**A Project Report**

**on**

# “An advanced image enhancement suite using machine learning”

Towards partial fulfillment of **Bachelor of Technology (B.Tech)** of

# “University College of Engineering & Technology

**Vinoba Bhave University, Hazaribagh, Jharkhand”**

**Under the supervision of**

**Mr. Ranjeet kumar**

Asst. Professor

ECED - UCET, VBU

**Submitted by**

**Sidharth Kumar – UCET-2006130**

**Chandan Kumar – UCET-2006111**

**Fagu Kumar – UCET-2006142**

**Paras Kumar – UCET-2006118**

**Ankit Ekka – UCET-2006138**



# Electronics & Communication Engineering Department

# University College of Engineering & Technology

**Vinoba Bhave University, Hazaribagh, Jharkhand**

**ABSTARCT**

In the fast-paced digital world, managing and enhancing images effectively is more crucial than ever. Our project introduces an advanced image processing tool that utilizes machine learning to offer a range of powerful functionalities directly through a user-friendly web interface. This tool allows users to compress images, reduce noise, sharpen details, and segment images into meaningful parts.

**Image Compression:** Our system uses machine learning to compress images efficiently. This helps in reducing the file size, which is essential for saving storage space and speeding up image transmission over the internet.

**Noise Reduction:** To enhance image clarity, our application provides techniques to reduce visual noise that often degrades image quality. This is especially useful in fields like medical imaging or security, where clear images are critical.

**Image Sharpening:** Our tool also includes a feature to sharpen images, enhancing edges and fine details that are important for applications requiring high-detail visibility such as satellite images or advanced photography.

**Image Segmentation:** By segmenting images into distinct parts, our tool aids in various advanced applications, including automated systems and diagnostic tools, where identifying specific regions within an image is necessary.

Our project aims to make advanced image processing accessible to everyone, from professionals in various industries to individuals, enhancing how we store, share, and utilize images in our daily lives. Whether for improving personal photos, professional media, or technical applications, our tool provides an essential suite of enhancements that cater to the needs of modern digital handling of images.

# DECLARATION

I undersigned, hereby declare that the project titled “**An Advanced Image Enhancement Suite Using Machine Learning”** submitted in partial fulfillment for the award of **Degree of Bachelor of Technology (B. Tech)** of “**University College of Engineering & Technology, Vinoba Bhave University, Hazaribagh, Jharkhand ”** is a bonafide record of work done by me under the guidance of “**Mr. Ranjeet Kumar, Asst. Professor” Department of Electronics and Communication Engineering, UCET, VBU Hazaribagh Jharkhand.** This report has not previously formed the basis for the award of any degree, diploma, or similar title of any University.

Date ………………

**Sidharth Kumar – UCET-2006130**

**Chandan Kumar – UCET-2006111**

**Fagu Kumar – UCET-2006142**

**Paras Kumar – UCET-2006118**

**Ankit Ekka – UCET-2006138**

# CERTIFICATE

This is to certify that the report titled “**An Advanced Image Enhancement Suite Using Machine Learning”** being submitted by “**Sidharth Kumar (UCET-2006130), Chandan Kumar (UCET-2006111), Fagu kumar (UCET-2006142), Paras kumar (UCET-2006118), &Ankit Ekka (UCET-2006138) ”** in partial fulfillment of the requirements for the award of the “**Degree of Bachelor of Technology (B. Tech)”**, of “**University College of Engineering & Technology, Vinoba Bhave University, Hazaribagh, Jharkhand ”** was prepared under my guidance. This report has not previously formed the basis for the award of any degree, diploma, or similar title of any University.

Date ………………

**Mr. Ranjeet Kumar**

**Asst. Professor ECED - UCET, VBU**

**ACKNOWLEDGEMENT**

We extend our heartfelt gratitude to our Head of the Department, Mr. Rakesh Kumar Singh, for his unwavering support and encouragement throughout the duration of this project. His insightful guidance has been instrumental in shaping our understanding and approach towards the development of "An advanced image enhancement suite using machine learning".

We are also deeply indebted to our project guide, Mr. Ranjeet Kumar, whose expertise and mentorship have been invaluable. His dedication and constructive feedback have propelled us towards achieving our goals and realizing the full potential of this project.

Furthermore, we express our appreciation to all faculty members of the Department of Electronics and Communication engineering for their continuous encouragement and assistance.

