Skin Lesion Analysis

By Umut Celik Email: umut.celik@cix.csi.cuny.edu

CSC 741 - Digital Image Processing Midterm Project

---

1. Introduction

The Challenge

- Automated analysis of dermoscopic images is crucial for early skin cancer detection.
- These images present challenges: varying illumination, hair, diverse skin tones, and subtle lesion characteristics.

Project Goal

- To develop an image processing pipeline to:
  1. Segment (isolate) skin lesions from dermoscopic images.
  2. Extract a comprehensive set of visual features from the segmented lesions.
- Apply fundamental Digital Image Processing techniques to a real-world medical imaging problem.

Dataset

- **Source:** SIIM-ISIC Melanoma Classification (Kaggle).
- **Content:** Dermoscopic images (JPEG format) and associated metadata (e.g., labels).

---

2. Image Processing Pipeline

A multi-step approach to process images and extract meaningful data:

Image Loading & Conversion -> Preprocessing -> Color Space Transformation -> Segmentation -> Feature Extraction

---

Step 1: Image Loading & Initial Conversion

- **Action:** Load JPEG images.
- **Conversion:** Convert from BGR (OpenCV default) to RGB for consistency in processing and display.
  - *Files involved:* `src/data_loader.py`

Step 2: Preprocessing

- **Grayscale Conversion:**
  - Convert RGB images to grayscale for intensity-based operations.
  - *Function:* `rgb_to_grayscale` *in* `src/color_utils.py`
- **Hair Removal:**
  - Technique: Morphological Black-Hat filtering to identify hair structures, followed by inpainting to remove them.
  - Reduces noise and artifacts caused by hair.
  - *Function:* `remove_hair` *in* `src/preprocessing.py`
- **(Illumination Correction):**
  - *Concept: Use morphological opening to estimate and correct non-uniform background illumination.*
  - *Status: Implemented (* `correct_illumination` *in* `src/preprocessing.py`*), but noted as potentially skipped in some display pipelines for directness to segmentation.*

Step 3: Color Space Transformation

- **RGB to HSV:**
  - Convert RGB images to Hue, Saturation, Value (HSV) color space.
  - HSV is often more intuitive for color-based feature extraction:
    - **H (Hue):** Dominant color.
    - **S (Saturation):** Purity/intensity of color.
    - **V (Value):** Brightness.
  - *Function:* `rgb_to_hsv` *in* `src/color_utils.py`

Step 4: Lesion Segmentation

- **Goal:** Create a binary mask that accurately isolates the lesion area.
- **Method: Otsu's Thresholding**
  - Applied to the preprocessed (hair-removed) grayscale image.
  - Automatically determines an optimal threshold value to separate lesion pixels from background.
  - *Function:* `otsu_threshold` *in* `src/segmentation.py`
- **Mask Refinement:**
  - Technique: Morphological operations.
    - **Opening:** Erosion followed by dilation (removes small noise/objects).
    - **Closing:** Dilation followed by erosion (fills small holes within the lesion).
  - Improves the quality and contiguity of the mask.
  - *Functions:* `opening`, `closing` *in* `src/morphology.py`, *utilized by* `apply_threshold` *in* `src/segmentation.py`

Step 5: Feature Extraction

Extracted from the segmented lesion area (defined by the refined mask):

Feature Extraction: Intensity Features

- **From Grayscale image:**
  - Mean pixel intensity
  - Standard deviation of pixel intensities
  - Minimum pixel intensity
  - Maximum pixel intensity
  - *Module:* `calculate_intensity_stats` *in* `src/feature_extraction.py`

Feature Extraction: Color Features (HSV)

- **From HSV channels:**
  - **Hue (H):**
    - Mean Hue (using circular mean calculation for angular data)
    - Standard deviation of Hue
  - **Saturation (S):**
    - Mean Saturation
    - Standard deviation of Saturation
  - **Value (V):**
    - Mean Value (Brightness)
    - Standard deviation of Value
  - *Module:* `calculate_hsv_stats` *in* `src/feature_extraction.py`

