

|  |  |  |
| --- | --- | --- |
| **Introduction to Image Processing** | | |
| Lab Manual | | |
| **Department of Computer Science and Engineering**  **The NorthCap University, Gurugram** | | |
|  | |
|  | |
|  | |
|  | |
|  |

**Introduction to Image Processing Lab Manual**

**CSL 316**

**Dr. Ashwini Rahangdale**

**Dr. Shaveta Arora**



Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2020-21

*Published by:*

**School of Engineering and Technology**

**Department of Computer Science & Engineering**

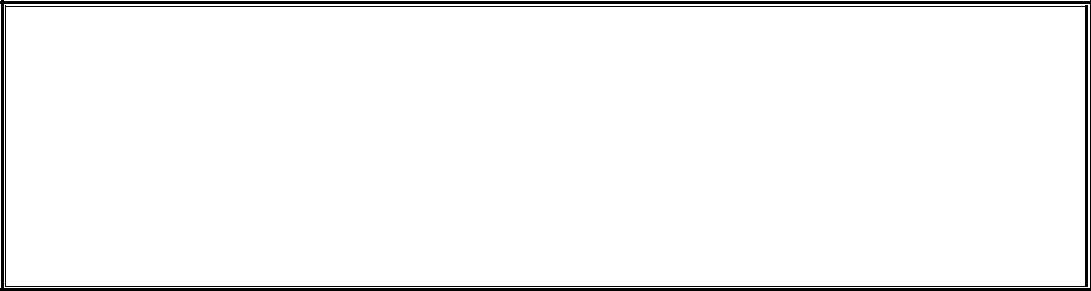
**The NorthCap University Gurugram**

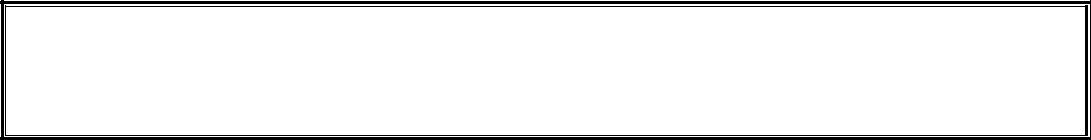
**• Laboratory Manual is for Internal Circulation only**

**© Copyright Reserved**

*No part of this Practical Record Book may be*

*reproduced, used, stored without prior permission of The NorthCap University*



Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

**PREFACE**

**Introduction to Image Processing** Lab Manual is designed to meet the course and program requirements of NCU curriculum for B.Tech third year students of CSE branch. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.

The Lab manual is written on the basis of “teach yourself pattern” and expected that students who come with proper preparation should be able to perform the experiments without any difficulty. Brief introduction to each experiment with information about self-study material is provided. The laboratory exercises will include the introduction to digital image and its visualization through python, arithmetic and bitwise operation on image, spatial and frequency transformation on image for image enhancement, edge detection, morphological operation on image, demonstration of pixel relationship within image, find connected component sets, region and boundary, segmentation techniques and watershed transformation on image. Experimentation also includes mini project based on face and object detection and project related to number, character recognition. Students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.

The lab manual is a part of curriculum for the TheNorthCap University, Gurugram. Teacher’s copy of the experimental results and answer for the questions are available as sample guidelines.

We hope that lab manual would be useful to students of CSE branch and author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

**Authors**

**The NorthCap University**

**Gurugram, India**

|  |  |  |  |
| --- | --- | --- | --- |
| **CONTENTS** | | | |
| S.N. | **Details** | Page No. | |
|  | **Syllabus** |  | |
| 1 | **Introduction** |  | |
| 2 | **Lab Requirement** |  |
| 3 | **General Instructions** |  |
| 4 | **List of Experiments** |  |
| 5 | **List of Flip Assignment** |  |
| 6 | **List of Projects** |  |
| 7 | **Rubrics** |  |
| 8 | **Annexure 1 (Format of Lab Report)** |  |
| 9 | **Annexure 2 (Format of Project Report)** |  |

**SYLLABUS**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **Department:** | | | | Department of Computer Science and Engineering | | | | | |
| 1. **Course Name:**   **Introduction to Image Processing and**  **Recognition** | | | | | | 1. **Course Code** | 1. **L-T-P** | | 1. **Credits** |
| CSL316 | 3-0-2 | | 4 |
| 1. **Type of Course (Check one):** | | | | **j✓d** **✓**  Programme Core  Programme Elective **✓** Open Elective | | | | | |
| 1. **Pre-requisite(s), if any:** None | | | | | | | | | |
| 1. **Frequency of offering (check one):** Odd **✓** Even Either semester Every semester | | | | | | | | | |
| 1. **Brief Syllabus:**   Elements of digital image processing, Image model, Sampling and quantization, Relationships between pixels, Image Transforms, Discrete Fourier Transform, Discrete Cosine Transform, Haar Transform, Hadamard Transform, Image Enhancement, Enhancement by point processing, Spatial filtering, Enhancement in the frequency domain, Color Image Processing, Image Segmentation, Discontinuity detection, Edge linking and boundary detection, Thresholding, Region oriented segmentation, Use of motion for segmentation, Introduction to CV, Introduction to Face Detection, Face Detection with OpenCV, Object Detection Introduction, Object Detection with SSD, Generative Adversarial Networks (GANs) Introduction. | | | | | | | | | |
| **Total lecture, Tutorial and Practical Hours for this course (Take 15 teaching weeks per semester):** 90 hours  The class size is maximum 30 learners. | | | | | | | | | |
| **Lectures:**  30 hours | | | | | **Practice** | | | | |
| **Tutorials :** 0 hours | | | **Lab Work:** 60 hours | |
| 1. **Course Outcomes (COs)**   On successful completion of this course students will be able to: | | | | | | | | | |
| **CO 1** | Implement fundamental image processing techniques required for computer vision. | | | | | | | | |
| **CO 2** | Analyze the different segmentation techniques and shape analysis | | | | | | | | |
| **CO 3** | Apply 3D vision techniques to images | | | | | | | | |
| **CO 4** | Develop projects that can detect faces and objects using Open CV | | | | | | | | |
| 1. **UNIT WISE DETAILS No. of Units: 4** | | | | | | | | | |
| **Unit Number: 1** | | **Title: Fundamentals of Image Processing** | | | | | | | **No. of hours: 6** |
| **Content Summary:**  Fundamentals of Image Formation, Transformation: Discrete Fourier Transform, Discrete Cosine Transform, Haar Transform, Hadamard Transform, Convolution and Filtering, Image Enhancement, Restoration, Image Segmentation -Discontinuity detection, Edge linking and boundary detection, Thresholding, Region oriented segmentation, Use of motion for segmentation , Histogram Processing. | | | | | | | | | |
| **Unit Number: 2** | | | **Shapes and Regions** | | | | | | **No. of hours: 6** |
| **Content Summary:**  Binary shape analysis, connectedness, object labeling and counting, size filtering, distance functions, skeletons and thinning, deformable shape analysis, boundary tracking procedures, active contours, shape models and shape recognition, centroidal profiles, handling occlusion, boundary length measures, boundary descriptors, chain codes, Fourier descriptors, region descriptors, moments. | | | | | | | | | |
| **Unit Number: 3** | | | **Title: 3D Vision and Motion** | | | | | | **No. of hours: 8** |
| **Content Summary:**  Methods for 3D vision, projection schemes, shape from shading, photometric stereo, shape from texture, shape from focus, active range finding, surface representations, point-based representation, volumetric representations, 3D object recognition, 3D reconstruction, introduction to motion, triangulation, bundle adjustment, translational alignment, parametric motion, spline-based motion, optical flow , layered motion. | | | | | | | | | |
| **Unit Number: 4** | | | **Title: Applications** | | | | | | **No. of hours: 10** |
| **Content Summary:**  Introduction to Face Detection**-** Face Detection with OpenCV. Object Detection Introduction**-** Object Detection with SSD, Generative Adversarial Networks (GANs) Introduction, Active appearance and 3D shape models of faces Application: Surveillance, foreground-background separation, particle filters,Chamfer matching, tracking, and occlusion, combining views from multiple camera, human gait analysis Application: In-vehicle vision system: locating roadway, road markings, identifying road signs, locating pedestrians. | | | | | | | | | |
| 1. **Brief Description of Self-learning components by students (through books/resource material etc.):**   Supplementary MOOC Courses   1. [**https://www.udemy.com/course/complete-python-based-image-processing-and-computer-vision/**](about:blank) 2. [**https://www.coursera.org/learn/computer-vision-basics**](about:blank) 3. [**https://www.classcentral.com/course/computer-vision-object-detection-19259**](about:blank) 4. **classcentral.com/course/edx-computer-vision-and-image-analysis-11378** | | | | | | | | | |
| 1. **Books Recommended :**   **Text Books:**   1. Szeliski, Richard , *Computer Vision Algorithms and Applications*, Microsoft, Fourth Edition, 2012 2. Jan Erik Solem, *Programming Computer Vision with Python: Tools and algorithms for analyzing images*, O'Reilly Media, First Edition, 2015 3. Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing*, Prentice, Third Edition, 2016 4. D. L. Baggio et al, *Mastering OpenCV with Practical Computer Vision Projects*, Packt Publishing, First Edition, 2012   **Reference Books:**   1. Mark Nixon and Alberto S. Aquado, ―*Feature Extraction & Image Processing for Computer Vision*, Academic Press, Third Edition,2012 2. Simon J. D. Prince, ―*Computer Vision: Models, Learning, and Inference*, Cambridge University Press, First Edition, 2012   **Ebooks**   1. [https://www.pdfdrive.com/image-operators-image-processing-in-python-e189690145.html](about:blank) 2. [https://www.pdfdrive.com/learning-image-processing-with-opencv-exploit-the-amazing-features-of-opencv-to-create-powerful-image-processing-applications-through-easy-to-follow-examples-e167899040.html](about:blank) 3. [https://www.pdfdrive.com/practical-machine-learning-and-image-processing-for-facial-recognition-object-detection-and-pattern-recognition-using-python-e188718832.html](about:blank)   **Reference Websites: (nptel, swayam, coursera, edx, udemy, lms, official documentation weblink)**   * [https://www.edx.org/course/computer-vision-image-analysis-1/](about:blank) * [http://www.cse.iitm.ac.in/~vplab/computer\_vision.html](about:blank) * [**www.lms.ncuindia.edu/lms**](about:blank)   **Interview/Placement related Commonly asked Questions:**   * [**https://engineeringinterviewquestions.com/digital-image-processing-viva-questions-and-answers-ece/**](about:blank) * [**https://www.exams99.com/interview-questions/digital-image-processing-interview-questions-and-answers**](about:blank) * [**https://www.sanfoundry.com/digital-image-processing-multiple-choice-questions-answers/**](about:blank) | | | | | | | | | |

