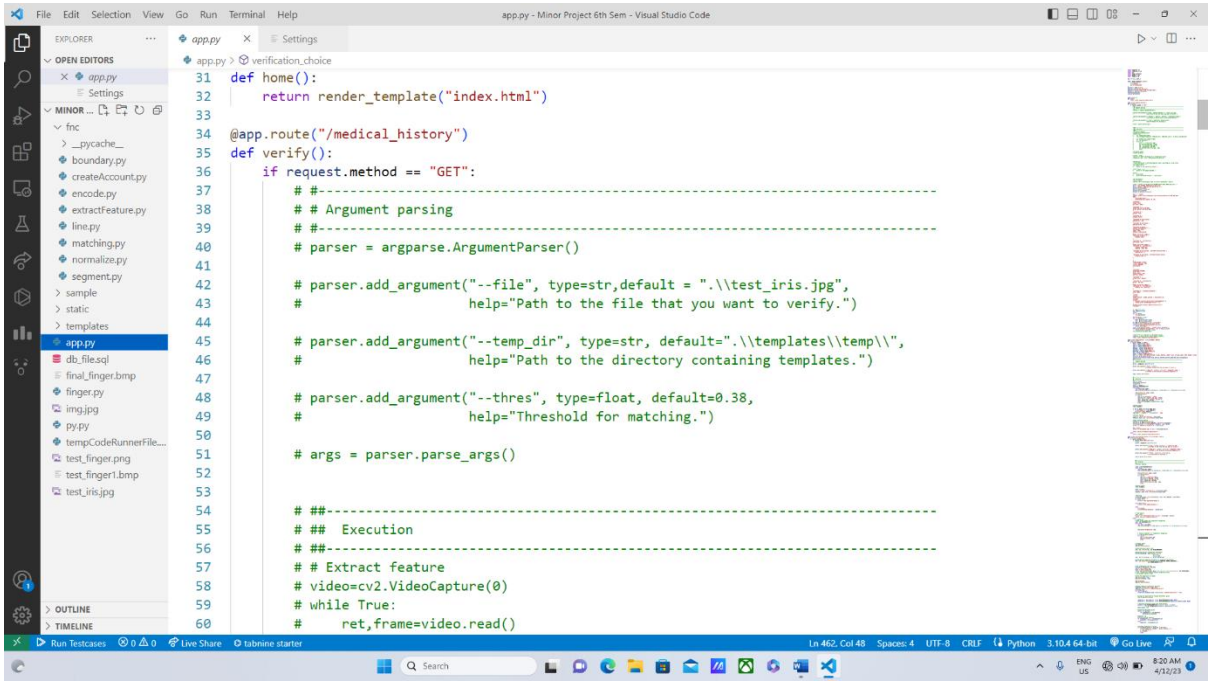
- Early in the project, when most effort estimates are needed, KDSI is difficult to estimate.
- The KDSI is a length measurement rather than a size measure.
- very prone to the development mode being misclassified.
- The model must be adjusted to the organization's requirements using past data, which is not always available. This is crucial for success.
- The cost of software is constrained in its precision.
- The abilities, collaboration, and knowledge of the consumer are also disregarded.

Implementation and Result

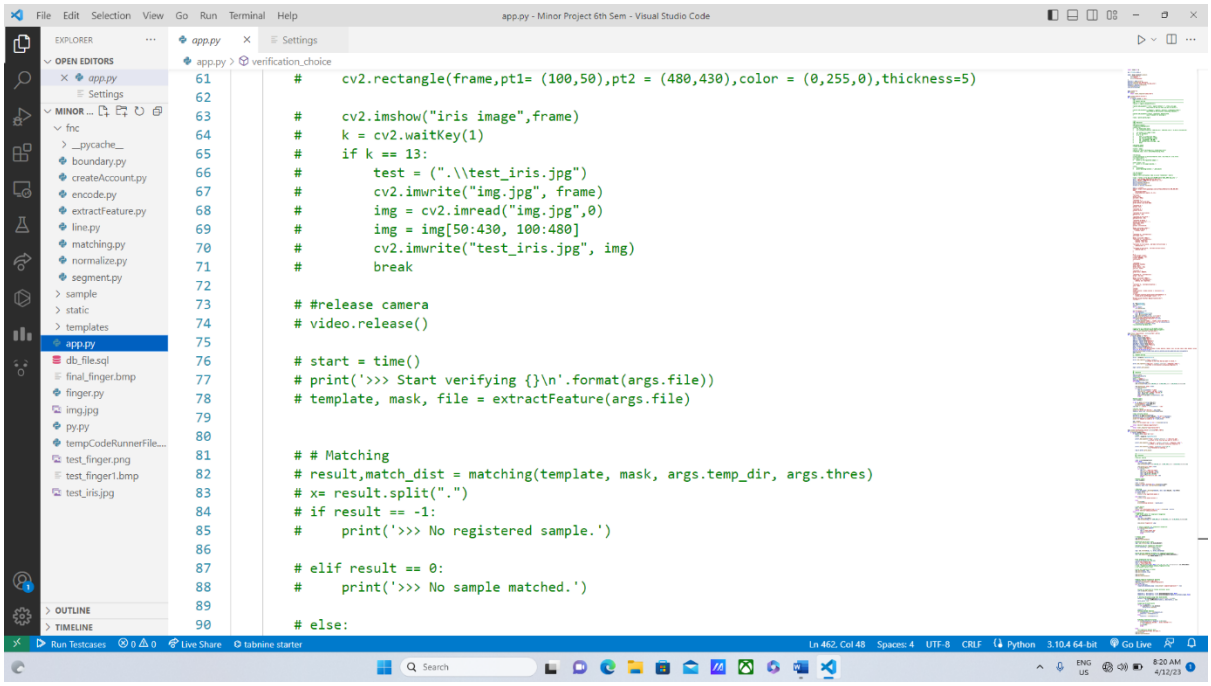
Source Code: -



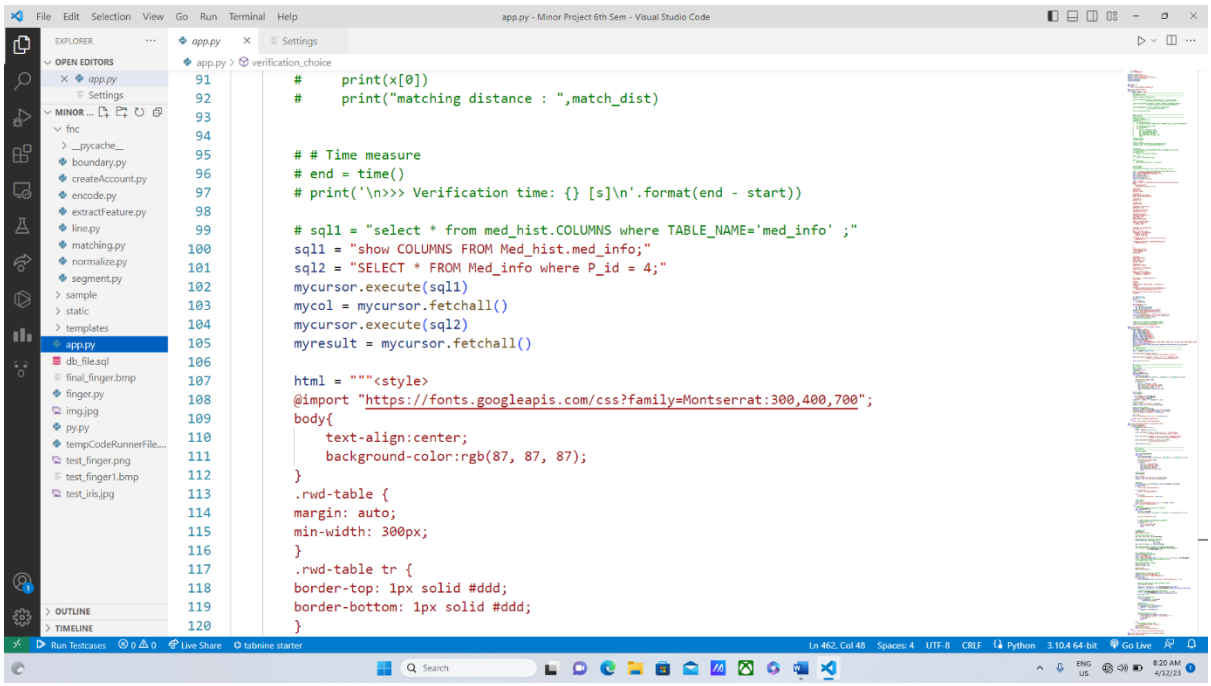
```python
from flask import Flask, redirect, url_for, render_template, request
from IPython.display import HTML
from fnc.extractFeature import extractFeature
from fnc.matching import matching
from scipy.io import savemat
from time import time

import argparse, os
import scipy.io as sio
import cv2
import mysql.connector
import pandas as pd
import numpy as np

app = Flask(__name__)

mydb = mysql.connector.connect(
    host="127.0.0.1",
    user="admin",
    password="abc12345"
)
mycursor = mydb.cursor()
mycursor.execute("use med_hist")
mycursor.execute("select P_id from med_info")
myresult = mycursor.fetchall()
last=(max(myresult))
last_enroll=last[0]


@app.route("/")
```



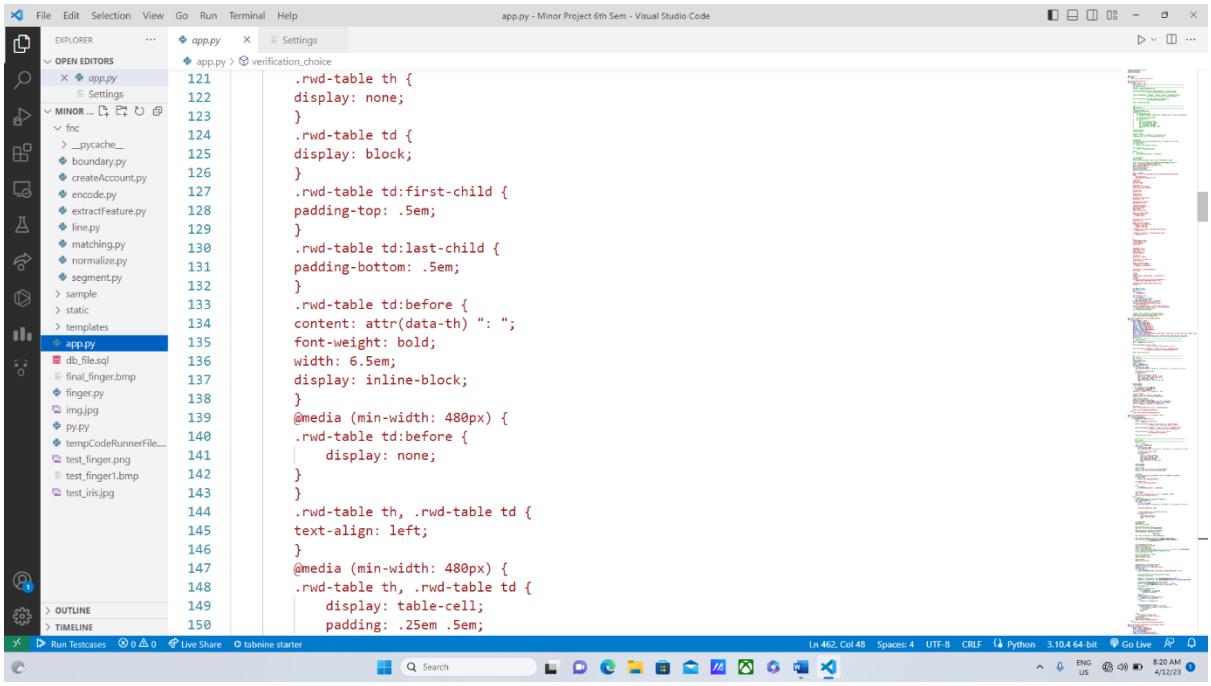
```python
def home():
    return render_template("index.html")

@app.route("/medical_history")
def verify():
    if request.method == "GET":
        # #--------------------------------------------------------------------------
        # # Argument parsing
        # #--------------------------------------------------------------------------
        # parser = argparse.ArgumentParser()

        # parser.add_argument("--file", type=str,default = ".\\test_iris.jpg",
        #                     help="Path to the file that you want to verify.")

        # parser.add_argument("--temp_dir", type=str, default=".\\templates\\temp\\",
        #                     help="Path to the directory containing templates.")

        # parser.add_argument("--thres", type=float, default=0.38,
        #                     help="Threshold for matching.")

        # args = parser.parse_args()


        # ##-------------------------------------------------------------------------
        # ##  Execution
        # ##-------------------------------------------------------------------------
        # # Extract feature
        # video=cv2.VideoCapture(0)
        # while True:
        #     ret,frame=video.read()
```



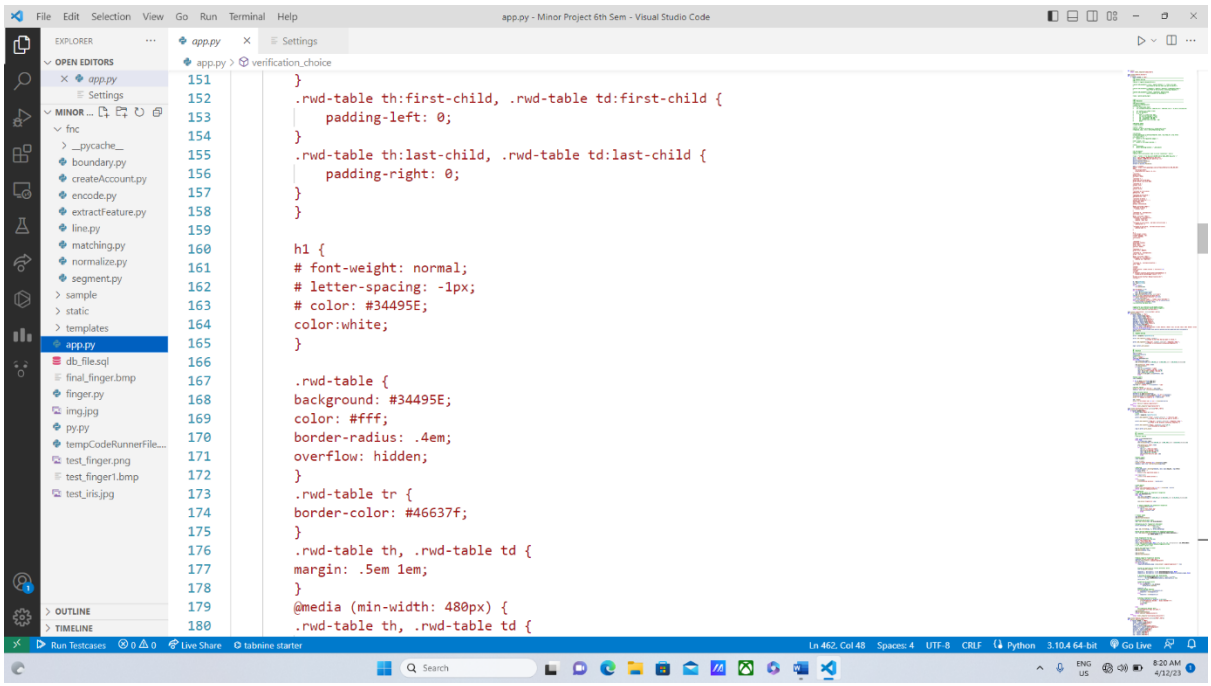
```
61      #     cv2.rectangle(frame,pt1= (100,50),pt2 = (480,430),color = (0,255,0),thickness=5)
62
63      #     cv2.imshow("iris image",frame)
64      #     k = cv2.waitKey(1)
65      #     if k == 13:
66      #         test = (".\\test_iris.jpg")
67      #         cv2.imwrite("img.jpg", frame)
68      #         img = cv2.imread("img.jpg",0)
69      #         img = img[50:430, 100:480]
70      #         cv2.imwrite("test_iris.jpg", img)
71      #         break
72
73      # #release camera
74      # video.release()
75
76      # start = time()
77      # print('>>> Start verifying {}\n'.format(args.file))
78      # template, mask, file = extractFeature(args.file)
79
80
81      # # Matching
82      # result,match_dist = matching(template, mask, args.temp_dir, args.thres)
83      # x= result.split(".")
84      # if result == -1:
85      #     print('>>> No registered sample.')
86
87      # elif result == 0:
88      #     print('>>> No sample matched.')
89
90      # else:
```



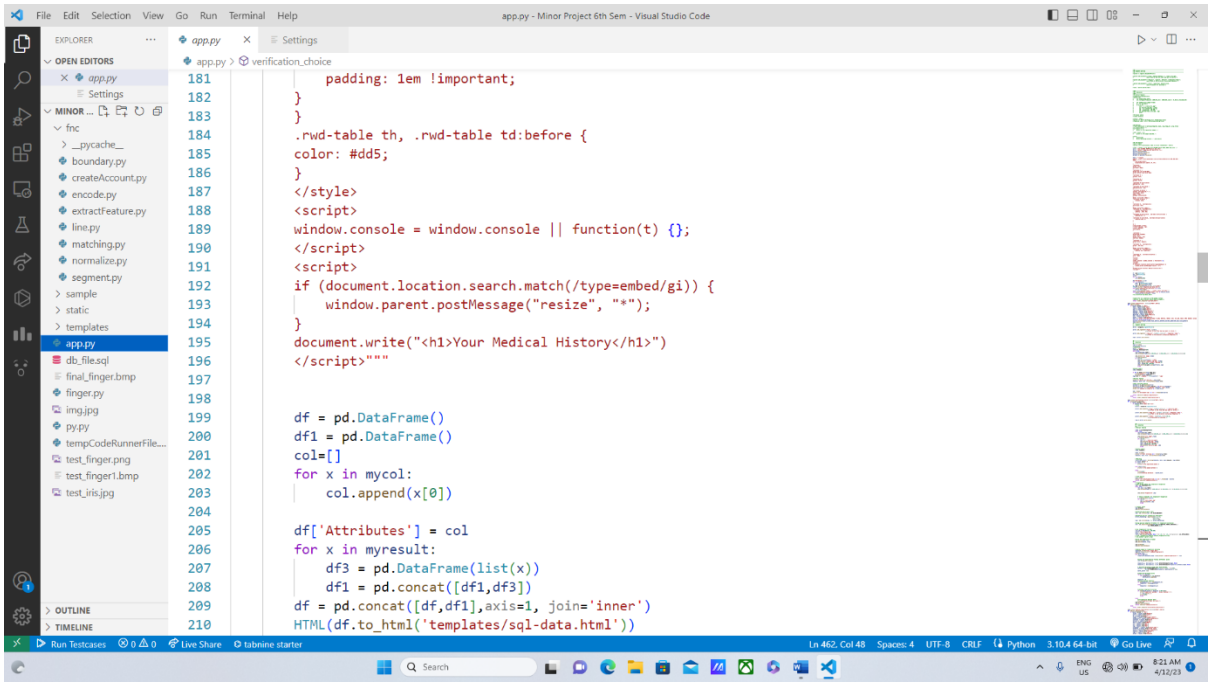
```
91      #     print(x[0])
92      #     print("matching distance : ",match_dist)
93
94
95      # # Time measure
96      # end = time()
97      # print('\n>>> Verification time: {} [s]\n'.format(end - start))
98
99      # sql1 = "select * from med_hist.COLUMNS where TABLE_NAME='med_info' ;"
100     sql1 = "show COLUMNS FROM Med_hist.med_info;"
101     sql2 = "SELECT * FROM Med_info where P_id = 4;"
102     mycursor.execute(sql1)
103     mycol = mycursor.fetchall()
104     mycursor.execute(sql2)
105     myresult = mycursor.fetchall()
106
107     html = """<style>
108     @import "https://fonts.googleapis.com/css?family=Montserrat:300,400,700";
109     body{
110         text-align:center;
111         background-color:rgb(87, 87, 87);
112     }
113     .rwd-table {
114     margin: auto;
115     min-width: 300px;
116     }
117     .rwd-table tr {
118     border-top: 1px solid #ddd;
119     border-bottom: 1px solid #ddd;
120     }
```



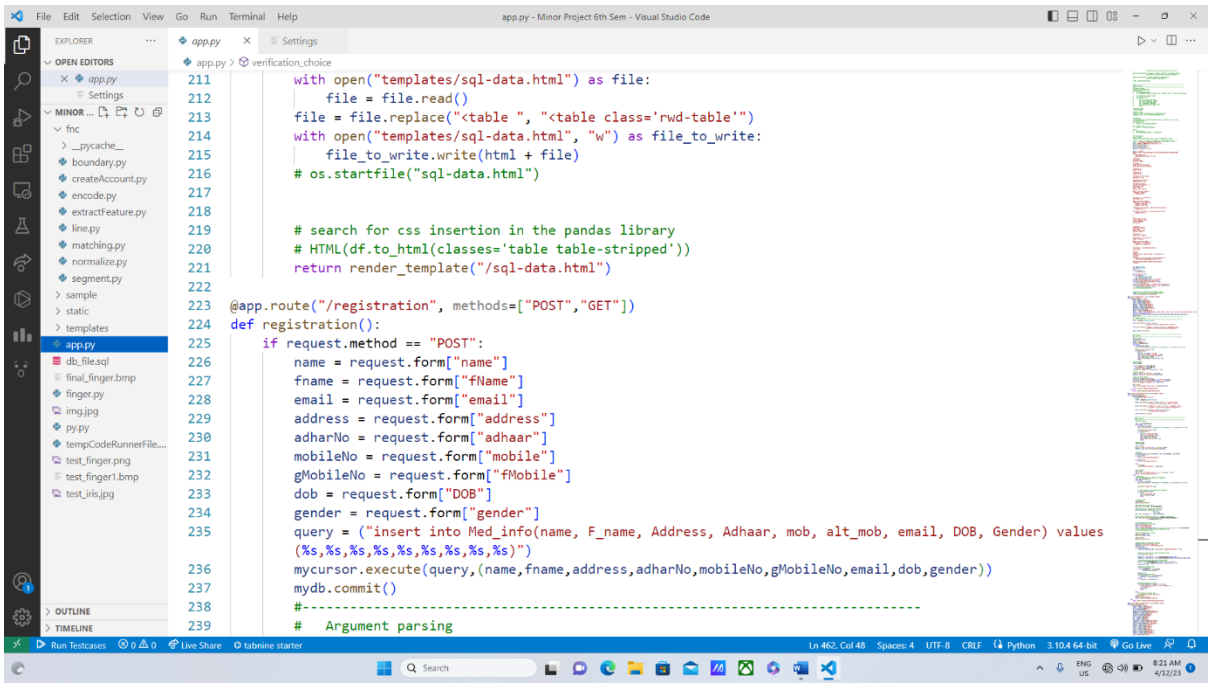
```
.rwd-table th {
display: none;
}
.rwd-table td {
display: block;
}
.rwd-table td:first-child {
padding-top: .5em;
}
.rwd-table td:last-child {
padding-bottom: .5em;
}
.rwd-table td:before {
content: attr(data-th) ": ";
font-weight: bold;
width: 6.5em;
display: inline-block;
}
@media (min-width: 480px) {
.rwd-table td:before {
    display: none;
}
}
.rwd-table th, .rwd-table td {
text-align: left;
}
@media (min-width: 480px) {
.rwd-table th, .rwd-table td {
    display: table-cell;
    padding: .25em .5em;
```



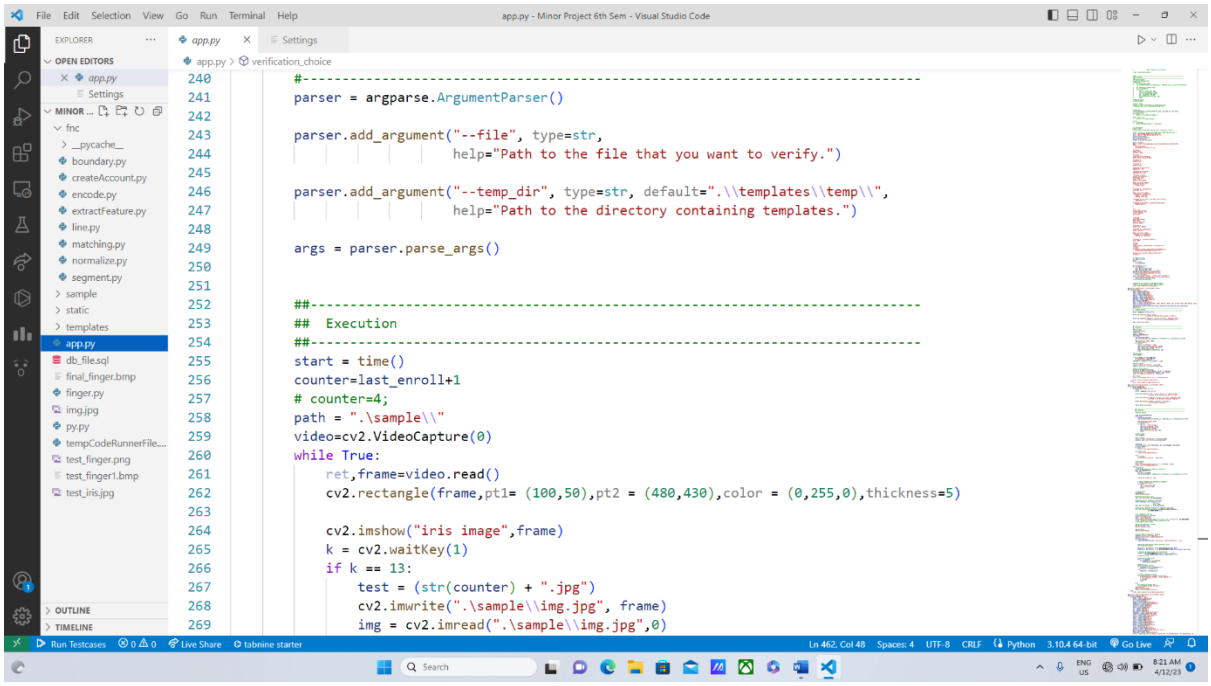
```
}
.rwd-table th:first-child, .rwd-table td:first-child {
    padding-left: 0;
}
.rwd-table th:last-child, .rwd-table td:last-child {
    padding-right: 0;
}
}

h1 {
# font-weight: normal;
# letter-spacing: -1px;
# color: #34495E;
color:white;
}

.rwd-table {
background: #34495E;
color: #fff;
border-radius: .4em;
overflow: hidden;
}
.rwd-table tr {
border-color: #46637f;
}
.rwd-table th, .rwd-table td {
margin: .5em 1em;
}
@media (min-width: 480px) {
.rwd-table th, .rwd-table td {
```



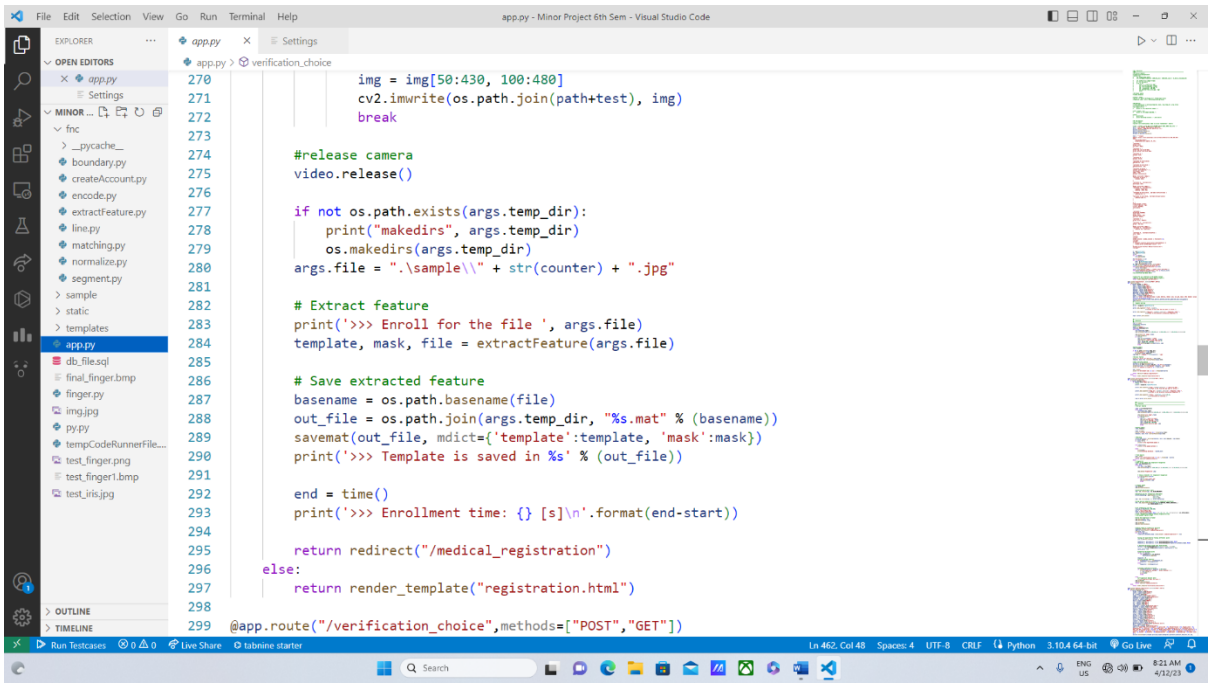
```
            padding: 1em !important;
        }
        }
        .rwd-table th, .rwd-table td:before {
        color: #dd5;
        }
        </style>
        <script>
        window.console = window.console || function(t) {};
        </script>
        <script>
        if (document.location.search.match(/type=embed/gi)) {
            window.parent.postMessage("resize", "*");
        }
        document.write("<h1>Your Medical History</h1>")
        </script>"""


        df = pd.DataFrame()
        df1 = pd.DataFrame()
        col=[]
        for x in mycol:
            col.append(x[0])

        df['Attributes'] = col
        for x in myresult:
            df3 = pd.DataFrame(list(x))
            df1 = pd.concat([df1,df3])
        df = pd.concat([df,df1],axis=1, join='inner')
        HTML(df.to_html('templates/sql-data.html'))
```



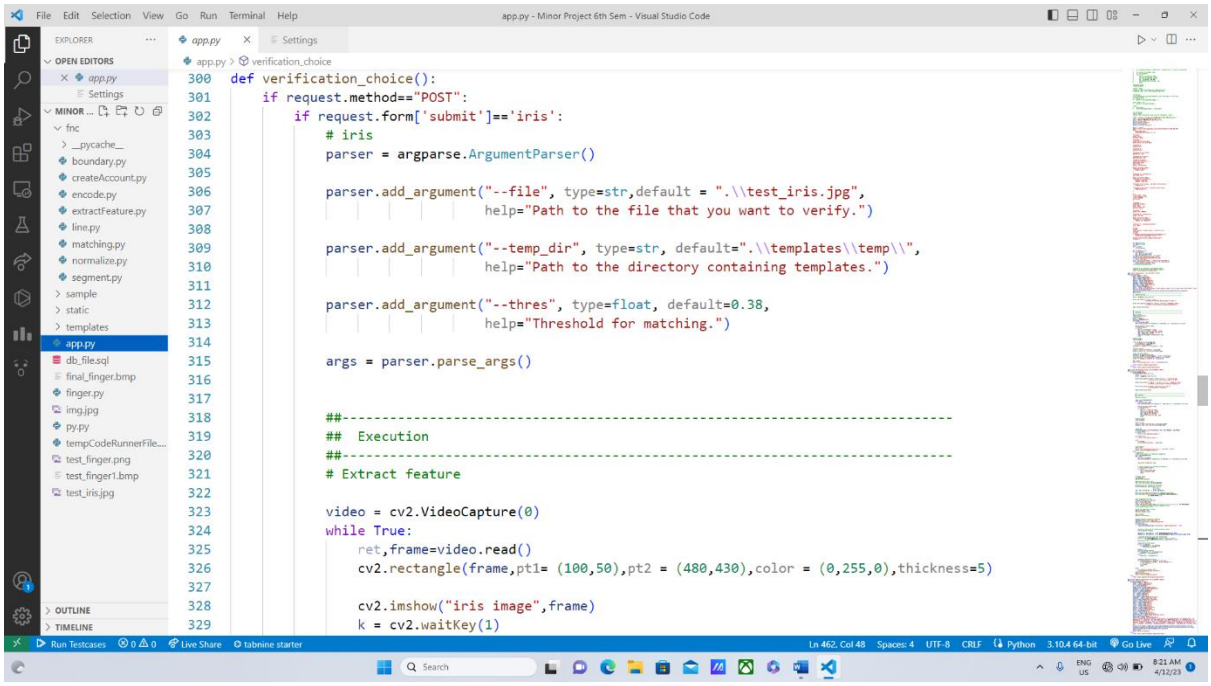
```
        with open("templates/sql-data.html") as file:
            file = file.read()
        file = file.replace("<table ", "<table class='rwd-table'")
        with open("templates/sql-data.html", "w") as file_to_write:
            file_to_write.write(html + file)
        # os.startfile("sql-data.html")


        # search for css insertion in the pandas library
        # HTML(df.to_html(classes='table table-stripped'))
        return render_template("/sql-data.html")

@app.route("/registration", methods=["POST","GET"])
def registration():
    if request.method == "POST":
        name = request.form["name"]
        fname = request.form["fName"]
        email = request.form["email"]
        address = request.form["address"]
        adharNo = request.form["adhaar"]
        mobileNo = request.form["mobile"]
        gMobileNo = request.form["fMobile"]
        dob = request.form["DOB"]
        gender = request.form["gender"]
        query = ("insert into Med_info(name, F_name, Address, Adhaar, mob, alt_mob, email, DOB, Gender) values
        (%s,%s,%s,%s,%s,%s,%s,%s,%s)")
        mycursor.execute(query,(name,fname,address,adharNo,mobileNo,gMobileNo,email,dob,gender))
        mydb.commit()
        #--------------------------------------------------------------------------
        #   Argument parsing
```