Feature Extraction: Border Features

- **From Grayscale image & mask:**
  - Calculated using morphological gradient on the lesion border.
  - Features include:
    - Mean, Standard Deviation, and Max of border gradient values.
    - Border Irregularity (ratio of border pixels to mask area).
  - *Module:* `calculate_gradient_features` *in* `src/border_texture_utils.py`

Feature Extraction: Texture Features

- **From Grayscale image & mask:**
  - Calculated using Morphological Top-Hat (highlights bright details) and Bottom-Hat (highlights dark details) transforms.
  - Features include:
    - Mean and Standard Deviation of Top-Hat transform values.
    - Mean and Standard Deviation of Bottom-Hat transform values.
    - Texture Contrast Index (sum of Top-Hat and Bottom-Hat means).
  - *Module:* `calculate_texture_features` *in* `src/border_texture_utils.py`

Feature Extraction: Histograms

- **Pixel intensity distributions calculated for:**
  - Grayscale channel
  - Hue (H) channel
  - Saturation (S) channel
  - Value (V) channel
- All calculated *within the masked lesion area*.
- *Module:* `calc_all_histograms` *in* `src/histogram_utils.py`

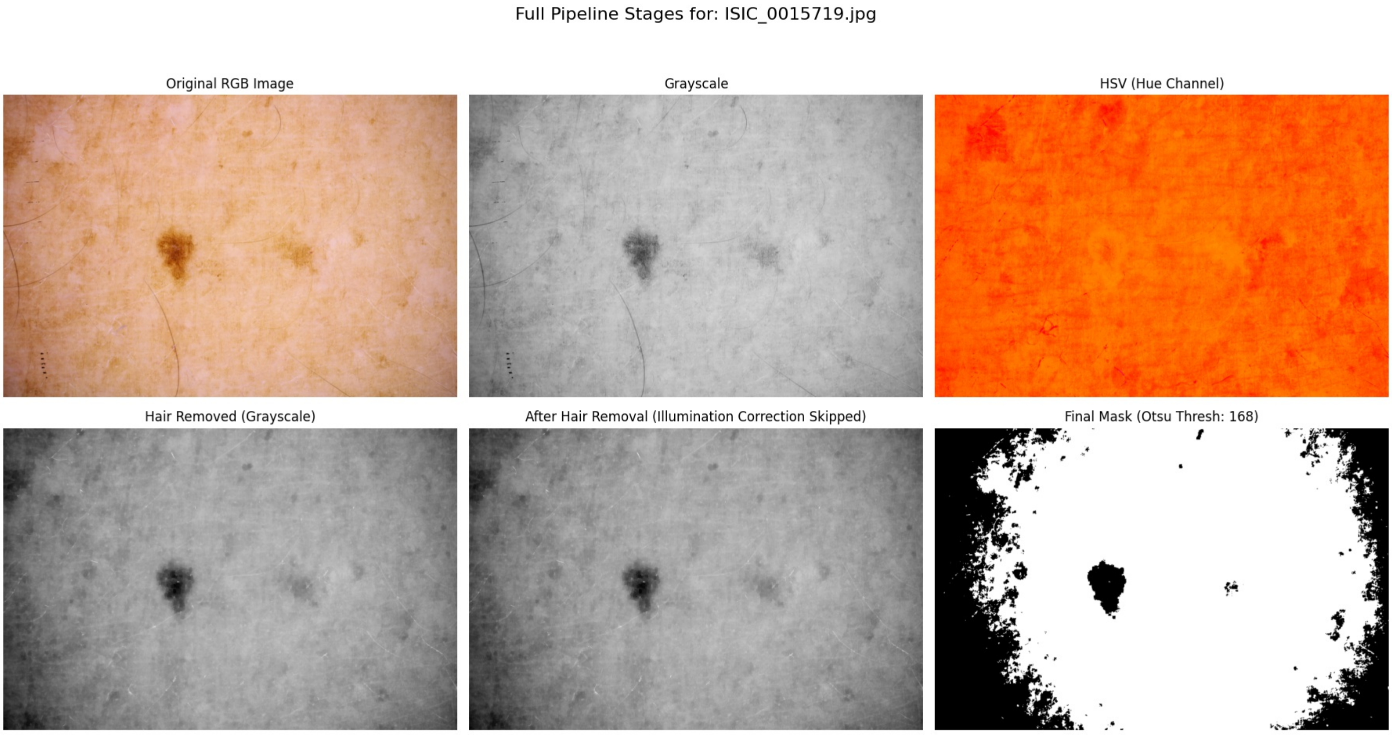
---

3. Results & Visualizations

This section showcases outputs from processing sample images. *(Assume images are in a local `img/` directory for this markdown)*

Pipeline Stages Visualization

- Illustrates key steps in processing an image.
- Example (`ISIC_0015719.jpg`):
    - Original RGB
    - Grayscale Conversion
    - HSV (Hue Channel shown)
    - Hair Removed (Grayscale)
    - After Hair Removal (Illumination Correction typically skipped for this view)
    - Final Binary Mask (from Otsu's thresholding)



<p align="center"> *Fig 1: Pipeline stages for ISIC_0015719.jpg (generated by `src/full_pipeline_display.py` or `src/all_features_display.py`)* </p>

Feature Summaries

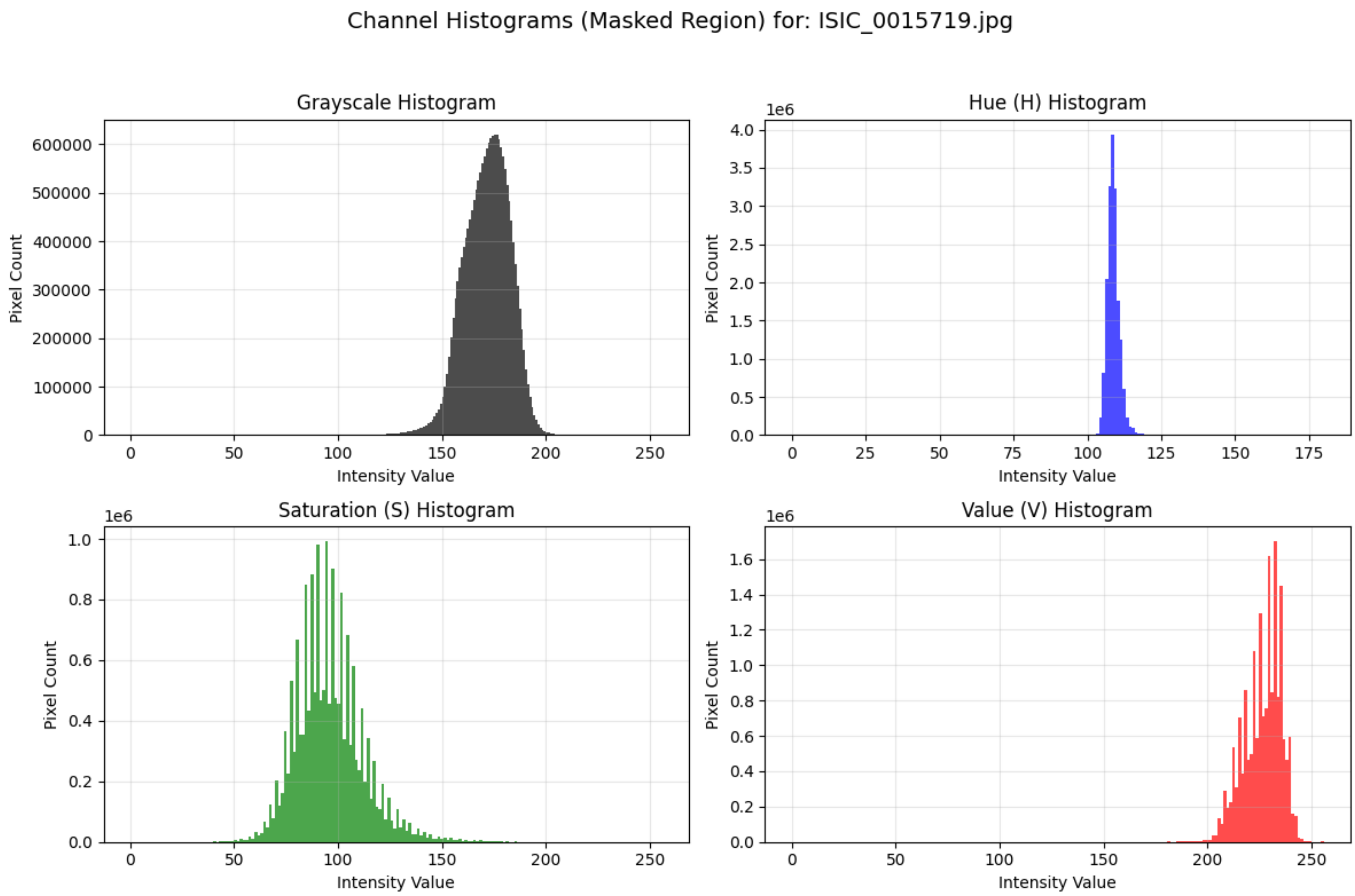
- Textual and visual representation of extracted *intensity and color features*.
- Example (`ISIC_0015719.jpg`):

```
Intensity Features for ISIC_0015719.jpg
========================================

Grayscale Intensity:
  Mean: 186.8315
  Std: 10.0121

HSV Channels:
  Hue (H):
    Mean: 0.0000
    Std: 0.0000
  Saturation (S):
    Mean: 0.0000
    Std: 0.0000
  Value (V):
    Mean: 0.0000
    Std: 0.0000
```

