Technical Documentation for Lumina Smart Editor

October 2025

1 Introduction

This document provides a technical overview of the Lumina Smart Editor, a web-based image editing application built using Flask for the backend and HTML5 Canvas for frontend image processing. The application supports a wide range of image editing features, including geometric transformations, color adjustments, linear and non-linear filters, morphological operations, enhancement tools, frequency analysis, and drawing capabilities. This documentation covers the code structure, algorithms, graphical user interface (GUI), and dependencies.

2 Code Structure

The application is divided into a backend (Python with Flask) and a frontend (HTML, CSS, JavaScript). No classes are used; the codebase relies on functional programming and direct DOM manipulation.

2.1 Backend (Python - Flask)

The backend is a single Python script that uses Flask to serve a web application. It renders an HTML template and automatically opens a browser to the application URL.

```python
from flask import Flask, render_template_string
import webbrowser
from threading import Timer

app = Flask(__name__)

@app.route('/')
def index():
    return render_template_string(HTML_TEMPLATE)

def open_browser():
    webbrowser.open('http://127.0.0.1:5004')

if __name__ == '__main__':
    Timer(1, open_browser).start()
    app.run(debug=False, host='127.0.0.1', port=5004)
```

2.2 Frontend (HTML, CSS, JavaScript)

The frontend consists of:

- HTML: Defines the structure with a header, sidebar (tabbed controls), and canvas area.
- CSS: Provides a responsive grid layout with a dark theme.
- JavaScript: Handles image processing, user interactions, and canvas manipulation.

```
<div class="container">
    <div class="header">
        <div class="header-content">
            <div class="header-title">
                <h1>Lumina Smart Editor</h1>
                <span class="badge">Complete Editor</span>
            </div>
            <div class="header-buttons">...</div>
        </div>
        <input type="file" id="fileInput" accept="image/*">
    </div>
    <div class="main-content">
        <div class="sidebar">...</div>
        <div class="canvas-area" id="canvasArea">...</div>
    </div>
</div>
```

Key JavaScript Functions:

- `switchTab(ev, tabName)`: Switches between sidebar tabs.
- `applyLiveAdjustments()`: Applies real-time color and geometric adjustments.
- `saveHistory()`, `undo()`, `redo()`: Manage edit history.

```
function switchTab(ev, tabName) {
    document.querySelectorAll('.tab-content').forEach(t =>
        t.classList.remove('active'));
    document.querySelectorAll('.tab-btn').forEach(b =>
        b.classList.remove('active'));
    document.getElementById(tabName).classList.add('active');
    ev.currentTarget.classList.add('active');
    drawMode = null;
    cropMode = false;
    canvas.style.cursor = 'default';
}
```

3 Algorithms

The application implements several image processing techniques using the HTML5 Canvas API.

3.1 Convolution

Used for linear filters (e.g., Gaussian Blur, Sharpen, Sobel). A 3x3 kernel is applied to modify pixels based on their neighbors.

```
function convolve(img, kernel, factor = 1, bias = 0) {
    const { width, height, data } = img;
    const out = new ImageData(width, height);
    for (let y = 1; y < height - 1; y++) {
        for (let x = 1; x < width - 1; x++) {
            let r = 0, g = 0, b = 0, n = 0;
            for (let ky = -1; ky <= 1; ky++) {
                for (let kx = -1; kx <= 1; kx++) {
                    const idx = ((y + ky) * width + (x + kx)) * 4;
                    r += data[idx] * kernel[n];
                    g += data[idx + 1] * kernel[n];
                    b += data[idx + 2] * kernel[n];
                    n++;
                }
            }
            const i = (y * width + x) * 4;
            out.data[i] = Math.max(0, Math.min(255, r * factor + bias));
            out.data[i + 1] = Math.max(0, Math.min(255, g * factor + bias));
            out.data[i + 2] = Math.max(0, Math.min(255, b * factor + bias));
            out.data[i + 3] = data[i + 3];
        }
    }
    return out;
}
```

3.2 Geometric Transformations

Rotation, scaling, and flipping are implemented using canvas transformations.

```
ctx.translate(canvas.width / 2, canvas.height / 2);
ctx.rotate(rotation * Math.PI / 180);
ctx.scale(scale, scale);
ctx.translate(-canvas.width / 2, -canvas.height / 2);
```

3.3 Color Adjustments

RGB manipulation for brightness, contrast, saturation, hue, and gamma.

```
for (let i = 0; i < d.length; i += 4) {
    let r = d[i], g = d[i + 1], b = d[i + 2];
    r += brightness * 2.55;
    g += brightness * 2.55;
    b += brightness * 2.55;
    r = cFactor * (r - 128) + 128;
    const gray = 0.2989 * r + 0.5870 * g + 0.1140 * b;
    r = gray + sFactor * (r - gray);
}
```