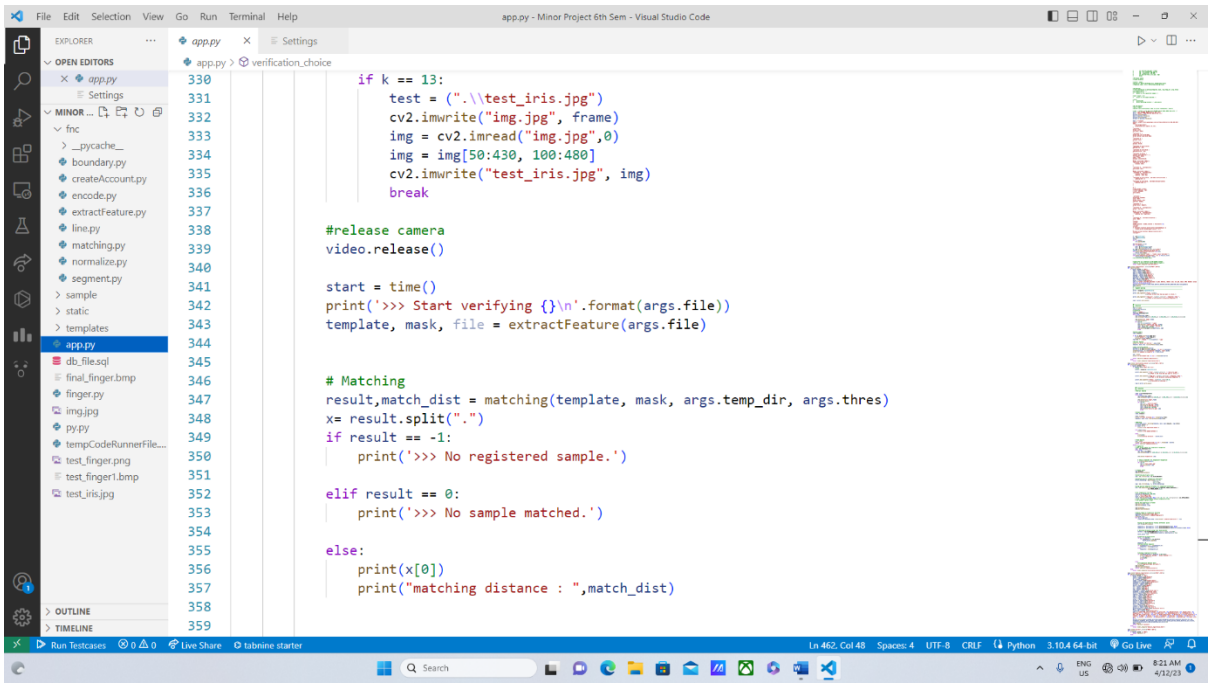
```
        #------------------------------------------------------------------------
        parser = argparse.ArgumentParser()

        parser.add_argument("--file", type=str,
                            help="Path to the file that you want to verify.")

        parser.add_argument("--temp_dir", type=str, default=".\\templates\\temp\\",
                            help="Path to the directory containing templates.")

        args = parser.parse_args()


        ##------------------------------------------------------------------------
        ##  Execution
        ##------------------------------------------------------------------------
        start = time()
        counter=last_enroll+1
        # counter=4;
        path = ".\sample\\"
        video=cv2.VideoCapture(0)
        while True:
            ret,frame=video.read()
            cv2.rectangle(frame,pt1= (100,50),pt2 = (480,430),color = (0,255,0),thickness=5)

            cv2.imshow("iris image",frame)
            k = cv2.waitKey(1)
            if k == 13:
                test = (str(counter) + ".jpg")
                cv2.imwrite(".\sample\\img.jpg", frame)
                img = cv2.imread(".\sample\\img.jpg",0)
```



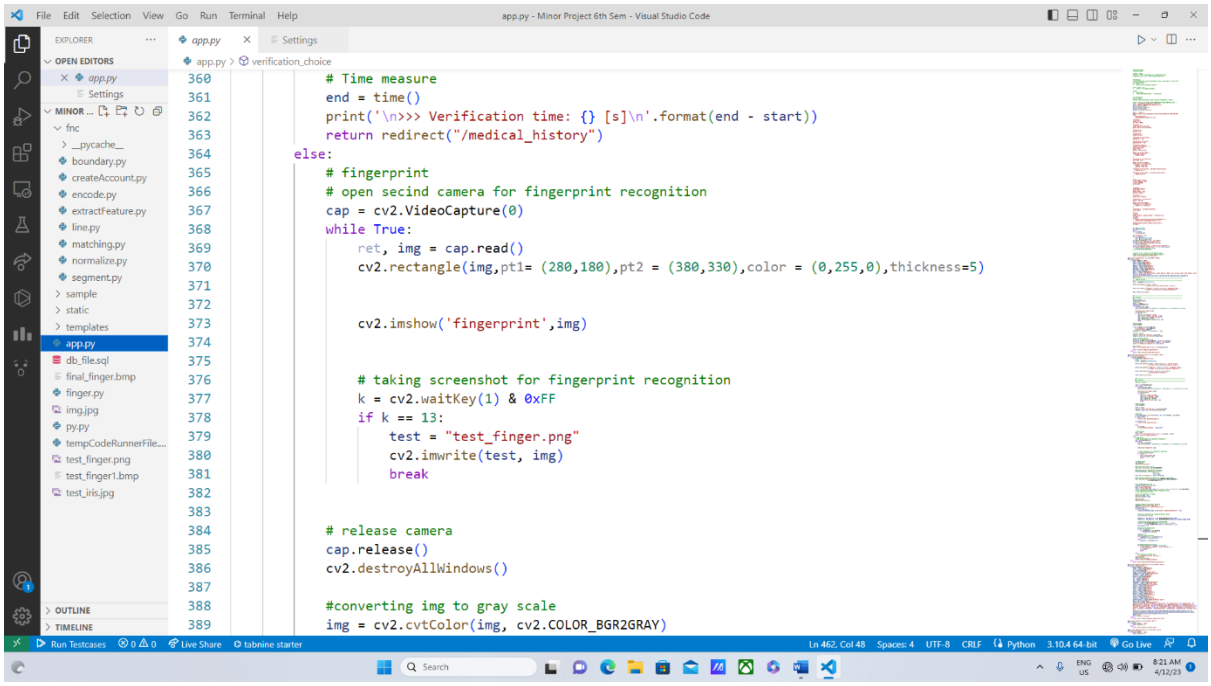
```
                img = img[50:430, 100:480]
                cv2.imwrite(os.path.join(path+test), img)
                break

        #release camera
        video.release()

        if not os.path.exists(args.temp_dir):
            print("makedirs", args.temp_dir)
            os.makedirs(args.temp_dir)
        args.file = ".\sample\\" + str(counter) + ".jpg"

        # Extract feature
        print('>>> Enroll for the file ', args.file)
        template, mask, file = extractFeature(args.file)

        # Save extracted feature
        basename = os.path.basename(file)
        out_file = os.path.join(args.temp_dir, "%s.mat" % (basename))
        savemat(out_file, mdict={'template':template, 'mask':mask})
        print('>>> Template is saved in %s' % (out_file))

        end = time()
        print('>>> Enrollment time: {} [s]\n'.format(end-start))

        return redirect("/medical_registration")
    else:
        return render_template("registration.html")

@app.route("/verification_choice",methods=["POST","GET"])
```



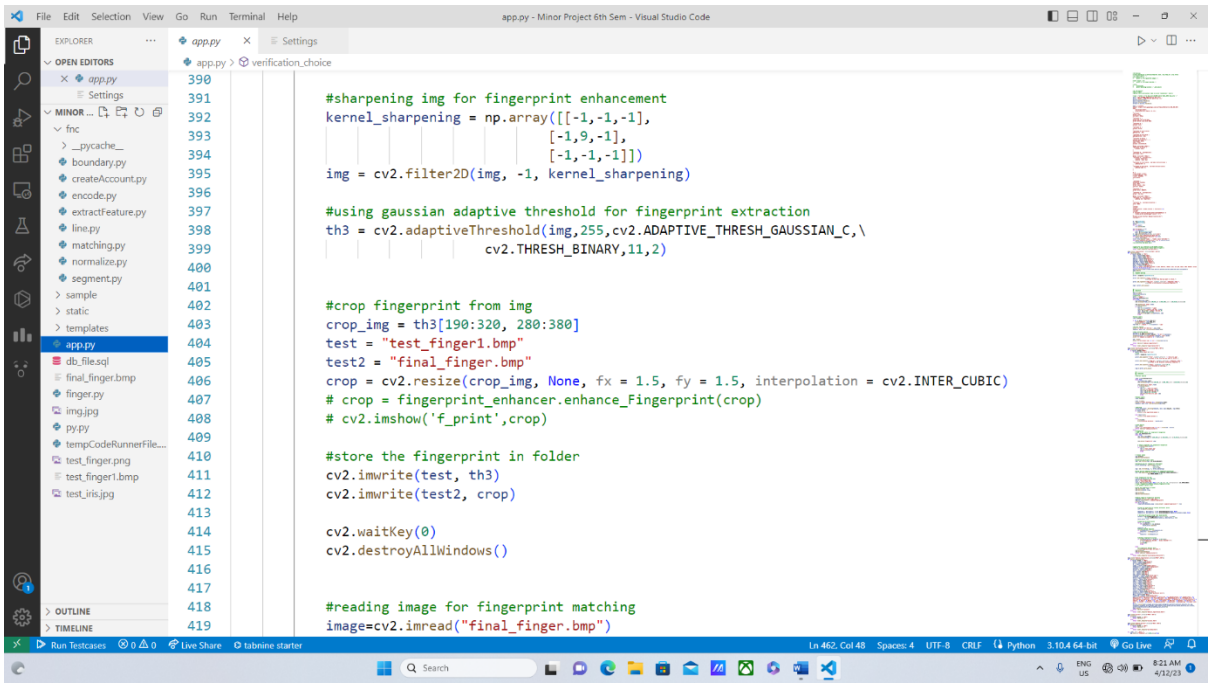
```
def verification_choice():
    if request.method=="POST":
        if request.form['submit']=='iris':
            # iris
            parser = argparse.ArgumentParser()

            parser.add_argument("--file", type=str,default = ".\\test_iris.jpg",
                                help="Path to the file that you want to verify.")

            parser.add_argument("--temp_dir", type=str, default=".\\templates\\temp\\",
                                help="Path to the directory containing templates.")

            parser.add_argument("--thres", type=float, default=0.38,
                                help="Threshold for matching.")

            args = parser.parse_args()


            ##------------------------------------------------------------------------
            ##  Execution
            ##------------------------------------------------------------------------
            # Extract feature

            video = cv2.VideoCapture(0)
            while True:
                ret,frame=video.read()
                cv2.rectangle(frame,pt1= (100,50),pt2 = (480,430),color = (0,255,0),thickness=5)

                cv2.imshow("iris image",frame)
                k = cv2.waitKey(1)
```



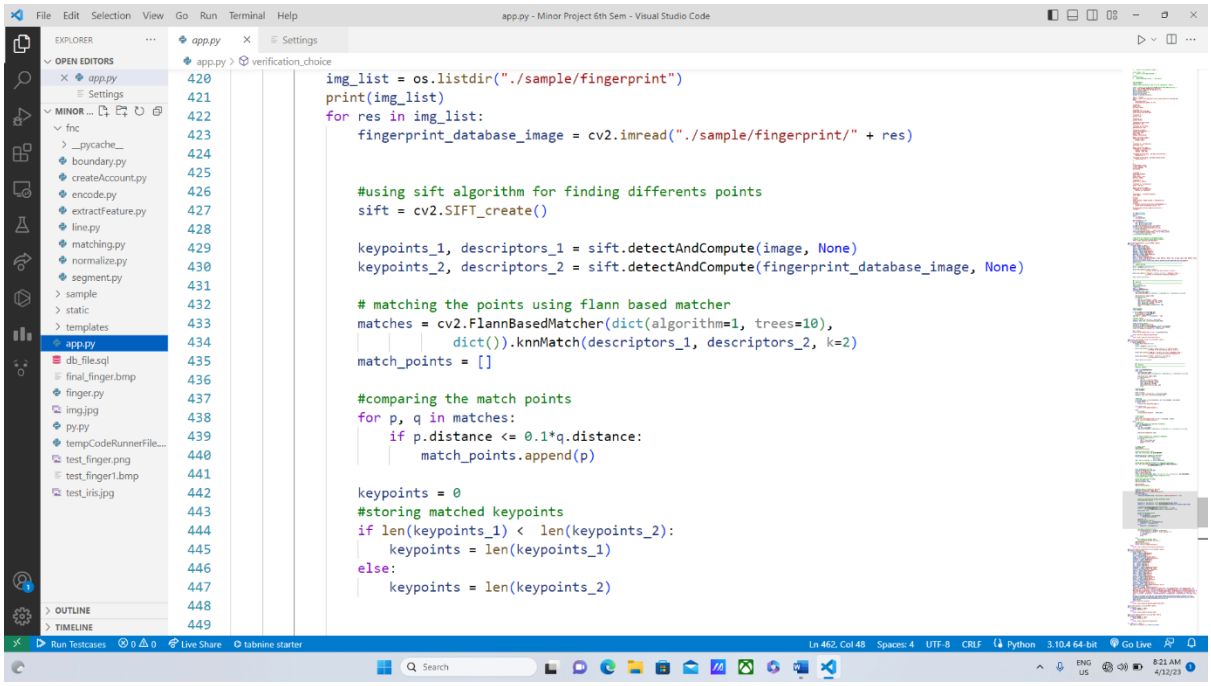
```
                if k == 13:
                    test = (".\\test_iris.jpg")
                    cv2.imwrite("img.jpg", frame)
                    img = cv2.imread("img.jpg",0)
                    img = img[50:430, 100:480]
                    cv2.imwrite("test_iris.jpg", img)
                    break

            #release camera
            video.release()

            start = time()
            print('>>> Start verifying {}\n'.format(args.file))
            template, mask, file = extractFeature(args.file)


            # Matching
            result,match_dist = matching(template, mask, args.temp_dir, args.thres)
            x= result.split(".")
            if result == -1:
                print('>>> No registered sample.')

            elif result == 0:
                print('>>> No sample matched.')

            else:
                print(x[0])
                print("matching distance : ",match_dist)
```



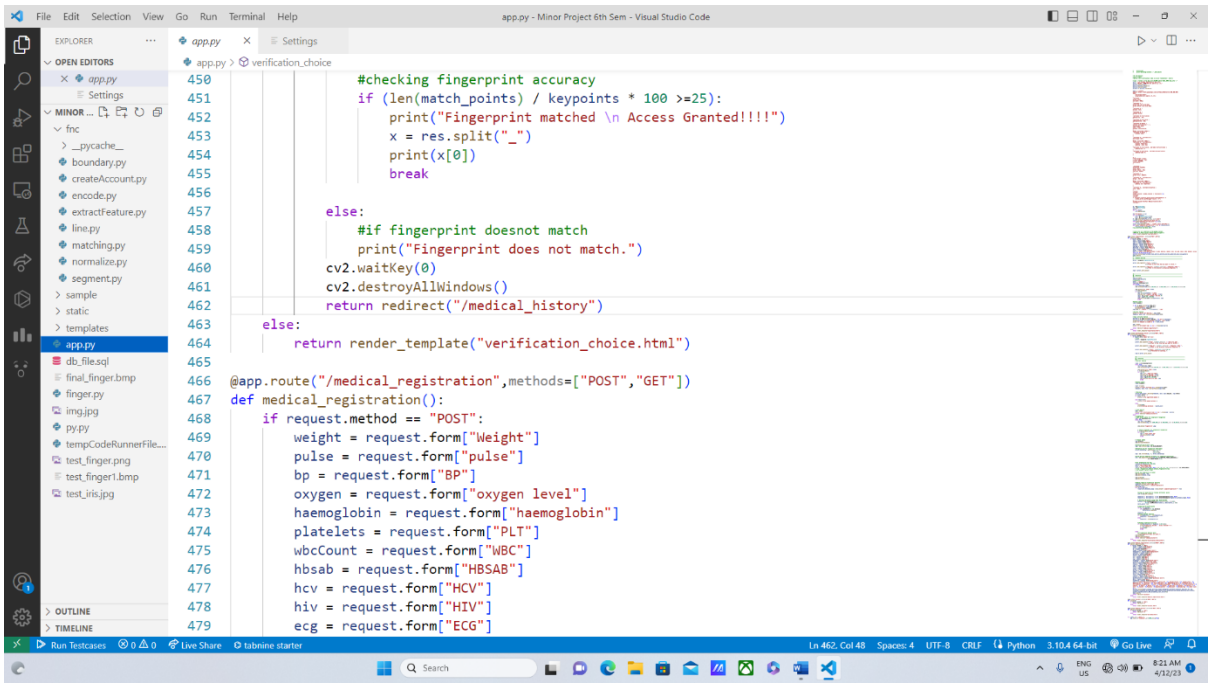
```
# Time measure
end = time()
print('\n>>> Verification time: {} [s]\n'.format(end - start))
return redirect("/medical_history")
else:
    # fingerprint
    # open secind camera for fingerprint recognition
    cap = cv2.VideoCapture(0)
    while True:
        ret, img = cap.read()
        cv2.rectangle(img,pt1= (280,180),pt2 = (380,330),color = (0,255,0),thickness=5)


        cv2.imshow('fingerprint',img)


        # taking screenshot for fingerprint recognition
        k = cv2.waitKey(1) & 0xFF
        if k == 13:
            test = "test_finger.png"
            cv2.imwrite(test, img)
            break


    # release camera
    cap.release()
    cv2.destroyAllWindows()

    #converting img to gray scale
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```



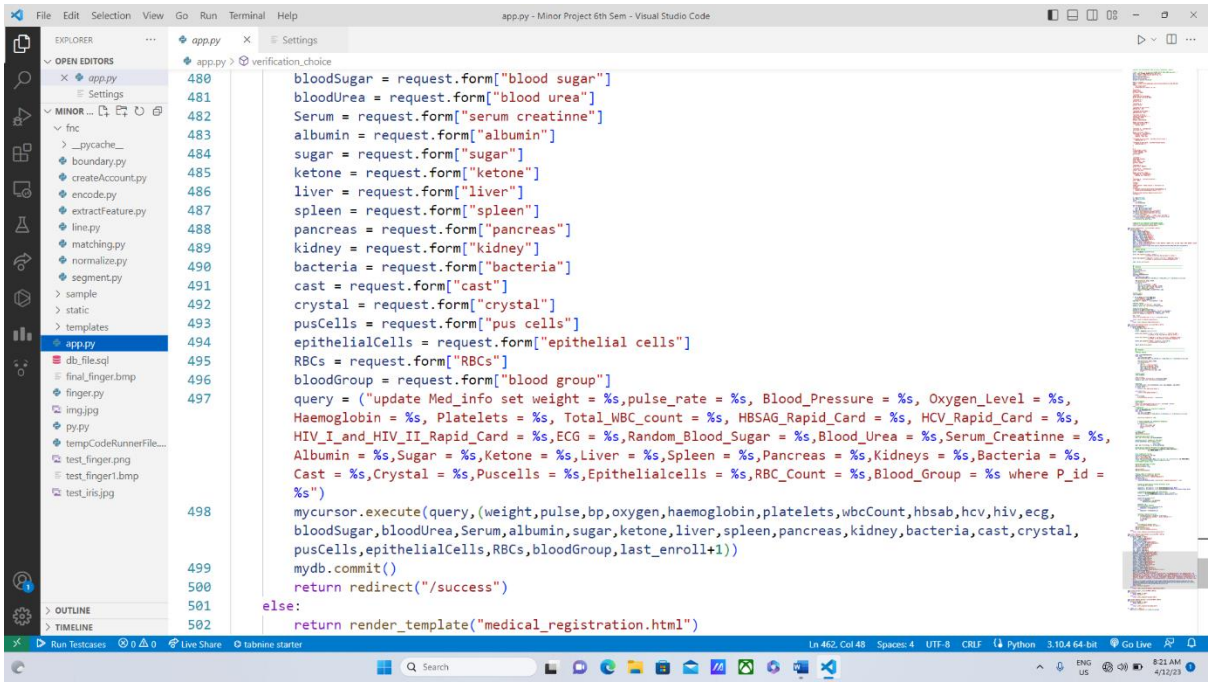
```
    #sharpening img for fingerprint enhancement
    kernel_sharpening = np.array([[-1,-1,-1],
                                  [-1,9,-1],
                                  [-1,-1,-1]])
    img = cv2.filter2D(img, -1, kernel_sharpening)

    #using gaussian adaptive threshold for fingerprint extraction
    th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
                cv2.THRESH_BINARY,11,2)


    #crop fingerprint from img
    crop_img = th3[190:320, 280:380]
    test = "test_finger1.bmp"
    test2 = "final_finger.bmp"
    crop = cv2.resize(crop_img, None, fx = 1.5, fy = 1.5, interpolation = cv2.INTER_CUBIC)
    # crop = fingerprint_enhancer.enhance_Fingerprint(crop)
    # cv2.imshow('f_print',crop)

    #store the fingerprint in folder
    cv2.imwrite(test, th3)
    cv2.imwrite(test2, crop)

    cv2.waitKey(0)
    cv2.destroyAllWindows()


    #reading image for fingerprint matching
    image=cv2.imread("final_finger.bmp")
```



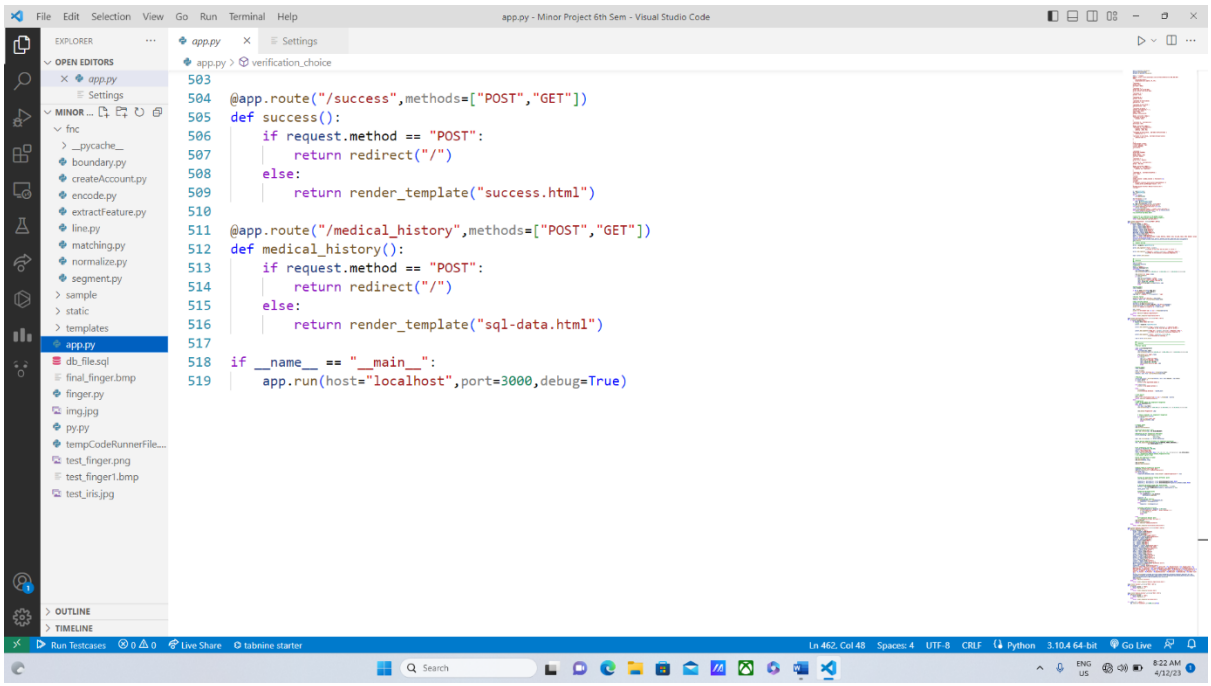
```
            img_list = os.listdir("./sample/fingerprint")
            print(img_list)
            for res in img_list:
                fingerprint_database_image = cv2.imread("./sample/fingerprint/" + res)


                #using sift algorithm for finding differents points
                sift = cv2.SIFT_create()

                keypoints_1, descriptors_1 = sift.detectAndCompute(image, None)
                keypoints_2, descriptors_2 = sift.detectAndCompute(fingerprint_database_image, None)

                # matching the points using flann based matcher
                matches = cv2.FlannBasedMatcher(dict(algorithm=1, trees=10),
                                    dict()).knnMatch(descriptors_1, descriptors_2, k=2)
                match_points = []

                #comparing the match points
                for p, q in matches:
                    if p.distance <= 0.1*q.distance:
                        match_points.append(p)

                keypoints = 0
                #storing matched keypoints
                if len(keypoints_1) <  len(keypoints_2):
                    keypoints = len(keypoints_1)
                else:
                    keypoints = len(keypoints_2)
```



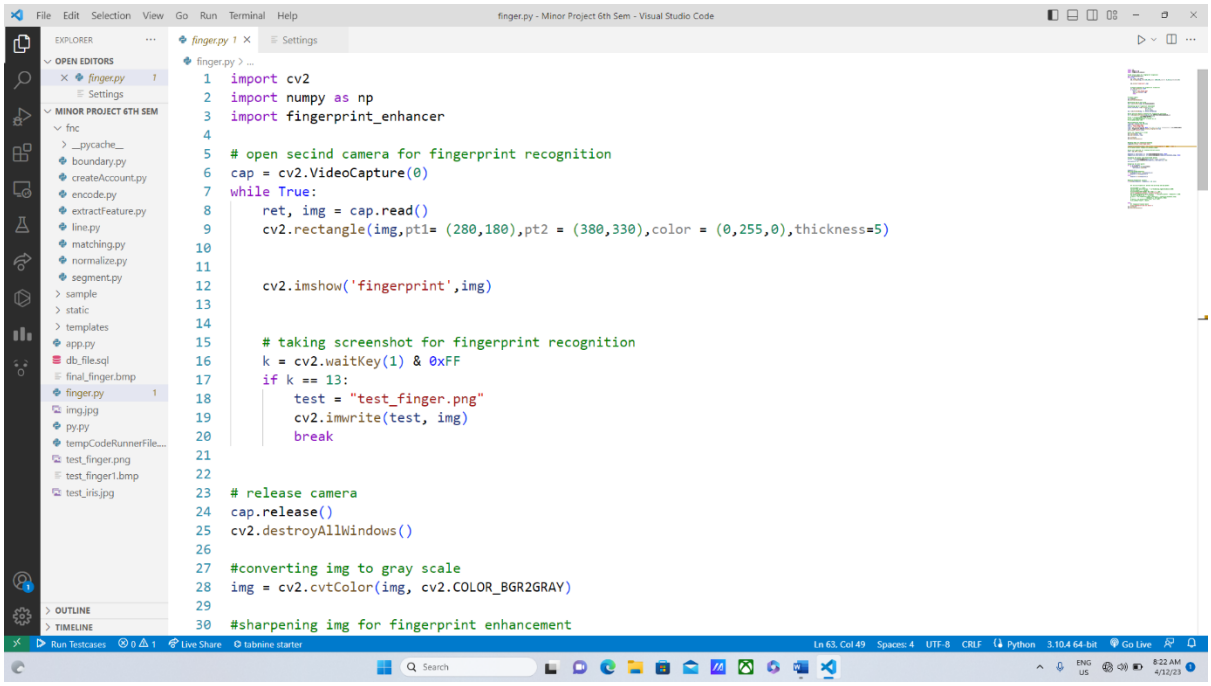
```
                #checking fingerprint accuracy
                if (len(match_points) / keypoints * 100 >=25):
                    print("Fingerprint matched \n Access Granted!!!!")
                    x = res.split("_")
                    print(x[0])
                    break

                else:
                    #if fingerprint doesnot match
                    print("Fingerprint does not match.")
                cv2.waitKey(0)
                cv2.destroyAllWindows()
                return redirect("/medical_history")
        else:
            return render_template("verification_choice.html")