`<p align="center">` *Fig 2: Intensity and HSV Color Features for ISIC_0015719.jpg (generated by* `src/intensity_features_display.py` *or similar)* `</p>`

Channel Histograms (Masked Region)

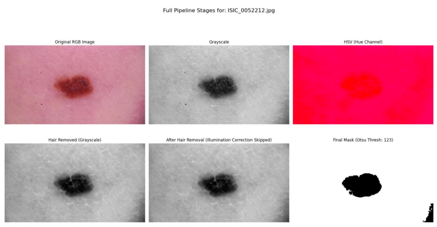
- Visualizes pixel distribution for different channels within the segmented lesion.
- Example (`ISIC_0015719.jpg`): Shows histograms for Grayscale, Hue, Saturation, and Value.



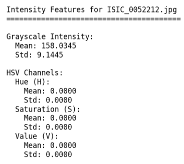
`<p align="center">` *Fig 3: Channel Histograms for ISIC_0015719.jpg (generated by* `src/masked_histogram_display.py` *or* `src/full_pipeline_display.py` *)* `</p>`

More Examples:

ISIC_0052212.jpg

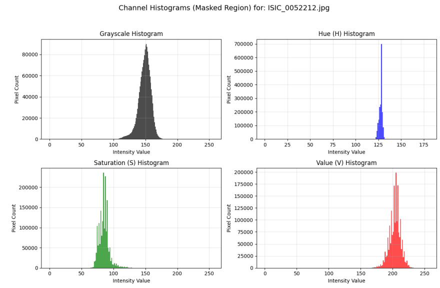


Pipeline Stages



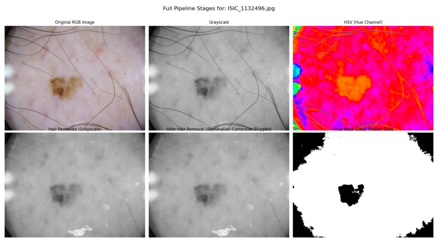
```
Intensity Features for ISIC_0052212.jpg
========================================

Grayscale Intensity:
  Mean: 158.0345
  Std: 9.1445

HSV Channels:
  Hue (H):
    Mean: 0.0000
    Std: 0.0000
  Saturation (S):
    Mean: 0.0000
    Std: 0.0000
  Value (V):
    Mean: 0.0000
    Std: 0.0000
```

