HACKATHON SUBMISSION

(LEVEL-1-SOLUTION)

Use Case Title :

COLOUR DETECTION FROM IMAGES

studentName :K.Nanthini

Register number : 5109231060028

Institution : GLOBAL INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department : BE-ECE

Date of Submission : 15.5.25

COLOUR DETECTION FROM IMAGES

1.Problem Statement:

In various industries—such as design, fashion, manufacturing, and even healthcare—accurate colour detection plays a crucial role in decision-making and quality control. However, manual colour identification is prone to inconsistencies due to variations in lighting, human perception, and device calibration. This inconsistency can lead to mismatches in product colours, misinterpretation in medical imaging, and inefficiencies in automated systems relying on colour recognition.

A reliable colour detection system that can accurately extract and classify colours from images would help mitigate these challenges. By automating this process using AI and image processing techniques, organizations can achieve consistent colour identification, improve workflow efficiency, and enhance user experience in applications such as augmented reality, smart shopping assistants, and automated grading in agriculture.

2.Proposed Solution:

To address the challenges of inconsistent colour identification, this project proposes an AI-driven colour detection system that accurately extracts and classifies colours from images. The solution leverages computer vision and machine learning techniques to ensure reliable colour identification across various lighting conditions and device calibrations.

CONCEPT AND APPROACH:-

❖ IMAGE PROCESSING PIPELINE:

1.Preprocessing techniques, such as noise reduction and contrast enhancement, to improve image quality.

2.Colour space conversion (e.g., RGB to HSV) for better segmentation of colour components.

❖ FEATURE EXTRACTION AND CLASSIFICATION:

1.Extract dominant colours using clustering algorithms like K-Means or DBSCAN.

2.Generate colour histograms for a structured representation of colour distribution.

3.Map detected colours to a predefined colour dictionary for standardized interpretation.

❖ AI-POWERED ENHANCEMENT:

1.Implement deep learning models (e.g., CNNs) for improved colour accuracy in complex images.

2.Fine-tune the model using synthetic and real-world datasets to enhance robustness.

❖ USER-FRIENDLY INTERFACE:

1.Develop an interactive web or mobile application using tools like Gradio for real-time colour detection.

2.Allow users to upload images or capture them live for instant analysis.

❖ INDUSTRY-SPECIFIC EXTENSIONS:

1.Retail & Fashion: Colour matching for product recommendations.

2.Agriculture: Identifying crop health based on leaf colour.

3.Healthcare: Standardizing colour interpretation in medical imaging.

3.Technologies & Tools Considered:-

For a Colour Detection from Images project, a variety of technologies and tools can be used depending on the complexity and goals of the solution. Here's a structured list of technologies, programming languages, frameworks, and APIs that could be considered:

➢ PROGRAMMING LANGUAGES:

Python – Widely used for image processing and computer vision due to rich libraries.

JavaScript – Useful for web-based colour detection tools.

C++ – For high-performance applications (commonly with OpenCV).

Java – Alternative for mobile (Android) or desktop applications.

➢ FRAMEWORKS & LIBRARIES:

1) PYTHON LIBRARIES

OpenCV – Core library for image processing and colour detection.

NumPy – For matrix and pixel-level operations.

Matplotlib – For visualizing colour distribution and histograms.

Pillow (PIL) – Image loading and basic manipulation.

scikit-image – Advanced image analysis.

2) JAVASCRIPT LIBRARIES

p5.js – For image analysis and creative coding.

fabric.js / Konva.js – For canvas manipulation.

Colour-thief – Extracts prominent colours from an image.

➢ APIS & TOOLS:

ColourThief API – Extract dominant colour from images.

Google Vision API – Offers colour properties as part of its image analysis.

Adobe Colour API – For colour extraction and palette generation.

Cloudinary – Image processing API with colour extraction capabilities.

➢ DEVELOPMENT TOOLS:

Jupyter Notebook – Ideal for prototyping and visualizing Python-based image processing.

VS Code / PyCharm – General-purpose IDEs for coding.

Google Colab – Cloud-based Python execution, great for demos and sharing.

➢ COLOUR DETECTION ALGORITHMS:

K-Means Clustering – Groups similar colours for dominant colour extraction.

DBSCAN – Identifies colour regions with density-based clustering.

Histogram Analysis – Helps in analysing pixel distribution to extract colour patterns.

➢ ADDITIONAL TOOLS:

Matplotlib & Seaborn – For visualizing colour distributions.

NumPy – Essential for image data manipulation and numerical computations.

Streamlit – Helps in building intuitive interfaces for colour detection applications

With these technologies, an accurate and scalable colour detection system can be developed, ensuring robust performance across different applications.