@app.route("/medical_registration",methods=["POST","GET"])
def medical_registration():
    if request.method == "POST":
        weight = request.form["Weight"]
        pulse = request.form["pulse"]
        bp = request.form["BP"]
        oxygen = request.form["oxygen level"]
        haemoglobin = request.form["haemoglobin"]
        platelets = request.form["PLT"]
        wbcCount = request.form["WBC"]
        hbsab = request.form["HBSAB"]
        hcv = request.form["HCV"]
        hiv = request.form["HIV"]
        ecg = request.form["ECG"]
```



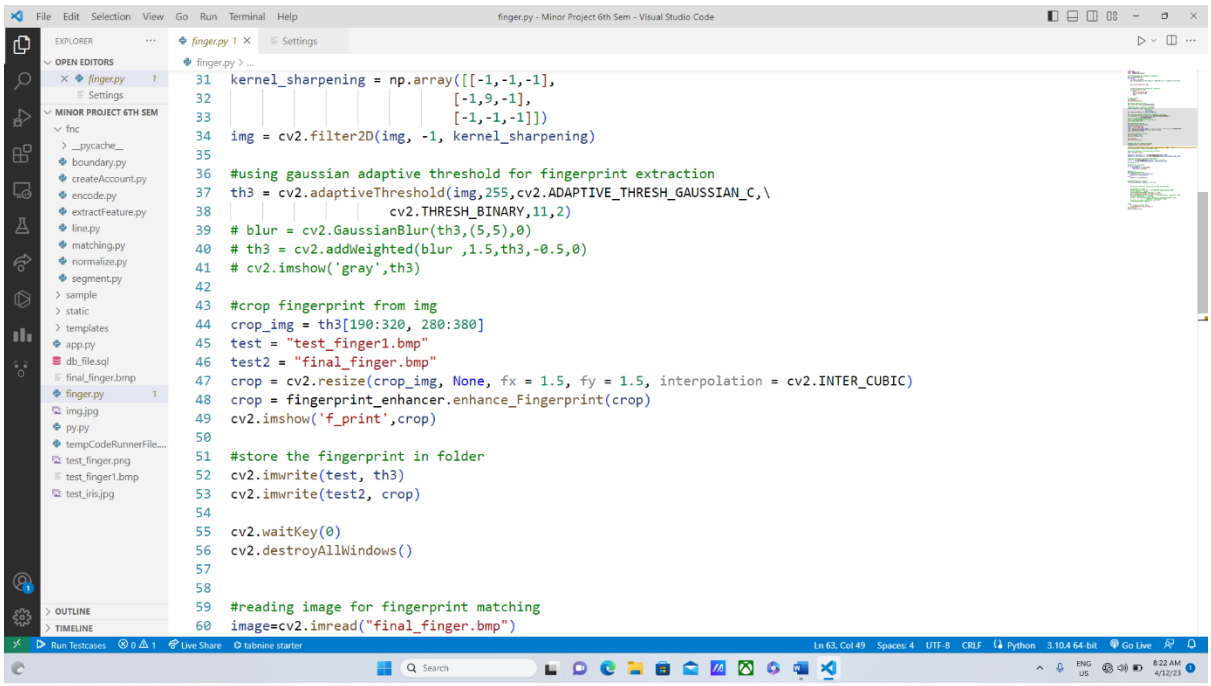
```
480         bloodSugar = request.form["blood sugar"]
481         bloodUrea = request.form["blood urea"]
482         Serum = request.form["serum creatinne"]
483         albumin = request.form["albumin"]
484         sugar = request.form["sugar"]
485         ketone = request.form["ketone"]
486         liver = request.form["liver"]
487         spleen = request.form["spleen"]
488         pancreas = request.form["pancreas"]
489         kidney = request.form["kidney"]
490         bacteria = request.form["bacteria"]
491         cast = request.form["cast"]
492         crystal = request.form["crystal"]
493         pusCells = request.form["pus cells"]
494         epithelialCells = request.form["epithelial cells"]
495         RBCs = request.form["RBCs"]
496         bloodGroup = request.form["blood group"]
497         query = ("update Med_info set weight = %s,pulse_rate = %s, Blood_Pressure = %s, Oxygen_Level = %s,
                  Haemoglobin = %s, Platelets = %s, Total_WBC_count = %s, HBSAG_Rapid_Card = %s, HCV_Rapid_Card = %s,
                  HIV_I_and_HIV_II_Rapid_Card = %s,ECG = %s,Random_Blood_Sugar = %s,Blood_Urea = %s,Serum_Creatinne = %s,
                  Albumin = %s,Sugar = %s,Ketone = %s,Liver = %s,Spleen = %s,Pancreas = %s,Kidneys = %s,Bacteria = %s,
                  Cast = %s,Crystal = %s,Puscells = %s,Epithelialcells = %s,RBC_Count = %s,Blood_Group = %s where P_id =
                  %s")
498         mycursor.execute(query,(weight,pulse,bp,oxygen,haemoglobin,platelets,wbcCount,hbsab,hcv,hiv,ecg,
                  bloodSugar,bloodUrea,Serum,albumin,sugar,ketone,liver,spleen,pancreas,kidney,bacteria,cast,crystal,
                  pusCells,epithelialCells,RBCs,bloodGroup,last_enroll+1))
499         mydb.commit()
500         return redirect("/success")
501     else:
502         return render_template("medical_registration.html")
```



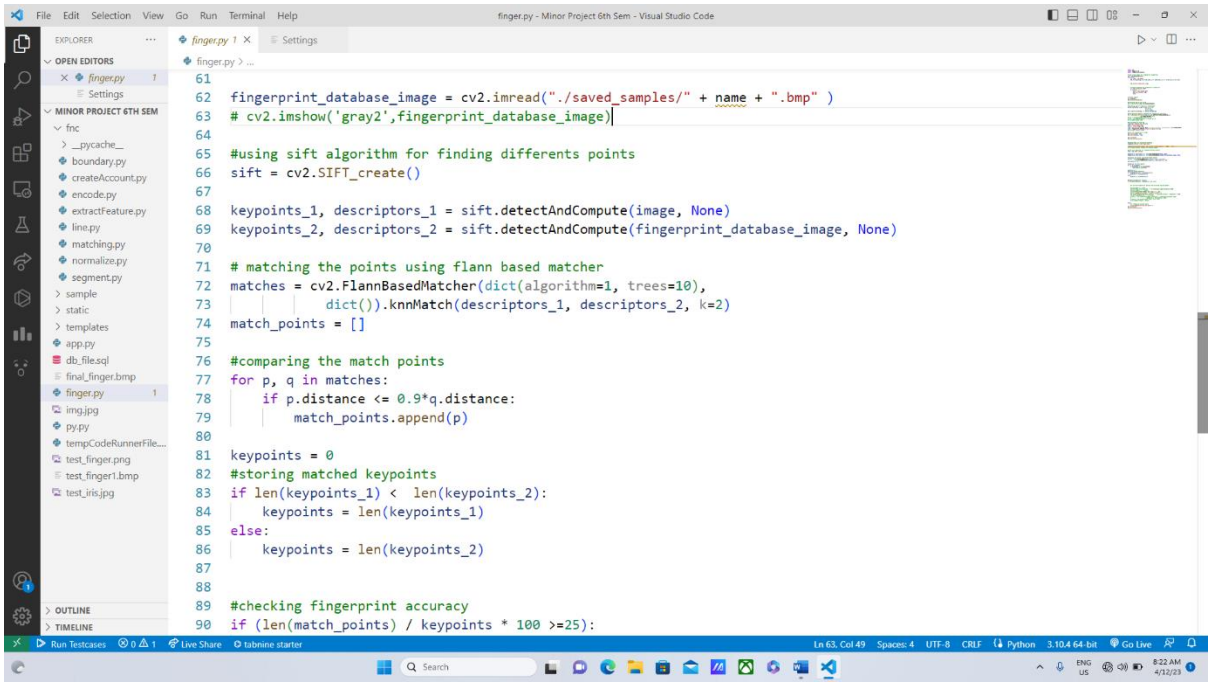
```
503
504 @app.route("/success",methods=["POST","GET"])
505 def success():
506     if request.method == "POST":
507         return redirect("/")
508     else:
509         return render_template("success.html")
510
511 @app.route("/medical_history",methods=["POST","GET"])
512 def medical_history():
513     if request.method == "POST":
514         return redirect("/")
515     else:
516         return render_template("sql-data.html")
517
518 if __name__ == "__main__":
519     app.run(host="localhost",port=3000,debug=True)
```

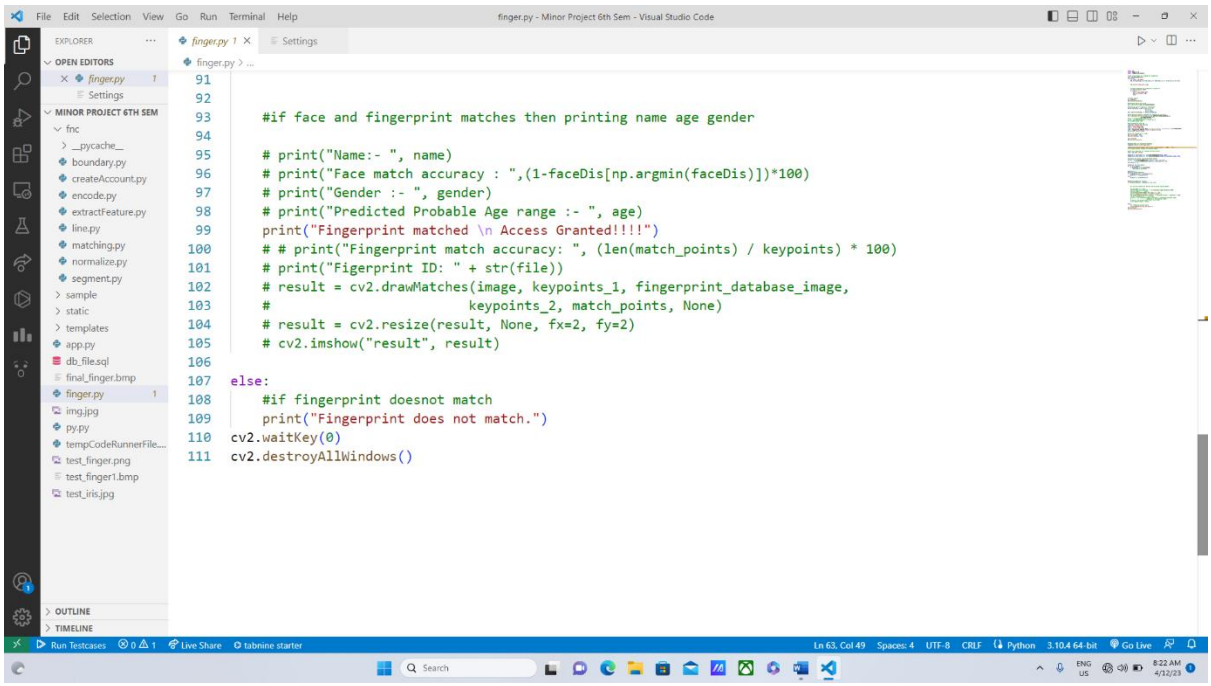


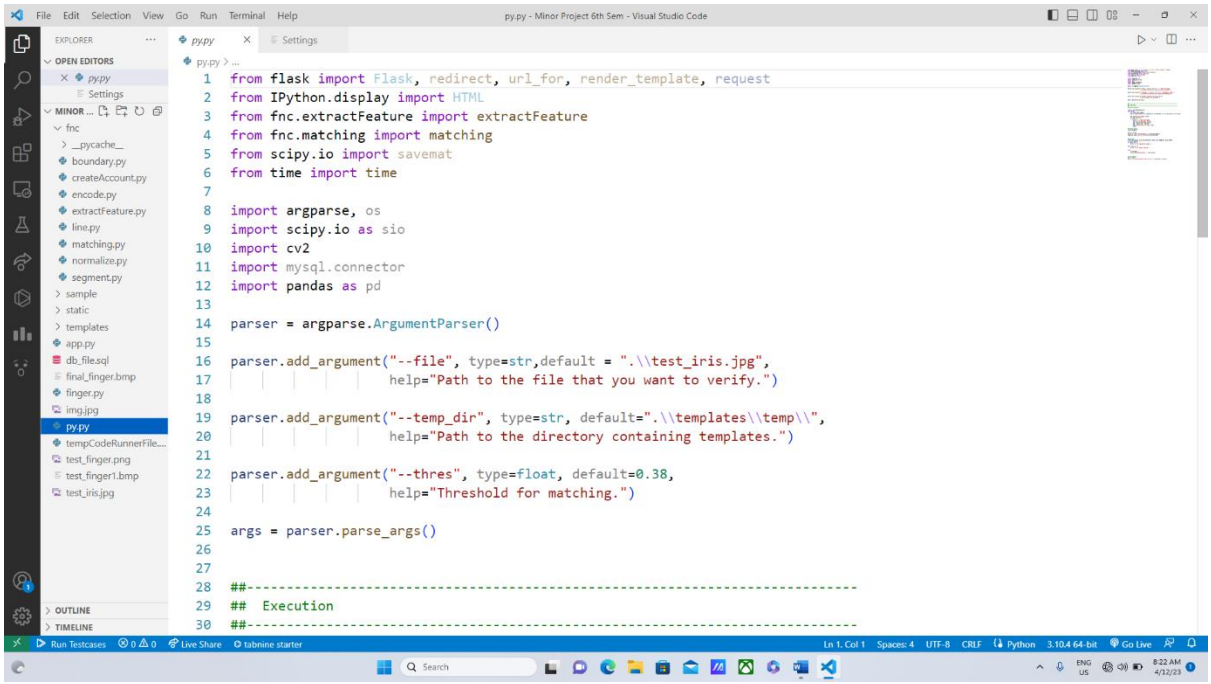
```python
import cv2
import numpy as np
import fingerprint_enhancer

# open secind camera for fingerprint recognition
cap = cv2.VideoCapture(0)
while True:
    ret, img = cap.read()
    cv2.rectangle(img,pt1= (280,180),pt2 = (380,330),color = (0,255,0),thickness=5)


    cv2.imshow('fingerprint',img)


    # taking screenshot for fingerprint recognition
    k = cv2.waitKey(1) & 0xFF
    if k == 13:
        test = "test_finger.png"
        cv2.imwrite(test, img)
        break


# release camera
cap.release()
cv2.destroyAllWindows()

#converting img to gray scale
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#sharpening img for fingerprint enhancement
```



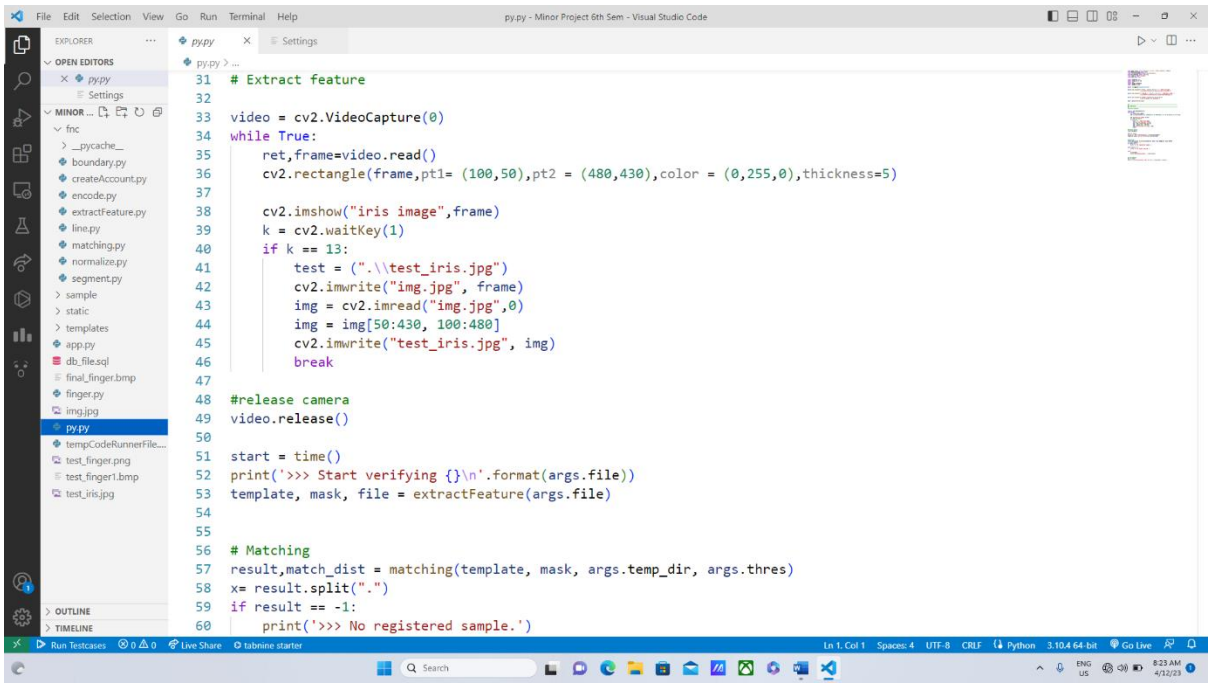
```python
kernel_sharpening = np.array([[-1,-1,-1],
                              [-1,9,-1],
                              [-1,-1,-1]])
img = cv2.filter2D(img, -1, kernel_sharpening)

#using gaussian adaptive threshold for fingerprint extraction
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
                    cv2.THRESH_BINARY,11,2)
# blur = cv2.GaussianBlur(th3,(5,5),0)
# th3 = cv2.addWeighted(blur ,1.5,th3,-0.5,0)
# cv2.imshow('gray',th3)

#crop fingerprint from img
crop_img = th3[190:320, 280:380]
test = "test_finger1.bmp"
test2 = "final_finger.bmp"
crop = cv2.resize(crop_img, None, fx = 1.5, fy = 1.5, interpolation = cv2.INTER_CUBIC)
crop = fingerprint_enhancer.enhance_Fingerprint(crop)
cv2.imshow('f_print',crop)

#store the fingerprint in folder
cv2.imwrite(test, th3)
cv2.imwrite(test2, crop)

cv2.waitKey(0)
cv2.destroyAllWindows()


#reading image for fingerprint matching
image=cv2.imread("final_finger.bmp")
```


finger.py - Minor Project 6th Sem - Visual Studio Code
finger.py
Settings
MINOR PROJECT 6TH SEM
fnc
__pycache__
boundary.py
createAccount.py
encode.py
extractFeature.py
line.py
matching.py
normalize.py
segment.py
sample
static
templates
app.py
db_file.sql
final_finger.bmp
finger.py
img.jpg
py.py
tempCodeRunnerFile....
test_finger.png
test_finger1.bmp
test_iris.jpg
OUTLINE
TIMELINE
61
62 fingerprint_database_image = cv2.imread("./saved_samples/" + name + ".bmp")
63 # cv2.imshow('gray2',fingerprint_database_image)
64
65 #using sift algorithm for finding differents points
66 sift = cv2.SIFT_create()
67
68 keypoints_1, descriptors_1 = sift.detectAndCompute(image, None)
69 keypoints_2, descriptors_2 = sift.detectAndCompute(fingerprint_database_image, None)
70
71 # matching the points using flann based matcher
72 matches = cv2.FlannBasedMatcher(dict(algorithm=1, trees=10),
73 dict()).knnMatch(descriptors_1, descriptors_2, k=2)
74 match_points = []
75
76 #comparing the match points
77 for p, q in matches:
78 if p.distance <= 0.9*q.distance:
79 match_points.append(p)
80
81 keypoints = 0
82 #storing matched keypoints
83 if len(keypoints_1) < len(keypoints_2):
84 keypoints = len(keypoints_1)
85 else:
86 keypoints = len(keypoints_2)
87
88
89 #checking fingerprint accuracy
90 if (len(match_points) / keypoints * 100 >=25):
Run Testcases
Live Share
tabnine starter
Ln 63, Col 49
Spaces: 4
UTF-8
CRLF
Python
3.10.4 64-bit
Go Live
Search

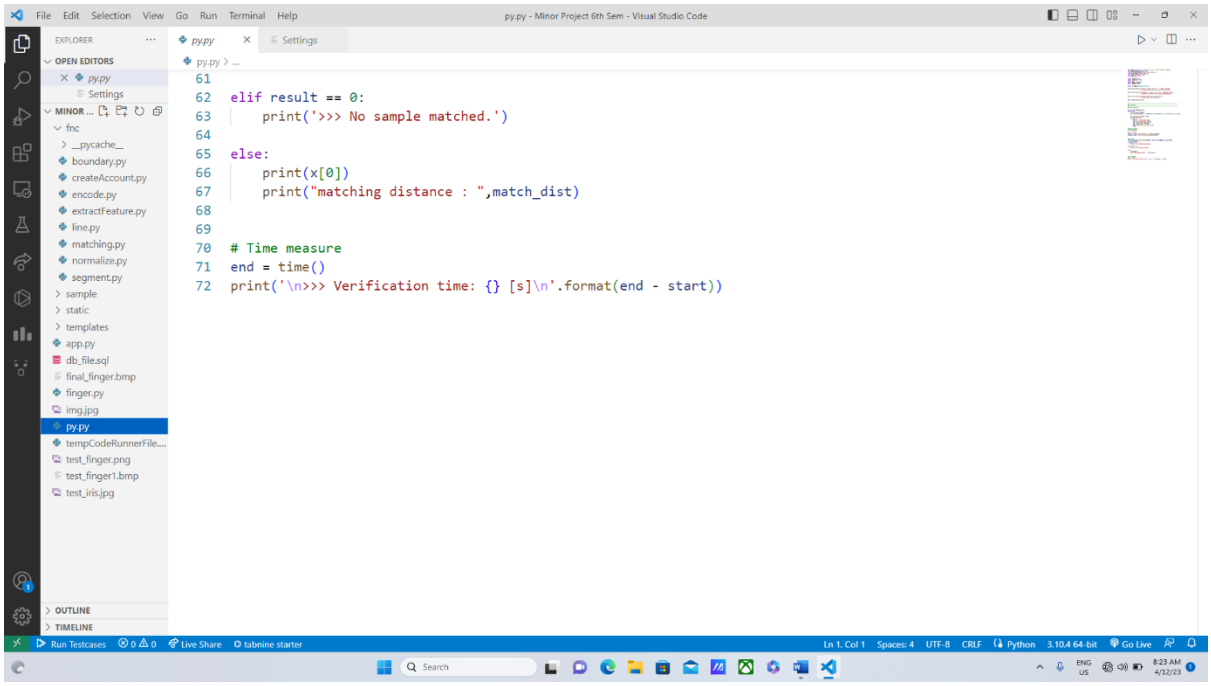

finger.py - Minor Project 6th Sem - Visual Studio Code
finger.py
Settings
MINOR PROJECT 6TH SEM
fnc
__pycache__
boundary.py
createAccount.py
encode.py
extractFeature.py
line.py
matching.py
normalize.py
segment.py
sample
static
templates
app.py
db_file.sql
final_finger.bmp
finger.py
img.jpg
py.py
tempCodeRunnerFile....
test_finger.png
test_finger1.bmp
test_iris.jpg
OUTLINE
TIMELINE
91
92
93 #if face and fingerprint matches then printing name age gender
94
95 # print("Name:- ", name)
96 # print("Face match accuracy : ",(1-faceDis[np.argmin(faceDis)])*100)
97 # print("Gender :- ", gender)
98 # print("Predicted Probable Age range :- ", age)
99 print("Fingerprint matched \n Access Granted!!!!")
100 # # print("Fingerprint match accuracy: ", (len(match_points) / keypoints) * 100)
101 # print("Figerprint ID: " + str(file))
102 # result = cv2.drawMatches(image, keypoints_1, fingerprint_database_image,
103 # keypoints_2, match_points, None)
104 # result = cv2.resize(result, None, fx=2, fy=2)
105 # cv2.imshow("result", result)
106
107 else:
108 #if fingerprint doesnot match
109 print("Fingerprint does not match.")
110 cv2.waitKey(0)
111 cv2.destroyAllWindows()
Run Testcases
Live Share
tabnine starter
Ln 63, Col 49
Spaces: 4
UTF-8
CRLF
Python
3.10.4 64-bit
Go Live
Search

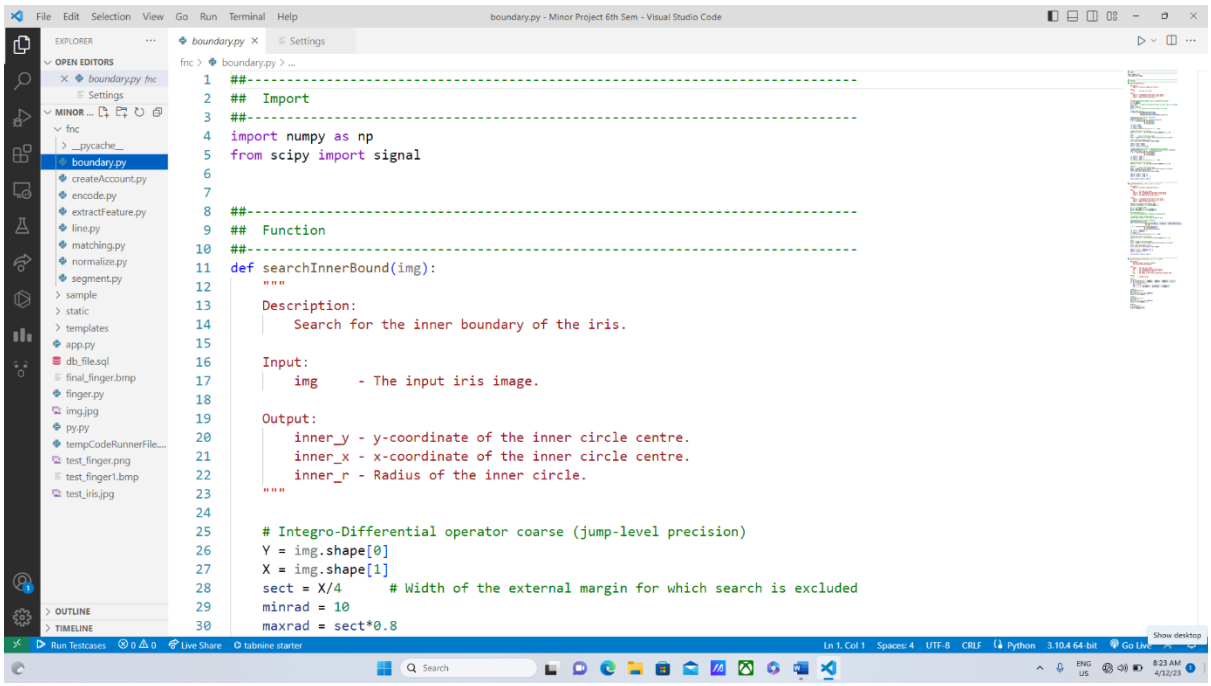


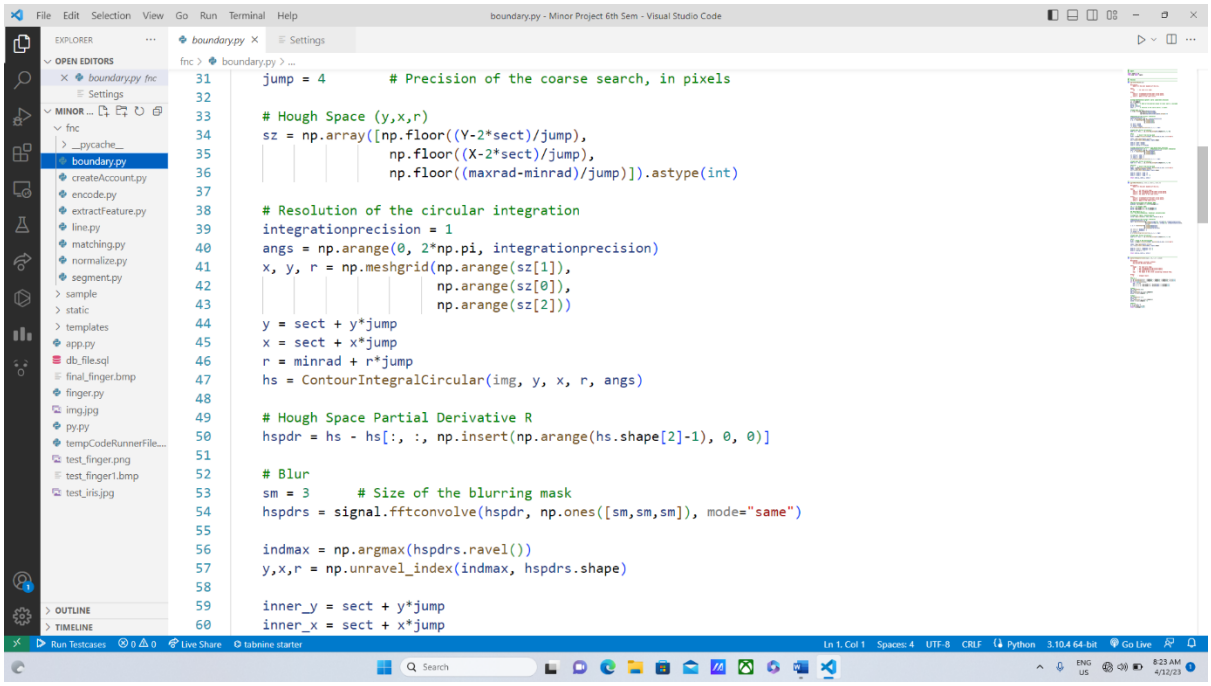
```
from flask import Flask, redirect, url_for, render_template, request
from IPython.display import HTML
from fnc.extractFeature import extractFeature
from fnc.matching import matching
from scipy.io import savemat
from time import time

import argparse, os
import scipy.io as sio
import cv2
import mysql.connector
import pandas as pd

parser = argparse.ArgumentParser()

parser.add_argument("--file", type=str,default = ".\\test_iris.jpg",
                    help="Path to the file that you want to verify.")

parser.add_argument("--temp_dir", type=str, default=".\\templates\\temp\\",
                    help="Path to the directory containing templates.")

parser.add_argument("--thres", type=float, default=0.38,
                    help="Threshold for matching.")

args = parser.parse_args()


##-------------------------------------------------------------------------
##  Execution
##-------------------------------------------------------------------------
```



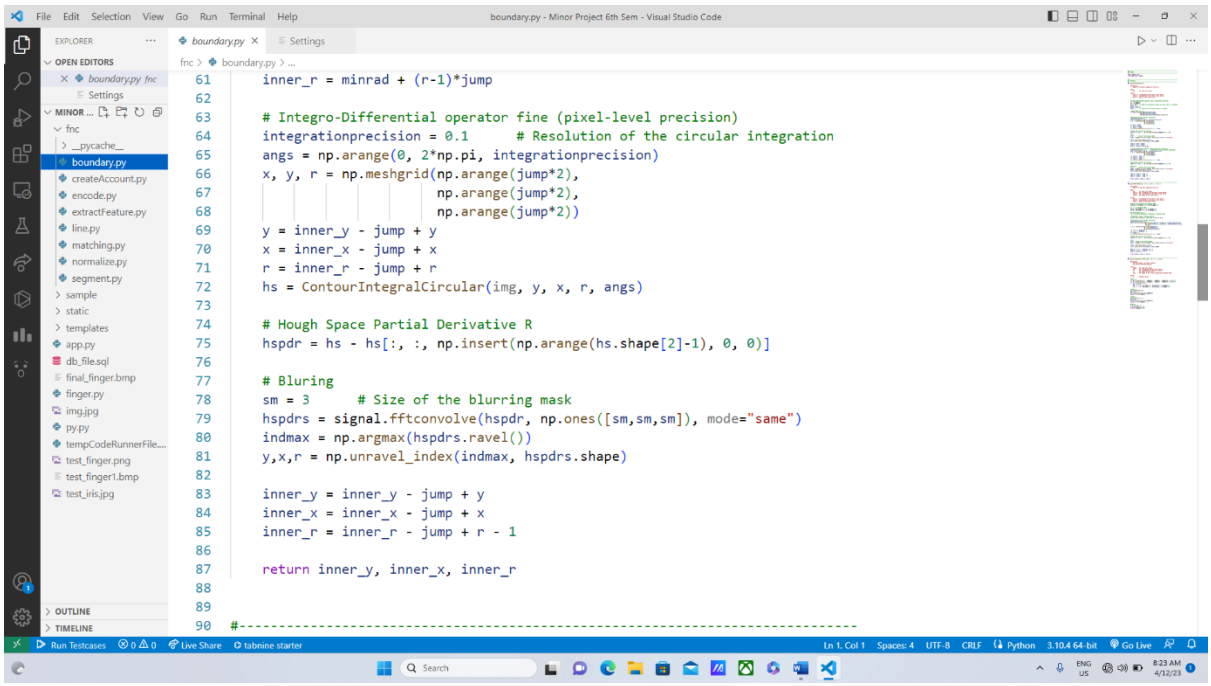
```
# Extract feature

video = cv2.VideoCapture(0)
while True:
    ret,frame=video.read()
    cv2.rectangle(frame,pt1= (100,50),pt2 = (480,430),color = (0,255,0),thickness=5)

    cv2.imshow("iris image",frame)
    k = cv2.waitKey(1)
    if k == 13:
        test = (".\\test_iris.jpg")
        cv2.imwrite("img.jpg", frame)
        img = cv2.imread("img.jpg",0)
        img = img[50:430, 100:480]
        cv2.imwrite("test_iris.jpg", img)
        break

#release camera
video.release()

start = time()
print('>>> Start verifying {}\n'.format(args.file))
template, mask, file = extractFeature(args.file)


# Matching
result,match_dist = matching(template, mask, args.temp_dir, args.thres)
x= result.split(".")
if result == -1:
    print('>>> No registered sample.')
```


File Edit Selection View Go Run Terminal Help
py.py - Minor Project 6th Sem - Visual Studio Code
EXPLORER
OPEN EDITORS
py.py
Settings
MINOR ...
fnc
__pycache__
boundary.py
createAccount.py
encode.py
extractFeature.py
line.py
matching.py
normalize.py
segment.py
sample
static
templates
app.py
db_file.sql
final_finger.bmp
finger.py
img.jpg
py.py
tempCodeRunnerFile....
test_finger.png
test_finger1.bmp
test_iris.jpg
OUTLINE
TIMELINE
61
62 elif result == 0:
63 print('>>> No sample matched.')
64
65 else:
66 print(x[0])
67 print("matching distance : ",match_dist)
68
69
70 # Time measure
71 end = time()
72 print('\n>>> Verification time: {} [s]\n'.format(end - start))
Run Testcases
Live Share
tabnine starter
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.10.4 64-bit
Go Live
Search

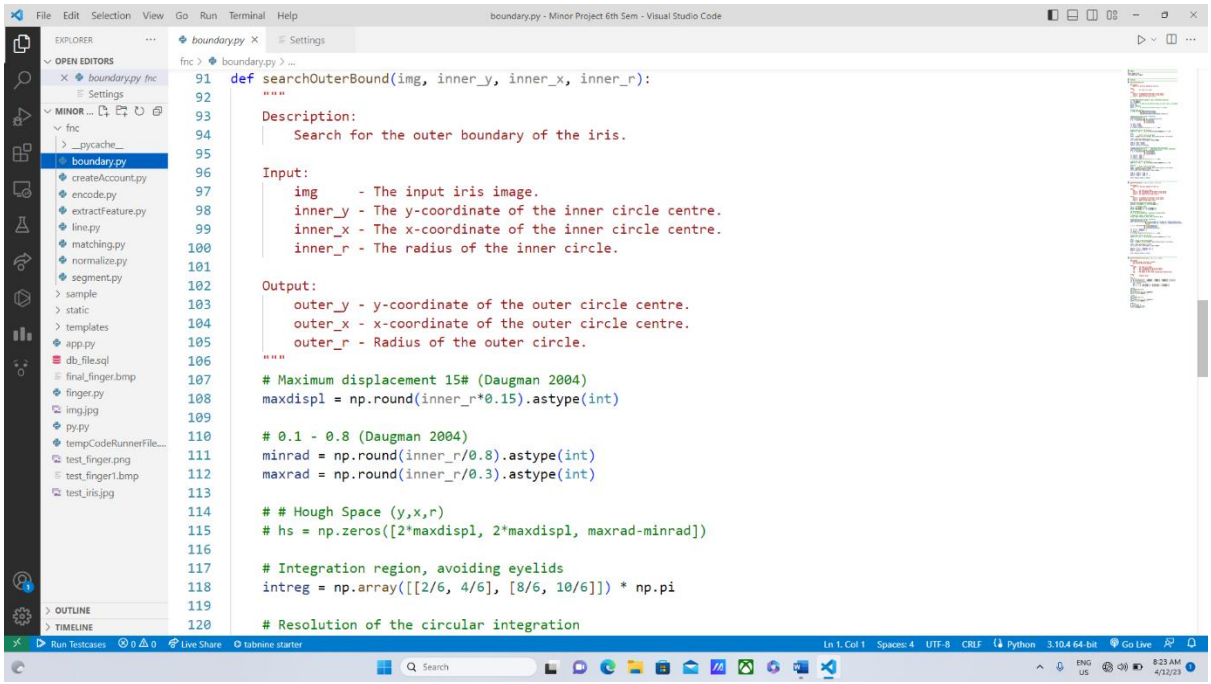

File Edit Selection View Go Run Terminal Help
boundary.py - Minor Project 6th Sem - Visual Studio Code
EXPLORER
OPEN EDITORS
boundary.py fnc
Settings
MINOR ...
fnc
__pycache__
boundary.py
createAccount.py
encode.py
extractFeature.py
line.py
matching.py
normalize.py
segment.py
sample
static
templates
app.py
db_file.sql
final_finger.bmp
finger.py
img.jpg
py.py
tempCodeRunnerFile....
test_finger.png
test_finger1.bmp
test_iris.jpg
OUTLINE
TIMELINE
1 ##---
2 ## Import
3 ##---
4 import numpy as np
5 from scipy import signal
6
7
8 ##---
9 ## Function
10 ##---
11 def searchInnerBound(img):
12 """
13 Description:
14 Search for the inner boundary of the iris.
15
16 Input:
17 img - The input iris image.
18
19 Output:
20 inner_y - y-coordinate of the inner circle centre.
21 inner_x - x-coordinate of the inner circle centre.
22 inner_r - Radius of the inner circle.
23 """
24
25 # Integro-Differential operator coarse (jump-level precision)
26 Y = img.shape[0]
27 X = img.shape[1]
28 sect = X/4 # Width of the external margin for which search is excluded
29 minrad = 10
30 maxrad = sect*0.8
Run Testcases
Live Share
tabnine starter
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.10.4 64-bit
Go Live
Search



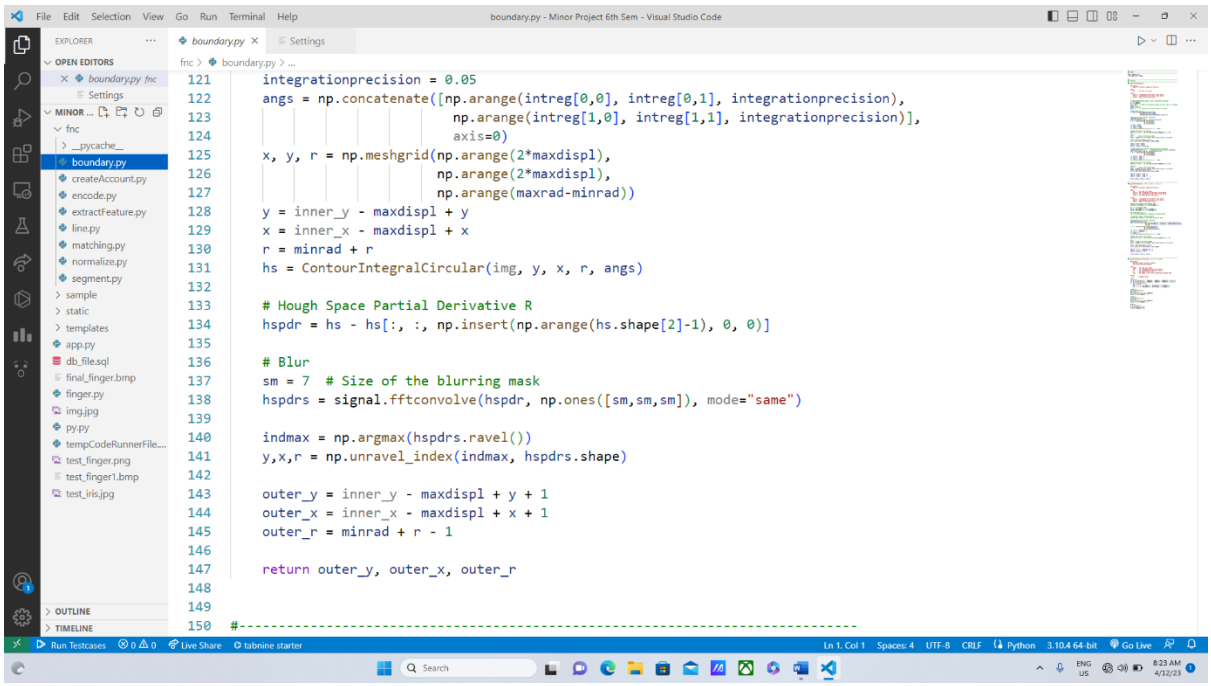
```
    jump = 4        # Precision of the coarse search, in pixels

