SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)
SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF MEDICAL ELECTRONICS

LABORATORY MANUAL

MD3664 – Medical Image Processing Laboratory

Regulation –2023

VI Semester

B.E. Medical Electronics

Academic Year 2025 – 26 (Even Semester)

Prepared by

Mr. M.Selvaraj, AP/MDE

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF MEDICAL ELECTRONICS

VISION OF THE INSTITUTE

- Educate to excel in social transformation

MISSION OF THE INSTITUTE

- To contribute to the development of human resources in the form of professional engineers and managers of international excellence and competence with high motivation and dynamism, who besides serving as ideal citizen of our country will contribute substantially to the economic development and advancement in their chosen areas of specialization.
- To build the institution with international repute in education in several areas at several levels with specific emphasis to promote higher education and research through strong institute-industry interaction and consultancy.

VISION OF THE DEPARTMENT

- To provide quality education for improving the healthcare and well-being of human kind.

MISSION OF THE DEPARTMENT

- **M1:** To inculcate students with fundamental knowledge, interdisciplinary problem-solving skills and confidence required to excel in Medical Electronics
- **M2:**. To up skill the students with the current technological trends and carryout quality research to meet the expectation of healthcare service sectors.
- **M3:** To instil creativity, responsibility, commitment and leadership qualities with professional ethics and moral values.

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF MEDICAL ELECTRONICS

PROGRAM OUTCOMES

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

By the completion of Medical Electronics program, the student will have following Program specific outcomes

1. Ability to apply the knowledge of engineering in solving the healthcare problems.

2. Ability to propose indigenous clinical solution through the application of their domain areas and emerging ICTs

Syllabus

MD3664 MEDICAL IMAGE PROCESSING LABORATORY

OBJECTIVES:

The student should be made:

- To practice the basic image processing techniques.
- To compute the spatial filtering techniques.
- To understand the concepts of frequency domain filtering.
- To know the concepts of image standards and analysis.
- To explore the applications of medical image processing techniques.

LIST OF EXPERIMENTS:

1. Display of color and grayscale Images.
2. Binary to Gray and Gray to binary conversion of images
3. Conversion between color spaces.
4. Histogram Equalization of image.
5. Image noise removal using spatial filtering process.
6. Non-linear Filtering of Image.
7. Edge detection using Operators.
8. Study of DICOM and NifTi standards.
9. Segmentation of Medical Image.
10. Feature extraction of medical images using entropy
11. Medical Image Compression techniques.
12. Medical image fusion.

COURSE OUTCOMES:

On completion of the course, the student will be able to

CO1: Perform fundamentals in the image processing operations.

CO2: Compute pixel level manipulations of the image.

CO3: Estimate the harmonic level analysis of the image.

CO4: Understand the medical image store and transmit procedures.

CO5: Apply image processing technique to medical standards.

List of Experiments:

CYCLE - I

1. Display of color and grayscale Images.
2. Binary to Gray and Gray to binary conversion of images
3. Conversion between color spaces.
4. Histogram Equalization of an image.
5. Image noise removal using a spatial filtering process.
6. Non-linear Filtering of an Image.

CYCLE - II

7. Edge detection using Operators.
8. Study of DICOM and NIfTI standards.
9. Segmentation of Medical Image.
10. Feature extraction of medical images using entropy
11. Medical Image Compression techniques.
12. Medical image fusion

TOTAL: 60 PERIO

INDEX

Experiment 1: Display of Color and Grayscale Images

Aim: To display color and grayscale images and visualize their channels and histograms.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory:
Images are represented as pixel matrices. Color images use three channels (RGB/BGR), and grayscale photos use a single intensity channel. Visualizing channels and histograms helps analyze color distribution and brightness variations.