Last but not least, we would like to acknowledge the support of our families and friends, whose unwavering belief in our abilities has been a constant source of motivation.

signature

SIDHARTH KUMAR (Roll no: UCET-2006130)

CHANDAN KUMAR (Roll no: UCET-2006111)

FAGU KUMAR (Roll no: UCET-2006142)

PARAS KUMAR (Roll no: UCET-2006118)

ANKIT EKKA (Roll no: UCET-2006138)

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No** | **Name** | **Pg No** |
| 1 | INTRODUCTION | 1 |
| 2 | LITERATURE REVIEW | 3 |
| 3 | IMAGE COMPRESSION USING K MEANS CLUSTERING | 5 |
| 4 | NOISE REDUCTION USING MEDIAN FILTER | 7 |
| 5 | IMAGE SEGMENTATION USING K MEANS CLUSTERING | 8 |
| 6 | IMAGE SHARPENING USING OPENCV | 10 |
| 7 | WORKING AND RESULT ANALYSIS | 12 |
| 8 | CONCLUSION | 22 |
| 9 | FUTURE WORK | 23 |
| 10 | REFERENCES | 24 |

**Chapter No – 1 1**

INTRODUCTION

In today's digital age, images form the backbone of numerous applications across various sectors including media, medical imaging, surveillance, and social networking. With the exponential increase in digital content, efficient image processing techniques have become crucial for optimizing storage and improving transmission speed while maintaining high image quality. Leveraging machine learning, our project offers a robust solution to tackle these challenges by providing advanced image processing capabilities directly accessible through a u web interface.

**Key Functionalities of Our Project**

1. **Image Compression**: As digital imagery becomes increasingly prevalent, the need for effective image compression techniques grows. Our application employs machine learning algorithms, specifically k-means clustering, to reduce the number of colour in images. This reduces the file size without significantly compromising image quality, making storage and transmission more efficient. This is particularly vital in web applications, where bandwidth conservation and quick loading times are crucial for user retention and satisfaction.
2. **Noise Reduction**: Digital images often suffer from various types of noise, which can degrade their quality. Noise can be introduced through sensor anomalies, low light conditions, or during the transmission of the image itself. Our project offers noise reduction functionality using median and Gaussian blurring techniques, which are highly effective at smoothing out image noise. This feature is essential for improving the usability of images in analysis and

**2**

presentation, especially in fields like medical imaging where clarity is paramount.

1. **Image Sharpening**: To counteract the blurring that might occur during capture or processing, our system provides an image sharpening feature. Using a convolutional kernel, the application enhances the visibility of edges and fine details in images, which is crucial for applications that rely on high levels of detail, such as satellite imaging and advanced photo editing.
2. **Image Segmentation**: Our application uses k-means clustering for image segmentation, which simplifies the image by dividing it into segments based on color similarity. This functionality is particularly useful in scenarios like object recognition, autonomous driving, and medical diagnostics, where identifying distinct sections of an image is essential.

**The Need for Our Project in Today's World**

The demand for real-time image processing continues to grow as industries and technologies evolve. With more devices connected to the internet and higher resolutions becoming standard, the data generated by images has increased dramatically. Efficiently managing this data through compression, noise reduction, and segmentation not only aids in storage management but also enhances the performance of machine learning models that rely on high-quality, detailed images. Our project addresses these needs by integrating powerful image processing tools with machine learning techniques.

**Chapter No – 2 3**

LITERATURE REVIEW

There The goal of image compression is to reduce the file size of an image. We can use K-means to select *k* colours to represent an entire image. This allows us to represent an image using only *k* colours, instead of the entire RGB space. This process is also referred to as *image quantization* [1].

The purpose of using k-means for image compression is to select *k* colours to represent a target image with the least approximation error. In other words, we will be using k-means to find the **best** *k* colour to represent a target image with.

The k-means algorithm is computationally efficient (linear time complexity), making it suitable for real-time image compression applications [2]. This also means it can handle large images.

NOISE REDUCTION Noise removal is a critical task in image processing, enhancing the visual quality of images by reducing unwanted variations or distortions known as noise. Two common techniques for noise removal are the Gaussian filter and the median filter, each with distinct characteristics and applications.

**Gaussian Filter**: This filter uses a Gaussian function to create a smoothing effect. It works by averaging the pixels under the normal curve centered around each target pixel in an image[3]. The Gaussian filter excels in reducing Gaussian noise, which is common in digital images, caused by poor illumination, high temperature, or transmission errors. This filter is particularly useful for environments where noise is consistent and spread normally throughout the image. However, it tends to blur sharp edges, reducing the clarity of details.

**Median Filter**: Unlike the Gaussian filter, the median filter sorts all the pixel values from the surrounding neighbourhood into numerical order and replaces the central pixel with the median of that sequence. This technique is highly effective at removing salt-and-pepper noise without reducing the sharpness of the image edges. Since the median filter replaces pixel values based on the ranking rather than averaging, it preserves useful details while effectively reducing noise, making it suitable for applications where maintaining edge integrity is important. **4**

Both filters are fundamental in image preprocessing, preparing images for further analysis or improving visual aesthetics. Choosing between them depends largely on the specific noise characteristics and the importance of preserving edge details in the application

Image sharpening using OpenCV's **filter2D()** function involves enhancing the edges and fine details of an image to make it appear more distinct and visually clearer. The process uses a convolution operation, where a kernel (a small matrix) is applied to the image. This kernel is designed to highlight edges by increasing the contrast between adjacent pixels that differ in intensity, which makes the image look less blurry and more detailed.