    # Hough Space (y,x,r)
    sz = np.array([np.floor((Y-2*sect)/jump),
                   np.floor((X-2*sect)/jump),
                   np.floor((maxrad-minrad)/jump)]).astype(int)

    # Resolution of the circular integration
    integrationprecision = 1
    angs = np.arange(0, 2*np.pi, integrationprecision)
    x, y, r = np.meshgrid(np.arange(sz[1]),
                          np.arange(sz[0]),
                          np.arange(sz[2]))
    y = sect + y*jump
    x = sect + x*jump
    r = minrad + r*jump
    hs = ContourIntegralCircular(img, y, x, r, angs)

    # Hough Space Partial Derivative R
    hspdr = hs - hs[:, :, np.insert(np.arange(hs.shape[2]-1), 0, 0)]

    # Blur
    sm = 3      # Size of the blurring mask
    hspdrs = signal.fftconvolve(hspdr, np.ones([sm,sm,sm]), mode="same")

    indmax = np.argmax(hspdrs.ravel())
    y,x,r = np.unravel_index(indmax, hspdrs.shape)

    inner_y = sect + y*jump
    inner_x = sect + x*jump
```



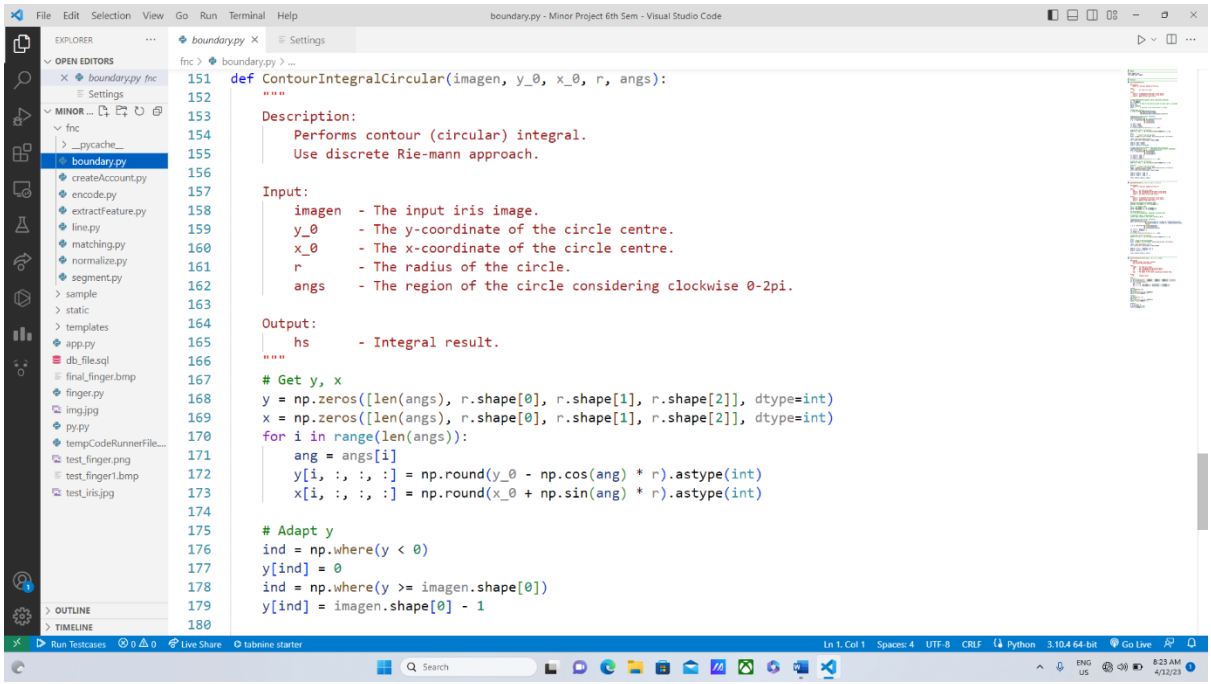
```
    inner_r = minrad + (r-1)*jump

    # Integro-Differential operator fine (pixel-level precision)
    integrationprecision = 0.1      # Resolution of the circular integration
    angs = np.arange(0, 2*np.pi, integrationprecision)
    x, y, r = np.meshgrid(np.arange(jump*2),
                          np.arange(jump*2),
                          np.arange(jump*2))
    y = inner_y - jump + y
    x = inner_x - jump + x
    r = inner_r - jump + r
    hs = ContourIntegralCircular(img, y, x, r, angs)

    # Hough Space Partial Derivative R
    hspdr = hs - hs[:, :, np.insert(np.arange(hs.shape[2]-1), 0, 0)]

    # Bluring
    sm = 3      # Size of the blurring mask
    hspdrs = signal.fftconvolve(hspdr, np.ones([sm,sm,sm]), mode="same")
    indmax = np.argmax(hspdrs.ravel())
    y,x,r = np.unravel_index(indmax, hspdrs.shape)

    inner_y = inner_y - jump + y
    inner_x = inner_x - jump + x
    inner_r = inner_r - jump + r - 1

    return inner_y, inner_x, inner_r


#---------------------------------------------------------------------------
```



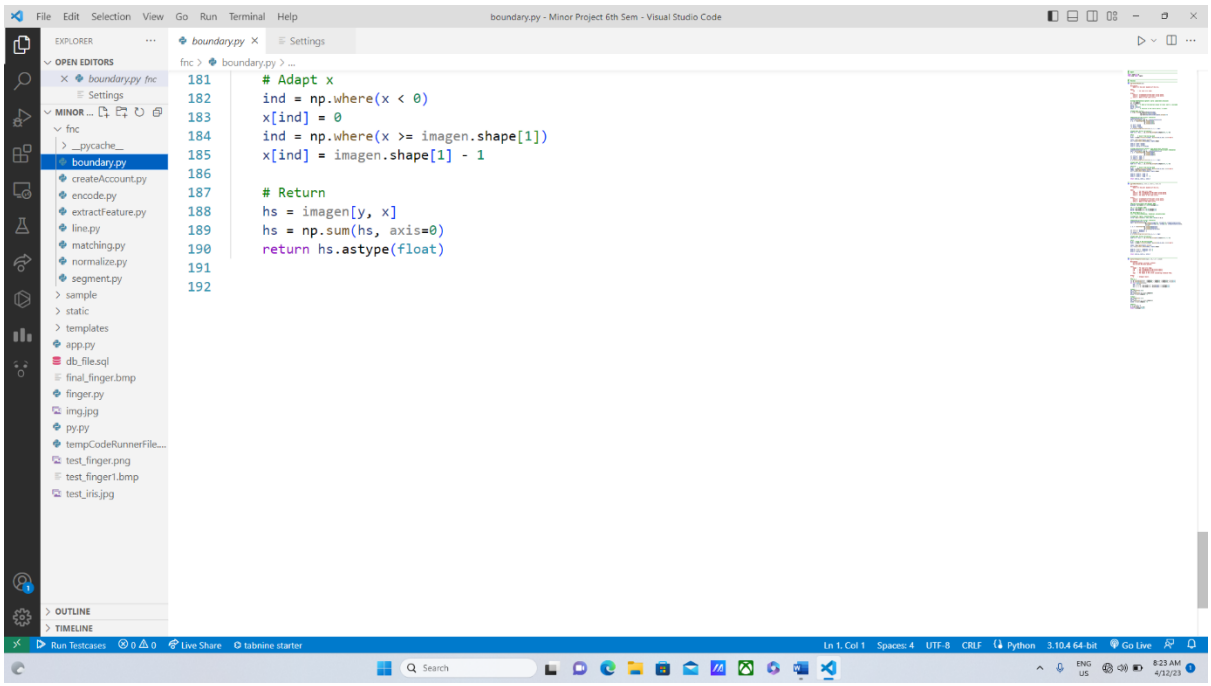
```python
def searchOuterBound(img, inner_y, inner_x, inner_r):
    """
    Description:
        Search for the outer boundary of the iris.

    Input:
        img     - The input iris image.
        inner_y - The y-coordinate of the inner circle centre.
        inner_x - The x-coordinate of the inner circle centre.
        inner_r - The radius of the inner circle.

    Output:
        outer_y - y-coordinate of the outer circle centre.
        outer_x - x-coordinate of the outer circle centre.
        outer_r - Radius of the outer circle.
    """
    # Maximum displacement 15# (Daugman 2004)
    maxdispl = np.round(inner_r*0.15).astype(int)

    # 0.1 - 0.8 (Daugman 2004)
    minrad = np.round(inner_r/0.8).astype(int)
    maxrad = np.round(inner_r/0.3).astype(int)

    # # Hough Space (y,x,r)
    # hs = np.zeros([2*maxdispl, 2*maxdispl, maxrad-minrad])

    # Integration region, avoiding eyelids
    intreg = np.array([[2/6, 4/6], [8/6, 10/6]]) * np.pi

    # Resolution of the circular integration
```



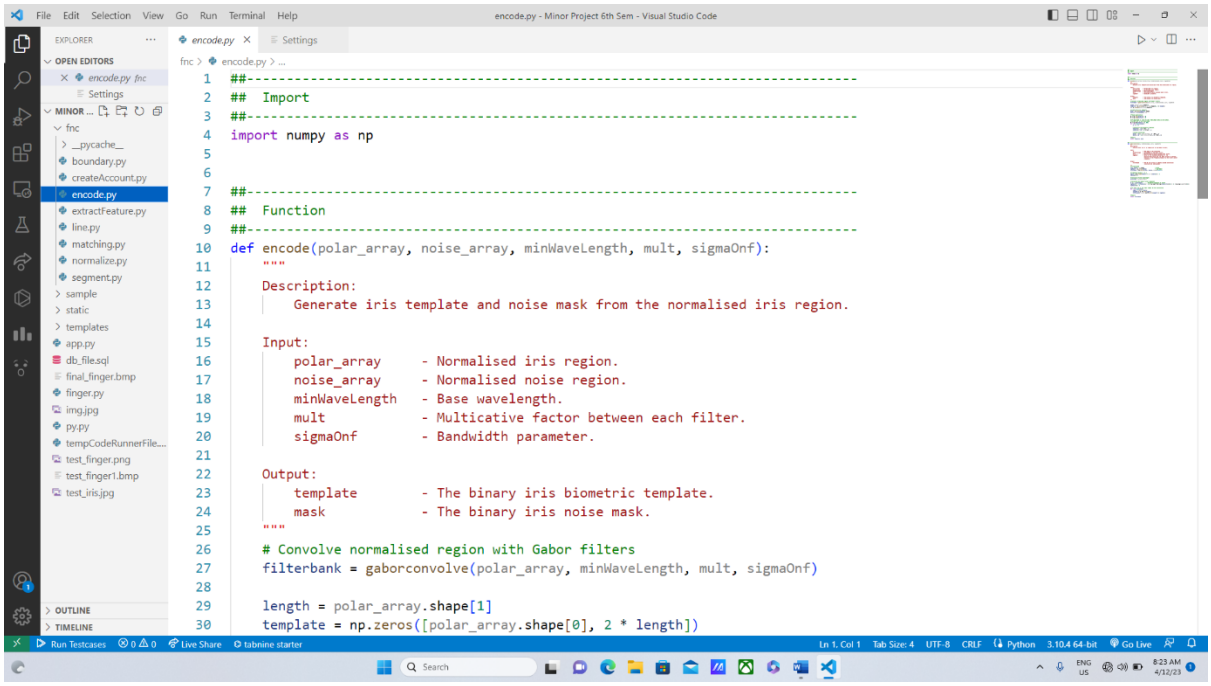
```python
    integrationprecision = 0.05
    angs = np.concatenate([np.arange(intreg[0,0], intreg[0,1], integrationprecision),
                           np.arange(intreg[1,0], intreg[1,1], integrationprecision)],
                           axis=0)
    x, y, r = np.meshgrid(np.arange(2*maxdispl),
                          np.arange(2*maxdispl),
                          np.arange(maxrad-minrad))
    y = inner_y - maxdispl + y
    x = inner_x - maxdispl + x
    r = minrad + r
    hs = ContourIntegralCircular(img, y, x, r, angs)

    # Hough Space Partial Derivative R
    hspdr = hs - hs[:, :, np.insert(np.arange(hs.shape[2]-1), 0, 0)]

    # Blur
    sm = 7  # Size of the blurring mask
    hspdrs = signal.fftconvolve(hspdr, np.ones([sm,sm,sm]), mode="same")

    indmax = np.argmax(hspdrs.ravel())
    y,x,r = np.unravel_index(indmax, hspdrs.shape)

    outer_y = inner_y - maxdispl + y + 1
    outer_x = inner_x - maxdispl + x + 1
    outer_r = minrad + r - 1

    return outer_y, outer_x, outer_r


#--------------------------------------------------------------------------
```



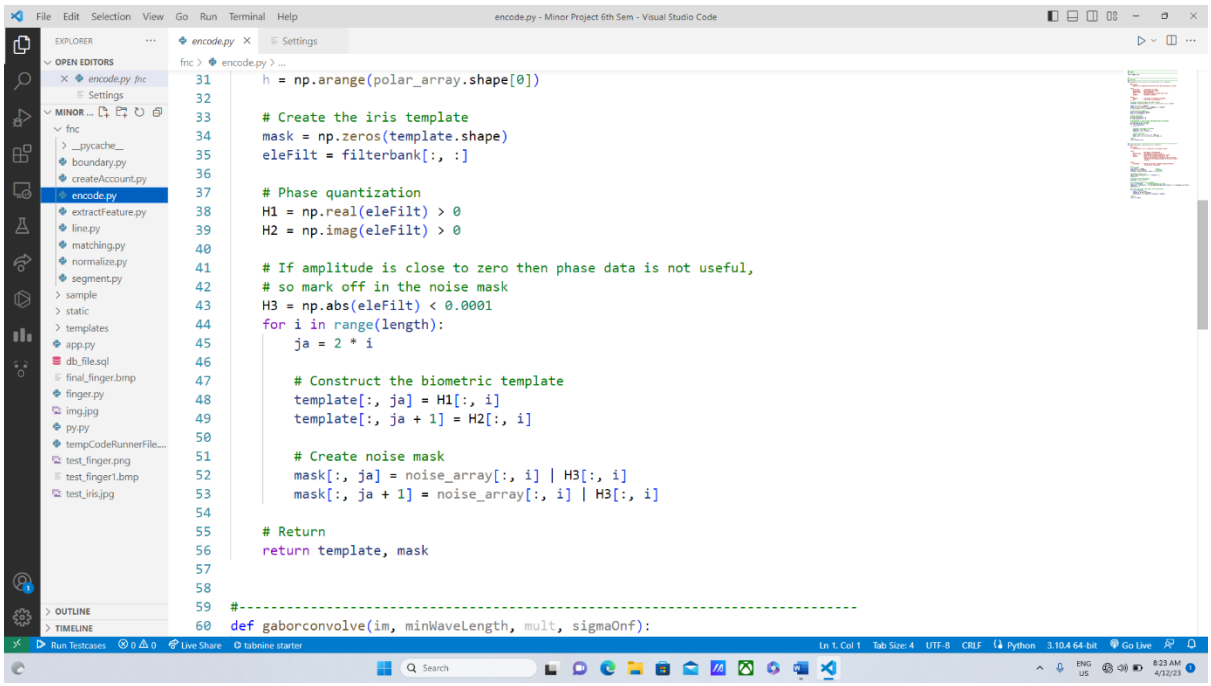
```
def ContourIntegralCircular(imagen, y_0, x_0, r, angs):
    """
    Description:
        Performs contour (circular) integral.
        Use discrete Rie-mann approach.

    Input:
        imagen  - The input iris image.
        y_0     - The y-coordinate of the circle centre.
        x_0     - The x-coordinate of the circle centre.
        r       - The radius of the circle.
        angs    - The region of the circle considering clockwise 0-2pi.

    Output:
        hs      - Integral result.
    """
    # Get y, x
    y = np.zeros([len(angs), r.shape[0], r.shape[1], r.shape[2]], dtype=int)
    x = np.zeros([len(angs), r.shape[0], r.shape[1], r.shape[2]], dtype=int)
    for i in range(len(angs)):
        ang = angs[i]
        y[i, :, :, :] = np.round(y_0 - np.cos(ang) * r).astype(int)
        x[i, :, :, :] = np.round(x_0 + np.sin(ang) * r).astype(int)

    # Adapt y
    ind = np.where(y < 0)
    y[ind] = 0
    ind = np.where(y >= imagen.shape[0])
    y[ind] = imagen.shape[0] - 1
```



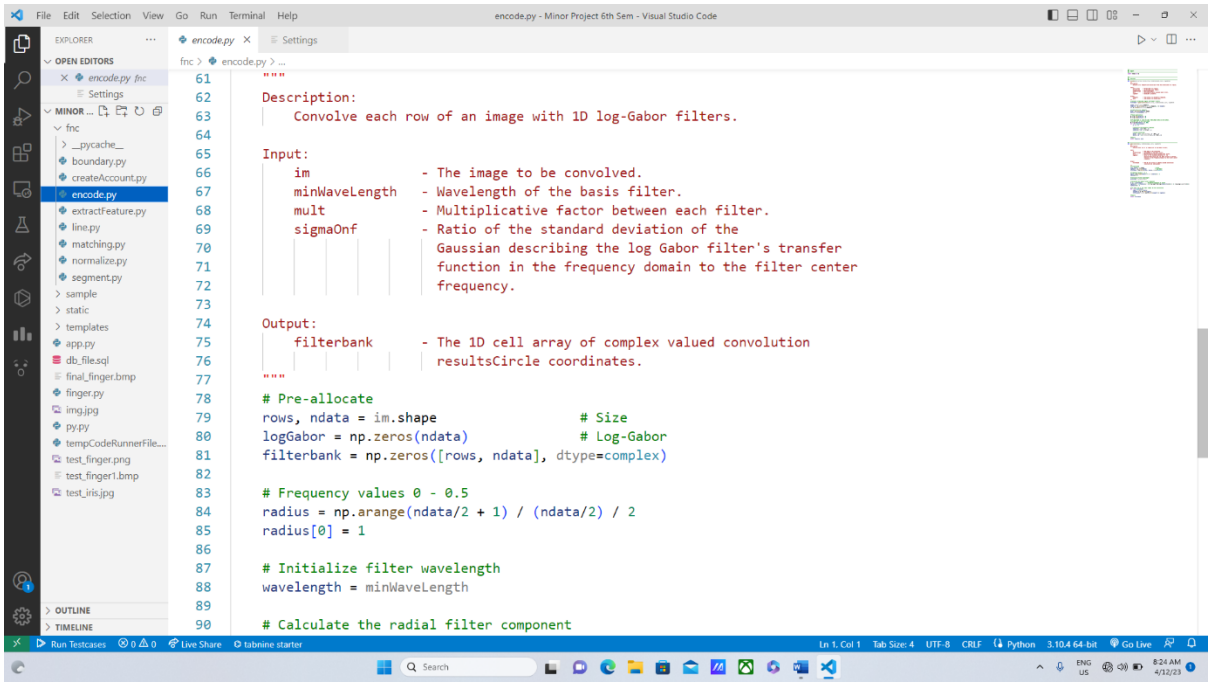
```
    # Adapt x
    ind = np.where(x < 0)
    x[ind] = 0
    ind = np.where(x >= imagen.shape[1])
    x[ind] = imagen.shape[1] - 1

    # Return
    hs = imagen[y, x]
    hs = np.sum(hs, axis=0)
    return hs.astype(float)
```



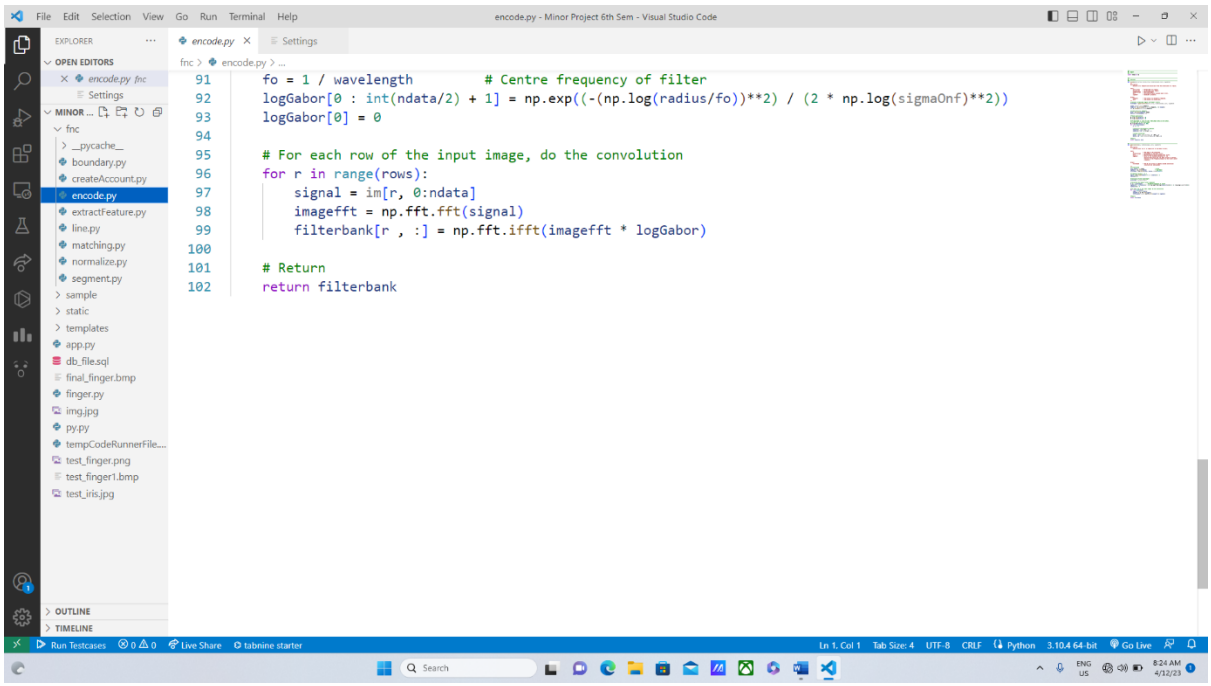
```python
##-------------------------------------------------------------------------
##  Import
##-------------------------------------------------------------------------
import numpy as np


##-------------------------------------------------------------------------
##  Function
##-------------------------------------------------------------------------
def encode(polar_array, noise_array, minWaveLength, mult, sigmaOnf):
    """
    Description:
        Generate iris template and noise mask from the normalised iris region.

    Input:
        polar_array     - Normalised iris region.
        noise_array     - Normalised noise region.
        minWaveLength   - Base wavelength.
        mult            - Multicative factor between each filter.
        sigmaOnf        - Bandwidth parameter.

    Output:
        template        - The binary iris biometric template.
        mask            - The binary iris noise mask.
    """
    # Convolve normalised region with Gabor filters
    filterbank = gaborconvolve(polar_array, minWaveLength, mult, sigmaOnf)

    length = polar_array.shape[1]
    template = np.zeros([polar_array.shape[0], 2 * length])
```



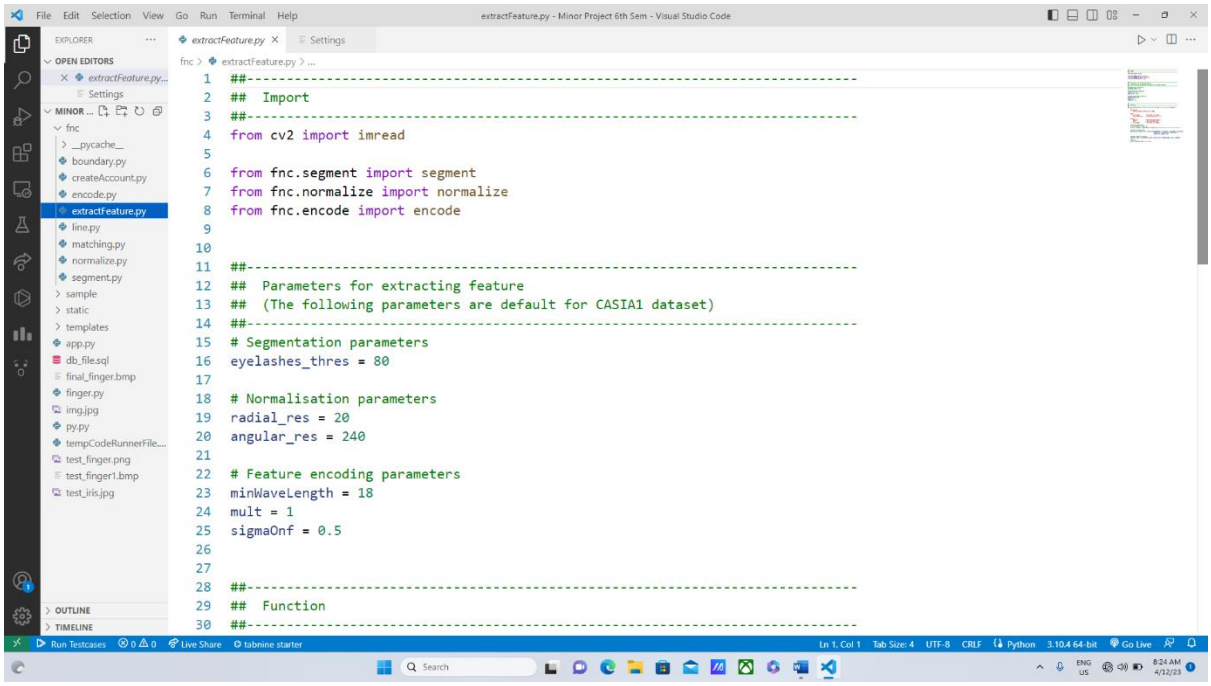
```python
    h = np.arange(polar_array.shape[0])

    # Create the iris template
    mask = np.zeros(template.shape)
    eleFilt = filterbank[:, :]

    # Phase quantization
    H1 = np.real(eleFilt) > 0
    H2 = np.imag(eleFilt) > 0

    # If amplitude is close to zero then phase data is not useful,
    # so mark off in the noise mask
    H3 = np.abs(eleFilt) < 0.0001
    for i in range(length):
        ja = 2 * i

        # Construct the biometric template
        template[:, ja] = H1[:, i]
        template[:, ja + 1] = H2[:, i]

        # Create noise mask
        mask[:, ja] = noise_array[:, i] | H3[:, i]
        mask[:, ja + 1] = noise_array[:, i] | H3[:, i]

    # Return
    return template, mask


#-------------------------------------------------------------------------
def gaborconvolve(im, minWaveLength, mult, sigmaOnf):
```



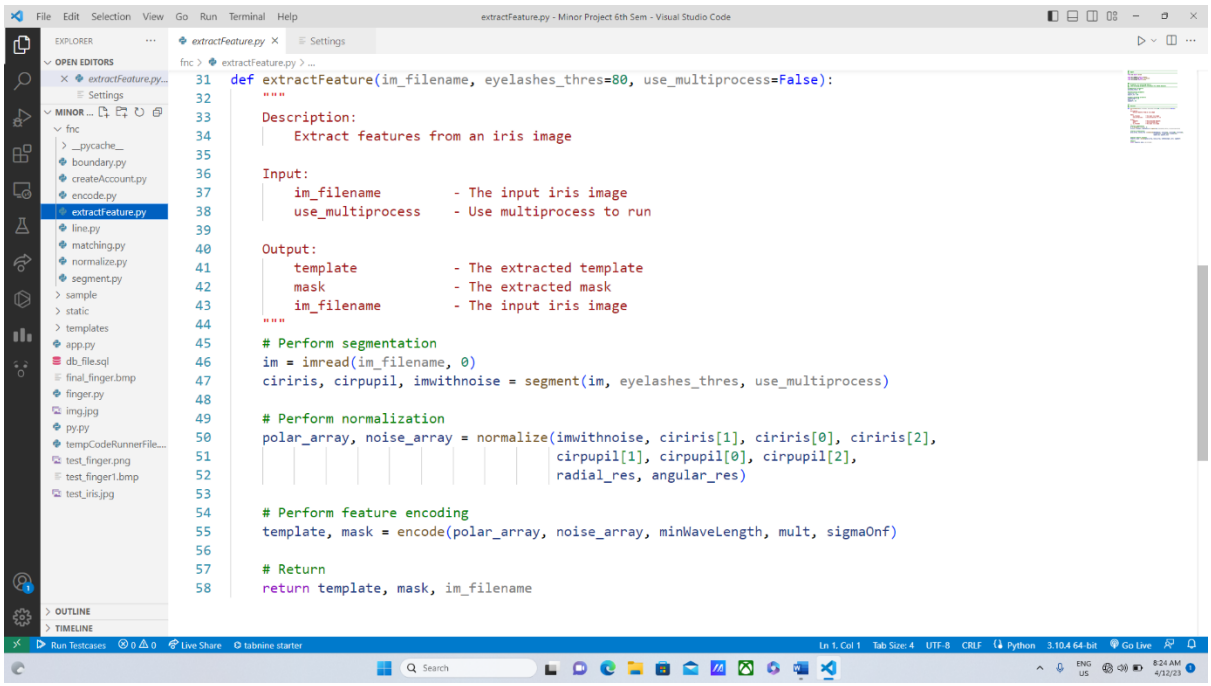
```
    """
    Description:
        Convolve each row of an image with 1D log-Gabor filters.

    Input:
        im              - The image to be convolved.
        minWaveLength   - Wavelength of the basis filter.
        mult            - Multiplicative factor between each filter.
        sigmaOnf        - Ratio of the standard deviation of the
                          Gaussian describing the log Gabor filter's transfer
                          function in the frequency domain to the filter center
                          frequency.