1. **INTRODUCTION**



That ‘learning is a continuous process’ cannot be over emphasized. The theoretical knowledge gained during lecture sessions need to be strengthened through practical experimentation. Thus, practical makes an integral part of a learning process.­­­­­­­­­­­­­­­­­­­­­­­

**COURSE OBJECTIVES:**

1. **Implement fundamental image processing techniques required for computer vision.**
2. **Demonstrate the different type operations and transformation on images for image enhancement.**
3. **Demonstrate the morphological operation on image.**
4. **Analyse the different segmentation techniques and shape analysis**
5. **LAB REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Requirements** | **Details** |
| **1** | **Software Requirements** | **Python 3.+, Opencv Libraries, Jupyter Notebook (Lab), Matplotlib** |
| **2** | **Operating System** | **Windows 10 (64-bit), Linux, Mac** |
| **3** | **Hardware Requirements** | **8GB RAM, 1TB hard-disk, 1-60GHz-1.80GHz processor,** |
| **4** | **Required Bandwidth** | Nil |

1. **GENERAL INSTRUCTIONS** 
   1. **General discipline in the lab**
   * Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.
   * Students will not be allowed to enter late in the lab.
   * Students will not leave the class till the period is over.
   * Students should come prepared for their experiment.
   * Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
   * Students must get the connection of the hardware setup verified before switching on the power supply.
   * Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
   * Violating the above code of conduct may attract disciplinary action.
   * Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.
   1. **Attendance**

* Attendance in the lab class is compulsory.
* Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
* On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**
  1. **Preparation and Performance**
* Students should come to the lab thoroughly prepared on the experiments they are assigned to perform on that day. Brief introduction to each experiment with information about self study reference is provided on LMS.
* Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.
* Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.
* Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
* Refer **Annexure 1** for Lab Report Format

1. **LIST OF EXPERIMENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Title of the Experiment** | **Software Used** | **No. of Hours** | **CO Covered** |
|  | Introduction to digital Image processing using PIL and cv2 libraries in python | Python 3.0 | 2 hrs | CO1 |
|  | Mathematical Operations and Image Transformation in Image Processing | Python 3.0 | 2 hrs | CO2 |
|  | Implementation of negative transformation, log transformation, power law transformation | Python 3.0 | 2 hrs | CO2 |
|  | To obtain histogram equalization of ab image, contrast stretching | Python 3.0 | 2 hrs | CO2 |
|  | Implementation of piecewise linear transformation: gray level slicing, Thresholding and Bit plane slicing | Python 3.0 | 2 hrs | CO2 |
|  | Implementation of correlation and convolution filters for Image processing | Python 3.0 | 2 hrs | CO2 |
|  | Implementation of smoothing filters: averaging filter, Median Filter, Mean Filter, Min-Max Filter in spatial domain | Python 3.0 | 2 hrs | CO2 |
|  | Implementation of Sharpening filter in spatial domain: Gaussian Filter (First Order Filter), Laplace Filter (Second Order Filter) | Python 3.0 | 2 hrs | CO2 |
|  | Implementation of Image Smoothening in Frequency Domain | Python 3.0 | 2 hrs | CO2 |
|  | Implementation of Image Sharpening in Frequency Domain | Python 3.0 | 2 hrs | CO2 |
|  | To perform morphological operations on images | Python 3.0 | 2 hrs | CO3 |
|  | To perform practical on edge detection using sobel, prewitt, Robert and Laplace operator |  |  |  |
|  | Perform Region growing for image segmentation | Python 3.0 | 2 hrs | CO3 |
|  | To perform practical to detect lines and circles using Hough transform | Python 3.0 | 2 hrs | CO4 |
|  | To perform the practical for contour Detection | Python 3.0 | 2 hrs | CO4 |
|  | To explore object detection using Deep learning | Python3.0 | 2 hrs | CO4 |

1. **LIST OF FLIP EXPERIMENTS**

|  |  |  |
| --- | --- | --- |
| **Exp. No.** | **Title of the Experiment** | **Mapped CO** |
|  | To perform linear and non-linear transformation on images | CO2 |
|  | Methods to model and process colour images | CO3 |
| 1. O | Thresholding-based segmentation technique | CO4 |
|  | The region growing technique for segmentation | CO4 |
|  | Demonstration of seed selection for segmentation | CO4 |
|  | Experiments with Memory-Based Object Recognition System | CO4 |

1. **LIST OF PROJECTS**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Project Title** | **Mapped CO** |
|  | **Licence Plate Recognition** | CO2, CO3, CO4 |
|  | Hand Gesture Detection and Recognition for Human-Computer Interaction | CO2, CO3 |
|  | Localized object detection | CO3, CO4 |
|  | Image Forensic for Digital Image Copy Move Forgery Detection | CO2, CO3 |
|  | Cancer Detection | CO2, CO3, CO4 |
|  | Lane detection for ADAS | CO3, CO4 |
|  | Face Emotion recognition | CO1, CO2, Co3, CO4 |
|  | Intelligent Traffic Light Control using Image Processing | CO1, CO2, CO3, CO4 |
|  | Identification of Human Act by Image Processing | CO1, CO2, CO3, CO4 |
|  | Real time Drowsy Driver Detection | CO1, CO2, CO3, CO4 |
|  | Currency Identification System | CO1, CO2, CO3, CO4 |
|  | Automatic Vehicle Parking System | CO1, CO2, CO3, CO4 |

**Project 1**: **License Plate Recognition**

License Plate recognition is one of the techniques used for vehicle identification purposes. The sole intention of this project is to find the most efficient way to recognize the registration information from the digital image (obtained from the camera).

**Project 2: Hand Gesture Detection and Recognition for Human-Computer Interaction**

This project deals with the detection and recognition of hand gestures. Gesture recognition is one of the essential techniques to build user-friendly interfaces. For example, a robot that can recognize hand gestures can take commands from humans, and for those who are unable to speak or hear, having a robot that can recognize sign language would allow them to communicate with it.

**Project 3: Localized object detection**

Multi-Object detection is one of the active fields of computer vision. The goal of this field is detecting all the objects of a given image.

**Project 4: Image Forensic for Digital Image Copy Move Forgery Detection**

In this day and age, digital images tampering has been made easy with widely available image editing software, such as Adobe Photoshop. The advancement of image editing software has reached a level such that image tampering can be done without degrading its quality or leaving obvious traces. This is alarming as images are now being presented as supported evidences and historical records in various fields, such as in forensic investigation, law enforcement, journalistic photography and medical images

**Project 5: Cancer Detection using MR Image**

In recent years the image processing mechanisms are used widely in several medical areas for improving earlier detection and treatment stages, in which the time factor is very important to discover the disease in the patient as possible as fast, especially in various cancer tumours such as the lung cancer, breast cancer, skin cancer, bone cancer, etc. The segmentation, detection, and extraction of infected tumour area from magnetic resonance (MR) images are a primary concern of this project.

