COMPARATIVE ANALYSIS OF IMAGE SEGMENTATION TECHNIQUES: GMM, KMEANS, DBSCAN, AND FUZZY C-MEANS

Supervisor

Dr Mohammed Abdelaziz

TEAM

Amira Yasser Ahmed	222100305
Nada Abdelkarim Ahmed	222101758
Abanoub Maged Ageb El Sayed	222100001
Ibrahim El Gaaly	A20000011
Abdelrahman Mohamed Negm	222200016
Mazen sherif Attia	222101273
Mohamed Rady salah	222200004

01 Introduction

In the ever-evolving field of computer vision, image segmentation and statistical analysis stand out as pivotal techniques for extracting meaningful information from visual data. Whether applied in medical diagnostics, autonomous driving, or digital image processing, these methods enable the partitioning of images into coherent regions and the extraction of valuable insights from pixel data.

This report outlines a robust Python-based framework designed to perform both image segmentation using advanced clustering algorithms and statistical analysis on grayscale images, with a particular focus on X-ray imagery. By leveraging powerful libraries such as OpenCV, scikit-learn, scikit-image, scikit-fuzzy, pandas, seaborn, and SciPy, the framework offers a versatile and efficient approach to comprehensive image processing tasks.

02 Framework Overview

Two primary pipelines:

1. Image Segmentation Pipeline: Uses clustering algorithms to partition images into distinct regions.
2. Statistical Analysis and Visualization Pipeline: Analyzes pixel intensity distributions, handles missing data, computes metrics, and generates visualizations.

03 Image Segmentation Pipeline

Objective

Divide images into meaningful segments for applications like object recognition and medical diagnostics.

Key Components

1. Image Loading and Preprocessing: Imports images, converts color spaces, and resizes images for uniform processing.
2. Feature Extraction: Extracts color, texture (using Local Binary Patterns), and edge information (via Sobel filters).
3. Dimensionality Reduction: Applies PCA to reduce the feature space to two dimensions for better clustering.
4. Clustering Algorithms: Implements algorithms like: Gaussian Mixture Models (GMM), K-Means, DBSCAN, Fuzzy C-Means
5. Visualization of Segmented Images: Creates color-coded images representing segmented regions.
6. Batch Processing: Processes multiple images in a directory, saving results for further analysis.

04 Statistical Analysis and Visualization Pipeline

Objective

Analyze pixel intensity distributions, particularly for X-ray images.

Key Components

1. Image Loading and Preprocessing: Imports images and converts to grayscale.
2. Handling Missing Values: Identifies missing or invalid pixel data and interpolates to fill gaps.
3. Statistical Calculations: Computes metrics like: Mean, median, mode, and skewness of pixel intensities.
4. Data Visualization: Generates visualizations including Histograms, boxplots, Q-Q plots, scatter plots, and heatmaps.

05 Implementation Details

Libraries and Dependencies:

1. OpenCV (cv2): For image loading, conversion, resizing, and saving segmented images.
2. NumPy: For efficient numerical operations and array manipulations.
3. Pandas: For data manipulation and structuring, especially in statistical analysis.
4. Matplotlib (plt) & Seaborn (sns): For creating a wide range of visualizations.
5. scikit-learn: For clustering and dimensionality reduction algorithms.
6. scikit-fuzzy: For fuzzy clustering techniques.
7. scikit-image (skimage): For advanced image processing features.
8. SciPy (stats and griddata): For statistical functions and interpolation.
9. Collections (Counter): For counting occurrences in pixel intensities.
10. Warnings: To manage and suppress warnings during execution.

Code Structure:

- Modular Design: Each component operates independently, promoting readability and scalability.
- Pipeline Separation: Image Segmentation and Statistical Analysis are kept separate for clarity.
- Function Encapsulation: Major processes are encapsulated within functions.
- Main Execution Blocks: Organized entry points using main() functions.
- Conditional Execution: Ensures scripts only execute main functions when run directly.

Usage Instructions

Prerequisites:

- Python 3.6 or higher is recommended.
- Install the required libraries via pip.