    Output:
        filterbank      - The 1D cell array of complex valued convolution
                          resultsCircle coordinates.
    """
    # Pre-allocate
    rows, ndata = im.shape                  # Size
    logGabor = np.zeros(ndata)              # Log-Gabor
    filterbank = np.zeros([rows, ndata], dtype=complex)

    # Frequency values 0 - 0.5
    radius = np.arange(ndata/2 + 1) / (ndata/2) / 2
    radius[0] = 1

    # Initialize filter wavelength
    wavelength = minWaveLength

    # Calculate the radial filter component
```



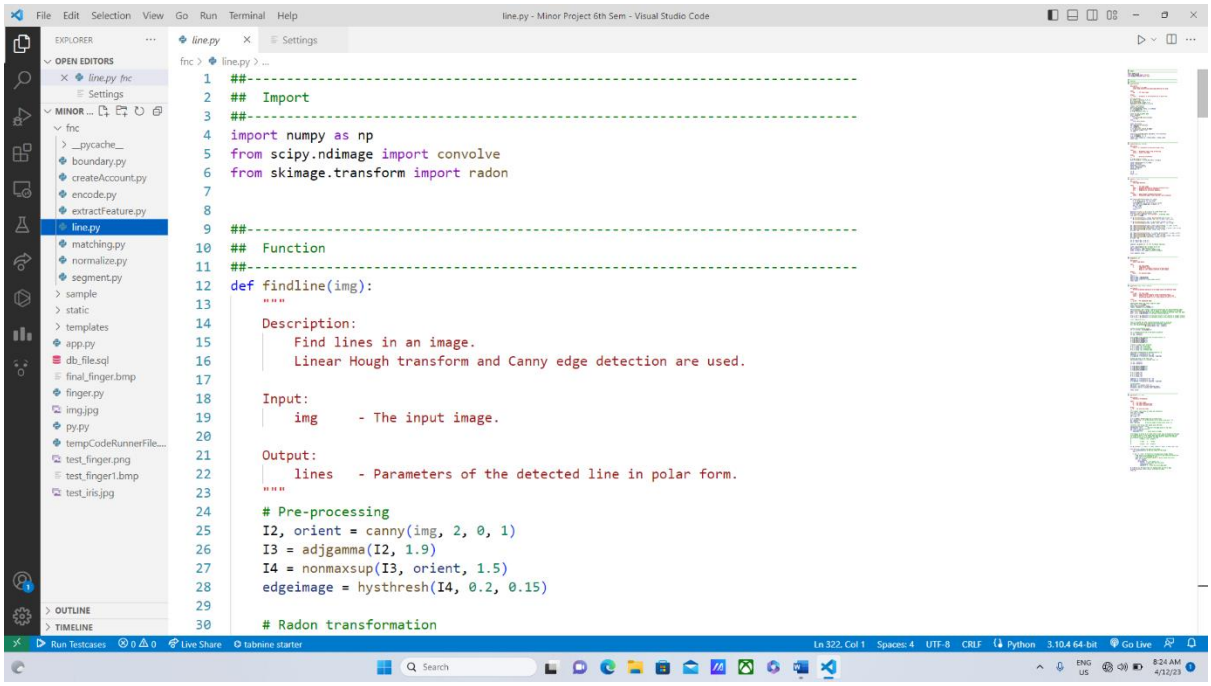
```
    fo = 1 / wavelength         # Centre frequency of filter
    logGabor[0 : int(ndata/2) + 1] = np.exp((-(np.log(radius/fo))**2) / (2 * np.log(sigmaOnf)**2))
    logGabor[0] = 0

    # For each row of the input image, do the convolution
    for r in range(rows):
        signal = im[r, 0:ndata]
        imagefft = np.fft.fft(signal)
        filterbank[r , :] = np.fft.ifft(imagefft * logGabor)

    # Return
    return filterbank
```



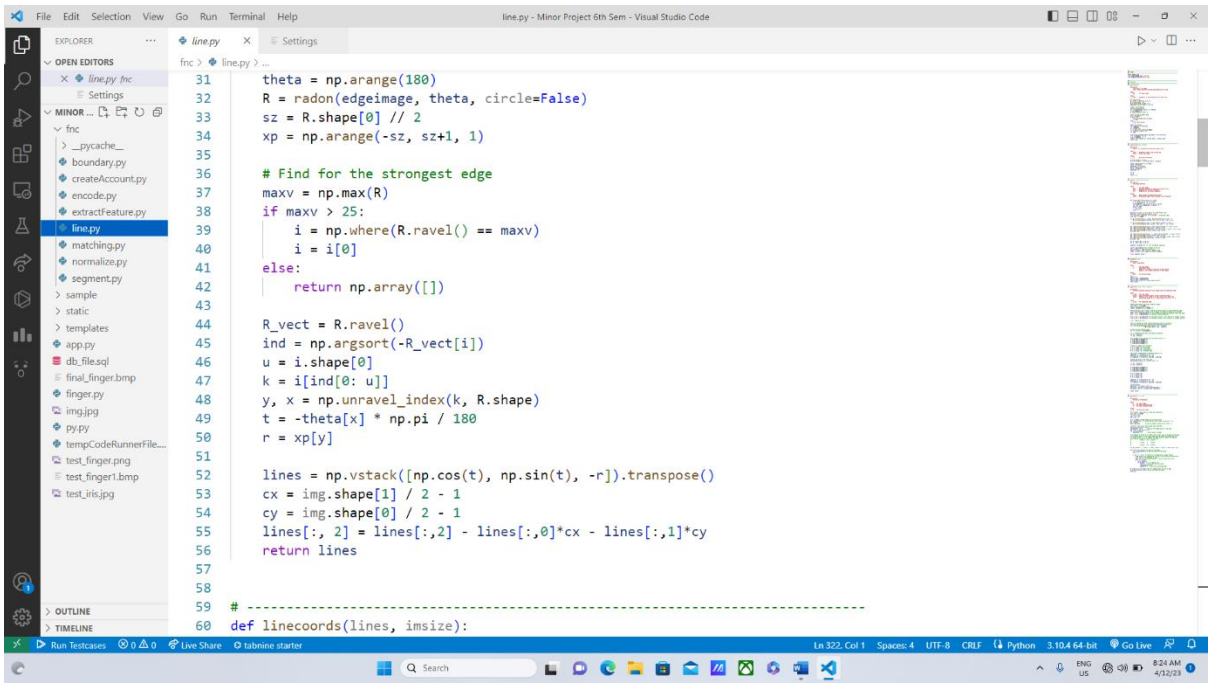
```python
##-------------------------------------------------------------------------
##  Import
##-------------------------------------------------------------------------
from cv2 import imread

from fnc.segment import segment
from fnc.normalize import normalize
from fnc.encode import encode


##-------------------------------------------------------------------------
##  Parameters for extracting feature
##  (The following parameters are default for CASIA1 dataset)
##-------------------------------------------------------------------------
# Segmentation parameters
eyelashes_thres = 80

# Normalisation parameters
radial_res = 20
angular_res = 240

# Feature encoding parameters
minWaveLength = 18
mult = 1
sigmaOnf = 0.5


##-------------------------------------------------------------------------
##  Function
##-------------------------------------------------------------------------
```



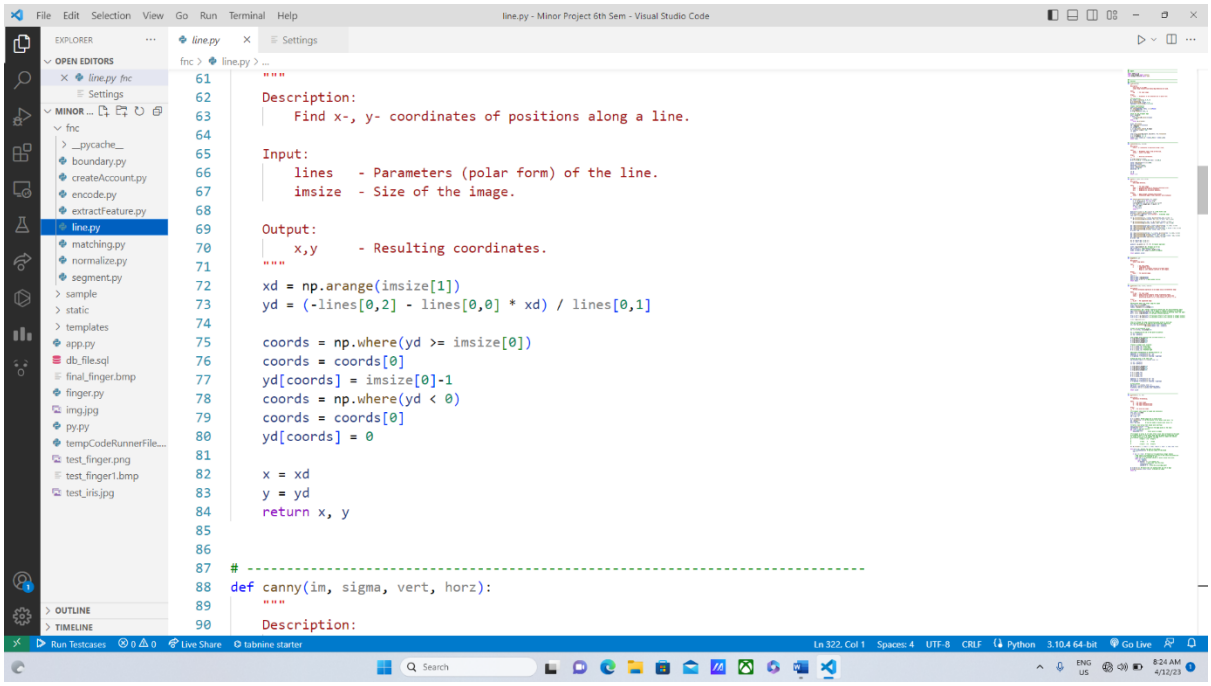
```python
def extractFeature(im_filename, eyelashes_thres=80, use_multiprocess=False):
    """
    Description:
        Extract features from an iris image

    Input:
        im_filename         - The input iris image
        use_multiprocess    - Use multiprocess to run

    Output:
        template            - The extracted template
        mask                - The extracted mask
        im_filename         - The input iris image
    """
    # Perform segmentation
    im = imread(im_filename, 0)
    ciriris, cirpupil, imwithnoise = segment(im, eyelashes_thres, use_multiprocess)

    # Perform normalization
    polar_array, noise_array = normalize(imwithnoise, ciriris[1], ciriris[0], ciriris[2],
                                         cirpupil[1], cirpupil[0], cirpupil[2],
                                         radial_res, angular_res)

    # Perform feature encoding
    template, mask = encode(polar_array, noise_array, minWaveLength, mult, sigmaOnf)

    # Return
    return template, mask, im_filename
```



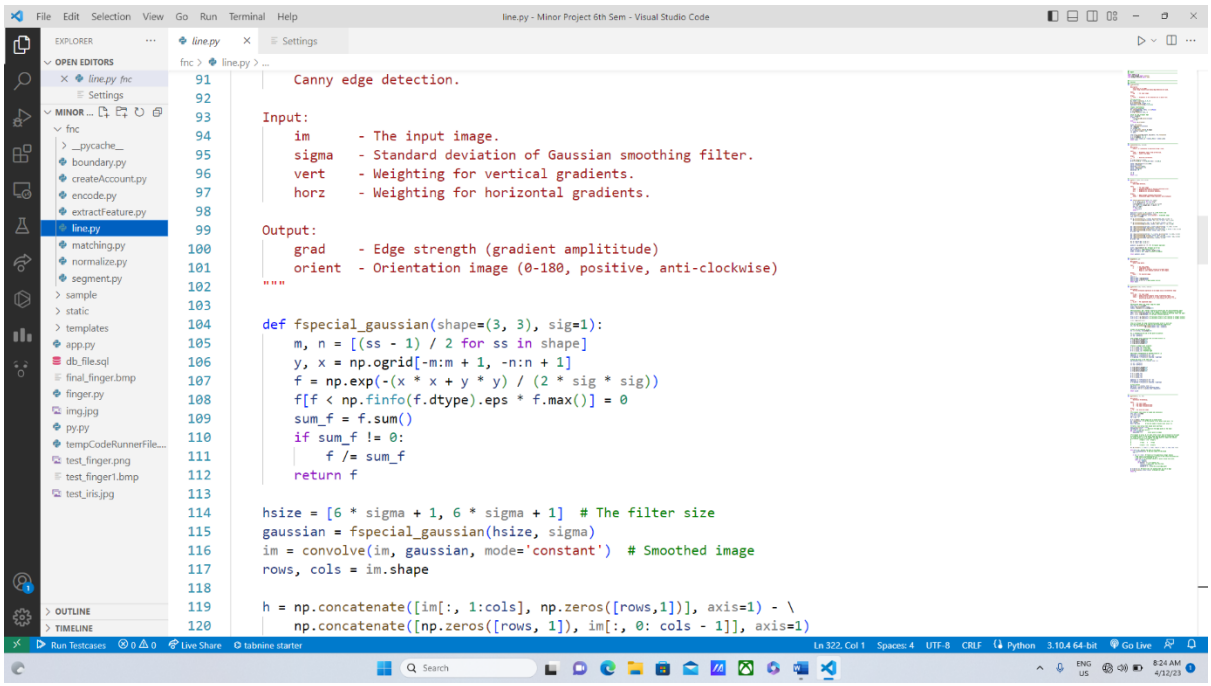
```
##-------------------------------------------------------------------------
##  Import
##-------------------------------------------------------------------------
import numpy as np
from scipy.ndimage import convolve
from skimage.transform import radon


##-------------------------------------------------------------------------
##  Function
##-------------------------------------------------------------------------
def findline(img):
    """
    Description:
        Find lines in an image.
        Linear Hough transform and Canny edge detection are used.

    Input:
        img     - The input image.

    Output:
        lines   - Parameters of the detected line in polar form.
    """
    # Pre-processing
    I2, orient = canny(img, 2, 0, 1)
    I3 = adjgamma(I2, 1.9)
    I4 = nonmaxsup(I3, orient, 1.5)
    edgeimage = hysthresh(I4, 0.2, 0.15)

    # Radon transformation
```



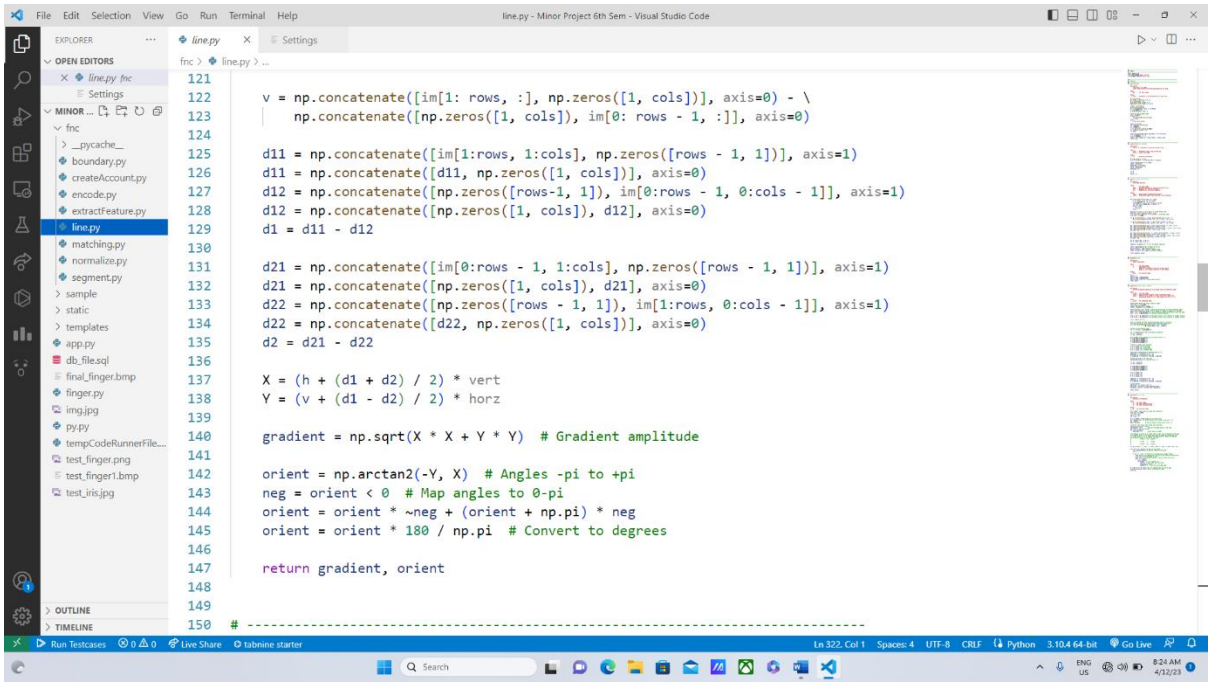
```
    theta = np.arange(180)
    R = radon(edgeimage, theta, circle=False)
    sz = R.shape[0] // 2
    xp = np.arange(-sz, sz+1, 1)

    # Find for the strongest edge
    maxv = np.max(R)
    if maxv > 25:
        i = np.where(R.ravel() == maxv)
        i = i[0]
    else:
        return np.array([])

    R_vect = R.ravel()
    ind = np.argsort(-R_vect[i])
    u = i.shape[0]
    k = i[ind[0: u]]
    y, x = np.unravel_index(k, R.shape)
    t = -theta[x] * np.pi / 180
    r = xp[y]

    lines = np.vstack([np.cos(t), np.sin(t), -r]).transpose()
    cx = img.shape[1] / 2 - 1
    cy = img.shape[0] / 2 - 1
    lines[:, 2] = lines[:,2] - lines[:,0]*cx - lines[:,1]*cy
    return lines


# ---------------------------------------------------------------------------
def linecoords(lines, imsize):
```



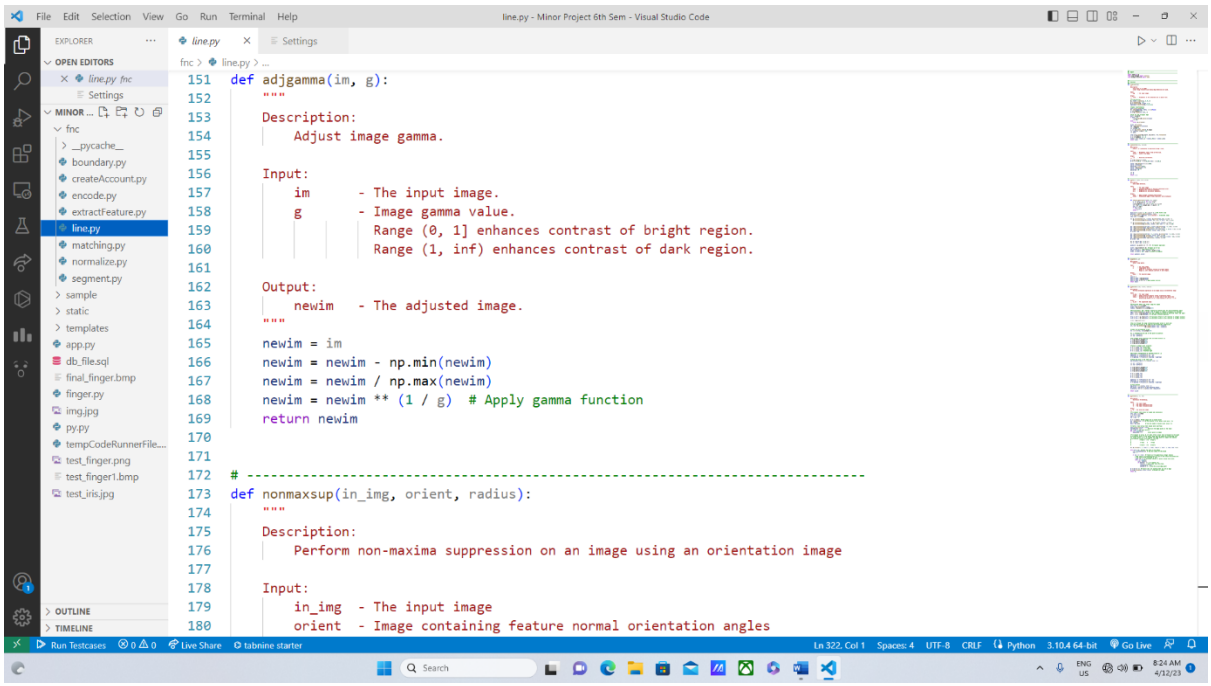
```
    """
    Description:
        Find x-, y- coordinates of positions along a line.

    Input:
        lines   - Parameters (polar form) of the line.
        imsize  - Size of the image.

    Output:
        x,y     - Resulting coordinates.
    """
    xd = np.arange(imsize[1])
    yd = (-lines[0,2] - lines[0,0] * xd) / lines[0,1]

    coords = np.where(yd >= imsize[0])
    coords = coords[0]
    yd[coords] = imsize[0]-1
    coords = np.where(yd < 0)
    coords = coords[0]
    yd[coords] = 0

    x = xd
    y = yd
    return x, y


# --------------------------------------------------------------------------
def canny(im, sigma, vert, horz):
    """
    Description:
```



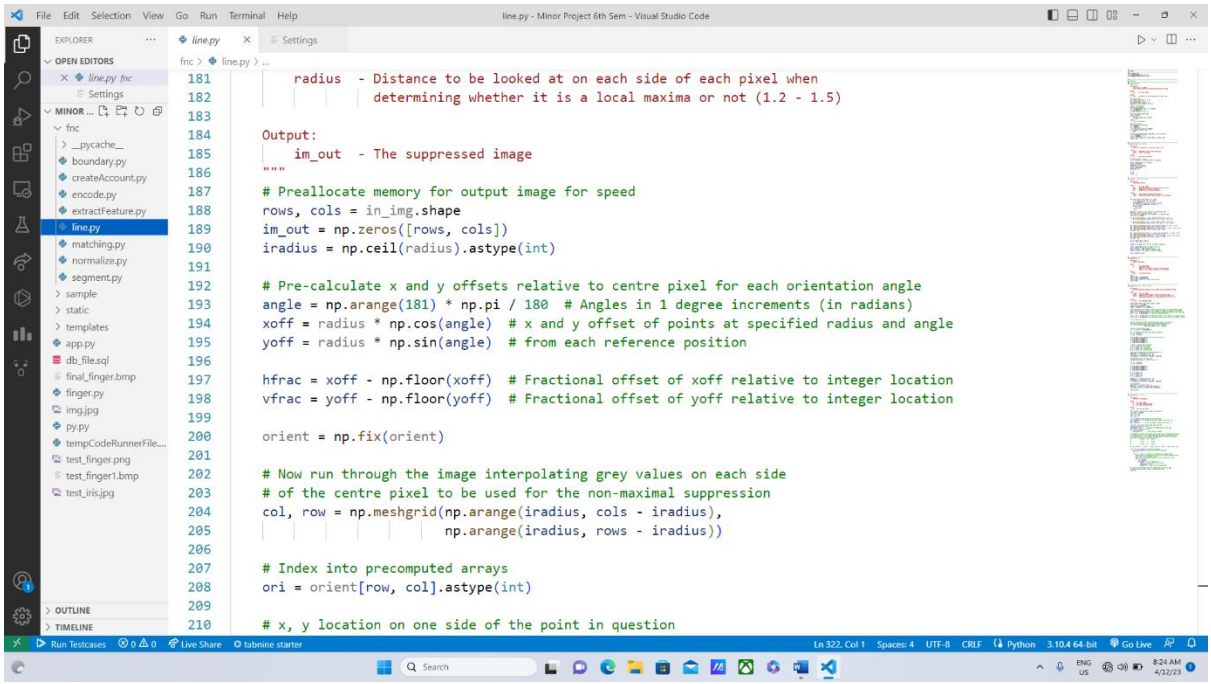
```
        Canny edge detection.

    Input:
        im      - The input image.
        sigma   - Standard deviation of Gaussian smoothing filter.
        vert    - Weighting for vertical gradients.
        horz    - Weighting for horizontal gradients.

    Output:
        grad    - Edge strength (gradient amplititude)
        orient  - Orientation image (0-180, positive, anti-clockwise)
    """

    def fspecial_gaussian(shape=(3, 3), sig=1):
        m, n = [(ss - 1) / 2 for ss in shape]
        y, x = np.ogrid[-m:m + 1, -n:n + 1]
        f = np.exp(-(x * x + y * y) / (2 * sig * sig))
        f[f < np.finfo(f.dtype).eps * f.max()] = 0
        sum_f = f.sum()
        if sum_f != 0:
            f /= sum_f
        return f

    hsize = [6 * sigma + 1, 6 * sigma + 1]  # The filter size
    gaussian = fspecial_gaussian(hsize, sigma)
    im = convolve(im, gaussian, mode='constant')  # Smoothed image
    rows, cols = im.shape

    h = np.concatenate([im[:, 1:cols], np.zeros([rows,1])], axis=1) - \
        np.concatenate([np.zeros([rows, 1]), im[:, 0: cols - 1]], axis=1)
```



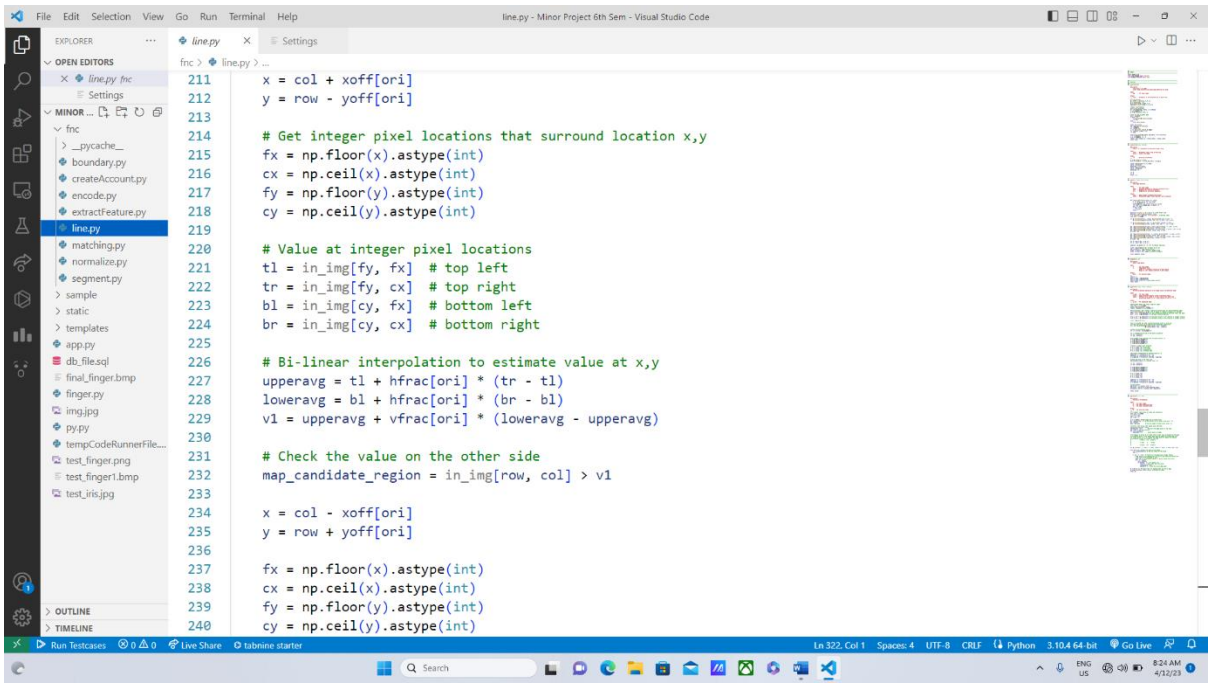
```
    v = np.concatenate([im[1: rows, :], np.zeros([1, cols])], axis=0) - \
        np.concatenate([np.zeros([1, cols]), im[0: rows - 1, :]], axis=0)

    d11 = np.concatenate([im[1:rows, 1:cols], np.zeros([rows - 1, 1])], axis=1)
    d11 = np.concatenate([d11, np.zeros([1, cols])], axis=0)
    d12 = np.concatenate([np.zeros([rows-1, 1]), im[0:rows - 1, 0:cols - 1]], axis=1)
    d12 = np.concatenate([np.zeros([1, cols]), d12], axis=0)
    d1 = d11 - d12

    d21 = np.concatenate([im[0:rows - 1, 1:cols], np.zeros([rows - 1, 1])], axis=1)
    d21 = np.concatenate([np.zeros([1, cols]), d21], axis=0)
    d22 = np.concatenate([np.zeros([rows - 1, 1]), im[1:rows, 0:cols - 1]], axis=1)
    d22 = np.concatenate([d22, np.zeros([1, cols])], axis=0)
    d2 = d21 - d22

    X = (h + (d1 + d2) / 2) * vert
    Y = (v + (d1 - d2) / 2) * horz

    gradient = np.sqrt(X * X + Y * Y)  # Gradient amplitude

    orient = np.arctan2(-Y, X)  # Angles -pi to +pi
    neg = orient < 0  # Map angles to 0-pi
    orient = orient * ~neg + (orient + np.pi) * neg
    orient = orient * 180 / np.pi  # Convert to degrees

    return gradient, orient


# ---------------------------------------------------------------------------
```



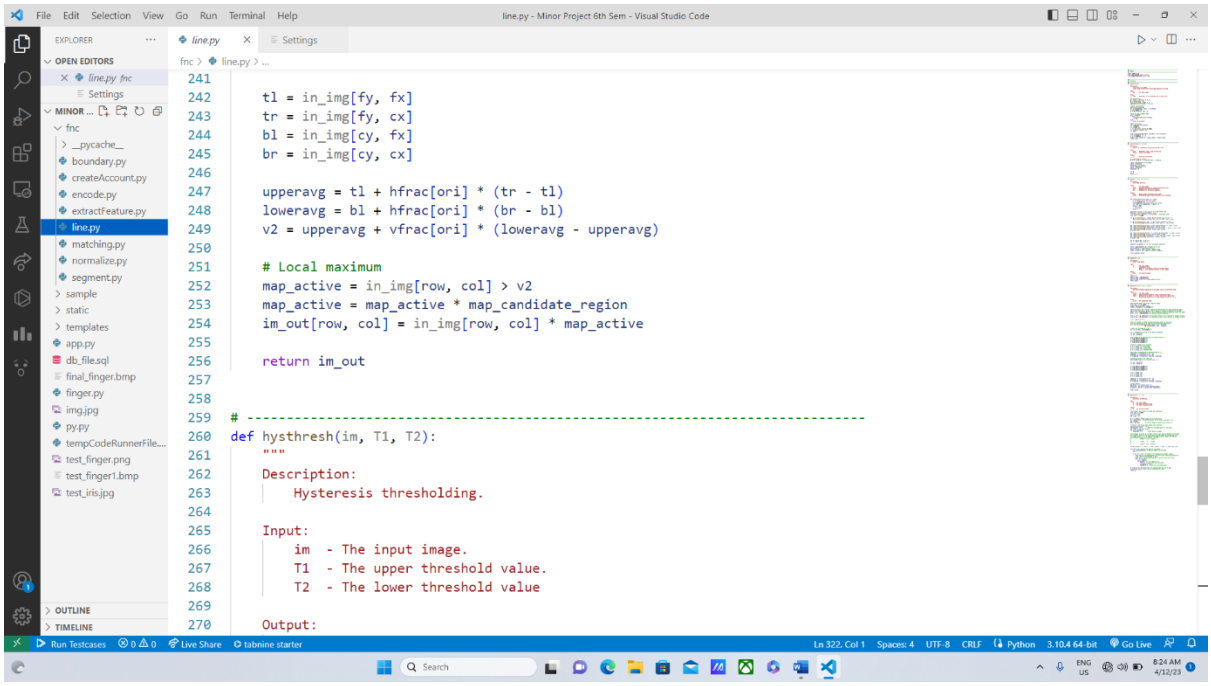
```
def adjgamma(im, g):
    """
    Description:
        Adjust image gamma.

    Input:
        im      - The input image.
        g       - Image gamma value.
                  Range (0, 1] enhances contrast of bright region.
                  Range (1, inf) enhances contrast of dark region.

    Output:
        newim   - The adjusted image.
    """
    newim = im
    newim = newim - np.min(newim)
    newim = newim / np.max(newim)
    newim = newim ** (1 / g)  # Apply gamma function
    return newim


# ---------------------------------------------------------------------------
def nonmaxsup(in_img, orient, radius):
    """
    Description:
        Perform non-maxima suppression on an image using an orientation image

    Input:
        in_img  - The input image
        orient  - Image containing feature normal orientation angles
```



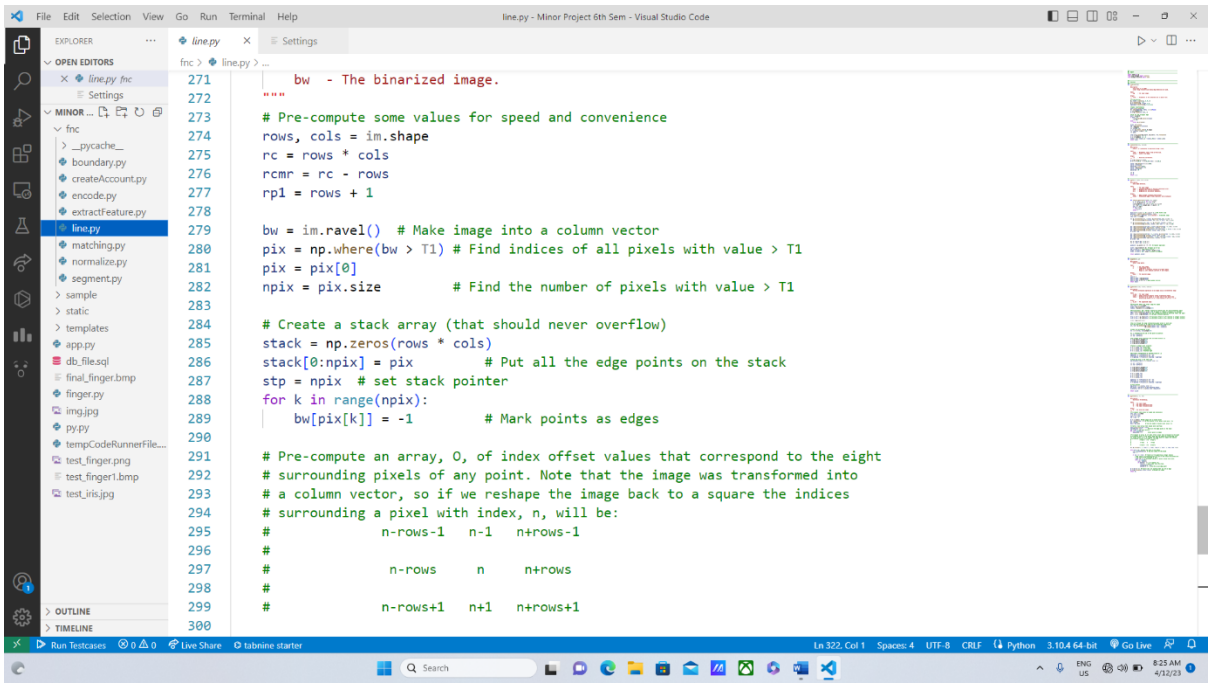
```
    radius  - Distance to be looked at on each side of each pixel when
              determining whether it is a local maxima or not (1.2 - 1.5)

Output:
    im_out  - The suppressed image
"""
# Preallocate memory for output image for speed
rows, cols = in_img.shape
im_out = np.zeros([rows, cols])
iradius = np.ceil(radius).astype(int)

# Pre-calculate x and y offsets relative to centre pixel for each orientation angle
angle = np.arange(181) * np.pi / 180  # Angles in 1 degree increments (in radians)
xoff = radius * np.cos(angle)  # x and y offset of points at specified radius and angle
yoff = radius * np.sin(angle)  # from each reference position

hfrac = xoff - np.floor(xoff)  # Fractional offset of xoff relative to integer location
vfrac = yoff - np.floor(yoff)  # Fractional offset of yoff relative to integer location

orient = np.fix(orient)

# Now run through the image interpolating grey values on each side
# of the centre pixel to be used for the non-maximal suppression
col, row = np.meshgrid(np.arange(iradius, cols - iradius),
                       np.arange(iradius, rows - iradius))

# Index into precomputed arrays
ori = orient[row, col].astype(int)

# x, y location on one side of the point in question
```



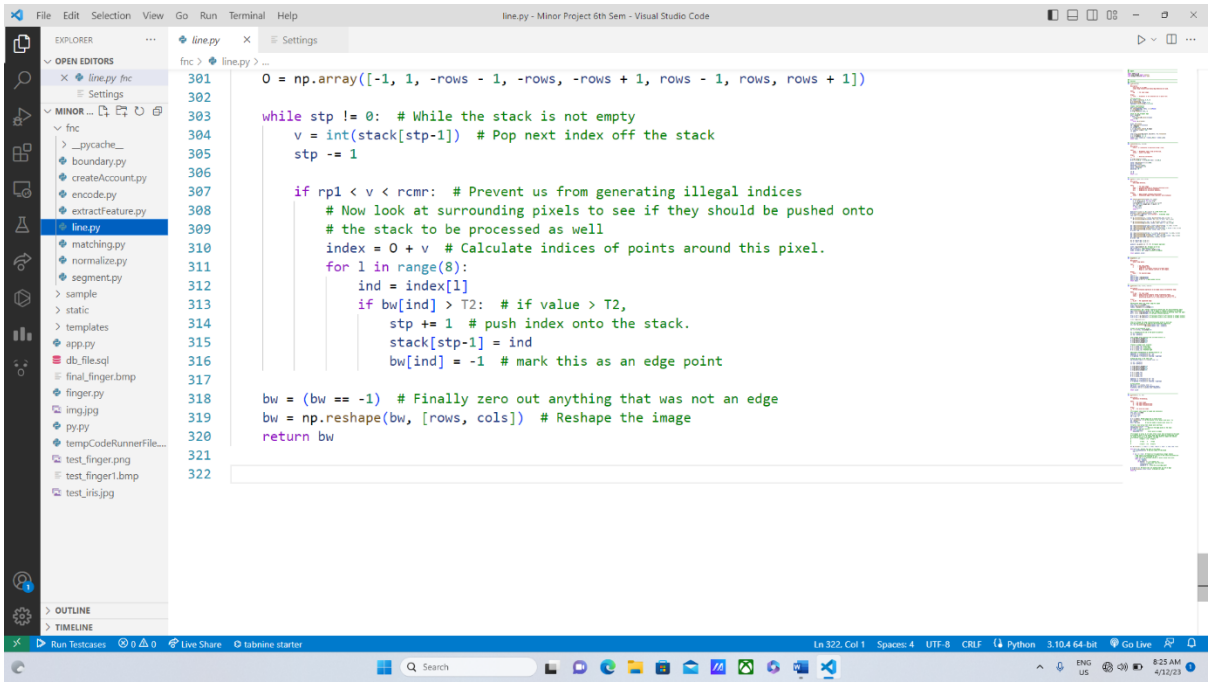
```
x = col + xoff[ori]
y = row - yoff[ori]

# Get integer pixel locations that surround location x,y
fx = np.floor(x).astype(int)
cx = np.ceil(x).astype(int)
fy = np.floor(y).astype(int)
cy = np.ceil(y).astype(int)

# Value at integer pixel locations
tl = in_img[fy, fx]  # top left
tr = in_img[fy, cx]  # top right
bl = in_img[cy, fx]  # bottom left
br = in_img[cy, cx]  # bottom right

# Bi-linear interpolation to estimate value at x,y
upperavg = tl + hfrac[ori] * (tr - tl)
loweravg = bl + hfrac[ori] * (br - bl)
v1 = upperavg + vfrac[ori] * (loweravg - upperavg)

# Check the value on the other side
map_candidate_region = in_img[row, col] > v1

x = col - xoff[ori]
y = row + yoff[ori]

fx = np.floor(x).astype(int)
cx = np.ceil(x).astype(int)
fy = np.floor(y).astype(int)
cy = np.ceil(y).astype(int)
```



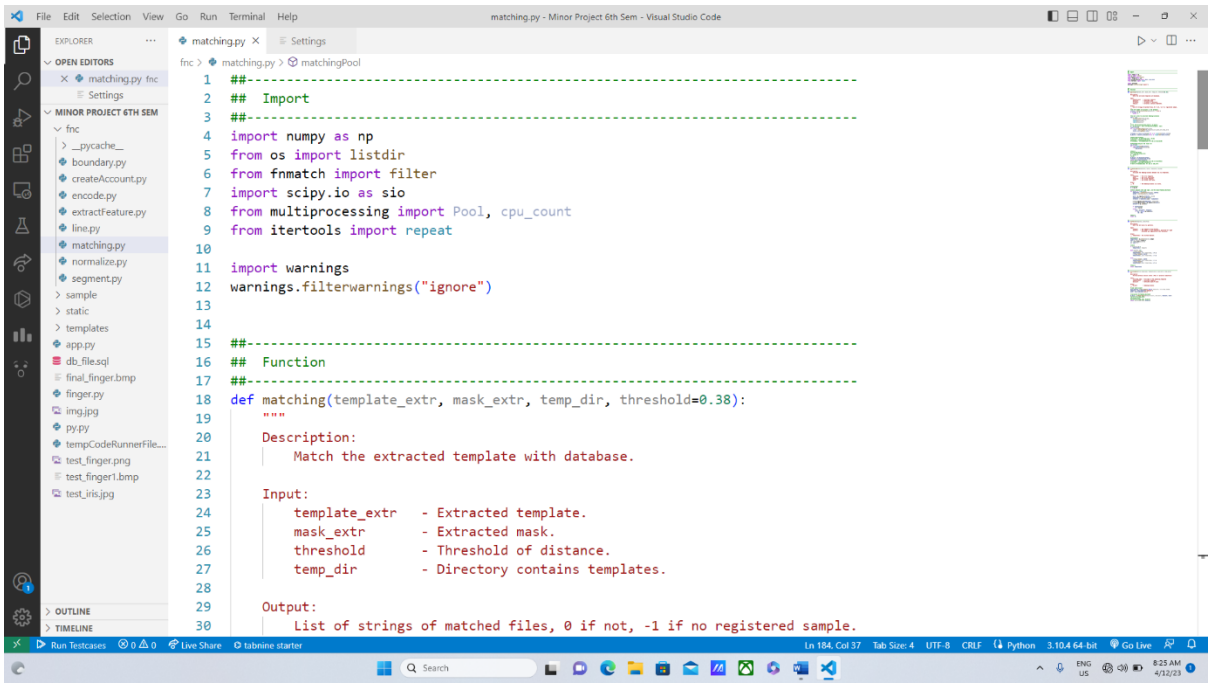
```
    tl = in_img[fy, fx]
    tr = in_img[fy, cx]
    bl = in_img[cy, fx]
    br = in_img[cy, cx]

    upperavg = tl + hfrac[ori] * (tr - tl)
    loweravg = bl + hfrac[ori] * (br - bl)
    v2 = upperavg + vfrac[ori] * (loweravg - upperavg)

    # Local maximum
    map_active = in_img[row, col] > v2
    map_active = map_active * map_candidate_region
    im_out[row, col] = in_img[row, col] * map_active

    return im_out


# ------------------------------------------------------------------------
def hysthresh(im, T1, T2):
    """
    Description:
        Hysteresis thresholding.