**Project 6: Lane detection for ADAS**

Advanced driver assistant systems (ADAS) have been implemented in many vehicles to help increase both the safety of drivers and pedestrian. The related technology is also used to develop self-driving cars.

**Project: Face Emotion recognition**

Emotions often mediate and facilitate interactions among human beings. Thus, understanding emotion often brings context to seemingly bizarre and/or complex social communication. Emotion can be recognized through a variety of means such as voice intonation, body language, and more complex methods such electroencephalography. However, the easier, more practical method is to examine facial expressions. There are seven types of human emotions shown to be universally recognizable across different cultures: anger, disgust, fear, happiness, sadness, surprise, contempt. Interestingly, even for complex expressions where a mixture of emotions could be used as descriptors, cross-cultural agreement is still observed. Therefore, a utility that detects emotion from facial expressions would be widely applicable.

**Project 8: Intelligent Traffic Light Control using Image Processing**

Day by day the traffic issue has become a major problem in India due to the rising number of motor vehicles. For this reason, one has to utilize the traffic signals which can do the real-time checking of compactness of traffic. This project employs an arrangement of image processing for controlling the traffic in an easy way by capturing images of traffic at crossroads. A step-by-step procedure for changing the duration of the traffic light depends on the traffic density of crossroads at a traffic signal.

**Project 9: Identification of Human Act by Image Processing**

This project is used to identify the human act by image processing in real-time, and the main intention is to communicate the identified gestures using the camera system. This system starts on recognizing the human act given in the database as it transmits the activate signs to the camera arrangement for recording & storing the video stream in the system. The process of pattern matching is utilized to now actions from the recorded video outline straight. The image from the video is intern evaluates by the database and finally, the output will get.

**Project 10: Real time Drowsy Driver Detection**

Driver fatigue is a significant factor in a large number of vehicle accidents. The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the eye movements of a driver in real-time. By monitoring the eye movements, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident.

**Project 11: Currency Identification System**

The identification of different countries’ currency is very difficult. The main intention of this project is to help citizens to resolve this problem. However, currency identification systems are based on image analysis and are completely not enough.

The process of this project makes automatic as well as strong, and this system uses as an example of the Chinese renminbi (RMB) and Sweden SEK to demonstrate the techniques.

**Project 12: Automatic Vehicle Parking System**

Nowadays, there are many cities worldwide facing a lot of problems with vehicle parking due to less availability of parking places, high land prices, etc. To overcome this issue here is a solution namely an automatic car parking system. The proposed system is used in public places like hotels, offices, theatres, homes, hospitals, stadiums, airports, etc. There are several advantages by using this system such as it occupies less space, takes less time for taking as well as delivering the car, safety, and security for the vehicle from thefts.

1. **RUBRICS (Only for Lab components)**

|  |  |
| --- | --- |
| **Marks Distribution (Total Marks 70)** | |
| **Continuous Evaluation (30Marks)** | **Project Evaluations with Industry Mentor (40 Marks)** |
| Each experiment shall be evaluated for 10 marks and at the end of the semester proportional marks shall be awarded out of total 20. And at the end of semester one final viva will be conducted on all topics of subject taught in lab and theory for 10 marks. | The project shall be evaluated for 30 marks. It will include the marks for defining clear problem statement, project objective, implementation and output achieved using various Image processing techniques. And at the end of the semester viva will be conducted related to the project review for 10 Marks. |
| Following is the breakup of 20 marks for each  **5 Marks**: Observation & conduct of experiment. Teacher may ask questions about experiment.  **10 Marks:** For report writing  **5 Marks:** For the 15 minutes quiz to be conducted in every lab. |

**Annexure 1**

**Introduction to Image Processing**

**(CSL316)**

Lab Practical Report



Faculty Name : Dr.Ashwini Rahangdale Student name: Chayan Gulati

Roll No.: 18CSU054

Semester: 6th

Group: A-2

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2020-21

**INDEX**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Experiment** | **Date of Experiment** | **Date of Submission** | **Marks** | **Signature** |
| **1.** | Introduction to digital Image processing using PIL and cv2 libraries in python | **04-02-2021** | **04-02-2021** |  |  |
| **2.** | Mathematical and Image Transformation on Image Processing | **11-02-2021** | **11-02-2021** |  |  |
| **3.** | Implementation of negative transformation, log transformation, power law transformation | **18-02-2021** | **18-02-2021** |  |  |
| **4.** | To obtain histogram equalization image, contrast stretching | **25-02-2021** | **25-02-2021** |  |  |
| **5.** | Implementation of piecewise linear transformation: gray level slicing, Thresholding and Bit plane slicing | **04-03-2021** | **04-03-2021** |  |  |
| **6.** | Implementation of correlation and convolution filters for Image processing | **18-03-2021** | **18-03-2021** |  |  |
| **7.** | Implementation of smoothing filters: averaging filter, Median Filter, Mean Filter, Min-Max Filter in spatial domain | **25-03-2021** | **25-03-2021** |  |  |
| **8.** | Implementation of Sharpening filter in spatial domain: Gaussian Filter (First Order Filter), Laplace Filter (Second Order Filter) | **01-04-2021** | **01-04-2021** |  |  |
| **9.** | Implementation of Image Smoothening in Frequency Domain | **08-04-2021** | **08-04-2021** |  |  |
| **10.** | Implementation of Image Sharpening in Frequency Domain | **08-04-2021** | **08-04-2021** |  |  |
| **11.** | Edge detection in images using canny algorithm | **29-04-2021** | **29-04-2021** |  |  |
| **12.** | To perform morphological operations on images | **15-04-2021** | **15-04-2021** |  |  |
| **13.** | To perform practical on edge detection using sobel, prewitt, Robert and Laplace operator | **22-04-2021** | **22-04-2021** |  |  |
| **14.** | Perform Region growing for image segmentation | **29-04-2021** | **29-04-2021** |  |  |
| **15.** | To perform practical to detect lines and circles using Hough transform | **06-05-2021** | **06-05-2021** |  |  |
| **16.** | To perform the practical for contour Detection | **13-05-2021** | **13-05-2021** |  |  |

**EXPERIMENT NO. 1**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th / DS-A-2** |
| **Link to Code:** **https://github.com/chayangulati321/IIPR** |
| **Date: 04-02-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring digital Image processing using PIL and cv2 libraries in python |
| **Outcome:**  Familiar with digital Image processing using PIL and cv2 libraries in python. |
| **Problem Statement:**  Introduction to digital Image processing using PIL and cv2 libraries in python |
| **Background Study:**  **Python Imaging Library** - is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.  **OpenCV** - Python makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib. |
| **Code(Solution):**  import skimage  from skimage import io  import numpy as np  import math  import matplotlib  import matplotlib.pyplot as plt  random\_image = np.random.random([500, 500])  random\_image  plt.figure(figsize = (10, 10))  plt.imshow(random\_image)  plt.colorbar();  plt.figure(figsize = (10, 10))  plt.imshow(random\_image, cmap='gray')  plt.axis('off')  plt.colorbar();  from skimage import data  print('Type:', type(camera))  print('dtype:', camera.dtype)  print('shape:', camera.shape)  print('size:', camera.size)  plt.figure(figsize = (10, 10))  plt.imshow(camera, cmap='gray');  print('Type:', type(chessboard))  print('dtype:', chessboard.dtype)  print('shape:', chessboard.shape)  print('size:', chessboard.size)  plt.figure(figsize = (10, 10))  plt.imshow(chessboard, cmap='gray');  chessboard[:, 250:510] = [0, 255, 0]  chessboard[1290:1530, 1020:1310, :] = [255, 255, 255]  plt.figure(figsize = (10, 10))  plt.imshow(chessboard)  plt.show() |
|  |