Program:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img_path = 'sample_color.jpg'
img_bgr = cv2.imread(img_path)
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(12,6))
plt.subplot(2,3,1); plt.imshow(img_rgb); plt.title('Color (RGB)'); plt.axis('off')
plt.subplot(2,3,2); plt.imshow(img_gray, cmap='gray'); plt.title('Grayscale'); plt.axis('off')
r,g,b = img_rgb[:,:,0], img_rgb[:,:,1], img_rgb[:,:,2]
plt.subplot(2,3,3); plt.imshow(r, cmap='Reds'); plt.title('Red channel'); plt.axis('off')
plt.subplot(2,3,4); plt.imshow(g, cmap='Greens'); plt.title('Green channel'); plt.axis('off')
plt.subplot(2,3,5); plt.imshow(b, cmap='Blues'); plt.title('Blue channel'); plt.axis('off')
plt.subplot(2,3,6); plt.hist(img_gray.ravel(), bins=256); plt.title('Gray Histogram')
plt.tight_layout(); plt.show()
```

Result: Thus, the display of color and grayscale images was successfully performed and verified.

Experiment 2: Binary to Gray and Gray to Binary Conversion

Aim: To convert grayscale images to binary and reconstruct grayscale from binary.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Binary images have two intensity levels. Thresholding converts grayscale images to binary; binary images are then mapped back to the intensity format. Useful in segmentation and feature extraction.

Program:

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('sample_gray.jpg', cv2.IMREAD_GRAYSCALE)
_, binary_otsu = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
binary_adapt = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 2)
reconstructed = cv2.GaussianBlur(binary_otsu, (7,7), 0)
plt.figure(figsize=(10,5))
plt.subplot(1,4,1); plt.imshow(img, cmap='gray'); plt.title('Original Gray'); plt.axis('off')
plt.subplot(1,4,2); plt.imshow(binary_otsu, cmap='gray'); plt.title('Binary (Otsu)'); plt.axis('off')
plt.subplot(1,4,3); plt.imshow(binary_adapt, cmap='gray'); plt.title('Binary (Adaptive)'); plt.axis('off')
plt.subplot(1,4,4); plt.imshow(reconstructed, cmap='gray'); plt.title('Smoothed (Binary->Gray)');
plt.axis('off')
plt.tight_layout(); plt.show()
```

Result: Thus, the gray-to-binary and binary-to-gray conversions were successfully implemented and verified.

Experiment 3: Conversion Between Color Spaces

Aim: To convert images between RGB, HSV, LAB, and YCrCb color spaces.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Color spaces separate intensity and color for processing tasks. RGB is for display, HSV separates hue, LAB separates lightness, and YCrCb separates luminance. Conversion aids segmentation, enhancement, and analysis.

Program:

```
import cv2
import matplotlib.pyplot as plt
bgr = cv2.imread('sample_color.jpg')
rgb = cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB)
hsv = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)
lab = cv2.cvtColor(bgr, cv2.COLOR_BGR2LAB)
ycrcb = cv2.cvtColor(bgr, cv2.COLOR_BGR2YCrCb)
plt.figure(figsize=(12,6))
plt.subplot(2,3,1); plt.imshow(rgb); plt.title('RGB'); plt.axis('off')
plt.subplot(2,3,2); plt.imshow(cv2.cvtColor(hsv, cv2.COLOR_HSV2RGB)); plt.title('HSV->RGB');
plt.axis('off')
plt.subplot(2,3,3); plt.imshow(cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)); plt.title('LAB->RGB');
plt.axis('off')
plt.subplot(2,3,4); plt.imshow(cv2.cvtColor(ycrcb, cv2.COLOR_YCrCb2RGB)); plt.title('YCrCb-
>RGB'); plt.axis('off')
plt.subplot(2,3,5); plt.imshow(hsv[:,:,0], cmap='gray'); plt.title('Hue'); plt.axis('off')
plt.subplot(2,3,6); plt.imshow(hsv[:,:,1], cmap='gray'); plt.title('Saturation'); plt.axis('off')
plt.tight_layout(); plt.show()
```

Result: Thus, the conversion between RGB, HSV, LAB, and YCrCb color spaces was successfully performed and verified.

Experiment 4: Histogram Equalization of an Image

Aim: Enhance image contrast using histogram equalization.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images
Theory: Histogram equalization redistributes intensity values to improve visibility. Global equalization works on the entire image; CLAHE works locally to avoid over-enhancement.

Program:

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('low_contrast.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
eq_global = cv2.equalizeHist(gray)
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
eq_clahe = clahe.apply(gray)
plt.figure(figsize=(12,6))
plt.subplot(1,3,1); plt.imshow(gray, cmap='gray'); plt.title('Original'); plt.axis('off')
plt.subplot(1,3,2); plt.imshow(eq_global, cmap='gray'); plt.title('Global Equalization'); plt.axis('off')
plt.subplot(1,3,3); plt.imshow(eq_clahe, cmap='gray'); plt.title('CLAHE'); plt.axis('off')
plt.tight_layout(); plt.show()
```

Result: Thus, histogram equalization was successfully performed, and the enhancement in image contrast was observed.

Experiment 5: Image Noise Removal Using Spatial Filtering

Aim: Remove noise using mean and Gaussian filters.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Mean filters average neighboring pixels; Gaussian filters weigh central pixels more. These remove random noise while maintaining structure.

Program:

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('noisy_image.jpg', cv2.IMREAD_GRAYSCALE)
mean = cv2.blur(img, (5,5))
gauss = cv2.GaussianBlur(img, (5,5), 1.0)
plt.subplot(1,3,1); plt.imshow(img, cmap='gray'); plt.title('Original'); plt.axis('off')
plt.subplot(1,3,2); plt.imshow(mean, cmap='gray'); plt.title('Mean Filter'); plt.axis('off')
plt.subplot(1,3,3); plt.imshow(gauss, cmap='gray'); plt.title('Gaussian Filter'); plt.axis('off')
plt.show()
```

Result: Thus, image noise was successfully removed using mean and Gaussian filters, improving the image smoothness.

Experiment 6: Non-Linear Filtering of an Image

Aim: Apply median and bilateral filters to remove noise while preserving edges.

Apparatus Required:

Computer with Python IDE

Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Median filter replaces pixel with neighborhood median (removes salt-pepper noise). A bilateral filter smooths while keeping edges sharp.

Program:

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread('salt_pepper.jpg', cv2.IMREAD_GRAYSCALE)
median = cv2.medianBlur(img,5)
bilateral = cv2.bilateralFilter(img,9,75,75)
plt.subplot(1,3,1); plt.imshow(img, cmap='gray'); plt.title('Original'); plt.axis('off')
plt.subplot(1,3,2); plt.imshow(median, cmap='gray'); plt.title('Median Filter'); plt.axis('off')
plt.subplot(1,3,3); plt.imshow(bilateral, cmap='gray'); plt.title('Bilateral Filter'); plt.axis('off')
plt.show()
```

Result: Thus, the image was successfully filtered using non-linear techniques, and edge preservation was observed.

Experiment 7: Edge Detection Using Operators

Aim: Detect edges using Sobel, Prewitt, Roberts, and Canny operators.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Edges mark intensity transitions. Gradient operators detect horizontal/vertical changes. Canny detects refined edges with smoothing and thresholding.

Program:

```
import cv2
import matplotlib.pyplot as plt
from skimage.filters import prewitt, roberts
img = cv2.imread('edges_sample.jpg', cv2.IMREAD_GRAYSCALE)
sobel = cv2.Sobel(img,cv2.CV_64F,1,1,ksize=3)
prew = prewitt(img)
rob = roberts(img)
canny = cv2.Canny(img,50,150)
plt.subplot(2,2,1); plt.imshow(sobel, cmap='gray'); plt.title('Sobel'); plt.axis('off')
plt.subplot(2,2,2); plt.imshow(prew, cmap='gray'); plt.title('Prewitt'); plt.axis('off')
plt.subplot(2,2,3); plt.imshow(rob, cmap='gray'); plt.title('Roberts'); plt.axis('off')
plt.subplot(2,2,4); plt.imshow(canny, cmap='gray'); plt.title('Canny'); plt.axis('off')
plt.show()
```

Result: Thus, the edges of the given image were successfully detected and verified using Sobel and Canny operators.

Experiment 8: Study of DICOM and NIfTI Standards

Aim: To study about DICOM and NIfTI medical images.

DICOM (Digital Imaging and Communications in Medicine)

DICOM is the international standard for medical images and related information, facilitating communication and exchange across different modalities and equipment manufacturers in a clinical setting.It is a complex, metadata-rich format where a single file typically contains one 2D image slice along with extensive header information including patient demographics, acquisition parameters, and clinical workflow details.

- **Key Features:**
  - **Comprehensive Metadata:** The rich header ensures data integrity and interoperability across hospital systems (PACS).
  - **Vendor Variations:** Different manufacturers (Siemens, Philips, GE) often use proprietary "private tags" within the standard, which can complicate data exchange and conversion.
  - **File Organization:** A single imaging study often results in a large number of individual DICOM files.
  - **Coordinate System:** Uses a patient coordinate system (PCS) with specific orientation standards (LPS: Left, Posterior, Superior positive).

NIfTI (Neuroimaging Informatics Technology Initiative)

NIfTI is a neuroimaging-specific format designed for analysis beyond clinical workflow, widely adopted in the research community for its simplicity and efficiency. It is a minimalist format that stores image data as a 3D or 4D volume (e.g., three spatial dimensions plus time) in a single file (`.nii` or compressed `.nii.gz`), along with a simpler, fixed-byte header. An optional JSON "sidecar" file can store additional metadata (following the BIDS standard) if needed.

- **Key Features:**
  - **Simplified Analysis:** NIfTI's straightforward structure enables faster processing and is compatible with most neuroimaging analysis software.
  - **Efficient Storage:** It generally results in smaller file sizes than large DICOM datasets, simplifying data management.
  - **Coordinate System:** Uses a right-handed coordinate system (RAS: Right, Anterior, Superior positive) that is more consistent for research purposes.

- **Origin:** The origin (0,0,0) is typically set to the anterior commissure after registration to a template, facilitating cross-study comparisons.

Comparison:

Feature	DICOM	NIfTI
Primary Use	Clinical workflow, storage, and transfer	Neuroimaging research and analysis
Metadata Richness	Extensive (patient info, acquisition details, etc.)	Minimalist, focused on image parameters
File Structure	Many files (one per slice), complex	Single file (volume), simple header
Image Dimension	Typically 2D slices	Typically 3D or 4D volumes
File Size	Can be large and complex	Smaller and more efficient for analysis

Result: Thus, the DICOM and NIfTI standards were successfully studied, and the formats were understood.

Experiment 9: Segmentation of Medical Image

Aim: Segment medical images using thresholding and watershed.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Segmentation partitions images into regions or objects. Thresholding separates by intensity. Watershed treats the image as topography, identifying boundaries. Used for organ/tissue isolation.

Program:

```
import cv2
import numpy as np
from skimage import morphology, segmentation, measure
from scipy import ndimage as ndi
img = cv2.imread('medical_slice.png', cv2.IMREAD_GRAYSCALE)
_, th = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
clean = morphology.remove_small_objects(th.astype(bool), min_size=500).astype('uint8')*255
dist = ndi.distance_transform_edt(clean)
markers = measure.label(morphology.local_maxima(dist))
labels = segmentation.watershed(-dist, markers, mask=clean)
plt.imshow(labels, cmap='nipy_spectral'); plt.title('Watershed Segmentation'); plt.axis('off'); plt.show()
```

Result: Thus, medical image segmentation was successfully performed and verified through thresholding.

Experiment 10: Feature Extraction Using Entropy

Aim: Extract features of medical images using entropy.

Apparatus Required:

Computer with Python IDE

Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Feature extraction reduces image data to measurable properties. Entropy quantifies intensity, randomness, or texture complexity. High entropy regions often indicate detailed structures or abnormalities. Useful in tissue classification and tumor detection.

Program:

```
import cv2
import numpy as np
from skimage.measure import shannon_entropy
img = cv2.imread('medical_slice.png', cv2.IMREAD_GRAYSCALE)
entropy_val = shannon_entropy(img)
print('Entropy of image:', entropy_val)
```

Result: Thus, entropy-based feature extraction was successfully performed, and the texture information was verified.

Experiment 11: Medical Image Compression Techniques

Aim: Compress medical images using lossless and lossy methods.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Compression reduces file size. Lossless preserves all data (diagnostic safe), lossy reduces size by discarding minor info. Wavelet-based methods maintain structural integrity while reducing storage needs.

Program:

```
from PIL import Image
img = Image.open('medical_image.png')
img.save('compressed_lossless.tiff', format='TIFF', compression='tiff_deflate')
img.save('compressed_lossy.jpg', format='JPEG', quality=50)
```

Result: Thus, medical image compression was successfully implemented, and the quality reduction was verified.

Experiment 12: Medical Image Fusion

Aim: Fuse multiple medical images into a single image.

Apparatus Required:
Computer with Python IDE
Python libraries: OpenCV, Matplotlib, NumPy - Sample color and grayscale images

Theory: Image fusion combines structural and functional information from multiple modalities. Methods include pixel averaging and wavelet fusion. Enhances diagnostic accuracy by merging complementary details.

Program:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
ct = cv2.imread('ct.png', cv2.IMREAD_GRAYSCALE)
mri = cv2.imread('mri.png', cv2.IMREAD_GRAYSCALE)
fused = cv2.addWeighted(ct,0.5,mri,0.5,0)
plt.imshow(fused, cmap='gray'); plt.title('Fused Image'); plt.axis('off'); plt.show()
```

Result: Thus, medical image fusion was successfully performed, and the fused output was verified visually.