    Input:
        im  - The input image.
        T1  - The upper threshold value.
        T2  - The lower threshold value

    Output:
```



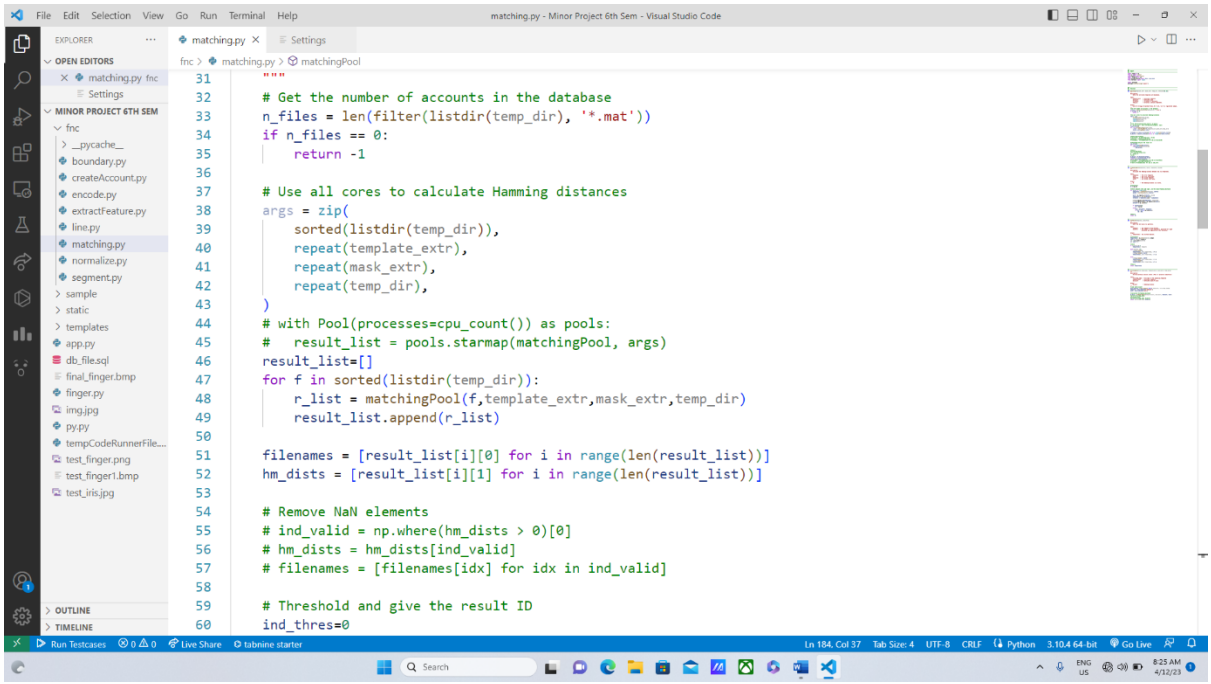
```
        bw  - The binarized image.
    """
    # Pre-compute some values for speed and convenience
    rows, cols = im.shape
    rc = rows * cols
    rcmr = rc - rows
    rp1 = rows + 1

    bw = im.ravel()  # Make image into a column vector
    pix = np.where(bw > T1) # Find indices of all pixels with value > T1
    pix = pix[0]
    npix = pix.size         # Find the number of pixels with value > T1

    # Create a stack array (that should never overflow)
    stack = np.zeros(rows * cols)
    stack[0:npix] = pix         # Put all the edge points on the stack
    stp = npix  # set stack pointer
    for k in range(npix):
        bw[pix[k]] = -1         # Mark points as edges

    # Pre-compute an array, O, of index offset values that correspond to the eight
    # surrounding pixels of any point. Note that the image was transformed into
    # a column vector, so if we reshape the image back to a square the indices
    # surrounding a pixel with index, n, will be:
    #             n-rows-1   n-1   n+rows-1
    #
    #              n-rows     n     n+rows
    #
    #             n-rows+1   n+1   n+rows+1
```



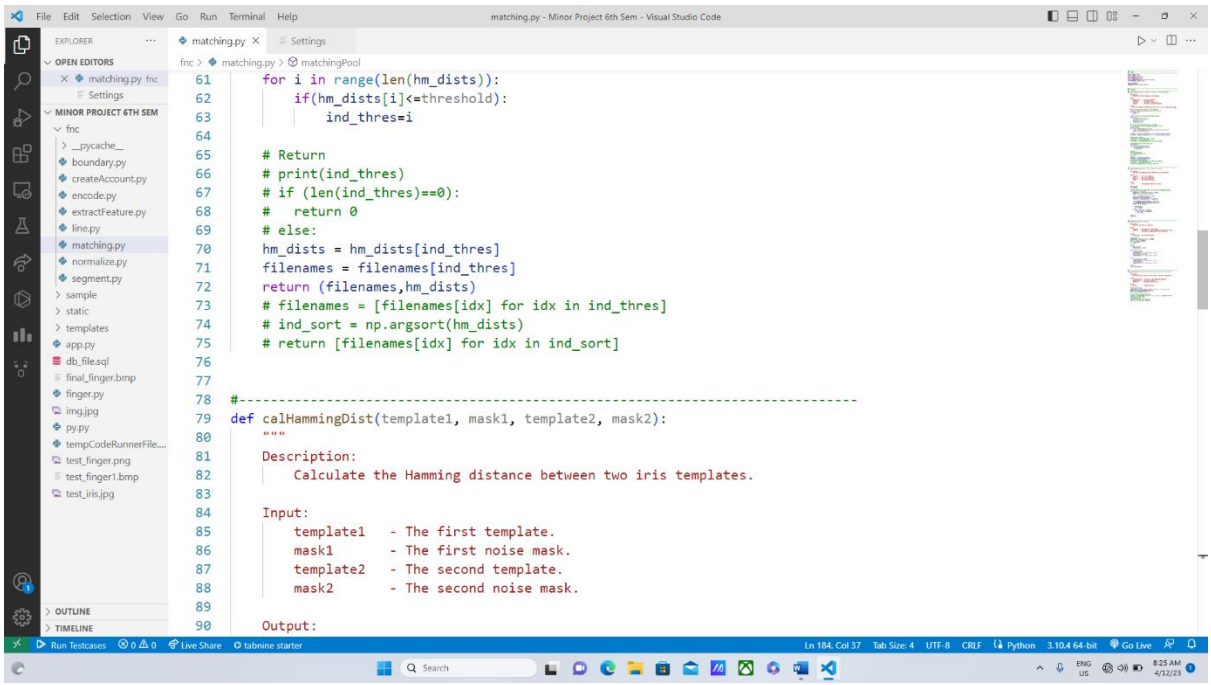
```
    O = np.array([-1, 1, -rows - 1, -rows, -rows + 1, rows - 1, rows, rows + 1])

    while stp != 0:  # While the stack is not empty
        v = int(stack[stp-1])  # Pop next index off the stack
        stp -= 1

        if rp1 < v < rcmr:  # Prevent us from generating illegal indices
            # Now look at surrounding pixels to see if they should be pushed onto
            # the stack to be processed as well
            index = O + v  # Calculate indices of points around this pixel.
            for l in range(8):
                ind = index[l]
                if bw[ind] > T2:  # if value > T2,
                    stp += 1  # push index onto the stack.
                    stack[stp-1] = ind
                    bw[ind] = -1  # mark this as an edge point

    bw = (bw == -1)  # Finally zero out anything that was not an edge
    bw = np.reshape(bw, [rows, cols])  # Reshape the image
    return bw
```



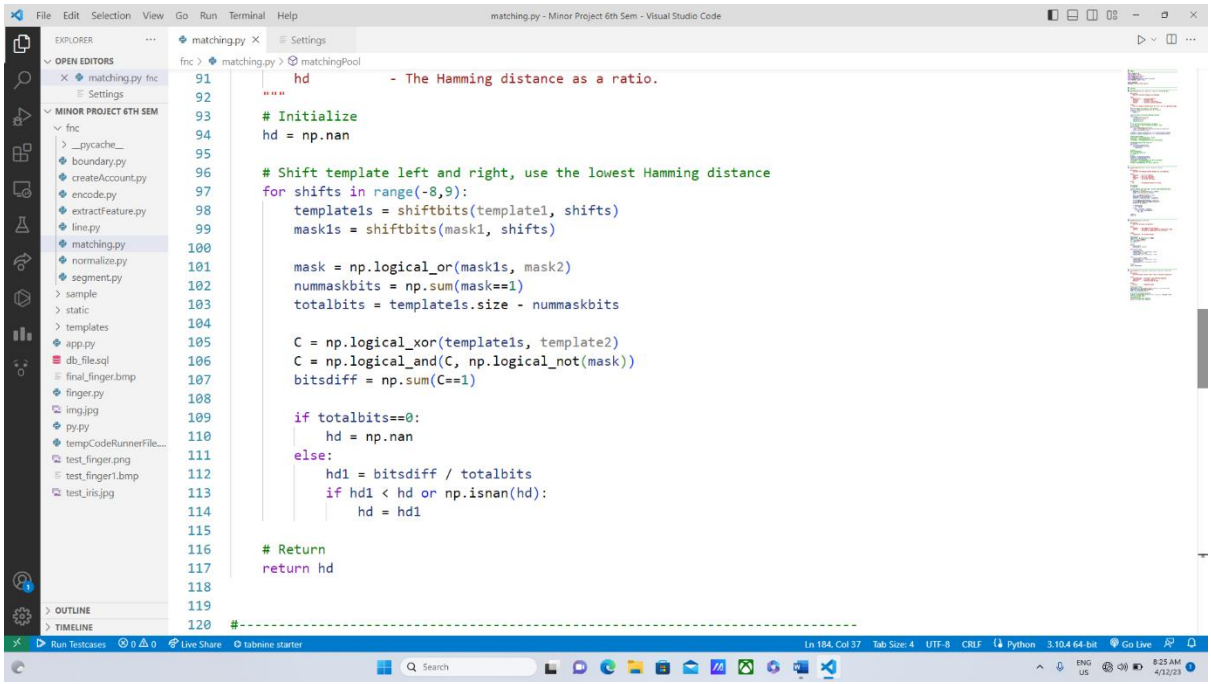
```
##------------------------------------------------------------------------
##  Import
##------------------------------------------------------------------------
import numpy as np
from os import listdir
from fnmatch import filter
import scipy.io as sio
from multiprocessing import Pool, cpu_count
from itertools import repeat

import warnings
warnings.filterwarnings("ignore")


##------------------------------------------------------------------------
##  Function
##------------------------------------------------------------------------
def matching(template_extr, mask_extr, temp_dir, threshold=0.38):
    """
    Description:
        Match the extracted template with database.

    Input:
        template_extr   - Extracted template.
        mask_extr       - Extracted mask.
        threshold       - Threshold of distance.
        temp_dir        - Directory contains templates.

    Output:
        List of strings of matched files, 0 if not, -1 if no registered sample.
```



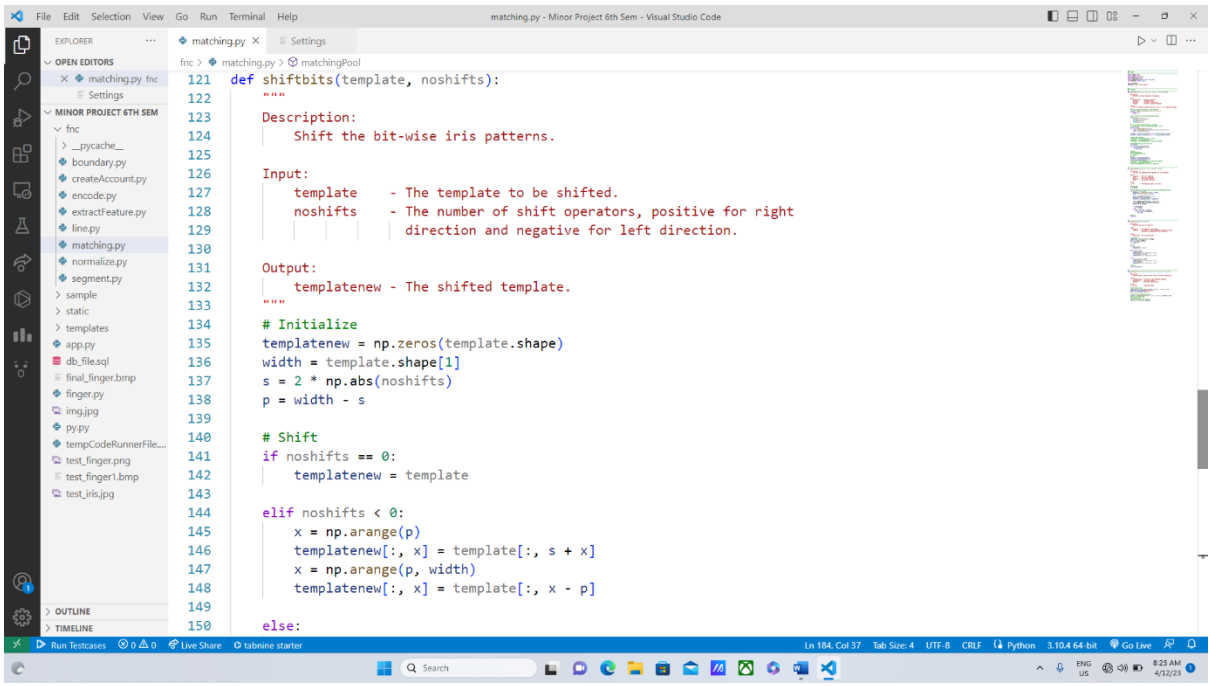
```
    """
    # Get the number of accounts in the database
    n_files = len(filter(listdir(temp_dir), '*.mat'))
    if n_files == 0:
        return -1

    # Use all cores to calculate Hamming distances
    args = zip(
        sorted(listdir(temp_dir)),
        repeat(template_extr),
        repeat(mask_extr),
        repeat(temp_dir),
    )
    # with Pool(processes=cpu_count()) as pools:
    #   result_list = pools.starmap(matchingPool, args)
    result_list=[]
    for f in sorted(listdir(temp_dir)):
        r_list = matchingPool(f,template_extr,mask_extr,temp_dir)
        result_list.append(r_list)

    filenames = [result_list[i][0] for i in range(len(result_list))]
    hm_dists = [result_list[i][1] for i in range(len(result_list))]

    # Remove NaN elements
    # ind_valid = np.where(hm_dists > 0)[0]
    # hm_dists = hm_dists[ind_valid]
    # filenames = [filenames[idx] for idx in ind_valid]

    # Threshold and give the result ID
    ind_thres=0
```



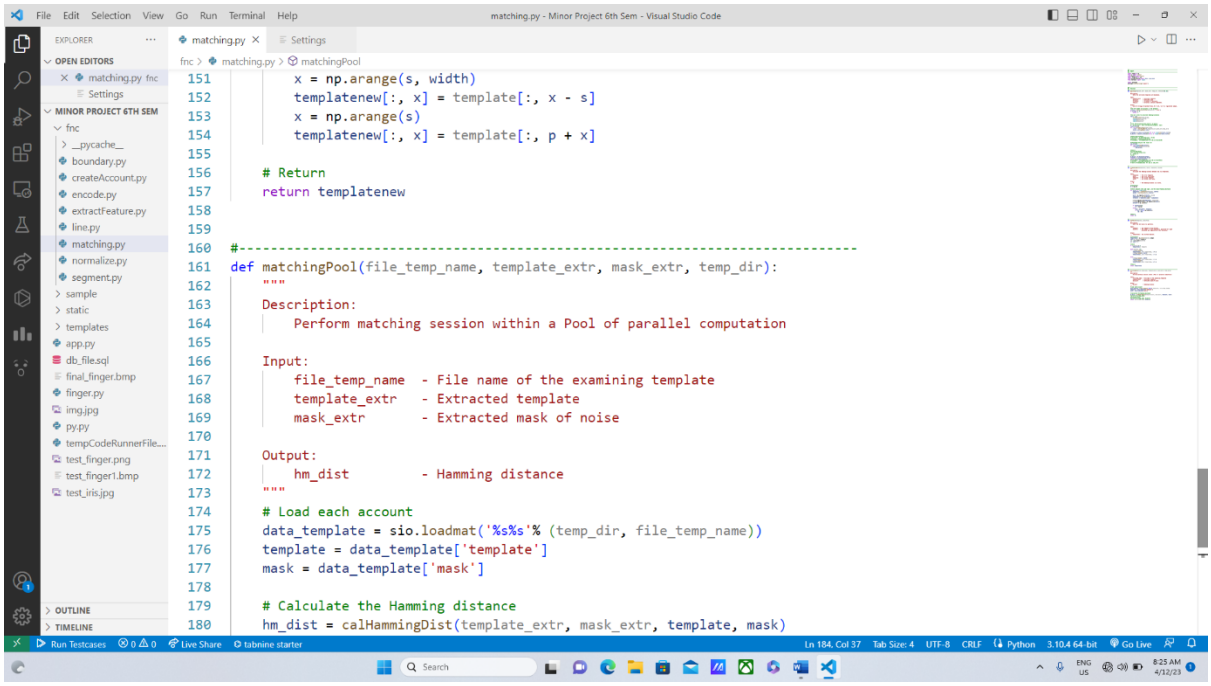
```
    for i in range(len(hm_dists)):
        if(hm_dists[i]<=threshold):
            ind_thres=i

    # Return
    # print(ind_thres)
    # if (len(ind_thres)==0):
    #   return 0
    # else:
    hm_dists = hm_dists[ind_thres]
    filenames = filenames[ind_thres]
    return (filenames,hm_dists)
    # filenames = [filenames[idx] for idx in ind_thres]
    # ind_sort = np.argsort(hm_dists)
    # return [filenames[idx] for idx in ind_sort]


#-----------------------------------------------------------------------------
def calHammingDist(template1, mask1, template2, mask2):
    """
    Description:
        Calculate the Hamming distance between two iris templates.

    Input:
        template1   - The first template.
        mask1       - The first noise mask.
        template2   - The second template.
        mask2       - The second noise mask.

    Output:
```



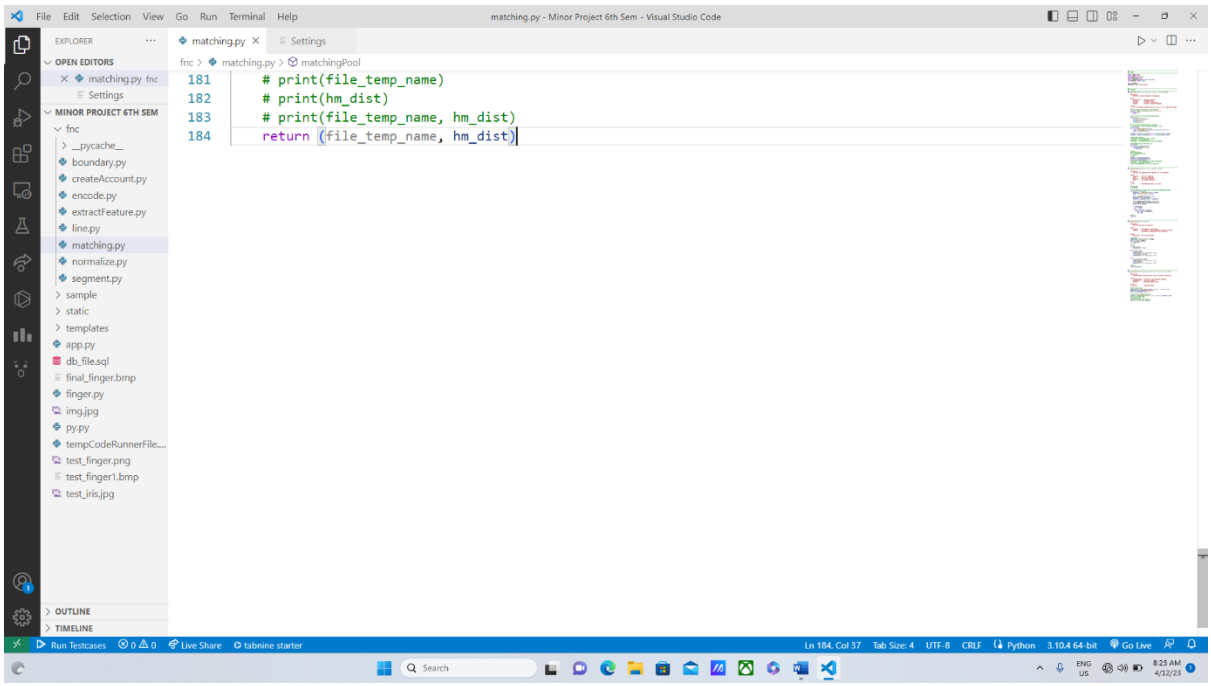
```
        hd          - The Hamming distance as a ratio.
    """
    # Initialize
    hd = np.nan

    # Shift template left and right, use the lowest Hamming distance
    for shifts in range(-8,9):
        template1s = shiftbits(template1, shifts)
        mask1s = shiftbits(mask1, shifts)

        mask = np.logical_or(mask1s, mask2)
        nummaskbits = np.sum(mask==1)
        totalbits = template1s.size - nummaskbits

        C = np.logical_xor(template1s, template2)
        C = np.logical_and(C, np.logical_not(mask))
        bitsdiff = np.sum(C==1)

        if totalbits==0:
            hd = np.nan
        else:
            hd1 = bitsdiff / totalbits
            if hd1 < hd or np.isnan(hd):
                hd = hd1

    # Return
    return hd


#---------------------------------------------------------------------------
```



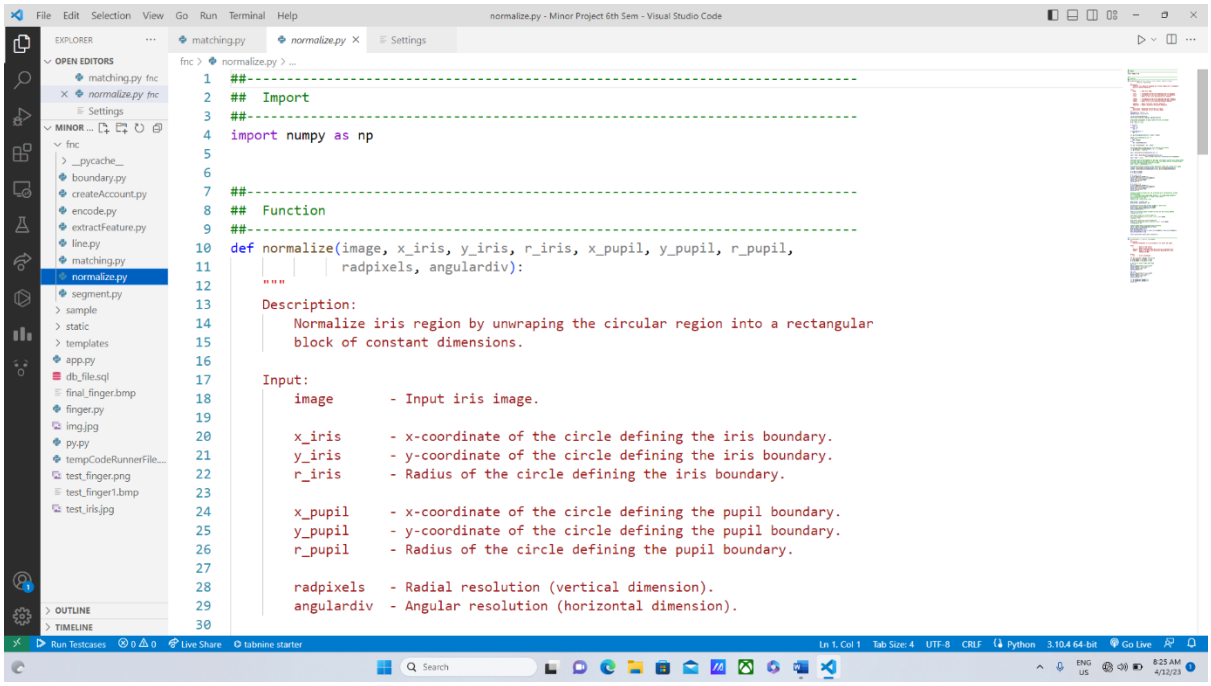
```
def shiftbits(template, noshifts):
    """
    Description:
        Shift the bit-wise iris patterns.

    Input:
        template    - The template to be shifted.
        noshifts    - The number of shift operators, positive for right
                      direction and negative for left direction.

    Output:
        templatenew - The shifted template.
    """
    # Initialize
    templatenew = np.zeros(template.shape)
    width = template.shape[1]
    s = 2 * np.abs(noshifts)
    p = width - s

    # Shift
    if noshifts == 0:
        templatenew = template

    elif noshifts < 0:
        x = np.arange(p)
        templatenew[:, x] = template[:, s + x]
        x = np.arange(p, width)
        templatenew[:, x] = template[:, x - p]

    else:
```



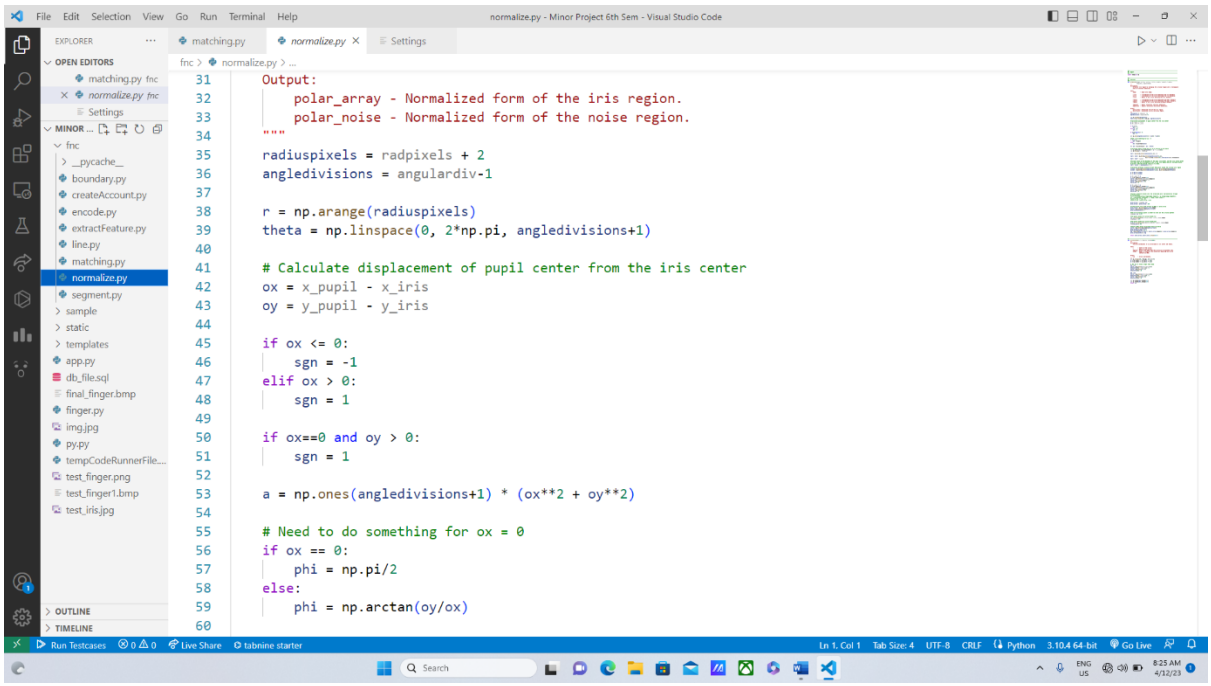
```
            x = np.arange(s, width)
            templatenew[:, x] = template[:, x - s]
            x = np.arange(s)
            templatenew[:, x] = template[:, p + x]

    # Return
    return templatenew


#------------------------------------------------------------------------------
def matchingPool(file_temp_name, template_extr, mask_extr, temp_dir):
    """
    Description:
        Perform matching session within a Pool of parallel computation

    Input:
        file_temp_name  - File name of the examining template
        template_extr   - Extracted template
        mask_extr       - Extracted mask of noise

    Output:
        hm_dist         - Hamming distance
    """
    # Load each account
    data_template = sio.loadmat('%s%s'% (temp_dir, file_temp_name))
    template = data_template['template']
    mask = data_template['mask']

    # Calculate the Hamming distance
    hm_dist = calHammingDist(template_extr, mask_extr, template, mask)
```



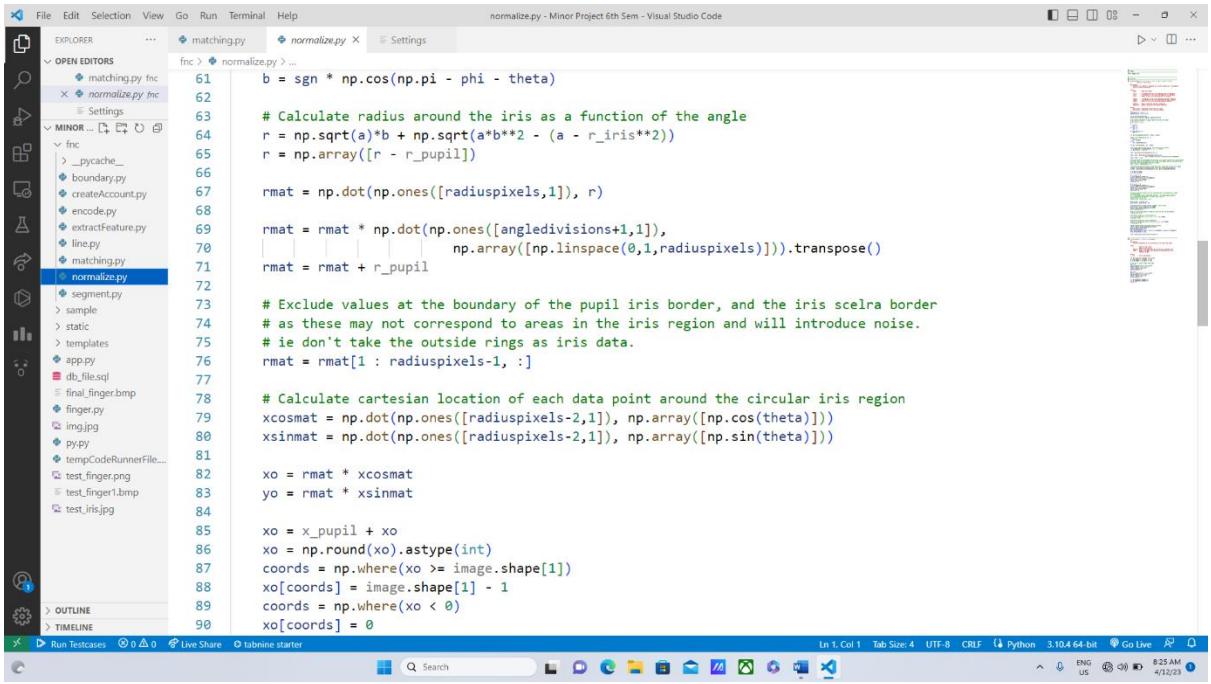
```
    # print(file_temp_name)
    # print(hm_dist)
    # print(file_temp_name, hm_dist)
    return (file_temp_name, hm_dist)
```



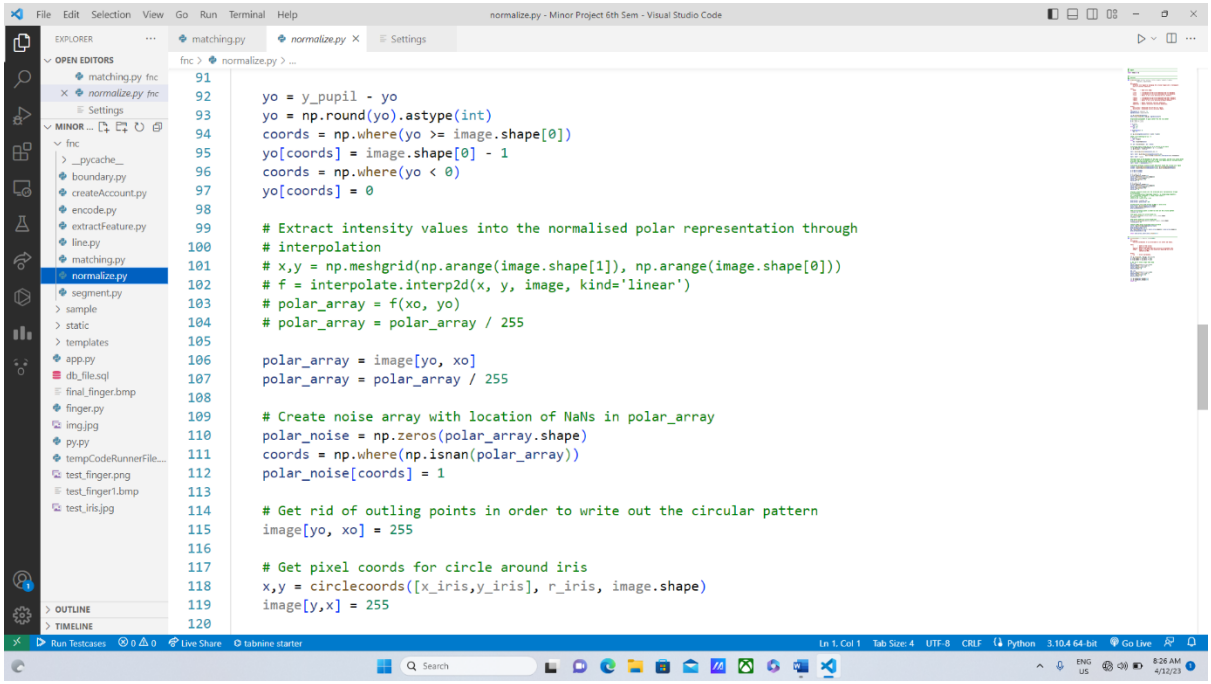

```python
##-------------------------------------------------------------------------
##  Import
##-------------------------------------------------------------------------
import numpy as np


##-------------------------------------------------------------------------
##  Function
##-------------------------------------------------------------------------
def normalize(image, x_iris, y_iris, r_iris, x_pupil, y_pupil, r_pupil,
              radpixels, angulardiv):
    """
    Description:
        Normalize iris region by unwraping the circular region into a rectangular
        block of constant dimensions.