Directory Setup:

1. Image Segmentation Pipeline:
   - Input Folder: Directory with images for segmentation (e.g., d:\paper code\data for try\test\Caries).
   - Output Folder: Segmented images will be saved in a subdirectory (segmented images).
2. Statistical Analysis and Visualization Pipeline:
   - Image Path: Path to the X-ray image (e.g., d:\paper code\Caries\4.bmp).

Configuration:

1. Image Segmentation Pipeline:
   - Modify input folder path in the script.
   - Adjust the number of clusters (n_clusters).

2.Statistical Analysis and Visualization Pipeline:

- Update the image path.
- Set missing value criteria (missing_value).

Usage Instructions

Execution:

1.Image Segmentation Pipeline.

2.Statistical Analysis and Visualization Pipeline.

Optional Configuration:

- To save plots, uncomment plt.savefig() lines in the visualization sections.

Results and Evaluation

- Image Segmentation Results:
  - Segmented images saved for each clustering algorithm.
  - Expected Outcomes:
    - i.Gaussian Mixture Model (GMM): Smooth segmentation boundaries.
    - ii.K-Means: Distinct clusters based on minimizing variance.
    - iii.DBSCAN: Density-based clustering handling noise.
    - iv.Fuzzy C-Means: Overlapping clusters with membership probabilities.
- Comparative Insights: Compare segmentation methods for different data characteristics (e.g., Gaussian distributions for GMM, density-based clustering for DBSCAN).

Statistical Analysis Results:

- Key Metrics:
  - a.Mean Pixel Intensity
  - b.Median Pixel Intensity
  - c.Mode Pixel Intensity

d.Skewness

- Visualizations:

1.Histogram: Displays pixel intensity distribution with KDE.
2.Boxplot: Shows spread and central tendency with outliers.
3.Q-Q Plot: Assesses normality.
4.Scatter Plot: Visualizes intensity variations across pixels.
5.Heatmap: Visualizes intensity gradients and spatial patterns.

Usage Instructions

Insights:

1.Distribution shape (skewness) reveals image brightness characteristics.
2.Boxplot outliers can indicate anomalies.
3.Q-Q plot informs on normality for further preprocessing.
4.Heatmap and scatter plot detect spatial patterns and regions of interest.

06 Conclusion

The Python-based framework provides a comprehensive solution for:

- Image segmentation using advanced clustering.
- Statistical analysis of grayscale images.

07 Key Benefits

1.Modularity: Easy to maintain and scale.

2.Versatility: Supports various clustering algorithms and statistical analyses.
3.Visualization: Enhances data interpretation with multiple plots.

08 Statistical Analysis and Visualization Pipeline

- Additional Clustering Algorithms: Integrate Agglomerative Clustering, Spectral Clustering, or Deep Learning-based methods.
- Automated Parameter Optimization: Implement Grid Search or Bayesian Optimization for optimal parameter selection.
- Performance Optimization: Use parallel processing or GPU acceleration.
- User-Friendly Interface: Develop a GUI for easier interaction.
- Comprehensive Evaluation: Benchmark performance on diverse datasets.
- Application-Specific Adaptations: Adapt the framework for specialized domains like medical imaging or satellite imagery.
- Advanced Statistical Analyses: Add more statistical measures and hypothesis testing.
- Automated Reporting: Generate automated reports based on segmentation and analysis.

09 References

- Bradski, G., & Kaehler, A. (2008). Learning OpenCV.
- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python.
- Jolliffe, I. T. (2002). Principal Component Analysis.
- Comaniciu, D., & Meer, P. (2002). Mean Shift: A Robust Approach Toward Feature Space Analysis.
- Krishnapuram, R., & Keller, J. M. (1996). Image Segmentation Using the C-Means Clustering Algorithm.
- Möller, C., et al. (2004). Scikit-Fuzzy: Fuzzy Logic Toolbox for SciPy.
- Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment.
- Waskom, M. (2021). Seaborn: Statistical Data Visualization.

McKinney, W. (2010). Data Structures for Statistical Computing in Python.
Virtanen, P., et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.