4.Solution Architecture & Workflow:-

This AI-powered colour detection system operates through a structured pipeline consisting of key components designed to ensure accurate colour extraction and classification. Below is a high-level breakdown of the solution architecture and its workflow.

- IMAGE ACQUISITION & PREPROCESSING:

Users can upload an image or capture one in real-time using a webcam.

The image undergoes preprocessing, including noise reduction, contrast enhancement, and resizing to optimize it for analysis.

Conversion to HSV or LAB colour space ensures better segmentation of colour regions.

- ## FEATURE EXTRACTION & COLOUR ANALYSIS:

K-Means Clustering identifies dominant colours by grouping similar pixels.

Histogram Analysis quantifies pixel distribution for accurate colour representation.

Mapping detected colours to a predefined colour dictionary ensures consistency

- ## MACHINE LEARNING & AI PROCESSING:

Pre-trained models (e.g., CNN-based colour classification) refine colour accuracy.

Integration of deep learning techniques enhances detection in complex images.

Model fine-tuning ensures robustness across different lighting conditions

- ## USER INTERFACE & INTERACTION:

The processed image, along with extracted colour details, is displayed on a web or mobile interface.

Users receive colour codes (Hex, RGB, HSV) for identified colours.

Option to download results or apply detected colours to industry-specific applications.

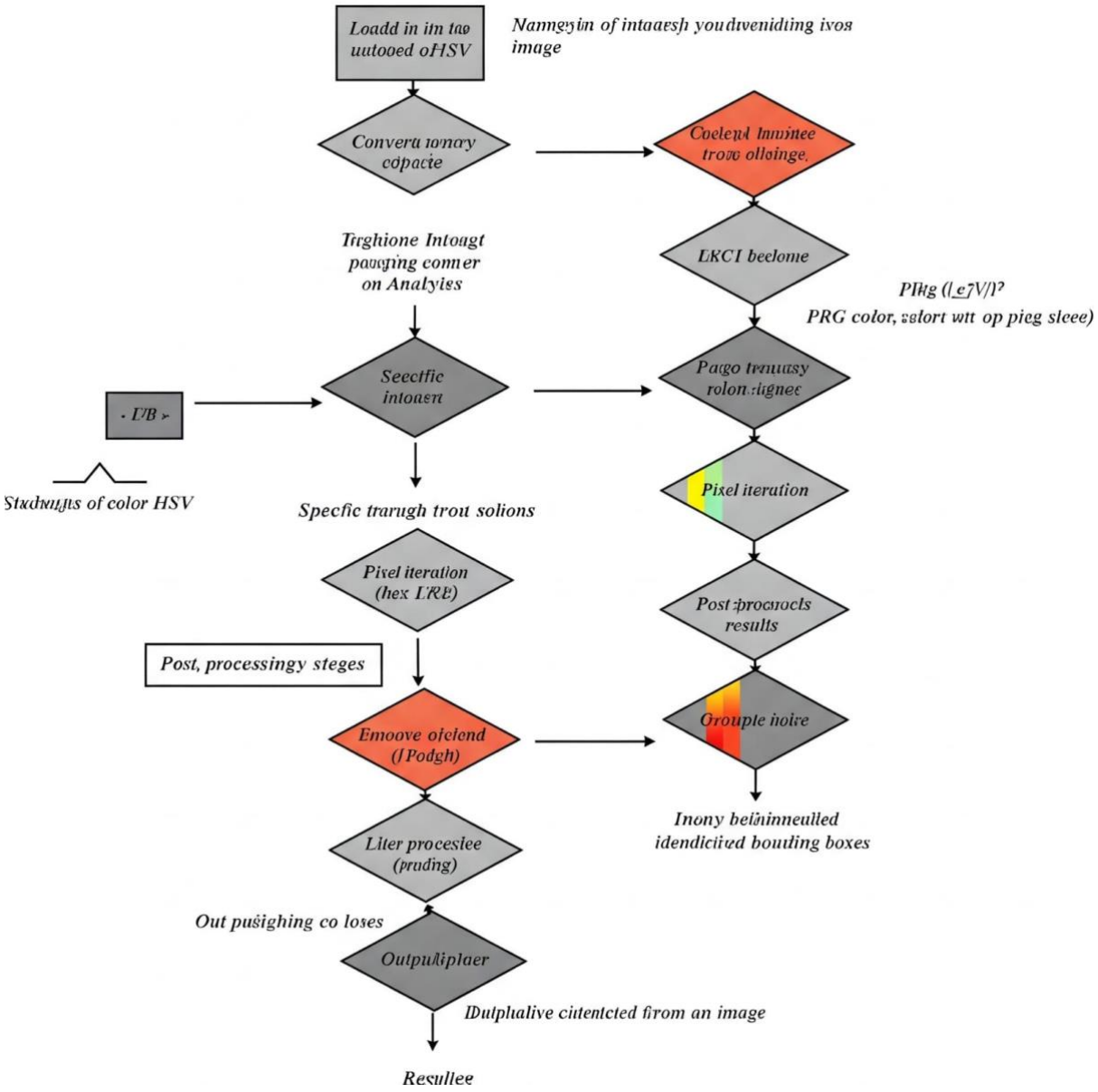
- ## DEPLOYMENT & OPTIMIZATION:

The system is deployed using Flask, FastAPI, or Streamlit for web-based interaction.