This kernel adjusts the central pixel by adding a weighted sum of itself and its neighbours. The negative weights reduce blurring by subtracting a fraction of adjacent pixel values, which emphasizes boundaries and transitions. The central value (often greater than 1) enhances the brightness of the pixel where sharp edges are detected.

When **filter2D()** is called, it convolves the image with this kernel, passing over each pixel, applying the sharpening effect based on the values defined in the kernel. This technique is particularly effective for images where detail clarity is required but can introduce noise in relatively uniform areas. Thus, it's often used judiciously to strike a balance between sharpness and overall image quality.

Image segmentation using K-means clustering involves partitioning an image into segments based on pixel color similarity. The K-means algorithm groups pixels into 'k' clusters where each cluster represents a segment. Each pixel is assigned to a cluster with the closest mean color value, effectively reducing the color palette of the image to 'k' colors. This process iterates, adjusting the cluster centers (mean values) until the clusters stabilize and no further changes occur. The result is an image segmented into regions of similar colors, which simplifies the image for analysis, such as in object recognition or boundary detection tasks.

**Chapter No – 3 5**

IMAGE COMPRESSION USING K MEANS CLUSTERING

Image compression using K-means clustering is a widely recognized technique for reducing the color space of an image, effectively lowering its storage size without excessively degrading its visual quality. The principle behind this method is to partition the set of colors in the image into a few clusters, each represented by a centroid, which then replaces all colors in that cluster in the image. This technique is grounded in vector quantization, a classic method for lossy data compression.

The Image compression using K-means clustering involves the following steps:

1. **Colour Space Conversion**: Convert the image from RGB to a different colour space if necessary (often to improve clustering results).
2. **Clustering**: Apply K-means clustering to the colour data points (pixels), where each pixel is treated as a point in a three-dimensional colour space.
3. **Quantization**: Each original colour in the image is replaced with the nearest cluster centroid. This significantly reduces the number of distinct colours in the image [4].

In the context of image compression, the centroids are the colours we are using to represent the compressed image. Therefore, our first step is to read in the image and select k random colours from the image to initialize our centroids. We read in the image using numpy. This produces a 2dimensional array where each element is a list of length 3 that represents the RGB values of that pixel. Remember to modify image path to your own. we define the function to initialize our centroids. We select a random pixel from the image and add its corresponding RGB value to the centroids init array.



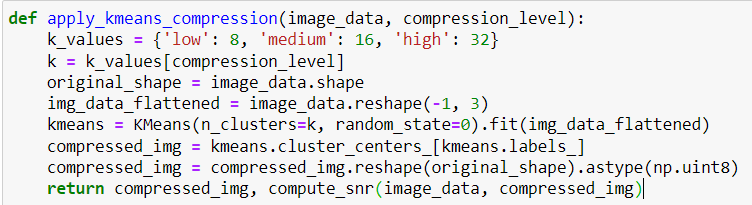
**6**

In the image compression, this means assigning each pixel of the target image to the centroid colour that is nearest to it.

we are creating the dictionary centroid RGBs. Each key corresponds to an index of the centroids and the values are a single numpy array that contain all of the colours assigned to the corresponding centroid. The assignment of each pixel to a centroid is done on line 13 using linalg.normto calculate the euclidean distance to each centroid and then using argminto find the index of the nearest centroid.

Creating the Compressed Image

Now that we have the finalized centroids, we can create the compressed image. We simply iterate through each pixel and change its colour to the nearest centroid.



**(i)Code for image compression using K means clustering**

The **apply\_kmeans\_compression** function in the provided code compresses an image using the K-means clustering algorithm based on a specified compression level. It maps compression levels ("low", "medium", "high") to different numbers of clusters (**k** values of 8, 16, and 32, respectively). The function first reshapes the image data into a 2-dimensional array where each row represents a pixel with three color values (RGB). Then, it applies K-means clustering to this array to group similar colors together. Each pixel’s color in the original image is replaced by the color of the centroid of its cluster, effectively reducing the color palette of the image. The compressed image is reshaped back to its original dimensions and converted to an unsigned 8-bit integer format. Additionally, the function calculates the Signal-to-Noise Ratio (SNR) to quantify the quality of the compressed image relative to the original.