**EXPERIMENT NO. 2**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th/ DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 11-02-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring Mathematical and Image Transformation on Image Processing |
| **Outcome:**  Familiar withMathematical and Image Transformation on Image Processing |
| **Problem Statement:**  Mathematical and Image Transformation on Image Processing |
| **Background Study:**  An Image transformation can be applied to an image to convert it from one domain to another. Viewing an image in domains such as frequency or Hough space enables the identification of features that may not be as easily detected in the spatial domain.    Function applied inside this digital system that process an image and convert it into output can be called as transformation function. As it shows transformation or relation, that how an image1 is converted to image2. |
| **Code(Solution):**  pip install opencv-python  import cv2  image2=cv2.imread("C:/Users/dell/Desktop/Semester 6/IIPR/rabbit.jpg")  image1=cv2.imread("C:/Users/dell/Desktop/Semester 6/IIPR/dog.jpg")  cv2.add(image1,image2)  cv2.addWeighted(image1,0.7,image2,0.3,0)  cv2.subtract(image1,image2)  cv2.multiply(image1,image2)  cv2.divide(image1,image2)  cv2.add(image1,32)  cv2.subtract(image2,15)  cv2.bitwise\_and(image1,image2)  cv2.bitwise\_or(image1,image2)  cv2.bitwise\_xor(image1,image2)  cv2.bitwise\_not(image1,image2)  import numpy as np  from PIL import Image  from PIL import ImageFilter  img=Image.open("C:/Users/dell/Desktop/Semester 6/IIPR/dog.jpg")  img.show()  img.size  img.size[0]  img.size[1]  import cv2  import matplotlib.pyplot as plt  # Read an image  img\_bgr = cv2.imread('C:/Users/dell/Desktop/Semester 6/IIPR/dog.jpg', 1)  plt.imshow(img\_bgr)  plt.show()  color = ('b', 'g', 'r')      height, width, \_ = img\_bgr.shape  for i in range(0, height - 1):    for j in range(0, width - 1):        pixel = img\_bgr[i, j]          pixel[0] = 255 - pixel[0]        pixel[1] = 255 - pixel[1]      pixel[2] = 255 - pixel[2]      img\_bgr[i, j] = pixel    plt.imshow(img\_bgr)  plt.show()  color = ('b', 'g', 'r')  **Output:** |
|  |

**EXPERIMENT NO. 3**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th/ DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 18-02-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring negative transformation, log transformation, power law transformation |
| **Outcome:**  Familiar with negative transformation, log transformation, power law transformation |
| **Problem Statement:**  Implementation of negative transformation, log transformation, power law transformation |
| **Background Study:**  **Negative Transformation**:- The second linear transformation is negative transformation, which is invert of identity transformation.  **Log Transformation**:- The log transformation is, arguably, the most popular among the different types of transformations used to transform skewed data to approximately conform to normality.  **Power law**:- a power law is a functional relationship between two quantities, where a relative change in one quantity results in a proportional relative change in the other quantity, independent of the initial size of those quantities: one quantity varies as a power of another. |
| **Code(Solution):**  import cv2  import numpy as np  import matplotlib.pyplot as plt    image = cv2.imread('bmw.jpg')  c = 255 / np.log(1 + np.max(image))  log\_image = c \* (np.log(image + 1))    log\_image = np.array(log\_image, dtype = np.uint8)  # Display both images  plt.imshow(image)  plt.show()  plt.imshow(log\_image)  plt.show()  import cv2  import numpy as np  import matplotlib.pyplot as plt  # Read an image  image = cv2.imread('bmw.jpg')  # Apply log transformation method  c = 65535 / np.log(1 + np.max(image))  log\_image = c \* (np.log(image + 1))  # Specify the data type so that  # float value will be converted to int  log\_image = np.array(log\_image, dtype = np.uint8)  # Display both images  plt.imshow(image)  plt.show()  plt.imshow(log\_image)  plt.show()  import cv2  import numpy as np    # Open the image.  img = cv2.imread('bmw.jpg')    # Trying 4 gamma values.  for gamma in [0.1]:        # Apply gamma correction.      gamma\_corrected = np.array(255\*(img / 255) \*\* gamma, dtype = 'uint8')        # Save edited images.      cv2.imwrite('gamma\_transformed'+str(gamma)+'.jpg', gamma\_corrected)  # Display both images  plt.imshow(img)  plt.show()  plt.imshow(gamma\_corrected)  plt.show()  import cv2  import numpy as np    # Open the image.  img = cv2.imread('bmw.jpg')    # Trying 4 gamma values.  for gamma in [0.5]:        # Apply gamma correction.      gamma\_corrected = np.array(255\*(img / 255) \*\* gamma, dtype = 'uint8')        # Save edited images.      cv2.imwrite('gamma\_transformed'+str(gamma)+'.jpg', gamma\_corrected)  # Display both images  plt.imshow(img)  plt.show()  plt.imshow(gamma\_corrected)  plt.show()  import cv2  import matplotlib.pyplot as plt  image = cv2.imread('dog.jpg')  for i, col in enumerate(['b', 'g', 'r']):      hist = cv2.calcHist([image], [i], None, [64], [0,64])      plt.plot(hist, color = col)      plt.xlim([0, 256])    plt.show()  import cv2  import matplotlib.pyplot as plt  image = cv2.imread('dog.jpg')  for i, col in enumerate(['b', 'g', 'r']):      hist = cv2.calcHist([image], [i], None, [512], [0, 256])      plt.plot(hist, color = col)      plt.xlim([0, 256])    plt.show()  **Output:**             |  | | --- | |  | |

**EXPERIMENT NO. 4**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th/ DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 25-02-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring histogram equalization image, contrast stretching |
| **Outcome:**  Familiar with histogram equalization image, contrast stretching |
| **Problem Statement:**  To obtain histogram equalization image, contrast stretching |
| **Background Study:**  **Histogram Equalization** - This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark.  **Contrast Stretching** - is a simple image enhancement technique that attempts to improve the contrast in an image by `stretching' the range of intensity values it contains to span a desired range of values, e.g. the the full range of pixel values that the image type concerned allows. |
| **Code(Solution):**  # import Opencv  import cv2  # import Numpy  import numpy as np  # read a image using imread  image = cv2.imread('rabbit.jpg', 0)  # creating a Histograms Equalization  # of a image using cv2.equalizeHist()  equ = cv2.equalizeHist(image)  # stacking images side-by-side  res = np.hstack((image, equ))  # Display the image  plt.imshow(image)  plt.show()  plt.imshow(res)  plt.show()  # Importing Image and ImageOps module from PIL package  from PIL import Image, ImageOps    # creating a image1 object  im1 = Image.open('rabbit.jpg')    # applying equalize method  im2 = ImageOps.equalize(im1, mask = None)  plt.imshow(im2)  plt.show()  hist = cv2.calcHist([img],[0],None,[256],[0,256])    plt.plot(hist)  plt.show()  minmax = np.zeros((img.shape[0],img.shape[1],3),dtype = 'uint8')  for i in range(img.shape[0]):      for j in range(img.shape[1]):            minmax[i,j] = 255\*(img[i,j]-np.min(img))/(np.max(img)-np.min(img))  plt.imshow(minmax)  plt.show()  hist = cv2.calcHist([img],[0],None,[256],[0,256])    plt.plot(hist)  plt.show()  hist = cv2.calcHist([minmax],[0],None,[256],[0,256])    plt.plot(hist)  plt.show() |