    Input:
        image       - Input iris image.

        x_iris      - x-coordinate of the circle defining the iris boundary.
        y_iris      - y-coordinate of the circle defining the iris boundary.
        r_iris      - Radius of the circle defining the iris boundary.

        x_pupil     - x-coordinate of the circle defining the pupil boundary.
        y_pupil     - y-coordinate of the circle defining the pupil boundary.
        r_pupil     - Radius of the circle defining the pupil boundary.

        radpixels   - Radial resolution (vertical dimension).
        angulardiv  - Angular resolution (horizontal dimension).
```



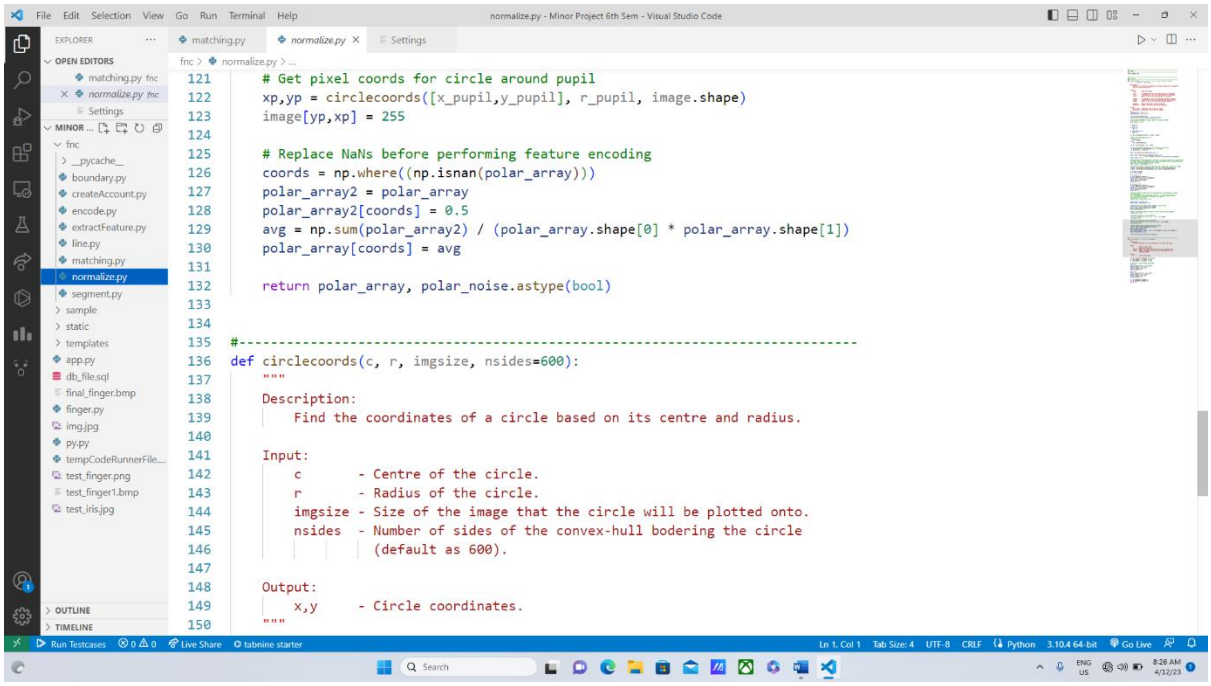

```python
    Output:
        polar_array - Normalized form of the iris region.
        polar_noise - Normalized form of the noise region.
    """
    radiuspixels = radpixels + 2
    angledivisions = angulardiv-1

    r = np.arange(radiuspixels)
    theta = np.linspace(0, 2*np.pi, angledivisions+1)

    # Calculate displacement of pupil center from the iris center
    ox = x_pupil - x_iris
    oy = y_pupil - y_iris

    if ox <= 0:
        sgn = -1
    elif ox > 0:
        sgn = 1

    if ox==0 and oy > 0:
        sgn = 1

    a = np.ones(angledivisions+1) * (ox**2 + oy**2)

    # Need to do something for ox = 0
    if ox == 0:
        phi = np.pi/2
    else:
        phi = np.arctan(oy/ox)
```



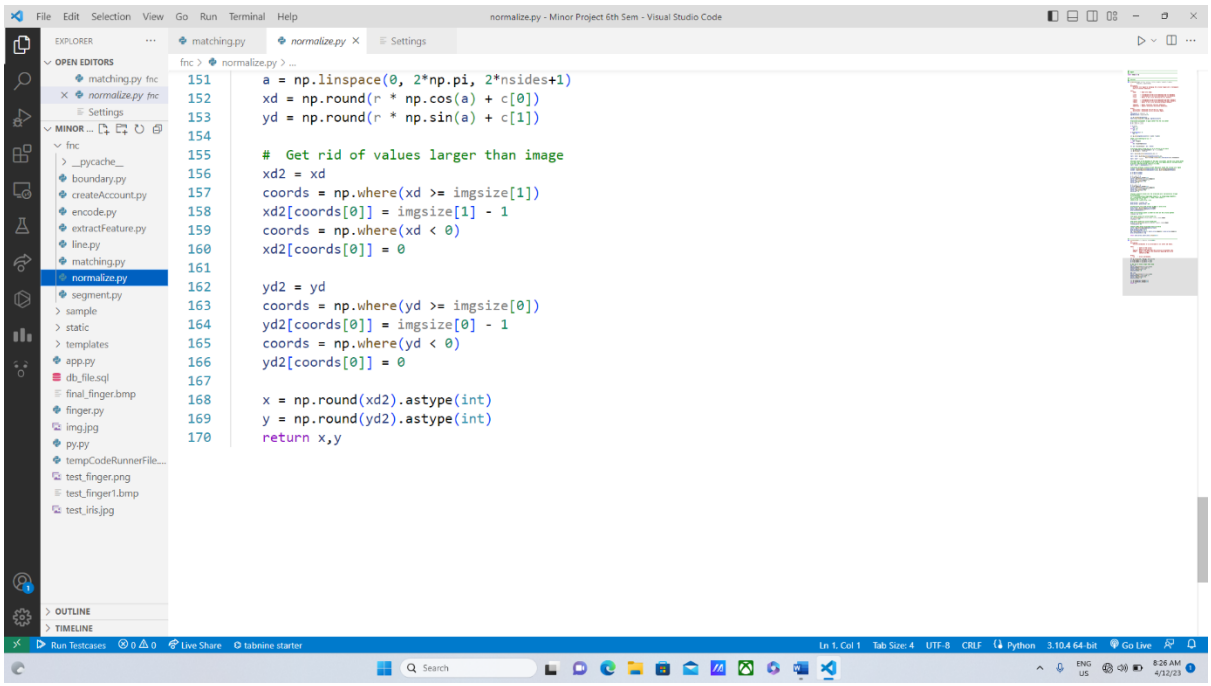
```
    b = sgn * np.cos(np.pi - phi - theta)

    # Calculate radius around the iris as a function of the angle
    r = np.sqrt(a)*b + np.sqrt(a*b**2 - (a - r_iris**2))
    r = np.array([r - r_pupil])

    rmat = np.dot(np.ones([radiuspixels,1]), r)

    rmat = rmat * np.dot(np.ones([angledivisions+1,1]),
                         np.array([np.linspace(0,1,radiuspixels)])).transpose()
    rmat = rmat + r_pupil

    # Exclude values at the boundary of the pupil iris border, and the iris scelra border
    # as these may not correspond to areas in the iris region and will introduce noise.
    # ie don't take the outside rings as iris data.
    rmat = rmat[1 : radiuspixels-1, :]

    # Calculate cartesian location of each data point around the circular iris region
    xcosmat = np.dot(np.ones([radiuspixels-2,1]), np.array([np.cos(theta)]))
    xsinmat = np.dot(np.ones([radiuspixels-2,1]), np.array([np.sin(theta)]))

    xo = rmat * xcosmat
    yo = rmat * xsinmat

    xo = x_pupil + xo
    xo = np.round(xo).astype(int)
    coords = np.where(xo >= image.shape[1])
    xo[coords] = image.shape[1] - 1
    coords = np.where(xo < 0)
    xo[coords] = 0
```



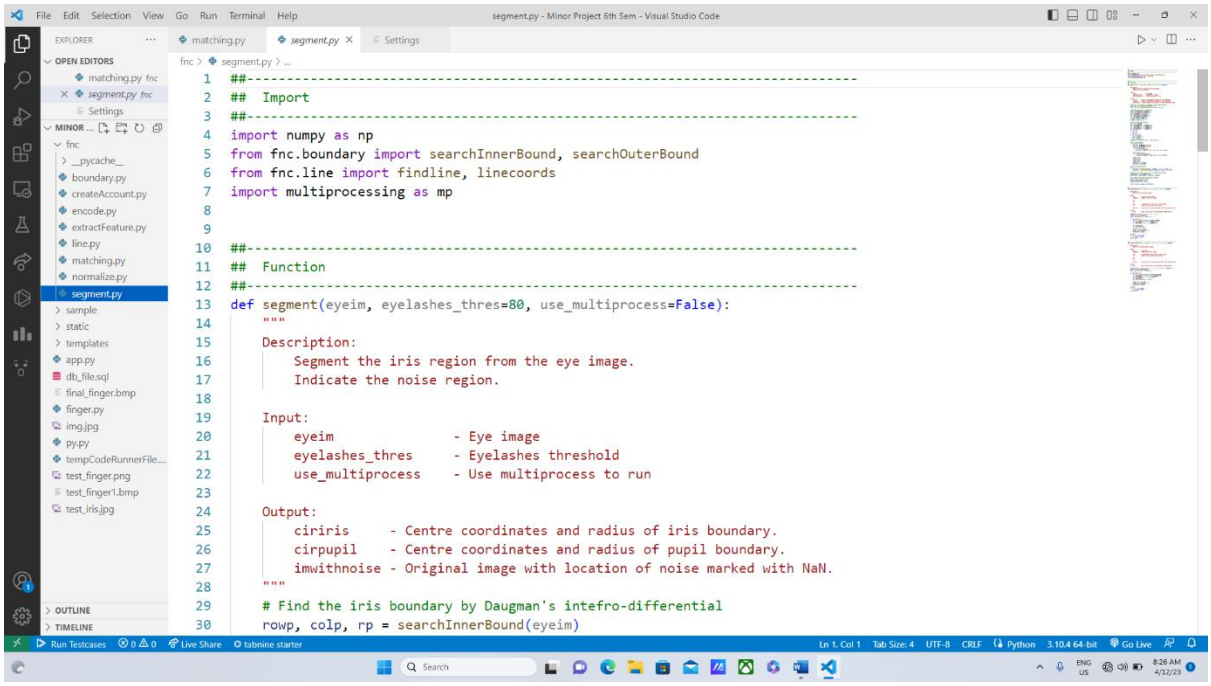
```
    yo = y_pupil - yo
    yo = np.round(yo).astype(int)
    coords = np.where(yo >= image.shape[0])
    yo[coords] = image.shape[0] - 1
    coords = np.where(yo < 0)
    yo[coords] = 0

    # Extract intensity values into the normalised polar representation through
    # interpolation
    # x,y = np.meshgrid(np.arange(image.shape[1]), np.arange(image.shape[0]))
    # f = interpolate.interp2d(x, y, image, kind='linear')
    # polar_array = f(xo, yo)
    # polar_array = polar_array / 255

    polar_array = image[yo, xo]
    polar_array = polar_array / 255

    # Create noise array with location of NaNs in polar_array
    polar_noise = np.zeros(polar_array.shape)
    coords = np.where(np.isnan(polar_array))
    polar_noise[coords] = 1

    # Get rid of outling points in order to write out the circular pattern
    image[yo, xo] = 255

    # Get pixel coords for circle around iris
    x,y = circlecoords([x_iris,y_iris], r_iris, image.shape)
    image[y,x] = 255
```



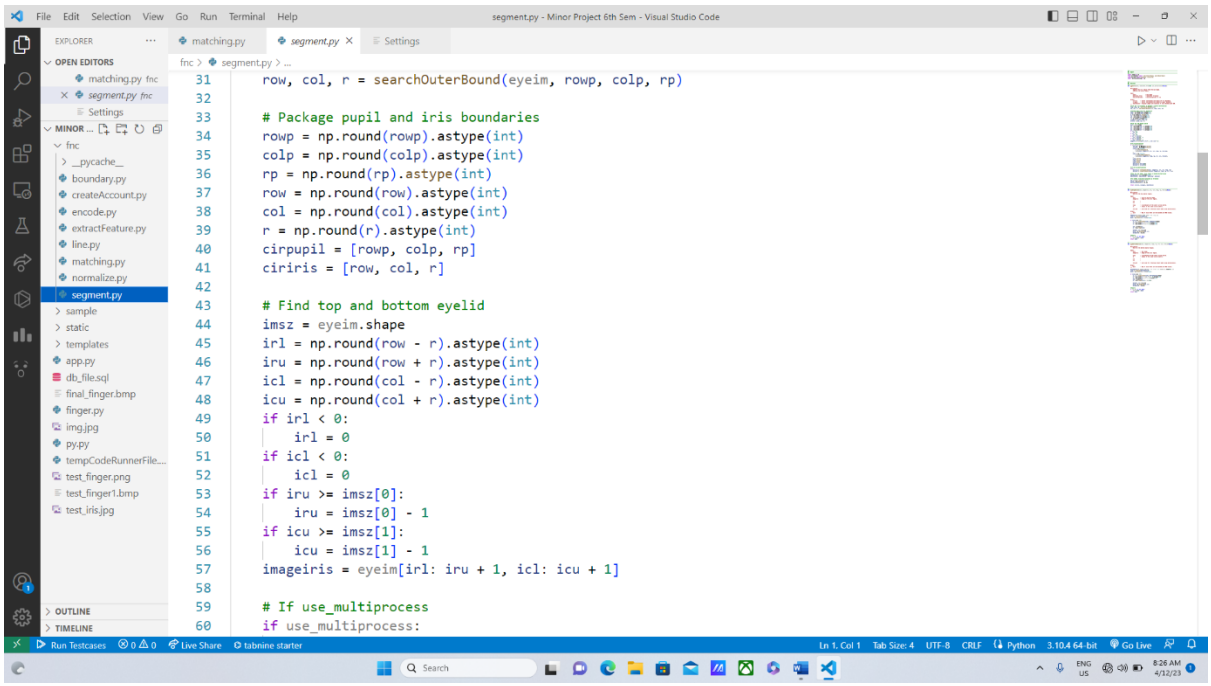
```
    # Get pixel coords for circle around pupil
    xp,yp = circlecoords([x_pupil,y_pupil], r_pupil, image.shape)
    image[yp,xp] = 255

    # Replace NaNs before performing feature encoding
    coords = np.where((np.isnan(polar_array)))
    polar_array2 = polar_array
    polar_array2[coords] = 0.5
    avg = np.sum(polar_array2) / (polar_array.shape[0] * polar_array.shape[1])
    polar_array[coords] = avg

    return polar_array, polar_noise.astype(bool)


#--------------------------------------------------------------------------
def circlecoords(c, r, imgsize, nsides=600):
    """
    Description:
        Find the coordinates of a circle based on its centre and radius.

    Input:
        c       - Centre of the circle.
        r       - Radius of the circle.
        imgsize - Size of the image that the circle will be plotted onto.
        nsides  - Number of sides of the convex-hull bodering the circle
                  (default as 600).

    Output:
        x,y     - Circle coordinates.
    """
```



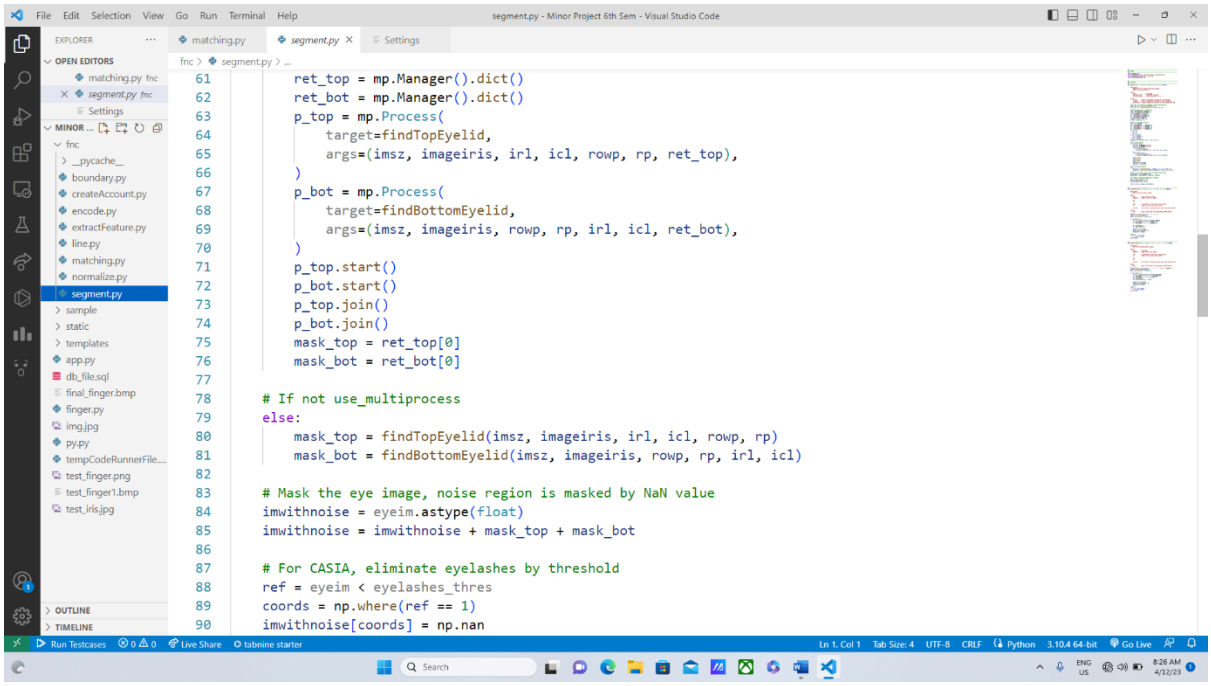
```
    a = np.linspace(0, 2*np.pi, 2*nsides+1)
    xd = np.round(r * np.cos(a) + c[0])
    yd = np.round(r * np.sin(a) + c[1])

    #  Get rid of values larger than image
    xd2 = xd
    coords = np.where(xd >= imgsize[1])
    xd2[coords[0]] = imgsize[1] - 1
    coords = np.where(xd < 0)
    xd2[coords[0]] = 0

    yd2 = yd
    coords = np.where(yd >= imgsize[0])
    yd2[coords[0]] = imgsize[0] - 1
    coords = np.where(yd < 0)
    yd2[coords[0]] = 0

    x = np.round(xd2).astype(int)
    y = np.round(yd2).astype(int)
    return x,y
```



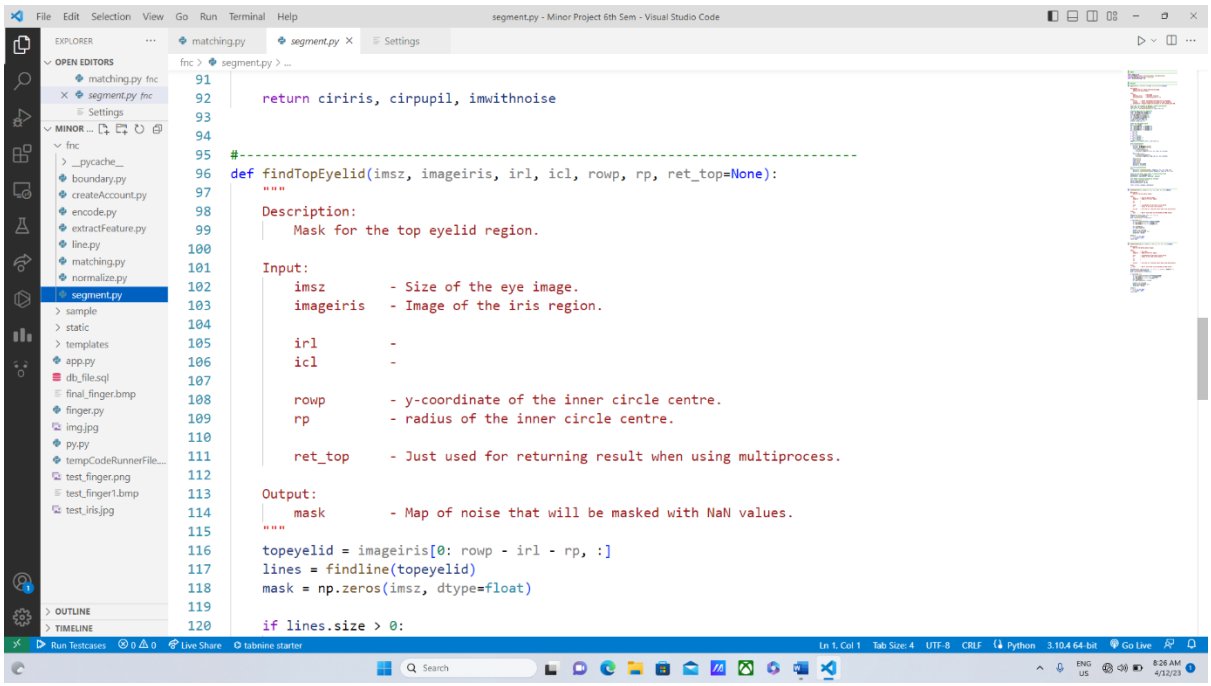
```
##-------------------------------------------------------------------------
##  Import
##-------------------------------------------------------------------------
import numpy as np
from fnc.boundary import searchInnerBound, searchOuterBound
from fnc.line import findline, linecoords
import multiprocessing as mp


##-------------------------------------------------------------------------
##  Function
##-------------------------------------------------------------------------
def segment(eyeim, eyelashes_thres=80, use_multiprocess=False):
    """
    Description:
        Segment the iris region from the eye image.
        Indicate the noise region.

    Input:
        eyeim              - Eye image
        eyelashes_thres    - Eyelashes threshold
        use_multiprocess   - Use multiprocess to run

    Output:
        ciriris     - Centre coordinates and radius of iris boundary.
        cirpupil    - Centre coordinates and radius of pupil boundary.
        imwithnoise - Original image with location of noise marked with NaN.
    """
    # Find the iris boundary by Daugman's intefro-differential
    rowp, colp, rp = searchInnerBound(eyeim)
```



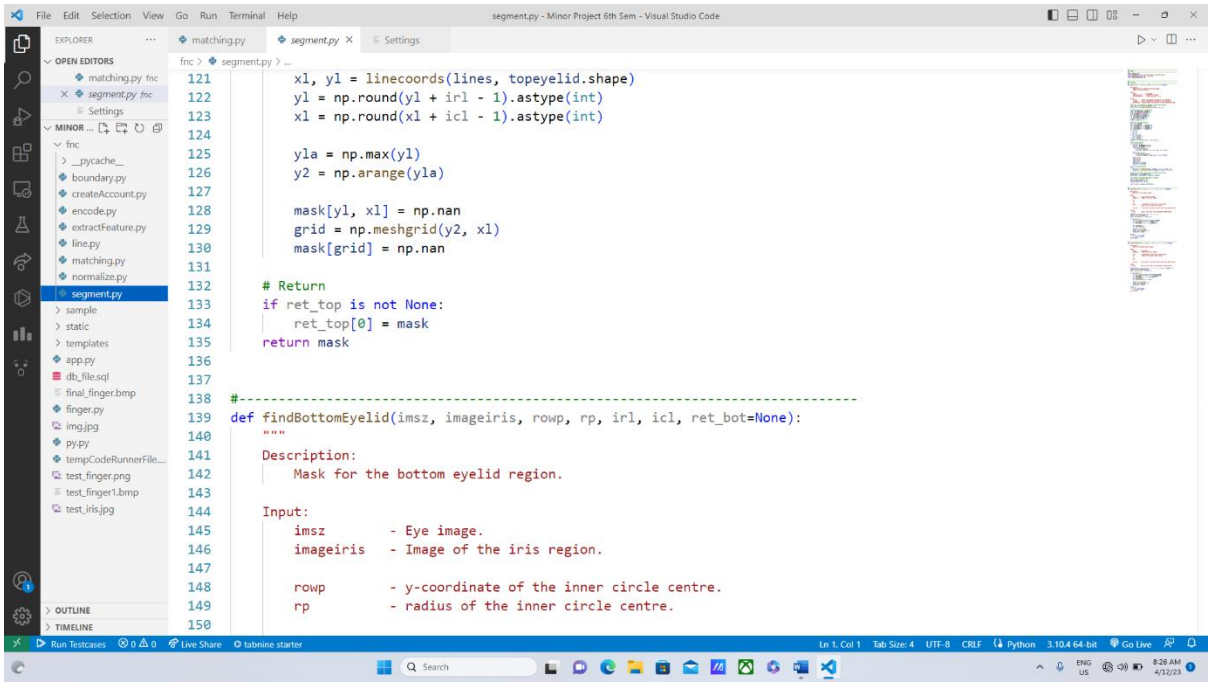
```
    row, col, r = searchOuterBound(eyeim, rowp, colp, rp)

    # Package pupil and iris boundaries
    rowp = np.round(rowp).astype(int)
    colp = np.round(colp).astype(int)
    rp = np.round(rp).astype(int)
    row = np.round(row).astype(int)
    col = np.round(col).astype(int)
    r = np.round(r).astype(int)
    cirpupil = [rowp, colp, rp]
    ciriris = [row, col, r]

    # Find top and bottom eyelid
    imsz = eyeim.shape
    irl = np.round(row - r).astype(int)
    iru = np.round(row + r).astype(int)
    icl = np.round(col - r).astype(int)
    icu = np.round(col + r).astype(int)
    if irl < 0:
        irl = 0
    if icl < 0:
        icl = 0
    if iru >= imsz[0]:
        iru = imsz[0] - 1
    if icu >= imsz[1]:
        icu = imsz[1] - 1
    imageiris = eyeim[irl: iru + 1, icl: icu + 1]

    # If use_multiprocess
    if use_multiprocess:
```



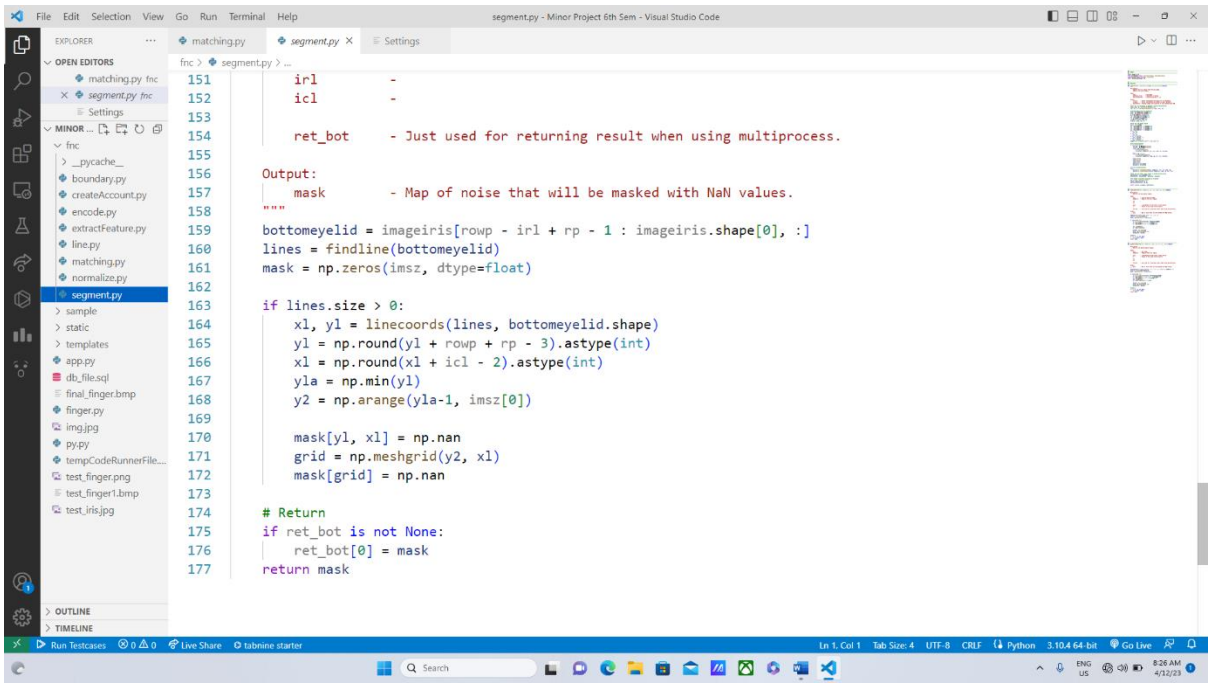
```
61          ret_top = mp.Manager().dict()
62          ret_bot = mp.Manager().dict()
63          p_top = mp.Process(
64              target=findTopEyelid,
65              args=(imsz, imageiris, irl, icl, rowp, rp, ret_top),
66          )
67          p_bot = mp.Process(
68              target=findBottomEyelid,
69              args=(imsz, imageiris, rowp, rp, irl, icl, ret_bot),
70          )
71          p_top.start()
72          p_bot.start()
73          p_top.join()
74          p_bot.join()
75          mask_top = ret_top[0]
76          mask_bot = ret_bot[0]
77
78      # If not use_multiprocess
79      else:
80          mask_top = findTopEyelid(imsz, imageiris, irl, icl, rowp, rp)
81          mask_bot = findBottomEyelid(imsz, imageiris, rowp, rp, irl, icl)
82
83      # Mask the eye image, noise region is masked by NaN value
84      imwithnoise = eyeim.astype(float)
85      imwithnoise = imwithnoise + mask_top + mask_bot
86
87      # For CASIA, eliminate eyelashes by threshold
88      ref = eyeim < eyelashes_thres
89      coords = np.where(ref == 1)
90      imwithnoise[coords] = np.nan
```



```
91
92      return ciriris, cirpupil, imwithnoise
93
94
95  #--------------------------------------------------------------------------
96  def findTopEyelid(imsz, imageiris, irl, icl, rowp, rp, ret_top=None):
97      """
98      Description:
99          Mask for the top eyelid region.
100
101     Input:
102         imsz        - Size of the eye image.
103         imageiris   - Image of the iris region.
104
105         irl         -
106         icl         -
107
108         rowp        - y-coordinate of the inner circle centre.
109         rp          - radius of the inner circle centre.
110
111         ret_top     - Just used for returning result when using multiprocess.
112
113     Output:
114         mask        - Map of noise that will be masked with NaN values.
115     """
116     topeyelid = imageiris[0: rowp - irl - rp, :]
117     lines = findline(topeyelid)
118     mask = np.zeros(imsz, dtype=float)
119
120     if lines.size > 0:
```

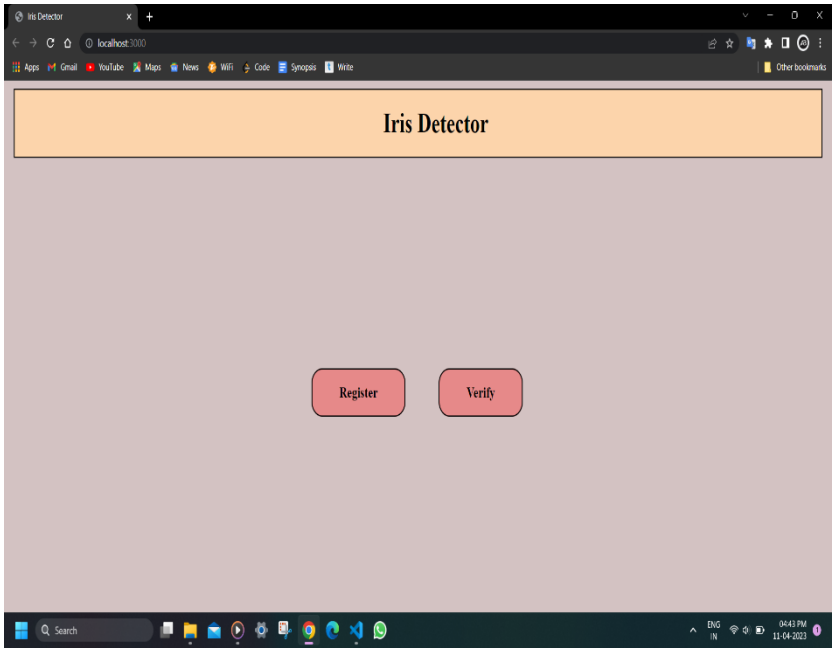


```
        xl, yl = linecoords(lines, topeyelid.shape)
        yl = np.round(yl + irl - 1).astype(int)
        xl = np.round(xl + icl - 1).astype(int)

        yla = np.max(yl)
        y2 = np.arange(yla)

        mask[yl, xl] = np.nan
        grid = np.meshgrid(y2, xl)
        mask[grid] = np.nan

    # Return
    if ret_top is not None:
        ret_top[0] = mask
    return mask


#---------------------------------------------------------------------------
def findBottomEyelid(imsz, imageiris, rowp, rp, irl, icl, ret_bot=None):
    """
    Description:
        Mask for the bottom eyelid region.