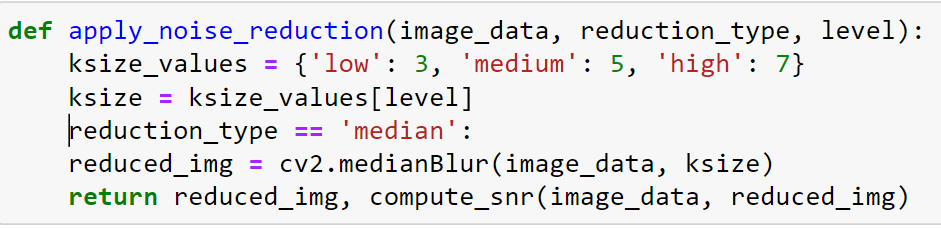
**Chapter No – 4 7**

NOISE REDUCTION USING MEDIAN FILTER

Noise reduction is a crucial preprocessing step in image processing, especially for improving the results of further image analysis such as edge detection, segmentation, and object recognition. One of the effective techniques for noise reduction is the application of the median filter, which is particularly good at removing "salt-and-pepper" noise from images. The median filter replaces each pixel's value with the median value of the intensities in the neighborhood defined by the kernel size.

[5] median filtering is a non-linear process useful in reducing impulsive, salt-and-pepper noise because the median is less sensitive than the mean to extreme values (outliers). This property makes median filtering better suited for certain types of noise that do not affect all pixels but instead cause extreme values in some pixels.

the median filter operates by moving through each element of the signal (in this case, the image pixels), replacing each element with the median of neighbouring elements. The neighbourhood is defined by the kernel size, and the median is calculated by first sorting all the pixel values from the surrounding neighbourhood into numerical order and then replacing the value of the central pixel with the median of that sequence.

** (ii) Code for image noise reduction using median filter**

the apply\_noise\_reduction function reduces noise in an image using OpenCV's medianBlur(). It selects the kernel size (ksize) based on the specified level ("low", "medium", "high"), applying a median filter to blur and denoise the image. The function also calculates and returns the Signal-to-Noise Ratio (SNR) to evaluate the quality of noise reduction

**Chapter No – 5 8**

IMAGE SEGMENTATION USING K MEANS CLUSTERING

Image segmentation using K-means clustering is a popular technique employed in computer vision for dividing an image into multiple segments (sets of pixels, also known as super pixels).

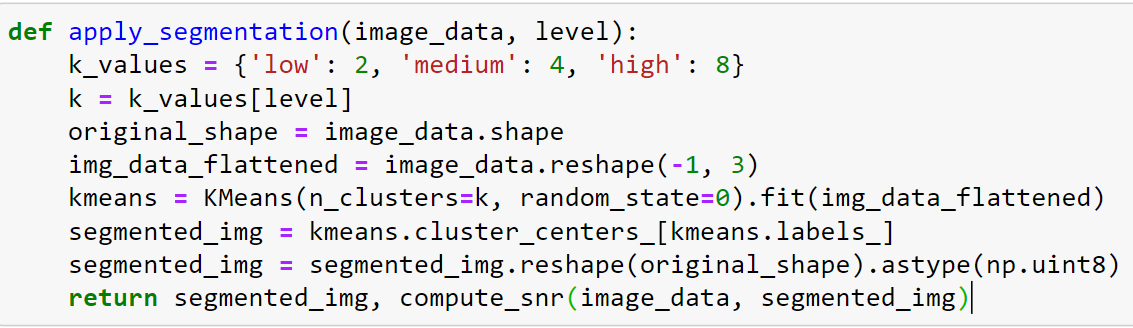
**Basic Concept of K-means Clustering for Image Segmentation:**

K-means clustering algorithm assigns each pixel in the image to the nearest cluster center based on the Euclidean distance between colour vectors or intensity. The centers of the clusters (centroids) are initially chosen randomly, and the algorithm iteratively refines these centers by minimizing the sum of squared differences between each pixel and its corresponding centroid. This process segments the image into k different regions, where each region represents pixels that have similar attributes.

It divides an image into several distinct regions, so that the pixels are

extremely similar In each region, and high regional contrast In several fields, it is a powerful resource like

health care, image processing, traffic signal, pattern recognition etc[6].



**(iii) code for image segmentation**

**9**

The **apply\_segmentation** function in the above code segmentates an image using the K-means clustering algorithm . The function allows the selection of different segmentation intensities, defined by **level**, which directly correlates to the number of clusters (**k**) used in the segmentation: "low" (2 clusters), "medium" (4 clusters), and "high" (8 clusters). This determines how many unique color groups the image will be divided into.