**EXPERIMENT NO. 5**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 04-03-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring piecewise linear transformation: gray level slicing, Thresholding and Bit plane slicing |
| **Outcome:**  Familiar with piecewise linear transformation: gray level slicing, Thresholding and Bit plane slicing |
| **Problem Statement:**  Implementation of piecewise linear transformation: gray level slicing, Thresholding and Bit plane slicing |
| **Background Study:**  **Gray-level Slicing**  Sometimes we need to highlight a specific range of gray levels in a image. Possible application areas are finding masses of water in satellite imagery, enhancement of flaws in x-ray images, etc.  There are two basic approaches towards gray-level slicing.  **Binary Thresholding:** all Gray levels in the range of interest are displayed using a high value and the rest using a low value.  **Gradual Thresholding:** desired range of gray levels are brightened but the background and the gray-level tonalities are preserved.  **Threshold Slicing**  So, we would consider a thresholding function that will search the image pixel by pixel. If the intensity of a pixel is greater than or equal to 150, it will be assigned a value of 255 and if its intensity falls below 150, a zero will be assigned in its place.  **Bit Plane Slicing**   1. Sometimes it is desirable to highlight the contribution made by specific bits to the total image appearance. 2. The image can be imagined to be composed of Eight 1-bit planes -- Plane 0 for the LSB plane and Plane 7 for the MSB. 3. The higher order bits contain visually significant data, the lower order plane contain more subtle details.   This can be accomplished by doing a **Bitwise AND** operation. For example, say the intensity of a pixel is 246 in decimal. In binary this would be (11110110)2(11110110)2. So in order to find the value of the 6th bit, one has to simply do this (11110110)2⊙(01000000)2(11110110)2⊙(01000000)2. The result will simply produce the value of the 6th bit. In other words, 246⊙64246⊙64 will give you the value of the 6th bit. |
| **Code(Solution):**  import cv2  import numpy as np  import matplotlib.pyplot as plt    image = cv2.imread('rabbit.jpg')  import cv2  import numpy as np  from matplotlib import pyplot as plt  img = cv2.imread('rabbit.jpg',0)  ret,thresh = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)  titles = ['Original Image','BINARY']  images = [img, thresh]  # Display both images  plt.imshow(img)  plt.show()  plt.imshow(thresh)  plt.show()  import cv2  import numpy as np  # Load the image  img = cv2.imread('rabbit.jpg',0)  # Find width and height of image  row, column = img.shape  # Create an zeros array to store the sliced image  img1 = np.zeros((row,column),dtype = 'uint8')    # Specify the min and max range  min\_range = 80  max\_range = 120    # Loop over the input image and if pixel value lies in desired range set it to 255 otherwise set it to 0.  for i in range(row):      for j in range(column):          if img[i,j]>min\_range and img[i,j]<max\_range:              img1[i,j] = 255          else:              img1[i,j] = 0  # Display the image  plt.imshow(img1)  plt.show()  #cv2.imshow('sliced image', img1)  #cv2.waitKey(0)  import numpy as np  import cv2  from matplotlib import pyplot as plt  # Read the image in greyscale  img = cv2.imread('city.jpg',0)    #Iterate over each pixel and change pixel value to binary using np.binary\_repr() and store it in a list.  lst = []  for i in range(img.shape[0]):      for j in range(img.shape[1]):           lst.append(np.binary\_repr(img[i][j] ,width=8)) # width = no. of bits    # We have a list of strings where each string represents binary pixel value. To extract bit planes we need to iterate over the strings and store the characters corresponding to bit planes into lists.  # Multiply with 2^(n-1) and reshape to reconstruct the bit image.  eight\_bit\_img = (np.array([int(i[0]) for i in lst],dtype = np.uint8) \* 128).reshape(img.shape[0],img.shape[1])  seven\_bit\_img = (np.array([int(i[1]) for i in lst],dtype = np.uint8) \* 64).reshape(img.shape[0],img.shape[1])  six\_bit\_img = (np.array([int(i[2]) for i in lst],dtype = np.uint8) \* 32).reshape(img.shape[0],img.shape[1])  five\_bit\_img = (np.array([int(i[3]) for i in lst],dtype = np.uint8) \* 16).reshape(img.shape[0],img.shape[1])  four\_bit\_img = (np.array([int(i[4]) for i in lst],dtype = np.uint8) \* 8).reshape(img.shape[0],img.shape[1])  three\_bit\_img = (np.array([int(i[5]) for i in lst],dtype = np.uint8) \* 4).reshape(img.shape[0],img.shape[1])  two\_bit\_img = (np.array([int(i[6]) for i in lst],dtype = np.uint8) \* 2).reshape(img.shape[0],img.shape[1])  one\_bit\_img = (np.array([int(i[7]) for i in lst],dtype = np.uint8) \* 1).reshape(img.shape[0],img.shape[1])    #Concatenate these images for ease of display using cv2.hconcat()  finalr = cv2.hconcat([eight\_bit\_img,seven\_bit\_img,six\_bit\_img,five\_bit\_img])  finalv =cv2.hconcat([four\_bit\_img,three\_bit\_img,two\_bit\_img,one\_bit\_img])    # Vertically concatenate  final = cv2.vconcat([finalr,finalv])    # Display the images  plt.imshow(img)  plt.show()  plt.imshow(final)  plt.show() |
| **Output:** |

**EXPERIMENT NO. 6**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th/ DS-A-2** |
| **Link to Code:** **https://github.com/chayangulati321/IIPR** |
| **Date: 18-03-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring correlation and convolution filters for Image processing |
| **Outcome:**  Familiar with correlation and convolution filters for Image processing |
| **Problem Statement:**  Implementation of correlation and convolution filters for Image processing |
| **Background Study:**  **Correlation Filters** – Correlation Filters are a class of classifiers, which are specifically optimized to produce sharp peaks in the correlation output, primarily to achieve accurate localization of targets in scenes.  **Convolution Filters -** A convolution filter passes over all the pixels of the image in such a manner that, at a given time, we take 'dot product' of the convolution filter and the image pixels to get one final value output. |
| **Code(Solution):**  img =cv2.imread('rabbit.jpg')  kernel =np.ones((5,5),np.float32)/25  dst = cv2.filter2D(img,-1,kernel)  plt.subplot(121),plt.imshow(img),plt.title('original')  plt.xticks([]), plt.yticks([])  plt.subplot(122),plt.imshow(dst),plt.title('Applying Kernel Averaging')  plt.xticks([]), plt.yticks([])  plt.show()  img =cv2.imread('rabbit.jpg')  kernel =np.ones((5,5),np.float32)/25  dst = cv2.filter2D(img,0,kernel)  plt.subplot(121),plt.imshow(img),plt.title('original')  plt.xticks([]), plt.yticks([])  plt.subplot(122),plt.imshow(dst),plt.title('Applying Kernel Averaging')  plt.xticks([]), plt.yticks([])  plt.show()  img =cv2.imread('rabbit.jpg')  kernel =np.ones((16,16),np.float32)/256  dst = cv2.filter2D(img,-1,kernel)  plt.subplot(121),plt.imshow(img),plt.title('original')  plt.xticks([]), plt.yticks([])  plt.subplot(122),plt.imshow(dst),plt.title('Applying Kernel Averaging')  plt.xticks([]), plt.yticks([])  plt.show()  img =cv2.imread('rabbit.jpg')  blur = cv2.blur(img,(5,5))  plt.subplot(121),plt.imshow(img),plt.title('original')  plt.xticks([]), plt.yticks([])  plt.subplot(122),plt.imshow(blur),plt.title('Blurred')  plt.xticks([]), plt.yticks([])  plt.show()  img =cv2.imread('rabbit.jpg')  blur = cv2.medianBlur(img,11)  plt.subplot(121),plt.imshow(img),plt.title('original')  plt.xticks([]), plt.yticks([])  plt.subplot(122),plt.imshow(blur),plt.title('Median Blurred')  plt.xticks([]), plt.yticks([])  plt.show()  img =cv2.imread('rabbit.jpg')  Gaussian = cv2.GaussianBlur(img, (21,21), 0)    plt.imshow(img)  plt.show()  plt.imshow(Gaussian)  plt.show()  **Output** |
|  |