    Input:
        imsz        - Eye image.
        imageiris   - Image of the iris region.

        rowp        - y-coordinate of the inner circle centre.
        rp          - radius of the inner circle centre.
```



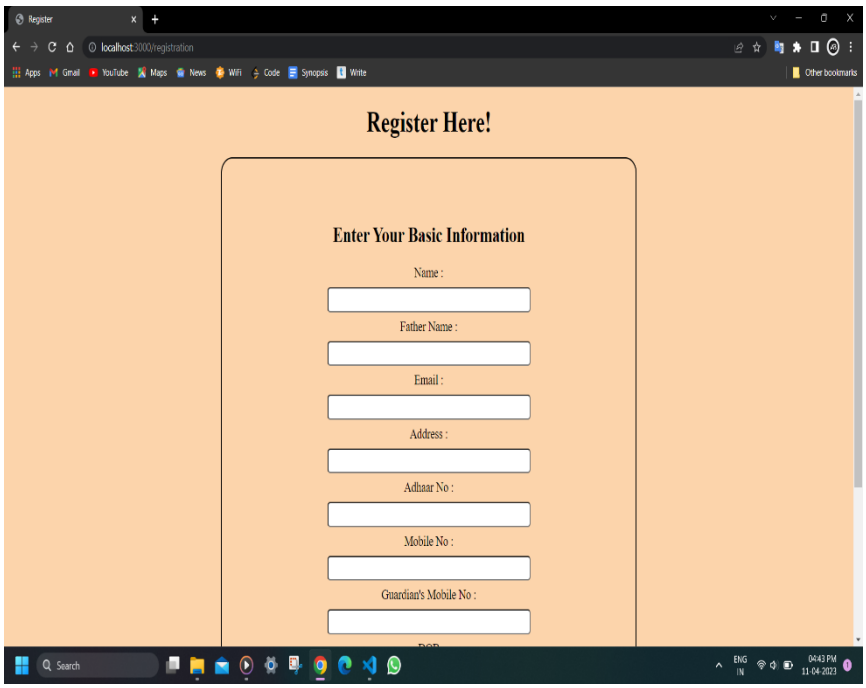
```
        irl         -
        icl         -

        ret_bot     - Just used for returning result when using multiprocess.

    Output:
        mask        - Map of noise that will be masked with NaN values.
    """
    bottomeyelid = imageiris[rowp - irl + rp - 1 : imageiris.shape[0], :]
    lines = findline(bottomeyelid)
    mask = np.zeros(imsz, dtype=float)

    if lines.size > 0:
        xl, yl = linecoords(lines, bottomeyelid.shape)
        yl = np.round(yl + rowp + rp - 3).astype(int)
        xl = np.round(xl + icl - 2).astype(int)
        yla = np.min(yl)
        y2 = np.arange(yla-1, imsz[0])

        mask[yl, xl] = np.nan
        grid = np.meshgrid(y2, xl)
        mask[grid] = np.nan

    # Return
    if ret_bot is not None:
        ret_bot[0] = mask
    return mask
```

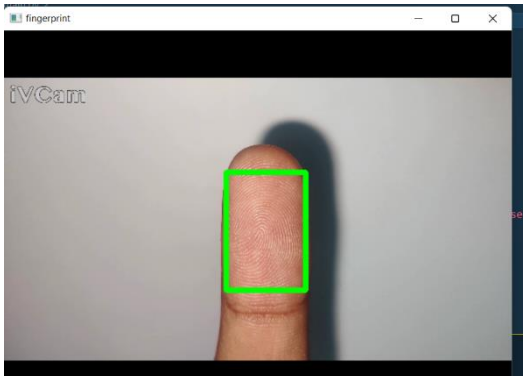
For registration: -



Home page.

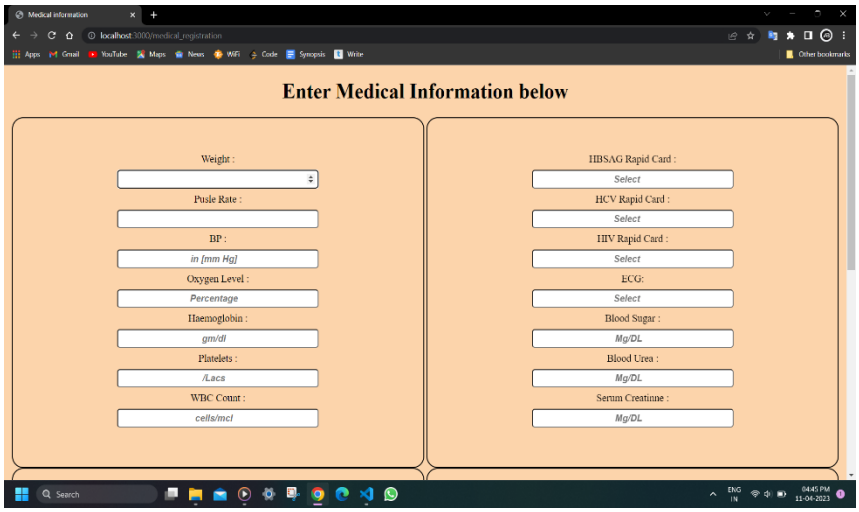
Basic Detail for registration



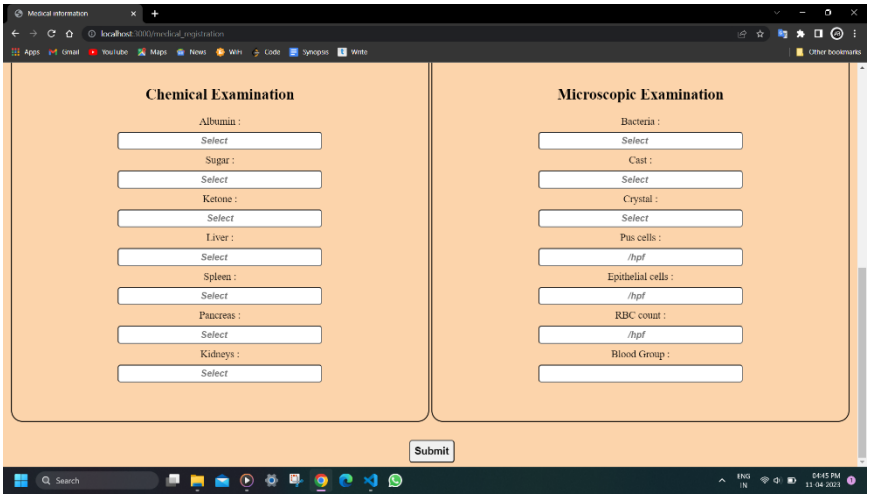


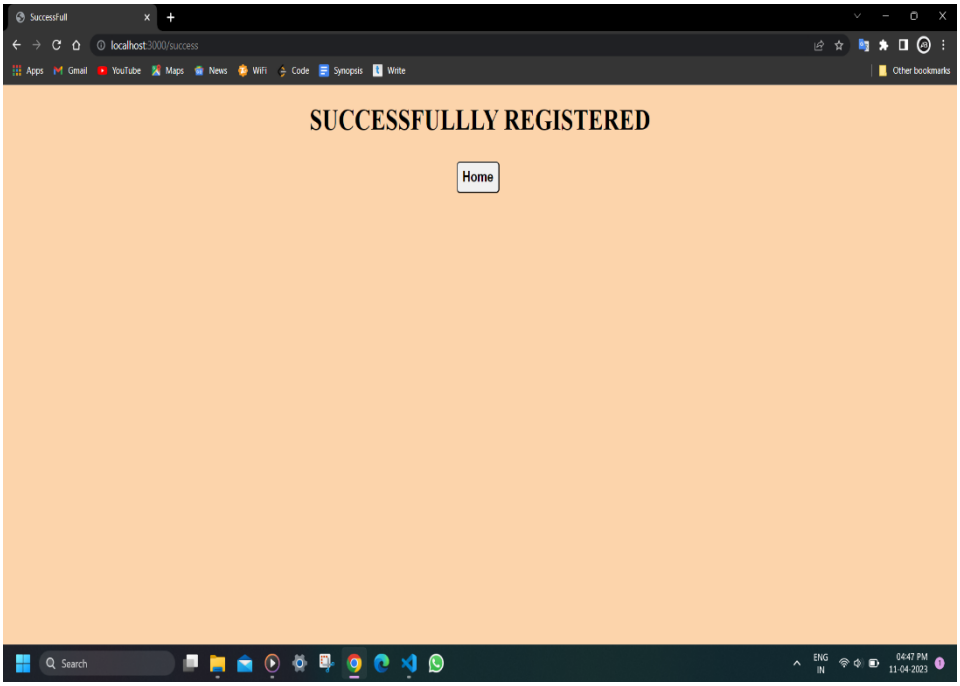
Iris enrollment

Fingerprint enrollment



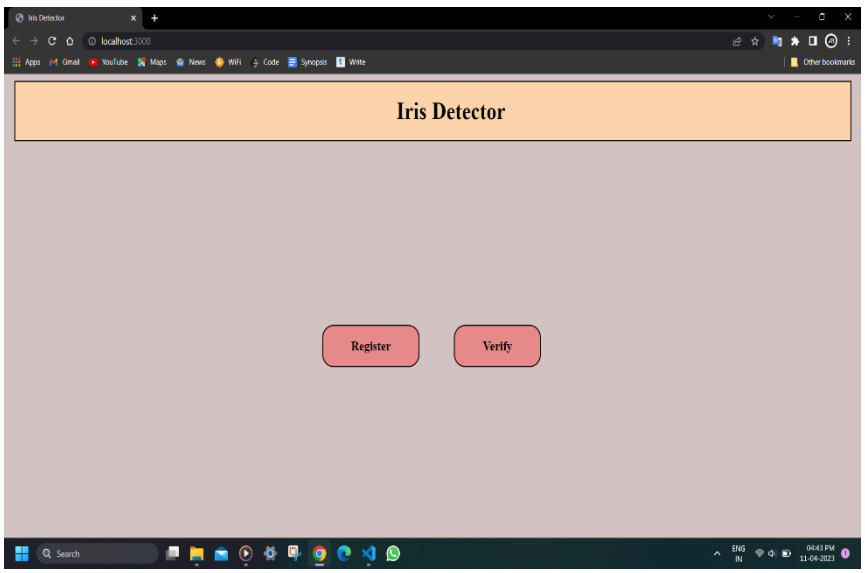
Medical History





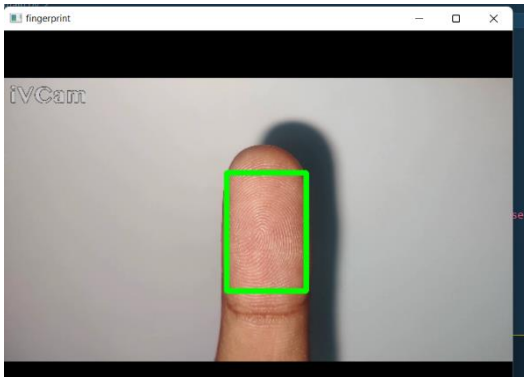
Successfully Registered

For verification: -



Home page.

Verification page

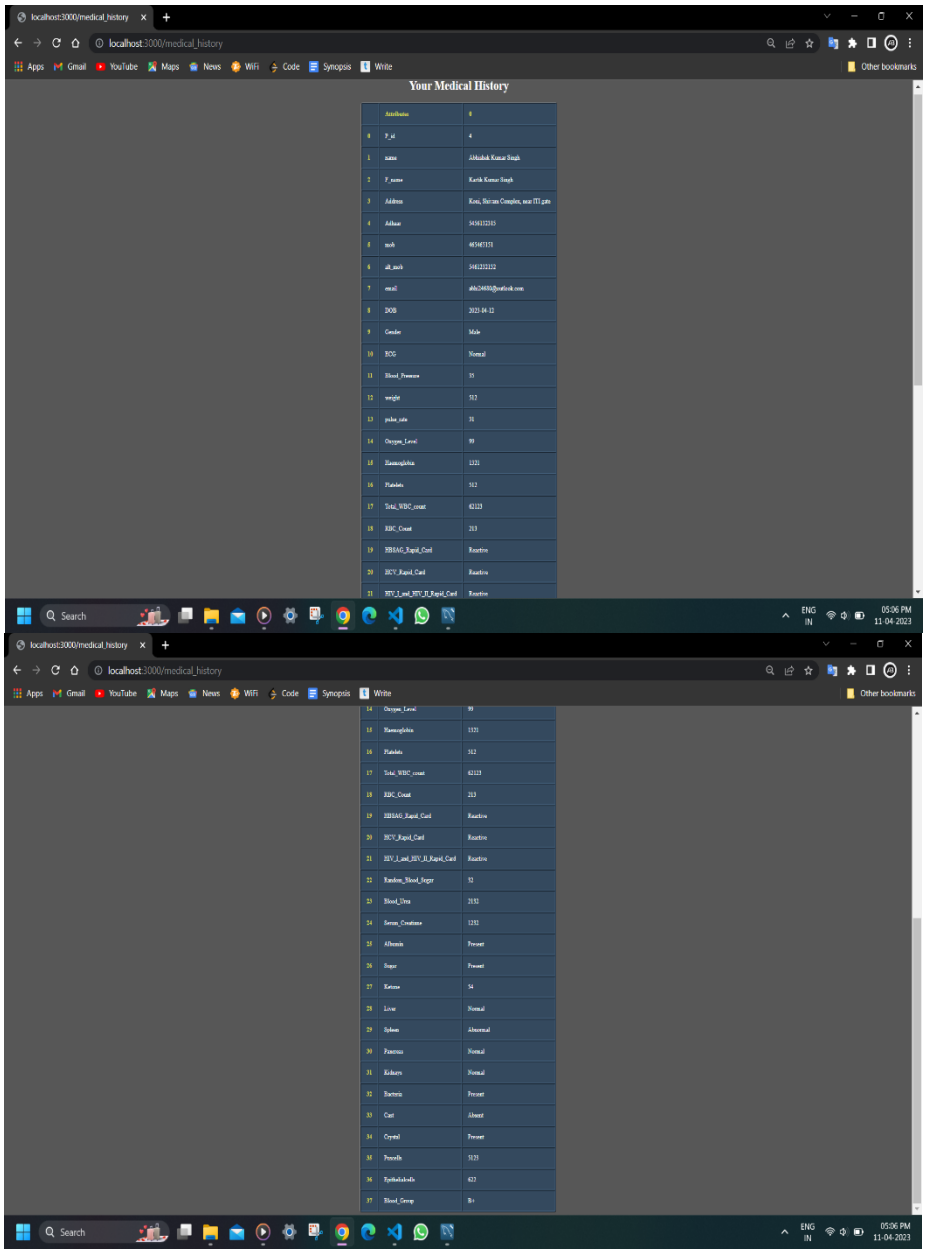


Iris verification

Fingerprint verification

After successful verification

Medical history of User



Conclusion and Future work

In our project we implanted iris recognition and fingerprint recognition for the verification of user. In iris recognition we can conclude that the accuracy of iris recognition is based on the quality of image taken from camera. For the better accuracy it is suggested to use high quality infrared camera. And for fingerprint verification accuracy can be increased by adding different image enhancement algorithms. Our prototype is working fine with basic features included in it.

In future we can add many features like registering doctors so that patients can directly take consultant from the doctor without going to their clinic. We can also add features to upload the reports of various tests so that the evaluation of patients becomes easy for the doctors.

References

[1].Parthasarathi De and Dibyendu Ghoshal (2016), National Institute of Technology, Agartala 799 046 “Human Iris Recognition for Clean Electoral Process in India by Creating a Fraud Free Voter Registration List” Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016) Procedia Computer Science 89 (2016) 850 – 855 © 2016 The Authors. Published by Elsevier B. V. DOI: 10.1016/j.procs.2016.06.071

[2].Ghanapriya Singha, Ratnesh Kumar Singha, Rajesh Sahaa, Nidhi Agarwal, (2020) “IWT Based Iris Recognition for Image Authentication” Third International Conference on Computing and Network Communications (CoCoNet’19) Procedia Computer Science 171 (2020) 1868–1876© 2016 The Authors. Published by Elsevier B. V. DOI: 10.1016/j.procs.2020.04.200

[3].M. Dobes, L. Machala, P. Tichavsky, J. Pospisil (2004) “Human eye iris recognition using the mutual information” International Journal for Light and Electron Optics, Optik 115, No. 9 (2004) 399–404, 0030-4026/04/115/09-399 $ 30.00/0

[4].Amena Khatun, A. K. M. Fazlul Haque, Sabbir Ahmed, Mohammad Mahfujur Rahman (2015) “Design and Implementation of Iris Recognition Based Attendance Management System” 2nd Int'l Conf. on Electrical Engineering and Information & Communication Technology (ICEEICT) 2015 Jahangirnagar University, Dhaka-1342, Bangladesh, 21-23 May 2015, 978-1-4673-6676-2115/$31.00 ©2015 IEEE

[5].Vahid Nazmdeh, Shaghayegh Mortazavi, Daniel Tajeddin, Hossein Nazmdeh, Morteza Modarresi Asem (2019) “Iris Recognition; From Classic to Modern Approaches” 978-1-7281-0554-3/19/$31.00 © 2019 IEEE

[6].Saša Adamović , Vladislav Miškovic , Nemanja Maček, Milan Milosavljević , Marko Šarac , Muzafer Saračević, Milan Gnjatović (2020) “An efficient novel approach for iris recognition based on stylometric features and machine learning techniques” Future Generation Computer Systems 107 (2020) 144–157, 0167-739X/© 2020 Elsevier B.V.

[7].R.M. Farouk, Department of Mathematics, Faculty of Science, Zagazig University, P.O. Box 44519, Zagazig, Egypt (2011) “Iris recognition based on elastic graph matching

and Gabor wavelets" Computer Vision and Image Understanding 115 (2011) 1239–1244, 1077-3142/$ -30.00 2011 © Elsevier Inc. DOI:10.1016/j.cviu.2011.04.002

[8].Ramadan Gad, Muhammad Talha, Ahmed A. Abd El-Latif , M. Zorkany, Ayman EL-SAYED, Nawal EL-Fishawy, Ghulam Muhammad (2018) "Iris Recognition Using Multi-Algorithmic Approaches for Cognitive Internet of things (CIoT) Framework" Future Generation Computer Systems 89 (2018) 178–191, DOI:10.1016/j.future.2018.06.020 0167-739X/© 2018 Elsevier B.V.

[9].Alfi Zuhriya Khoirunnisaa, Lutfi Hakim, Adhi Dharma Wibawa (2019) "The Biometrics System Based on Iris Image Processing: A Review" 2019 2nd International Conference of Computer and Informatics Engineering (IC2IE) 10-11 September, Indonesia-Banyuwangi, East Java, 978-1-7281-2384-4/19/$31.00 ©2019 IEEE

[10].Hasimah Ali, Momoh J. E. Salami, Wahyudi (2008) "Iris Recognition System by Using Support Vector Machines" Proceedings of the International Conference on Computer and Communication Engineering 2008 May 13-15, 2008 Kuala Lumpur, Malaysia 978-1-4244-1692-9/08/$25.00 ©2008 IEEE

[11].Rohini Shelke, Prof. S. B. Bagal (2017) "Iris Recognition System: A Novel Approach For Biometric Authentication" 2017 Third International Conference on Computing, Communication, Control And Automation (ICCUBEA) 978-1-5386-4008-1/17/$31.00 ©2017 IEEE

[12].Kirti Gupta, Rashmi Gupta, Department of Electronics & Communication Engineering Ambedkar Institute of Advanced Communication Technologies & Research, Delhi, India (2014) "Iris Recognition System for Smart Environments" 978-1-4799-4674-7/14/$31.00 ©2014 IEEE

[13].Jagadeesh N, Dr. Chandrasekhar M. Patil (2017) "A Brief Review of the Iris Recognition Systems for Developing a User-Friendly Biometric Application" International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS-2017), 978-1-5386-1887-5/17/$31.00 ©2017 IEEE

[14].Sandeep Patil, Shreya Gudasalamani, Nalini C. Iyer (2016) "A Survey on Iris Recognition System" International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016, 978-1-4673-9939-5/16/$31.00 ©2016 IEEE

[15].Cheng-Shun Hsiao, Chih-Peng Fan, and Yin-Tsung Hwang (2021) "Design and Analysis of Deep-Learning Based Iris Recognition Technologies by Combination of U-

Net and EfficientNet" 2021 9th International Conference on Information and Education Technology, 978-1-6654-1933-8/21/$31.00 ©2021 IEEE, DOI: 10.1109/ICIET51873.2021.9419589

[16].Eri Prasetyo Wibowo, Wisnu Sukma maulana (2009) "Real-Time Iris Recognition System Using A Proposed Method" 2009 International Conference on Signal Processing Systems 978-0-7695-3654-5/09 $25.00 © 2009 IEEE DOI 10.1109/ICSPS.2009.9

[17].Rishmita Saha, Mahasweta Kundu, Madhuparna Dutta, Rahul Majumder, Debosmita Mukherjee, Sayak Pramanik, Uttam Narendra Thakur, Chiradeep Mukherjee, Dipta Mukherjee (2017) "A Brief Study on Evolution of Iris Recognition System" 978-1-5386-3371-7/17/$31.00 ©2017 IEEE

[18].Neha Kak, Rishi Gupta, Sanchit Mahajan (2010) "Iris Recognition System" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 1, No. 1, 2010

[19].Navjot Kaur, Mamta Juneja (2014) "A Review on Iris Recognition" Proceedings of 2014 RAECS UIET Panjab University Chandigarh, 06 – 08 March, 2014, 978-1-4799-2291-8/14/$31.00 ©2014 IEEE

[20].Jinghui Lia, Bairui Tao, Yanchun Wang,Xibing Li (2012) "Research and Implementation of Iris Recognition Algorithm" 2012 International Workshop on Information and Electronics Engineering (IWIEE) © 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of Harbin University of Science and Technology

[21].Xiaonan Liu, Yuchen Bai, Yuwen Luo, Zhengwei Yang, Yan Liu (2019) "Iris recognition in visible spectrum based on multi-layer analogous convolution and collaborative representation" Pattern Recognition Letters 117 (2019) 66–73 © 2018 Elsevier B.V. All rights reserved

[22].Neal S. Latman, Emily Herb (2013) "A field study of the accuracy and reliability of a biometric iris recognition system" Science and Justice 53 (2013) 98–102 © 2012 Forensic Science Society. Published by Elsevier Ireland Ltd. All rights reserved.

[23].Kien Nguyen, Clinton Fookes, Sridha Sridharan, Simon Denman (2013) "Feature-domain super-resolution for iris recognition" Computer Vision and Image Understanding 117 (2013) 1526–1535 © 2013 Elsevier Inc. All rights reserved.

[24].Lubos Omelina , Jozef Goga, Jarmila Pavlovicova Milos Oravec , Bart Jansen (2021) “A survey of iris datasets” Image and Vision Computing 108 (2021) 104109 © 2021 The Author(s). Published by Elsevier B.V.

[25].Padma Polash Paul, Md. Maruf Monwar, Ahsanullah University of Science and Technology, Dhaka, Bangladesh (2007) “Human Iris Recognition for Biometric Identification” 1-4244-1551-9/07/$25.00 ©2007 IEEE.

[26].Fadi N. Sibai, Hafsa I. Hosani, Raja M. Naqbi, Salima Dhanhani, Shaikha Shehhi (2011) “Iris recognition using artificial neural networks” Expert Systems with Applications 38 (2011) 5940–5946 © 2010 Elsevier Ltd. All rights reserved

[27].Xiaoshuang Li, Shenhua Hu, Chengkan Lv, Haojun Qin, Fenghua Zhu, Feiyue Wang (2019) “A Brief Research and Lookback of the Development of Iris Recognition” supported by National Key R&D Program of China 2018YFB1700200, National Natural Science Foundation of China U1811463, 61533019, 61773381 and 91720000, the Beijing Municipal Science and Technology Commission Program under Grant D171100000317002/ 978-1-7281-2853-5/19/$31.00©2019 IEEE

[28].John Daugman (2004) “How Iris Recognition Works” IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 14, NO. 1, JANUARY 2004 1051-8215/04$20.00 © 2004 IEEE

[29].Yuan Zhuang, Joon Huang Chuah, Chee Onn Chow and Marcus Guozong Lim (2020) “Iris Recognition Using Convolutional Neural Network” 2020 IEEE 10th International Conference on System Engineering and Technology (ICSET), 9 November 2020, Shah Alam, Malaysia 978-1-7281-9910-8/20/$31.00 ©2020 IEEE

[30].Gerald 0. Williams (1996) “Iris Recognition Technology” 0-7803-3537-6-9/96/$4.00 ©1996 IEEE

[31].Shimaa M. Elsherief, Mahmoud E. Allam, Mohamed W. Fakhr (2006) “Biometric Personal Identification Based on Iris Recognition” 1-4244-0272-7/06/$20.00 ©2006 IEEE

[32].Maulisa Oktiana, Takahiko Horiuchi, Keita Hirai, Khairun Saddami, Fitri Arnia, Yuwaldi Away, Khairul Munadi (2020) “Cross-spectral iris recognition using phase-based matching and homomorphic filtering” Heliyon 6 (2020) e03407 DOI:10.1016/j.heliyon.2020.e03407 2405-8440/© 2020 Published by Elsevier Ltd.

[33].XiaoZhou Chen, Fan Yang, ChangYin Wu, LiangLin Xiong (2012) “A Novel Unified-Noise Algorithm for Iris Recognition” 2012 International Conference on Future Energy,

Environment, and Materials DOI :10.1016/j.egypro.2012.01.139 1876-6102 © 2011 Published by Elsevier B.V

[34].Qingqiao Hu, Siyang Yin, Huiyang Ni, Yisiyuan Huang (2019) "An End-to-End Deep Neural Network for Iris Recognition" 2019 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI2019) Procedia Computer Science 174 (2020) 505–517 DOI: 10.1016/j.procs.2020.06.118 1877-0509 © 2020 The Authors. Published by Elsevier B.V

[35].Niranjan C. Kundur & M.R. Prasad, Dr. T.C. Manjunath (2018) "Iris recognition systems – A review" Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS 2018) IEEE Xplore Compliant Part Number: CFP18K74-ART; ISBN:978-1-5386-2842-3 978-1-5386-2842-3/18/$31.00 ©2018 IEEE

[36].Kamal Hajari, Ujwalla Gawande, Yogesh Golhar (2016) "Neural Network Approach to Iris Recognition in Noisy Environment" International Conference on Information Security & Privacy (ICISP2015), DOI: 10.1016/j.procs.2016.02.11611-12 December 2015, Nagpur, INDIA © 2016 The Authors. Published by Elsevier B.V.

[37].Izem Hamouchene, Saliha Aouat (2014) "A New Texture Analysis Approach for Iris Recognition" 2014 AASRI Conference on Circuit and Signal Processing (CSP 2014) © 2014 The Authors. Published by Elsevier B.V.

[38].D. Harikrishnan, N. Sunilkumar, Joseph Shelby, Nair Kishor, Gopalakrishnan Remya (2023) "An effective authentication scheme for a secured IRIS recognition system based on a novel encoding technique" DOI: 10.1016/j.measen.2022.100626 2665-9174/© 2022 The Authors. Published by Elsevier Ltd.

[39].Kathleen M. Brelsford, Susan E. Spratt and Laura M. Beskow (2018) "Research use of electronic health records: patients' perspectives on contact by researchers" Journal of the American Medical Informatics Association, 25(9), Vol. 25, No. 9, 2018, 1122–1129 DOI: 10.1093/jamia/ocy087 VC The Author(s) 2018. Published by Oxford University Press on behalf of the American Medical Informatics Association.

[40].Hao Zhu, Mengshu Hou (2018) "Research on an Electronic Medical Record System Based on the Internet" 2018 2nd International Conference on Data Science and Business Analytics DOI: 10.1109/ICDSBA.2018.0010, 978-1-5386-8431-3/18/$31.00 ©2018 IEEE

[41].Richard P. Wildes (1997) “Iris Recognition: An Emerging Biometric Technology” PROCEEDINGS OF THE IEEE, VOL. 85, NO. 9, SEPTEMBER 1997 0018–9219/97$10.00 © 1997 IEEE

[42].Min Beom Lee, Jin Kyu Kang, Hyo Sik Yoon, And Kang Ryoung Park (2021) Division of Electronics and Electrical Engineering, Dongguk University, Seoul 04620, South Korea “Enhanced Iris Recognition Method by Generative Adversarial Network-Based Image Reconstruction” Received December 16, 2020, accepted January 6, 2021, date of publication January 11, 2021, date of current version January 20, 2021. Digital Object Identifier 10.1109/ACCESS.2021.3050788 ©2021 IEEE

[43].Abhishek Gangwar, Akanksha Joshi, Centre for Development of Advanced Computing (CDAC), India (2016) “DeepIrisNet: DEEP IRIS REPRESENTATION WITH APPLICATIONS IN IRIS RECOGNITION AND CROSS-SENSOR IRIS RECOGNITION” ICIP 2016 978-1-4673-9961-6/$31.00 ©2016 IEEE

[44].Shirke, Swati D.; RajaBhushnam, C. (2017) “Review of IRIS Recognition Techniques” 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET) - Review of IRIS recognition techniques, (), 1–5. doi:10.1109/ICAMMAET.2017.8186651 ©2017 IEEE

[45].Halim, Robby Alphonsus; Wahju Rahardjo Emanuel, Andi (2020) “A Review of Iris Recognition System ROI and Accuracy” 2020 International Conference on Smart Technology and Applications (ICoSTA) - A Review of Iris Recognition System ROI and Accuracy, (), 1–6. doi:10.1109/ICoSTA48221.2020.1570615087 ©2020 IEEE

[46].Trokielewicz, Mateusz; Czajka, Adam; Maciejewicz, Piotr (2018) “Iris Recognition After Death” IEEE Transactions on Information Forensics and Security, (), 1–1. doi:10.1109/TIFS.2018.2881671 ©2018 IEEE

[47].Zhang, Wentao; Jiang, Shaohua; Zhao, Shan; Hou, Kai; Liu, Yang; Zhang, Li (2019) “A Unifying Framework of Mining Trajectory Patterns of Various Temporal Tightness” 2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA)166-169 doi:10.1109/ICICTA49267.2019.00043 ©2019 IEEE

[48].Firas Yaqoob Yousif Alabdullah, Abdullahi Abdu ibrahim (2020) “Iris Detection and Recognition by Image Segmentation Using K-Means Algorithm and Artificial Neural Network” 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), (), –. doi:10.1109/ismsit50672.2020.9255002 ©2020 IEEE

[49].Xue, Wenyuan, Li, Qingyong; Xue, Qiyuan (2020) “Text Detection and Recognition for Images of Medical Laboratory Reports With a Deep Learning Approach” IEEE Access, 8(), 407–416. doi:10.1109/ACCESS.2019.2961964 ©2020 IEEE

[50].Chen, Yangyu; Zhang, Weigang (2018) “Iris Liveness Detection: A Survey” Conference on Multimedia Big Data (BigMM) - Xi'an, China (2018.9.13-2018.9.16)] 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM) ,(), 1–7. doi:10.1109/BigMM.2018.8499061 ©2018 IEEE