The function starts by reshaping the image into a 2-dimensional array where each row represents a pixel and the columns represent RGB values. K-means clustering is then applied to group these pixels into **k** clusters based on their colour similarity. Each pixel is assigned the colour of the centroid of the cluster it belongs to, effectively segmenting the image into regions of uniform colour.

After segmentation, the image is reshaped back to its original dimensions and converted to 8-bit unsigned integer format. Additionally, the function calculates the Signal-to-Noise Ratio (SNR) to assess the preservation of the original image's content relative to the noise introduced by the segmentation process. The function returns the segmented image and its SNR, providing both the processed image and a metric for evaluating the segmentation quality.

**Chapter No – 6 10**

IMAGE SHARPENING USING OPENCV

Image sharpening is a common image processing technique that enhances the visibility of edges and fine details in an image. In OpenCV, one effective method to achieve image sharpening is by using a convolution operation with a specific sharpening kernel. This operation is often implemented using the **filter2D()** function from OpenCV.

**Concept of Image Sharpening**

The idea behind image sharpening is to accentuate the transitions in intensity at the edges of objects within an image. This is typically done by applying a high-pass filter that amplifies high-frequency components of the image. A common approach involves subtracting a portion of a blurred (low-pass filtered) version of the image from the original image, emphasizing edges and other high-frequency components.

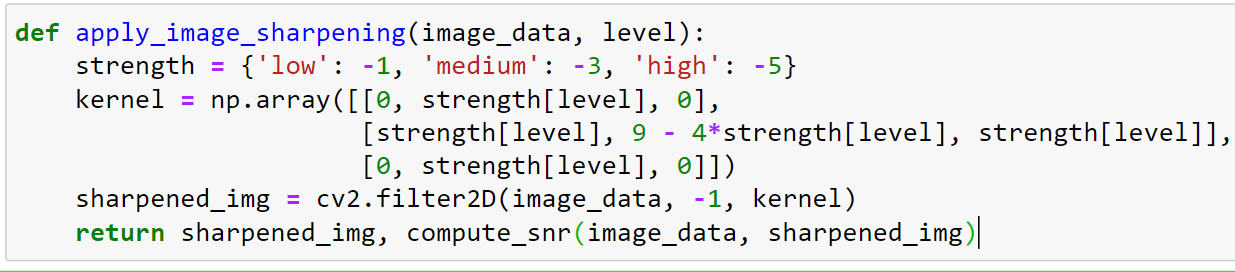
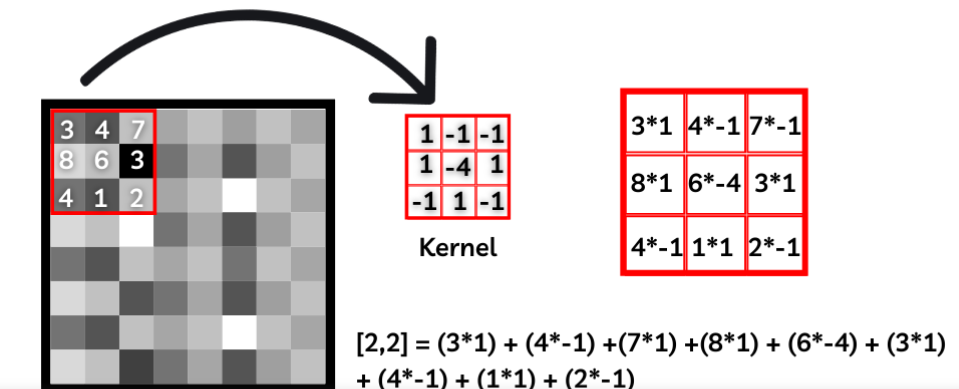
While dealing with images in Image Processing, filter2D() function is used to change the pixel intensity value of an image based on the surrounding pixel intensity values. This method can enhance or remove certain features of an image to create a new image. Using this function, we can create a convolution between the image and the given kernel for creating filters like smoothing and blurring, sharpening, and edge detection in an image. This function will simply convolute the 2d matrix with the image at pixel level and produce an output image. To understand this concept, we shall first skim through the concept of the kernel.

**Kernel:** A simple 2d matrix used in convolution or Convolution Matrix or a mask used to blur, sharpen and edge detect an image[7].

**Working of the kernel**: So, how this kernel works? Let’s see, we all know that images are represented as pixel values in OpenCV. These pixels are arranged as a matrix to form an image and as we know that a kernel is a simple 2d matrix with specific values in it based on the function of the kernel like if the kernel is used for blurring and sharpening the images are different. Let us take an example, In this image take the first 3 rows and columns like a matrix and we have a kernel of 3 by 3 matrix. Now the

**11**

convolution is done by multiplying the values of each pixel value with the kernel value in the respective places and adding all the values which were just weighted by the kernel by multiplying and forming one pixel (the center pixel which in this case it is [2,2]). And this method is repeated for the rest of the pixel value matrices in the image.