**EXPERIMENT NO. 7**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 25-03-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring smoothing filters: averaging filter, Median Filter, Mean Filter, Min-Max Filter in spatial domain |
| **Outcome:**  Familiar with smoothing filters: averaging filter, Median Filter, Mean Filter, Min-Max Filter in spatial domain |
| **Problem Statement:**  Implementation of smoothing filters: averaging filter, Median Filter, Mean Filter, Min-Max Filter in spatial domain |
| **Background Study:**  **Spatial Filtering -** technique is used directly on pixels of an image. Mask is usually considered to be added in size so that it has a specific centre pixel. This mask is moved on the image such that the centre of the mask traverses all image pixels.  **Mean Filter -** Linear spatial filter is simply the average of the pixels contained in the neighbourhood of the filter mask. The idea is replacing the value of every pixel in an image by the average of the grey levels in the neighbourhood define by the filter mask.  **Averaging filter -** It is used in reduction of the detail in image. All coefficients are equal.  **Median filter -** Each pixel in the image is considered. First neighbouring pixels are sorted and original values of the pixel is replaced by the median of the list.  **Minimum filter -** 0th percentile filter is the minimum filter. The value of the center is replaced by the smallest value in the window.  **Maximum filter -** 100th percentile filter is the maximum filter. The value of the center is replaced by the largest value in the window**.** |
| **Code(Solution):**  import cv2  import numpy as np  import matplotlib.pyplot as plt    img = cv2.imread('rabbit.jpg',0)  m, n = img.shape  mask = np.ones([5, 5], dtype = int)  mask = mask / 25  img\_new = np.zeros([m, n])  for i in range(1, m-2):      for j in range(1, n-2):            temp = img[i-2, j-2]\*mask[0, 0]+img[i-2, j-1]\*mask[0, 1]+img[i-2, j]\*mask[0, 2]+img[i-2, j+1]\*mask[0, 3]+ img[i-2, j+2]\*mask[0, 4]+img[i-1, j - 2]\*mask[1, 0]+img[i - 1, j-1]\*mask[1, 1]+img[i-1 , j]\*mask[1, 2]+img[i-1 , j +1]\*mask[1, 3]+img[i-1 , j+2]\*mask[1, 4]+img[i , j -2]\*mask[2, 0] +img[i , j-1]\*mask[2, 1]+img[i , j ]\*mask[2, 2]+img[i , j+1]\*mask[2, 3]+img[i , j +2]\*mask[2, 4]+img[i+1 , j-2]\*mask[3, 0]+img[i+1 , j -1]\*mask[3, 1]+img[i+1 , j]\*mask[3, 2]+img[i+1 , j +1]\*mask[3, 3]+img[i+1 , j+2]\*mask[3, 4]+img[i+2 , j -2]\*mask[4, 0]+img[i+2 , j-1]\*mask[4, 1]+img[i+2 , j]\*mask[4, 2]+img[i+2 , j+1]\*mask[4, 3]+img[i+2 , j +2]\*mask[4, 4]        img\_new[i, j]= temp    img\_new = img\_new.astype(np.uint8)    plt.imshow(img)  plt.show()  plt.imshow(img\_new)  plt.show()  m, n = img.shape  mask = np.ones([5, 5], dtype = int)  mask = mask / 25  img\_new1 = np.zeros([m, n])  for i in range(1, m-2):      for j in range(1, n-2):       temp = img[i-2, j-2]+img[i-2, j-1]+img[i-2, j]+img[i-2, j+1]+ img[i-2, j+2]+img[i-1, j - 2]+img[i - 1, j-1]+img[i-1 , j]+img[i-1 , j +1]+img[i-1 , j+2]+img[i , j -2]+img[i , j-1]+img[i , j ]+img[i , j+1]+img[i , j +2]+img[i+1 , j-2]+img[i+1 , j -1]+img[i+1 , j]+img[i+1 , j +1]+img[i+1 , j+2]+img[i+2 , j -2]+img[i+2 , j-1]+img[i+2 , j]+img[i+2 , j+1]+img[i+2 , j +2]         temp = sorted(temp)     img\_new1[i, j]= temp[6]    img\_new = img\_new.astype(np.uint8)  m, n = img.shape  mask = np.ones([5, 5], dtype = int)  mask = mask / 25  img\_min = np.zeros([m, n])  for i in range(1, m-2):      for j in range(1, n-2):     temp = img[i-2, j-2]+img[i-2, j-1]+img[i-2, j]+img[i-2, j+1]+ img[i-2, j+2]+img[i-1, j - 2]+img[i - 1, j-1]+img[i-1 , j]+img[i-1 , j +1]+img[i-1 , j+2]+img[i , j -2]+img[i , j-1]+img[i , j ]+img[i , j+1]+img[i , j +2]+img[i+1 , j-2]+img[i+1 , j -1]+img[i+1 , j]+img[i+1 , j +1]+img[i+1 , j+2]+img[i+2 , j -2]+img[i+2 , j-1]+img[i+2 , j]+img[i+2 , j+1]+img[i+2 , j +2]         img\_min[i, j]= min(temp)    img\_min = img\_new.astype(np.uint8)  # Median Spatial Domain Filtering  import cv2  import numpy as np  # Read the image  img\_noisy1 = cv2.imread('rabbit.jpg', 0)  # Obtain the number of rows and columns  # of the image  m, n = img\_noisy1.shape  # Traverse the image. For every 3X3 area,  # find the median of the pixels and  # replace the ceter pixel by the median  img\_new1 = np.zeros([m, n])  for i in range(1, m-2):    for j in range(1, n-2):      temp = [img\_noisy1[i-2, j-2],        img\_noisy1[i-1, j],        img\_noisy1[i-1, j + 1],        img\_noisy1[i, j-1],        img\_noisy1[i, j],        img\_noisy1[i, j + 1],        img\_noisy1[i + 1, j-1],        img\_noisy1[i + 1, j],        img\_noisy1[i + 1, j + 1]]        temp = sorted(temp)      img\_new1[i, j]= temp[4]  img\_new1 = img\_new1.astype(np.uint8)  cv2.imwrite('new\_median\_filtered.png', img\_new1) |
|  |

**EXPERIMENT NO. 8**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6 / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 01-04-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring Sharpening filter in spatial domain: Gaussian Filter (First Order Filter), Laplace Filter (Second Order Filter) |
| **Outcome:**  Familiar with Sharpening filter in spatial domain: Gaussian Filter (First Order Filter), Laplace Filter (Second Order Filter) |
| **Problem Statement:**  Implementation of Sharpening filter in spatial domain: Gaussian Filter (First Order Filter), Laplace Filter (Second Order Filter) |
| **Background Study:**  **Sharpening filter in spatial domain -** It is also known as derivative filter. The purpose of the sharpening spatial filter is just the opposite of the smoothing spatial filter. Its main focus in on the removal of blurring and highlight the edges. It is based on the first and second order derivative.  **Gaussian Filter (First Order Filter) –** have the properties of having no overshoot to a step function input while minimizing the rise and fall time. This behavior is closely connected to the fact that the Gaussian filter has the minimum possible group delay. It is considered the ideal time domain filter, just as the sinc is the ideal frequency domain filter.  **Laplace Filter (Second Order Filter) -** The Laplacian of an image highlights regions of rapid intensity change and is an example of a second order or a second derivative method of enhancement . It is particularly good at finding the fine details of an image. Any feature with a sharp discontinuity will be enhanced by a Laplacian operator. The Laplacian is a well-known linear differential operator approximating the second derivative. |
| **Code(Solution):**  img =cv2.imread('rabbit.jpg')  Gaussian = cv2.GaussianBlur(img, (21,21), 0)    plt.imshow(img)  plt.show()  plt.imshow(Gaussian)  plt.show()  img =cv2.imread('rabbit.jpg')  Gaussian = cv2.GaussianBlur(img, (7, 7), 0)    plt.imshow(img)  plt.show()  plt.imshow(Gaussian)  plt.show() |
| **Output:** |

**EXPERIMENT NO. 9**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th/ DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 08-04-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring Image Smoothening in Frequency Domain |
| **Outcome:**  Got familiar with Image Smoothening in Frequency Domain |
| **Problem Statement:**  Implementation of Image Smoothening in Frequency Domain |
| **Background Study:**  **Image Smoothing (Low-pass Frequency Domain Filters)**  A low-pass filter that attenuates (suppresses) high frequencies while passing the low frequencies which results in creating a blurred (smoothed) image. It leaves the low frequencies of the Fourier transform relatively unchanged and ignores the high frequency noise components. Three main low-pass filters are: |
| **Code(Solution):** |
|  |

**EXPERIMENT NO. 10**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6th / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 08-04-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Exploring Image Sharpening in Frequency Domain |
| **Outcome:**  Got Familiar with Image Sharpening in Frequency Domain |
| **Problem Statement:**  Implementation of Image Sharpening in Frequency Domain |
| **Background Study:**  **Image Sharpening (High-pass Frequency Domain Filters)**  Sharpening of an image in the frequency domain can be achieved by high pass filtering process which attenuates (suppress) low frequency components without disturbing high frequency information in the Fourier transform of the image.  The high-pass filter Hhp is often represented by its relationship  to the low-pass filter (Hlp) as:  Hhp (u, v) =1- Hlp (u, v)i.  Ideal High-Pass Filter (IHPF)  The ideal high pass filter simply cuts off all the low frequencies  lower than the specified cut-off frequency. |
| **Code(Solution):** |
|  |

**EXPERIMENT NO. 11**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6 / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 29-04-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Edge detection in images using canny algorithm |
| **Outcome:**  Student will come to know about the Edge detection in images using canny algorithm |
| **Problem Statement:**  Edge detection in images using canny algorithm |
| **Background Study:**  The **Canny edge detector** is an **edge detection operator** that uses a multi-stage **algorithm** to **detect** a wide range of **edges** in **images**. It was developed **by** John F. **Canny** in 1986. **Canny** also produced a computational theory of **edge detection** explaining why the technique works.  The effect of the **Canny** operator is determined by **three** parameters --- the width of the Gaussian kernel used in the smoothing phase, and the upper and lower thresholds used by the tracker. |
| **Code(Solution):**  import numpy as np  import cv2 as cv  from matplotlib import pyplot as plt  img = cv.imread('Map.jpg',0)  edges = cv.Canny(img,100,200)  plt.subplot(121),plt.imshow(img,cmap = 'gray')  plt.title('Original Image'), plt.xticks([]), plt.yticks([])  plt.subplot(122),plt.imshow(edges,cmap = 'gray')  plt.title('Edge Image'), plt.xticks([]), plt.yticks([])  plt.show()   Try Canny using "wide" and "tight" thresholds  wide = cv2.Canny(img, 30, 100)  tight = cv2.Canny(img, 200, 240)      # Display the images  f, (ax1, ax2) = plt.subplots(1, 2, figsize=(20,10))  ax1.set\_title('wide')  ax1.imshow(wide, cmap='gray')  ax2.set\_title('tight')  ax2.imshow(tight, cmap='gray')  # Read in the image  image = cv2.imread('Map.jpg')  # Change color to RGB (from BGR)  image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  plt.imshow(image)  # Convert the image to grayscale  gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)  ## TODO: Define lower and upper thresholds for hysteresis  # right now the threshold is so small and low that it will pick up a lot of noise  lower = 0  upper = 50  edges = cv2.Canny(gray, lower, upper)  plt.figure(figsize=(20,10))  plt.imshow(edges, cmap='gray')  # Convert the image to grayscale  gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)  lower = 0  upper = 200  edges = cv2.Canny(gray, lower, upper)  plt.figure(figsize=(20,10))  plt.imshow(edges, cmap='gray') |
| **Outputs:** |