3.4 Morphological Operations

Erosion and dilation use min/max operations on a 3x3 window.

```
function applyErosion() {
    const { width, height } = canvas;
    const src = ctx.getImageData(0, 0, width, height);
    const out = new ImageData(width, height);
    for (let y = 1; y < height - 1; y++) {
        for (let x = 1; x < width - 1; x++) {
            let minR = 255, minG = 255, minB = 255;
            for (let ky = -1; ky <= 1; ky++) {
                for (let kx = -1; kx <= 1; kx++) {
                    const idx = ((y + ky) * width + (x + kx)) * 4;
                    minR = Math.min(minR, src.data[idx]);
                    minG = Math.min(minG, src.data[idx + 1]);
                    minB = Math.min(minB, src.data[idx + 2]);
                }
            }
            const i = (y * width + x) * 4;
            out.data[i] = minR;
            out.data[i + 1] = minG;
            out.data[i + 2] = minB;
            out.data[i + 3] = 255;
        }
    }
    ctx.putImageData(out, 0, 0);
}
```

3.5 Flood Fill

A stack-based algorithm for filling areas with a selected color.

```
function floodFill(sx, sy) {
    const img = ctx.getImageData(0, 0, canvas.width, canvas.height);
    const { width, height, data } = img;
    const stack = [[sx, sy]];
    const visited = new Set();
    while (stack.length > 0) {
        const [x, y] = stack.pop();
        if (x < 0 || x >= width || y < 0 || y >= height) continue;
        const key = y * width + x;
        if (visited.has(key)) continue;
        visited.add(key);
        const i = key * 4;
        if (Math.abs(data[i] - startR) < 30 && Math.abs(data[i + 1] - startG)
            < 30 && Math.abs(data[i + 2] - startB) < 30) {
            data[i] = r;
            data[i + 1] = g;
            data[i + 2] = b;
            stack.push([x + 1, y], [x - 1, y], [x, y + 1], [x, y - 1]);
        }
    }
    ctx.putImageData(img, 0, 0);
}
```

4 GUI

The GUI consists of a header, sidebar with tabbed controls, and a canvas area. It uses a responsive grid layout with a dark theme.

4.1 Header

Contains the title, badge, and action buttons (Upload, Undo, Redo, Download).

```html
<div class="header">
    <div class="header-content">
        <div class="header-title">
            <h1>Lumina Smart Editor</h1>
            <span class="badge">Complete Editor</span>
        </div>
        <div class="header-buttons">
            <button class="btn-primary"
                onclick="document.getElementById('fileInput').click()">Upload</button>
            <button class="btn-secondary" onclick="undo()" id="undoBtn"
                disabled>Undo</button>
            <button class="btn-secondary" onclick="redo()" id="redoBtn"
                disabled>Redo</button>
            <button class="btn-success" onclick="downloadImage()"
                id="downloadBtn" disabled>Download</button>
        </div>
    </div>
    <input type="file" id="fileInput" accept="image/*">
</div>
```

4.2 Sidebar

Contains tabs for Geometric, Color, Linear Filters, Non-Linear, Morphology, Enhancement, Frequency, and Drawing, with sliders, buttons, and color inputs.

```html
<div class="sidebar">
    <div class="tab-buttons">
        <button class="tab-btn active" onclick="switchTab(event,
            'geometric')">Geometric</button>
        <button class="tab-btn" onclick="switchTab(event,
            'color')">Color</button>
    </div>
    <div id="geometric" class="tab-content active">
        <div class="control-group">
            <label class="control-label">
                <span>Rotation</span>
                <span id="rotationValue">0</span>
            </label>
            <input type="range" id="rotation" min="0" max="360" value="0"
                oninput="updateLiveAdjust()">
        </div>
    </div>
</div>
```

4.3 Canvas Area

Displays a placeholder or the editable image canvas.

```
<div class="canvas-area" id="canvasArea">
    <div class="upload-placeholder" id="placeholder">
        <svg>...</svg>
        <h2>Upload Image to Start</h2>
        <p>Supports JPG, PNG, GIF</p>
        <button class="btn-primary"
            onclick="document.getElementById('fileInput').click()">Choose
            File</button>
    </div>
    <canvas id="canvas" style="display:none;"></canvas>
</div>
```

4.4 Event Handling

Handles user interactions like image upload and mouse events for drawing/cropping.

```
fileInput.addEventListener('change', (e) => {
    const file = e.target.files?.[0];
    if (!file) return;
    const reader = new FileReader();
    reader.onload = (ev) => {
        const img = new Image();
        img.onload = () => {
            baseImage = img;
            currentImage = img;
            canvas.width = img.width;
            canvas.height = img.height;
            ctx.drawImage(img, 0, 0);
            placeholder.style.display = 'none';
            canvas.style.display = 'block';
            history = [];
            historyIndex = -1;
            resetAdjustments();
            saveHistory();
        };
        img.src = ev.target.result;
    };
    reader.readAsDataURL(file);
});
```

5 Dependencies

The application has minimal dependencies, relying on Python standard libraries and Flask for the backend, and vanilla JavaScript for the frontend.

- Flask: Web framework for serving the application.
  - Installation: `pip install flask`
  - Usage:

    ```
    from flask import Flask, render_template_string
    ```

- webbrowser: Standard Python library for opening the browser.
- threading: Standard Python library for delayed browser opening.
- Frontend: Uses HTML5 Canvas API and vanilla JavaScript, with no external libraries.

6 Conclusion

The Lumina Smart Editor is a feature-rich web-based image editor built with Flask and HTML5 Canvas. It supports a variety of image processing techniques, a responsive GUI, and minimal dependencies, making it lightweight and portable. Potential improvements include integrating WebGL for better performance or adding more advanced image processing libraries like OpenCV.js.