Cloud-based model hosting (AWS, Azure, GCP) ensures scalability.

Continuous model refinement and optimization enhance real-time performance.

FLOWCHART:



SOURCE CODE:

```
from graphviz import Digraph

# Create a new directed graph

dot = Digraph(comment='Colour Detection Flowchart')

dot.attr(rankdir='TB') # TB = Top to Bottom, LR = Left to Right


# Define nodes (shapes can be specified)

# Oval for Start/End

dot.node('A', 'Start', shape='oval')

dot.node('J', 'End', shape='oval')


# Parallelogram for Input/Output

dot.node('B', 'Input Image', shape='parallelogram')

dot.node('I', 'Output Results', shape='parallelogram')


# Rectangle for Processes
```

```
dot.node('C', 'Pre-processing\n(Resize, Noise Reduction, Colour Space Conversion)', shape='box')

dot.node('D', 'Select Region of Interest (ROI)\n(Optional)', shape='box')

dot.node('F', 'Extract Colour Information', shape='box')

dot.node('G', 'Process/Analyze Colour Data\n(Averaging, Counting, Mapping to Names)', shape='box')


# Diamond for Decision/Method Choice

dot.node('E', 'Colour Identification/Quantization Method\n(Pixel Iteration, Masking, Clustering, Histogram)', shape='diamond')



# Define edges (connections between nodes)

dot.edge('A', 'B')  # Start -> Input Image

dot.edge('B', 'C')  # Input Image -> Pre-processing

dot.edge('C', 'D')  # Pre-processing -> Select ROI

dot.edge('D', 'E')  # Select ROI -> Colour Identification Method
```

 # If ROI is skipped, Pre-processing can go directly to Colour Identification

 # You could add an alternative path or just keep it sequential for simplicity here.

 # For a more complex flowchart, you'd add more decision logic.

dot.edge('E', 'F') # Colour Identification Method -> Extract Colour Info

dot.edge('F', 'G') # Extract Colour Info -> Process/Analyze Data

dot.edge('G', 'I') # Process/Analyze Data -> Output Results

dot.edge('I', 'J') # Output Results -> End

 # --- Optional: Add a "bypass" for the ROI step if it's truly optional ---

 # This makes the graph slightly more complex to draw simply.

 # For this example, we'll keep the main flow.

 # An alternative is to have an edge from 'C' directly to 'E' labeled "Skip ROI"

```
# Specify output format and render the graph
# You can render to PDF, PNG, SVG, etc.
try:
    dot.render('colour_detection_flowchart', view=True, format='png')
    print("Flowchart 'colour_detection_flowchart.png' generated successfully.")
    print("It should open automatically if view=True and a viewer is available.")
except Exception as e:
    print(f"Error rendering flowchart: {e}")
    print("Make sure Graphviz is installed and in your system's PATH.")
```

5.Feasibility & Challenges:-

✅ FEASIBILITY:

The solution is highly practical and implementable, especially given the wide availability of open-source tools and libraries for image processing. Here's why:

1.Technological Maturity: Libraries like OpenCV, Pillow, and scikit-image in Python provide all necessary functions for image reading, colour space conversion, and pixel-level operations.

2.Lightweight Processing: For basic colour detection (e.g., dominant colour extraction), the computational load is low, allowing it to run on mobile devices or in-browser with JavaScript.

3.Cross-Platform Options: Can be developed as a desktop app, mobile app, or web app using common tech stacks (Python, JavaScript, etc.).

4.Availability of Cloud Resources: If required, APIs like Google Vision or Azure Computer Vision

can offer advanced capabilities without building everything from scratch

⚠ CHALLENGES & SUGGESTED SOLUTIONS:

❖ COLOUR ACCURACY & VARIABILITY:

Challenge: Lighting, shadows, and image quality can affect colour accuracy.

Solution: Normalize lighting conditions using preprocessing techniques like histogram equalization, or apply colour constancy algorithms

❖ REAL-TIME PERFORMANCE (IF NEEDED):

Challenge: Real-time colour detection in video streams can be computationally intensive.