**EXPERIMENT NO. 12**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6 / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 15-04-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  To perform morphological operations on images |
| **Outcome:**  Student will learn how To perform morphological operations on images |
| **Problem Statement:**  To perform morphological operations on images |
| **Background Study:**  **Morphological operations** apply a structuring element to an input **image**, creating an output **image** of the same size. In a **morphological operation**, the value of each pixel in the output **image** is based on a comparison of the corresponding pixel in the input **image** with its neighbors.  **Morphological Operations** is a broad set of **image processing operations** that **process** digital **images** based on their shapes. In a **morphological operation**, each **image** pixel is corresponding to the value of other pixel in its neighborhood |
| **Code(Solution):**  import numpy as np  import matplotlib.pyplot as plt  import cv2  import numpy as np  img = cv2.imread('dog.jpg')    kernel = np.ones((3,3), np.uint8)    img\_erosion = cv2.erode(img, kernel, iterations=1)  img\_dilation = cv2.dilate(img, kernel, iterations=1)  plt.imshow(img)  plt.show()  plt.imshow(img\_erosion)  plt.show()  plt.imshow(img\_dilation)  plt.show()  kernel = np.ones((5,5), np.uint8)    img\_erosion = cv2.erode(img, kernel, iterations=1)  img\_dilation = cv2.dilate(img, kernel, iterations=1)  kernel = np.ones((7,7), np.uint8)    img\_erosion = cv2.erode(img, kernel, iterations=1)  img\_dilation = cv2.dilate(img, kernel, iterations=1)  rect\_kernel=cv2.getStructuringElement(cv2.MORPH\_RECT,(3,3))  ellipse\_kernel=cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(3,3))  cs\_kernel=cv2.getStructuringElement(cv2.MORPH\_CROSS,(3,3))  opening1 = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)  opening2 = cv2.morphologyEx(img, cv2.MORPH\_OPEN, rect\_kernel)  opening3 = cv2.morphologyEx(img, cv2.MORPH\_OPEN, ellipse\_kernel)  opening4 = cv2.morphologyEx(img, cv2.MORPH\_OPEN, cs\_kernel)  plt.imshow(opening1)  plt.show()  plt.imshow(opening2)  plt.show()  plt.imshow(opening3)  plt.show()  plt.imshow(opening4)  plt.show()  rect\_kernel=cv2.getStructuringElement(cv2.MORPH\_RECT,(3,3))  ellipse\_kernel=cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(3,3))  cs\_kernel=cv2.getStructuringElement(cv2.MORPH\_CROSS,(3,3))  closing1 = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel)  closing2 = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, rect\_kernel)  closing3 = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, ellipse\_kernel)  closing4 = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, cs\_kernel)  plt.imshow(closing1)  plt.show()  plt.imshow(closing2)  plt.show()  plt.imshow(closing3)  plt.show()  plt.imshow(closing4)  plt.show()  gradient = cv2.morphologyEx(img, cv2.MORPH\_GRADIENT, kernel)  plt.imshow(gradient)  plt.show() |
| **Outputs:** |

**EXPERIMENT NO. 13**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6 / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 22-04-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  To perform practical on edge detection using sobel, prewitt, Robert and Laplace operator |
| **Outcome:**  Student will come To perform practical on edge detection using sobel, prewitt, Robert and Laplace operator |
| **Problem Statement:**  To perform practical on edge detection using sobel, prewitt, Robert and Laplace operator |
| **Background Study:**  In digital **image** processing and computer vision, **image segmentation** is the process of partitioning a digital **image** into multiple segments (sets of pixels, also known as **image** objects). . **Image segmentation** is typically used to locate objects and boundaries (lines, curves, etc.) in **images**.  The **image segmentation** can be classified into two basic. **types**: Local **segmentation** (concerned with specific part or region of **image**) and Global **segmentation** (concerned with. **segmenting** the whole **image**, consisting of large number of pixels). The **image segmentation** approaches can be categorized into. |
| **Code(Solution):**  import cv2 as cv  import matplotlib.pyplot as plt  import numpy as np  img = cv.imread('puppy.jpg', cv.COLOR\_BGR2GRAY)  rgb\_img = cv.cvtColor(img, cv.COLOR\_BGR2RGB)  # Grayscale processing image  grayImage = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)  kernelx = np.array([[-1, 0], [0, 1]], dtype=int)  kernely = np.array([[0, -1], [1, 0]], dtype=int)  x = cv.filter2D(grayImage, cv.CV\_16S, kernelx)  y = cv.filter2D(grayImage, cv.CV\_16S, kernely)  absX = cv.convertScaleAbs(x)  absY = cv.convertScaleAbs(y)  Roberts = cv.addWeighted(absX, 0.5, absY, 0.5, 0)  titles = ['The original image', 'Roberts operator']  images = [rgb\_img, Roberts]  for i in range(2):      plt.subplot(1, 2, i + 1), plt.imshow(images[i], 'gray')      plt.title(titles[i])      plt.xticks([]), plt.yticks([])  plt.show()  import cv2  import numpy as np  import matplotlib.pyplot as plt  # Load the image  image\_original = cv2.imread('dog.jpg', cv2.IMREAD\_COLOR)  # Convert image to gray scale  image\_gray = cv2.cvtColor(image\_original, cv2.COLOR\_BGR2GRAY)  # 3x3 Y-direction  kernel  sobel\_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])  # 3 X 3 X-direction kernel  sobel\_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])  # Filter the image using filter2D, which has inputs: (grayscale image, bit-depth, kernel)  filtered\_image\_y = cv2.filter2D(image\_gray, -1, sobel\_y)  filtered\_image\_x = cv2.filter2D(image\_gray, -1, sobel\_x)  img\_sobelx = cv2.Sobel(img\_gaussian,cv2.CV\_8U,1,0,ksize=5)  img\_sobely = cv2.Sobel(img\_gaussian,cv2.CV\_8U,0,1,ksize=5)  img\_sobel = img\_sobelx + img\_sobely  plt.imshow(img\_prewittx)  plt.show()  plt.imshow(img\_prewitty)  plt.show()  plt.imshow(img\_prewittx + img\_prewitty)  plt.show()  #cv2.imshow("Prewitt X", img\_prewittx)  #cv2.imshow("Prewitt Y", img\_prewitty)  #cv2.imshow("Prewitt", img\_prewittx + img\_prewitty)  import cv2  import numpy as np  from matplotlib import pyplot as plt  img = cv2.imread('dog.jpg',0)  laplacian = cv2.Laplacian(img,cv2.CV\_64F)  plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')  plt.title('Original'), plt.xticks([]), plt.yticks([])  plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')  plt.title('Laplacian'), plt.xticks([]), plt.yticks([]) |
| **Sample Outputs:** |

**EXPERIMENT NO. 14**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6 / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 29-04-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Perform Region growing for image segmentation |
| **Outcome:**  Students will come to know about Perform Region growing for image segmentation |
| **Problem Statement:**  Perform Region growing for image segmentation |
| **Background Study:**  **Region growing** is a simple **region**-based **image segmentation** method. ... This approach to **segmentation** examines neighboring pixels of initial seed points and determines whether the pixel neighbors should be added to the **region**. The process is iterated on, in the same manner as general data clustering algorithms.  A **region** in an **image** is a group of connected pixels with similar properties. Regions are important for the interpretation of an **image** because they may correspond to objects in a scene. |
| **Code(Solution):**  import cv2 as cv  import numpy as np  from matplotlib import pyplot as plt  import sys  def on\_mouse(event, x, y, flags, params):  if event == cv.EVENT\_LBUTTONDOWN:  print('Start Mouse Position: ' + str(x) + ', ' + str(y))  s\_box = x, y  boxes.append(s\_box)  def region\_growing(img, seed):  #Parameters for region growing  neighbors = [(-1, 0), (1, 0), (0, -1), (0, 1)]  region\_threshold = 0.2  region\_size = 1  intensity\_difference = 0  neighbor\_points\_list = []  neighbor\_intensity\_list = []  #Mean of the segmented region  region\_mean = img[seed]  #Input image parameters  height, width = img.shape  image\_size = height \* width  #Initialize segmented output image  segmented\_img = np.zeros((height, width, 1), np.uint8)  #Region growing until intensity difference becomes greater than certain threshold  while (intensity\_difference < region\_threshold) & (region\_size < image\_size):  #Loop through neighbor pixels  for i in range(4):  #Compute the neighbor pixel position  x\_new = seed[0] + neighbors[i][0]  y\_new = seed[1] + neighbors[i][1]  #Boundary Condition - check if the coordinates are inside the image  check\_inside = (x\_new >= 0) & (y\_new >= 0) & (x\_new < height) & (y\_new < width)  #Add neighbor if inside and not already in segmented\_img  if check\_inside:  if segmented\_img[x\_new, y\_new] == 0:  neighbor\_points\_list.append([x\_new, y\_new])  neighbor\_intensity\_list.append(img[x\_new, y\_new])  segmented\_img[x\_new, y\_new] = 255  #Add pixel with intensity nearest to the mean to the region  distance = abs(neighbor\_intensity\_list-region\_mean)  pixel\_distance = min(distance)  index = np.where(distance == pixel\_distance)[0][0]  segmented\_img[seed[0], seed[1]] = 255  region\_size += 1  #New region mean  region\_mean = (region\_mean\*region\_size + neighbor\_intensity\_list[index])/(region\_size+1)  #Update the seed value  seed = neighbor\_points\_list[index]  #Remove the value from the neighborhood lists  neighbor\_intensity\_list[index] = neighbor\_intensity\_list[-1]  neighbor\_points\_list[index] = neighbor\_points\_list[-1]  return segmented\_img  boxes = []  filename = 'C:/Users/dell/Desktop/Semester 6/IIPR/8.png'  img = cv.imread(filename, 0)  resized = cv.resize(img,(256,256))  cv.namedWindow('input')  cv.setMouseCallback('input', on\_mouse, 0,)  cv.imshow('input', resized)  cv.waitKey()  print("Starting region growing based on last click")  seed = boxes[-1]  cv.imshow('input', region\_growing(resized, seed))  print("Done. Showing output now")  cv.waitKey()  cv.destroyAllWindows() |
| **Outputs:** |