**(iv) Code for image sharpening**

The **apply\_image\_sharpening** function sharpens an image by applying a convolutional kernel, which enhances the contrast at the edges between differing areas of brightness. It accepts an **image\_data** input and a **level** of sharpening ('low', 'medium', 'high'). The function defines the strength of the sharpening through a dictionary that adjusts based on the selected level. This strength modifies a predefined kernel array, which accentuates edges by increasing pixel contrast relative to their neighbors. Using OpenCV’s **filter2D()** method, this kernel is convolved with the original image, thereby enhancing edges and making the image appear more defined. Additionally, the function calculates the Signal-to-Noise Ratio (SNR) of the sharpened image compared to the original, providing a metric for the effectiveness of the sharpening .

**Chapter No – 7 12**

WORKING AND RESULT ANALYSIS

**Working of the model consist of following Image Processing Functions:**

* **compute\_snr**: Calculates the Signal-to-Noise Ratio (SNR) of images to evaluate the quality of processing.
* **apply\_kmeans\_compression**: Compresses the image by reducing the color space using k-means clustering.
* **apply\_noise\_reduction**: Reduces noise from the image using either median or Gaussian blurring.
* **apply\_image\_sharpening**: Enhances image sharpness using a convolution kernel.
* **apply\_segmentation**: Segments the image by clustering pixel colours into fewer distinct colours, again using k-means.

This Flask web application provides an interactive web interface where users can upload images and apply different image processing operations such as compression, noise reduction, sharpening, and segmentation. Here's how a user can perform various operations using the web interface, step by step:

**Step 1: Accessing the Application**

* **Open the Application**: The user needs to navigate to the URL where the Flask application is hosted (typically something like **http://localhost:5000** if it's running on a local machine).

**Step 2: Uploading an Image**

* **Upload an Image**: On the main page (**index.html**), the user will find an option to upload an image file. There should be a file input field where users can browse their computer and select an image file to upload.

**13**

* **Select Operation**: Below or next to the upload field, there are options (likely in the form of dropdown menus or radio buttons) to choose the desired image processing operation:
* Compression
* Noise Reduction
* Sharpening
* Segmentation

**Step 3: Configuring the Operation**

* **Choose the Level of Processing**: Depending on the operation selected, the user might need to specify the intensity or level of the processing (low, medium, high). This choice affects how aggressive the processing will be.
* **Additional Options**: For noise reduction, the user also chooses the type of noise reduction technique (either median or Gaussian blur).

**Step 4: Submitting the Request**

* **Submit the Form**: After configuring all options, the user submits the form by clicking a submit button. This sends the image and the selected options to the server.

**Step 5: Processing the Image**

* **Server-Side Processing**: The server receives the image and the user's selections, processes the image as specified, and saves the processed image in the designated output directory. The Signal-to-Noise Ratio (SNR) of the processed image is also calculated to provide feedback on the quality of the processing.

**Step 6: Viewing Results**

* **Displaying the Processed Image**: Once processing is complete, the user is redirected to a results page (likely **result.html**). This page displays the processed image along with the SNR value. There should be a link or an embedded image showing the processed result.

**14**

* **Download or View**: The user has the option to view the processed image in their browser or download it directly by clicking on the provided link.

**Step 7: Additional Interactions**

* **Return and Process New Image**: If the user wishes to process another image, they can navigate back to the home page (using a provided link or button) and repeat the process with a new image and different settings.

**RESULT AND ANALYSIS**

IMAGE COMPRESSION RESULTS:



Original image (337kb)