Intensity Features

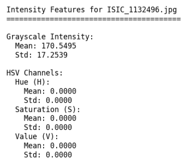


Channel Histograms

ISIC_1132496.jpg

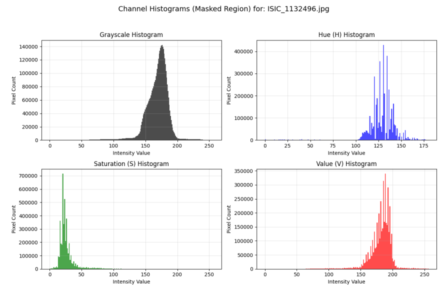


Pipeline Stages



```
Intensity Features for ISIC_1132496.jpg
========================================

Grayscale Intensity:
  Mean: 170.5495
  Std: 17.2539

HSV Channels:
  Hue (H):
    Mean: 0.0000
    Std: 0.0000
  Saturation (S):
    Mean: 0.0000
    Std: 0.0000
  Value (V):
    Mean: 0.0000
    Std: 0.0000
```

Intensity Features



Channel Histograms

4. Code Structure & Key Modules

The project is organized into several Python modules within the `src/` directory:

- `color_utils.py` : Color space conversions.
- `preprocessing.py` : Hair removal, illumination correction.
- `segmentation.py` : Thresholding methods, mask application.
- `morphology.py` : Morphological operations (opening, closing).
- `histogram_utils.py` : Masked histogram calculations.
- `feature_extraction.py` : Core logic for calculating intensity, color features.
- `border_texture_utils.py` : Calculates border and texture specific features.
- `data_loader.py` : Handles image and metadata loading.
- `batch_processor.py` : Enables processing of multiple images.

Display Scripts: Numerous scripts like `full_pipeline_display.py` , `all_features_display.py` , `intensity_features_display.py` , etc., allow visualization of different pipeline stages and extracted features.

5. Challenges & Learnings

- **Segmentation Accuracy:** Achieving perfect lesion segmentation is challenging due to image variability (hair, bubbles, skin lines, low contrast). Otsu's method provides a good baseline, and morphological cleanup helps significantly.
- **Feature Relevance:** Understanding how different features (intensity, color, texture, border) capture various aspects of lesion appearance.
- **Pipeline Integration:** Ensuring each step correctly feeds into the next and that data formats are consistent.
- **Modularity:** Structuring the code into reusable functions and modules (e.g., for color conversion, preprocessing, feature calculation) was key for development and testing.

6. Conclusion & Future Work

Achievements

- Successfully implemented an end-to-end image processing pipeline for skin lesion analysis.

- Developed modules for key DIP tasks: preprocessing, segmentation, and feature extraction.
- Extracted a diverse set of features:
    - Intensity (Grayscale statistics)
    - Color (HSV channel statistics)
    - Border (Gradient-based metrics)
    - Texture (Top-hat/Bottom-hat based metrics)
- Created comprehensive visualization tools to inspect pipeline stages and results.

Future Work

- **Advanced Segmentation:** Explore more robust segmentation techniques (e.g., active contours, watershed, machine learning-based segmentation).
- **Implement PRD Feature Set (f1-f28):**
    - Introduce the "Ida (Darkness)" channel.
    - Implement the specific Brightness, Saturation, and Darkness features (f1-f28) involving higher-order statistics (skewness, kurtosis), entropy, etc., as outlined in the project's PRD.
- **Machine Learning Integration:** Use the extracted features to train classification models (e.g., SVM, Random Forest, Neural Networks) to predict lesion malignancy.
- **Quantitative Evaluation:** If detailed ground truth masks become available, perform quantitative evaluation of segmentation accuracy (e.g., Dice coefficient, Jaccard index).
- **Expanded Dataset Processing:** Utilize the `batch_processor.py` to analyze a larger subset of the dataset.

Thank You & Questions?