**EXPERIMENT NO. 15**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6 / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 06-05-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  To perform practical to detect lines and circles using Hough transform |
| **Outcome:**  Students will come to know how To perform practical to detect lines and circles using Hough transform |
| **Problem Statement:**  To perform practical to detect lines and circles using Hough transform |
| **Background Study:**  f two edge points lay on the same **line**, their corresponding cosine curves will intersect each other on a specific (ρ, θ) pair. Thus, the **Hough Transform** algorithm detects **lines** by finding the (ρ, θ) pairs that has a number of intersections larger than a certain threshold.  The **Hough transform** (HT) can be **used** to detect lines circles or • The **Hough transform** (HT) can be **used** to detect lines, circles or other parametric curves. It was introduced in 1962 (**Hough** 1962) and first **used** to find lines in images a decade later (Duda 1972). The goal is to find the location of lines in images. |
| **Code(Solution):**  import cv2  import numpy as np  import matplotlib.pyplot as plt  image = cv2.imread('C:/Users/dell/Desktop/Semester 6/IIPR/sudoko.jpg')  plt.imshow('Image', image)  #cv2.waitKey(0)  #cv2.destroyAllWindows()  gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  cv2.imshow('Gray Image', gray)  cv2.waitKey(0)  cv2.destroyAllWindows()  canny = cv2.Canny(gray, 100, 170, apertureSize=3)  cv2.imshow('Canny Image', canny)  cv2.waitKey(0)  cv2.destroyAllWindows()  lines = cv2.HoughLines(canny, 1, np.pi/180, 50)  print(lines.shape)  for rho, theta in lines[0]:  a = np.cos(theta)  b = np.sin(theta)  x0 = a \* rho  y0 = b \* rho  x1 = int(x0 + 1000 \* (-b))  y1 = int(y0 + 1000 \* (a))  x2 = int(x0 - 1000 \* (-b))  y2 = int(y0 - 1000 \* (a))  cv2.line(image, (x1, y1), (x2, y2), (255, 0, 0), 2)    cv2.imshow('Detection on Image(Hough Lines)', image)  cv2.waitKey(0)  cv2.destroyAllWindows()  import cv2  import numpy as np  image = cv2.imread('tic\_tac\_toe.png')  gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  edges = cv2.Canny(gray, 30, 120, apertureSize = 3)  lines = cv2.HoughLinesP(edges, 1, np.pi/180, 240, 20, 50)  print(lines.shape)  for x1, y1, x2, y2 in lines[0]:  cv2.line(image, (x1, y1), (x2, y2), (0,0,255), 2)  cv2.imshow('Probalistic Hough Lines', image)  cv2.waitKey(0)  cv2.destroyAllWindows()  import cv2 as cv  import numpy as np  img = cv.imread(cv.samples.findFile('sudoko.png'))  gray = cv.cvtColor(img,cv.COLOR\_BGR2GRAY)  edges = cv.Canny(gray,50,150,apertureSize = 3)  lines = cv.HoughLines(edges,1,np.pi/180,200)  print(lines) |
| **Sample Outputs:** |

**EXPERIMENT NO. 16**

|  |
| --- |
| **Student Name and Roll Number: Chayan Gulati 18csu054** |
| **Semester /Section: 6 / DS-A-2** |
| **Link to Code: https://github.com/chayangulati321/IIPR** |
| **Date: 13-05-2021** |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  To perform the practical for contour Detection |
| **Outcome:**  Student will come to know how To perform the practical for contour Detection |
| **Problem Statement:**  To perform the practical for contour Detection |
| **Background Study:**  To do **contours detection** OpenCV provide a function called FindContours which intent to find **contours** in the image. ... Then we apply the FindContours function to find **contours** and print them on the colour image even though we **work** on a grayscale version of the image.   1. Read **image** as grey scale **image**. 2. Use cv2. threshold() function to obtain the threshold **image**. 3. Use cv2. **findContours**() and pass the threshold **image** and necessary parameters. 4. **findContours**() returns **contours**. You can draw it on the original **image** or a blank **image**. |
| **Code(Solution):**  import cv2  import numpy as np  import matplotlib.pyplot as plt  %matplotlib inline  img = cv2.imread('C:/Users/dell/Desktop/Semester 6/IIPR/bunch\_shape.jpg')  cv2.imshow('Image', img)  cv2.waitKey(0)  cv2.destroyAllWindows()  gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  cv2.imshow('Gray Image', gray)  cv2.waitKey(0)  cv2.destroyAllWindows()  img.shape  canny = cv2.Canny(gray, 20, 50)  contours, hierarchy = cv2.findContours(canny, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  cv2.imshow('Contour Image', canny)  cv2.waitKey(0)  cv2.destroyAllWindows()  print(f'No. of contours: {len(contours)}')  cv2.drawContours(img, contours, -1, (0,0,255), 2)  cv2.imshow('Image with contours', img)  cv2.waitKey(0)  cv2.destroyAllWindows()  contours, hierarchy = cv2.findContours(canny, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)  cv2.imshow('Contour Image', canny)  cv2.waitKey(0)  cv2.destroyAllWindows()  print(f'No. of contours: {len(contours)}')  cv2.drawContours(img, contours, -1, (255,0,0), 2)  cv2.imshow('Image with contours', img)  cv2.waitKey(0)  cv2.destroyAllWindows()  contours, hierarchy = cv2.findContours(canny, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  cv2.imshow('Contour Image', canny)  cv2.waitKey(0)  cv2.destroyAllWindows()  print(f'No. of contours: {len(contours)}')  cv2.drawContours(img, contours, -1, (0,255,0), 2)  cv2.imshow('Image with contours', img)  cv2.waitKey(0)  cv2.destroyAllWindows()  image = cv2.imread('various\_shapes\_0.jpg')  cv2.imshow('Input Image', image)  cv2.waitKey(0)  cv2.destroyAllWindows()  gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  cv2.imshow('Gray Image', gray)  cv2.waitKey(0)  cv2.destroyAllWindows()  canny = cv2.Canny(gray, 20, 50)  cv2.imshow('Canny Image', canny)  cv2.waitKey(0)  cv2.destroyAllWindows()  contour, hierarchy = cv2.findContours(canny, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)  print(f'No. of contours: {len(contour)}')  def contour\_area(contour):  Area = []    for cnt in contour:  area = cv2.contourArea(cnt)  Area.append(area)  return Area  sorted\_contour = sorted(contour, key=cv2.contourArea, reverse=True)  for i in sorted\_contour:  cv2.drawContours(image, [i], -1, (125,125,125), 2)  cv2.imshow('Image after sorting contours', image)  cv2.waitKey(0)  cv2.destroyAllWindows() |
| **Sample Outputs:** |
|  |
|  |

**Annexure 2**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** |  | **Page No.** |
| **1.** | **Project Description** |  |
| **2.** | **Problem Statement** |  |
| **3.** | **Analysis**  **3.1 Hardware Requirements**  **3.2 Software Requirements** |  |
| **4.** | **Design**  **4.1 Data/Input Output Description:**  **4.2 Algorithmic Approach / Algorithm / DFD / ER diagram/Program Steps** |  |
| **5.** | **Implementation and Testing (stage/module wise)** |  |
| **6.** | **Output (Screenshots)** |  |
| **7.** | **Conclusion and Future Scope** |  |