Solution: Optimize using low-resolution frames, region-of-interest analysis, or use C++/OpenCV for performance-critical applications.

❖ MULTIPLE COLOUR DETECTION & CLUSTERING ACCURACY:

Challenge: Clustering (e.g., K-means) can give inconsistent results depending on the image.

Solution: Use adaptive clustering (e.g., Elbow method to determine optimal K) and post-processing filters to refine results.

❖ USER INTERFACE COMPLEXITY:

Challenge: Designing an intuitive UI that visually represents colours in an informative and appealing way.

Solution: Use colour palettes, visual bars, and live previews; frontend libraries like React and Chart.js can help

❖ BROWSER AND DEVICE COMPATIBILITY (WEB/MOBILE):

Challenge: Not all browsers or devices handle image files and canvas operations consistently.

Solution: Use polyfills, responsive frameworks, and thorough cross-browser testing.

❖ COLOUR NAMING AND HUMAN INTERPRETATION:

Challenge: RGB values don't always map neatly to known colour names.

Solution: Use predefined colour maps (e.g., XKCD colour survey dataset or CSS3 named colours) for approximate human-friendly naming

❖ LARGE IMAGE SIZES:

Challenge: High-resolution images may slow down processing.

Solution: Down sample images before processing or limit the upload size

6. Expected Outcome & Impact:-

✅ EXPECTED OUTCOMES:

- **Accurate Colour Identification: The system will successfully detect and extract dominant or**

specific colours from any image with high precision.

- **Colour Information Output: Users will receive detailed information such as RGB/Hex values, colour names, and percentage distribution of each detected colour.**

- **Interactive Visualization: A user-friendly interface will display the extracted colours in the form of swatches or palettes, enhancing usability.**

- **Optional Features: Depending on implementation, the tool may allow users to compare images, export colour palettes, or search based on colour.**

IMPACT & BENEFITS:

1. DESIGNERS & CREATIVES:

Benefit: Helps graphic designers, UI/UX experts, and artists extract colour schemes from photos for inspiration or consistency.

Impact: Accelerates design workflows and supports brand colour development.

2. DEVELOPERS:

Benefit: Useful for frontend and web developers who want to build colour-based themes or tools.

Impact: Simplifies integration of colour data into style guides or CSS files.

3. E-COMMERCE & MARKETING:

Benefit: Retailers can auto-detect dominant product colours to improve product categorization and search.

Impact: Enhances visual merchandising, filtering, and customer engagement.

4. EDUCATION & RESEARCH:

Benefit: Assists students and researchers in understanding image processing, computer vision, and colour theory.

Impact: Acts as an educational tool or project base in tech and design fields.

5. GENERAL USERS:

Benefit: Enables casual users to find colour names for personal projects (e.g., home painting, fashion coordination).

Impact: Empowers non-technical users with a practical image analysis tool.

7.Future Enhancements:-

Here's a detailed section on Future Enhancements for a Colour Detection from Images solution:

FUTURE ENHANCEMENTS:

1. REAL-TIME COLOUR DETECTION:

Description: Enable live colour detection from video streams (e.g., webcam or phone camera).

Benefit: Useful for stylists, designers, or users testing colour combinations on the fly.

2. COLOUR NAME SUGGESTIONS USING AI:

Description: Integrate machine learning to suggest human-friendly or brand-specific colour names (e.g., “Sunset Coral” instead of just “#FF6E6E”).

Benefit: Makes colour data more intuitive and marketing-ready.

3. COLOUR HARMONIZATION TOOLS:

Description: Suggest complementary, analogous, or triadic colour palettes based on detected colours.

Benefit: Supports design and fashion use cases by aiding colour-matching decisions.

4. SKIN TONE DETECTION & MATCHING:

Description: Add modules to detect skin tones in fashion photos and suggest colours that complement them.

Benefit: Personalized fashion advice and better styling for different audiences.

5. INTEGRATION WITH E-COMMERCE APIS:

Description: Connect to platforms like Shopify, Amazon, or Zalando to auto-tag and filter clothing based on detected colours.

Benefit: Enhances product discoverability and improves visual search features.

6. COLOUR TREND ANALYSIS:

Description: Use colour detection across datasets (e.g., social media, fashion week photos) to track emerging colour trends.

Benefit: Valuable for fashion forecasting and inventory planning.

7. MULTI-LANGUAGE COLOUR DESCRIPTIONS:

Description: Translate colour names and descriptions into multiple languages.

Benefit: Useful for global users and international fashion platforms.

8. ACCESSIBILITY ENHANCEMENTS:

Description: Offer features like colour-blind-safe palettes or auditory descriptions of colours.

Benefit: Makes the tool inclusive for users with visual impairments.