****

**Low K value Compressed image (222kb)** SNR: 2.489220207484472Db

**15**

****

**Medium K value Compressed image (242kb), SNR:** 5.4816042060 Db

****

**High K value Compressed image (252kb), SNR:** 5.48160420604809 dB

NOISE REDUCTION RESULTS: **16**

****

**Original image with noise**

****

**After Low level noise reduction**

**17**

****

**After Medium level noise reduction**

****

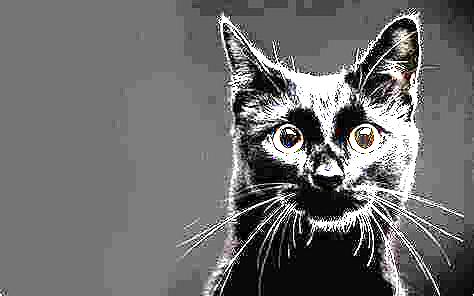
**After high level noise reduction**

Top of Form

IMAGE SHARPENING RESULTS: **18**

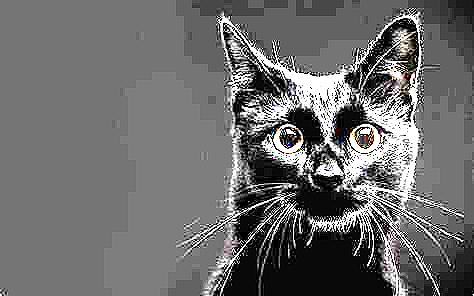


Original image



After Low level image sharpening SNR: 2.6033451667368146 dB

**19**



After medium level image sharpening SNR: 2.6709456184304843 dB



After high level image sharpening SNR: 2.743466031018242 dB

IMAGE SEGMENTATION RESULTS: **20**



Original M.R.I image



after low level image segmentation SNR: -0.7302878321882954 dB

**21**



after medium level image segmentation SNR: 0.39528384268923406 dB



**After high level image segmentation SNR**:1.5278748596560765 dB

CHAPTER – 8 **22**

CONCLUSION

The image processing process can extract some information from image, or get enhanced version of an image. But in order to optimize your workflow and avoid waste of time, it is very important to capture the image and process it afterwards, followed by image detection as a post-processing step. The combination of K-means clustering with OpenCV's median blur and filter2D functions presents a comprehensive approach to image enhancement, offering both noise reduction and sharpening capabilities. The results demonstrate the benefits from compressing the images by reducing the size of the images while preserving the acceptable quality of the compressed images.

* **K-means Clustering**: Provides a powerful method for segmenting of images into clusters based on pixel similarity, facilitating various image enhancement tasks by isolating regions of interest.
* **Median Blur**: Effectively reduces noise, particularly salt-and-pepper noise, preserving edges and fine details while enhancing the overall quality of the image.
* **Filter2D (Image Sharpening)**: Enhances the edges and details in the image, making it appear sharper and clearer by emphasizing high-frequency components.

**Benefits of the Combined Approach:**

* Comprehensive Enhancement: By integrating K-means clustering with noise reduction (median blur) and image sharpening (filter2D), the overall image enhancement process becomes more comprehensive, addressing both noise reduction and edge enhancement.
* Preservation of Structural Information: The combination ensures that important structural information in the image is preserved while enhancing its visual quality.
* Adaptability: The approach can be adapted to different types of images and applications, providing flexibility in image enhancement tasks.
* Image segmentation have wide range of use in medical field such as in MRI.

CHAPTER – 9  **23**

FUTURE WORKS

* Parameter Optimization: Investigate optimal parameter settings for K-means clustering (e.g., number of clusters), median blur (kernel size), and filter2D (sharpening kernel) to achieve the best results for different types of images and noise characteristics.
* Real-time Implementation: Explore techniques for optimizing the computational efficiency of the combined approach, enabling real-time image enhancement applications, such as video processing or live streaming.
* Integration with Deep Learning: Investigate the integration of deep learning techniques, such as convolutional neural networks (CNNs), for more advanced image enhancement tasks, leveraging the power of learned representations to further improve visual quality and noise reduction.
* Evaluation Metrics: Develop quantitative evaluation metrics to assess the effectiveness of the combined approach in terms of image quality, noise reduction, and edge enhancement, enabling systematic comparison with existing methods.
* Features addition : more feature addition such as Image restoration , Wavelength and multicolour processing, Morphological processing ,Representation and Description and Object recognition for making a complete digital image processing tool.

CHAPTER – 10 **24**

REFERENCES

[1] A. Munshi et al., “Image compression using K-mean clustering algorithm,” IJCSNS International Journal of Computer Science and Network Security, vol. 21, no. 9, p. 275, 2021, https://doi:10.22937/IJCSNS.2021.21.9.36

[2]. Chiou, Paul T., et al. “A Complexity Analysis of the JPEG Image Compression Algorithm.” 2017 9th Computer Science and Electronic Engineering (CEEC), 2017, https://doi.org/10.1109/ceec.2017.8101601.

[3]OpenCV Official documentation

<https://docs.opencv.org/3.4/d4/d13/tutorial_py_filtering.html>

[4] Gonzalez, R. C., & Woods, R. E. (2002). Digital Image Processing (2nd ed.). Prentice Hall. ISBN: 0-201-18075-8

[5] Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2009). Digital Image Processing Using MATLAB. Gatesmark Publishing. ISBN: 9780977858234.

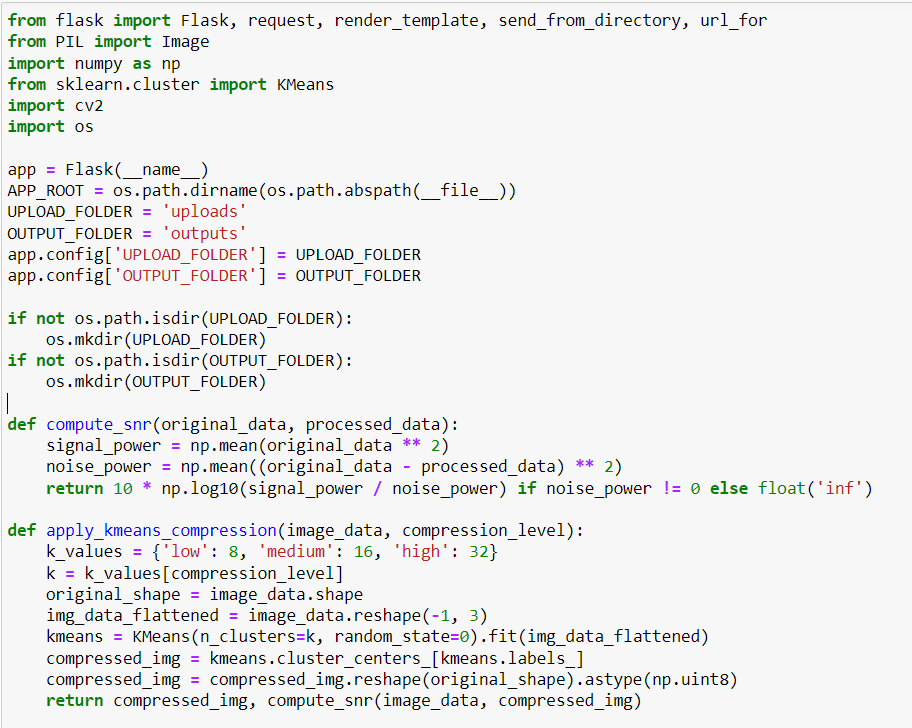
[6].F. jian. and M.R.Frater, “Threshold-based image segmentation through an improved particle swarm optimization,” in Digital image computing techniques and applications (DICTA),2012.

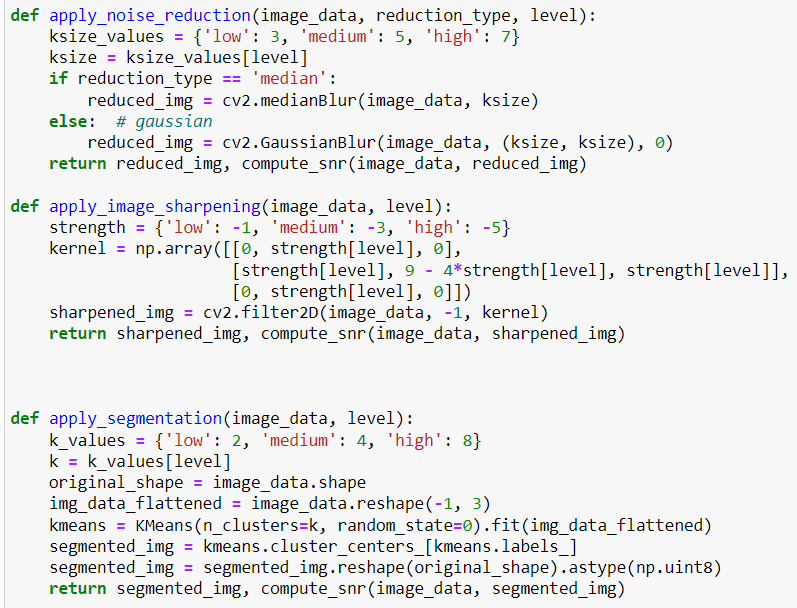
[7]. filter2D()Official Documentation

https://docs.opencv.org/3.4/d4/dbd/tutorial\_filter\_2d.html

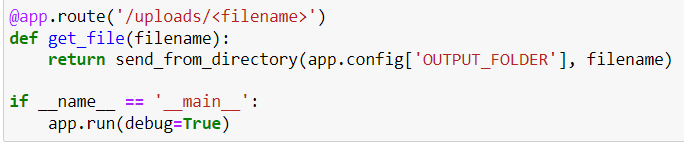
APPENDIX

app.py code:



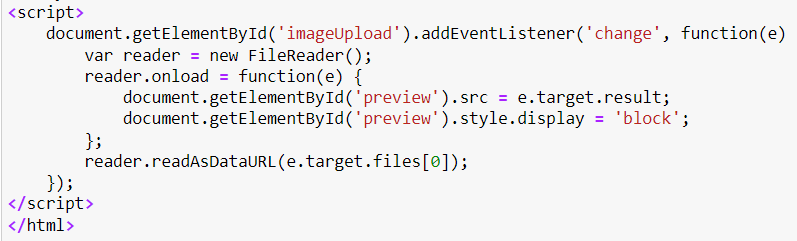






Index.html code:





Result.html:



Style.css:

