Coloring Page Processing Pipeline (Potrace Edition)

Overview

This project provides a high-performance pipeline for converting raster images (PNG, JPG, WEBP, etc.) into clean vector SVGs and print-ready PNG/PDF files.

It leverages powerful open-source tools—**ImageMagick**, **Potrace**, and **Inkscape**—to clean, trace, and export your images. The output is optimized for:

1. **Laser Cutting/Engraving:** Compatible with LightBurn, XTool Creative Space (S1/D1), and other laser software.
2. **CAD/CNC:** Optimized to reduce node counts for smoother import into **Autodesk Fusion** sketches.
3. **Print:** Generates high-resolution, centered PDFs.

📂 The Two Versions

This repository contains two scripts. Both process images into vectors, but they use different logic to get there.

1. The Base Pipeline (`process_coloring_pipeline.py`)

"The Fast & Strict"

- **Logic:** Applies a simple brightness threshold. Pixels darker than 60% become black; everything else becomes white.
- **Best For:** High-quality scans, crisp line art, or images that are already black ink on white paper.
- **Pros:** Fast, predictable results on clean input.

2. The Advanced Pipeline (`process_coloring_pipeline_adv.py`)

"The Smart & Flexible"

- **Logic:** Uses **Auto-Leveling** to stretch contrast (fixing muddy/gray inputs), detects color saturation, and uses "Posterization" to handle gradients.

- **Best For:** Photos of drawings (bad lighting), colored comic book panels, "muddy" scans, or inverted images (white lines on black).
- **Pros:** Handles almost any input quality; includes `--invert` and `--mode` flags.

⚖️ Decision Guide: Which script should I use?

Feature	Base Version	Advanced Version
Input Formats	PNG, JPG, WEBP, BMP, TIFF	PNG, JPG, WEBP, BMP, TIFF
Processing Speed	⚡ Fast	🐢 Slower (performs analysis)
Contrast Fix	No (Simple Threshold)	✅ **Auto-Levels** (Stretches contrast)
Color Support	Naive (Turns colors gray)	✅ **Smart** (Posterizes gradients first)
Invert Colors	No	✅ Yes (`--invert on`)
Command Flags	Basic (`--overwrite`)	Full (`--mode` , `--invert`)

Folder Structure

The project expects the following folder structure (auto-created if missing):

```
<project folder>/
├── input_png/      # Source images (auto-populated if images are next to
├── cleaned_png/    # Intermediate black/white PNGs (useful for debugging
├── svg/            # Traced vector output files (Optimized for CAD/Laser
├── png/            # Exported PNGs for printing (High Res)
├── pdf/            # Exported PDFs for printing
│
├── process_coloring_pipeline.py       # Base Script
├── process_coloring_pipeline_adv.py   # Advanced Script
│
├── run_pipeline.bat          # Windows "One-Click" for Base Version
├── run_pipeline.sh           # Mac/Linux "One-Click" for Base Version
├── run_pipeline_adv.bat      # Windows "One-Click" for Advanced Version
├── run_pipeline_adv.sh       # Mac/Linux "One-Click" for Advanced Versio
│
└── README.md
```

> **Note:** Simply drop your source images (PNG, JPG, etc.) in the root folder next to the scripts; the pipeline will automatically move them into `input_png` and process them.

🚀 Usage

Method 1: The Easy Way (One-Click Wrappers)

These scripts allow you to run the pipeline without opening a terminal manually. They also automatically detect and use a Python virtual environment (`venv`) if you have created one.

For Windows Users:

- Double-click `run_pipeline.bat` to run the **Base** version.
- Double-click `run_pipeline_adv.bat` to run the **Advanced** version.

For macOS / Linux Users:

1. **First time setup:** You must tell the system the scripts are safe to execute. Open your Terminal, navigate to the folder, and run:

   ```
   chmod +x run_pipeline.sh run_pipeline_adv.sh
   ```

2. **Running:** Double-click the `.sh` file (or run `./run_pipeline.sh` from the terminal).

Method 2: Command Line (Manual)

Open your terminal/command prompt in the project folder to run Python directly. This allows you to use specific flags.

Run Base Version:

```
python process_coloring_pipeline.py
```

Run Advanced Version:

```
python process_coloring_pipeline_adv.py
```

Advanced Flags (Manual Mode Only)

- **Force Reprocessing:**

    - `python process_coloring_pipeline_adv.py --overwrite`

- **Color Mode (Gradient Flattening):**
    - *Advanced Script Only*
    - `--mode auto` (Default): Auto-detects if an image is color or B&W.
    - `--mode color` : Forces the gradient flattening logic (posterize -> 8 colors -> threshold). Use this if colored images look noisy.
    - `--mode bw` : Forces standard black/white thresholding.

- **Invert Colors:**
    - *Advanced Script Only*
    - `--invert on` : Turns white lines black (useful for negative scans).
    - `--invert off` (Default): Standard processing.

⚙️ Configuration & Tuning

You can open the Python scripts in VSCode (or any text editor) to adjust these settings at the top of the file:

Common Settings (Both Scripts)

- `EXPORT_WIDTH_PX` : Resolution of the final PNG export (default: 3000).
- `THRESHOLD_PERCENT` : The cutoff for converting gray pixels to black or white.
- `OPTTOLERANCE` (Inside `trace_svg` function): Default is `0.5` .
    - **Increase (e.g., 1.0)** for simpler curves and fewer nodes (better for Fusion sketches and faster processing in XTool).
    - **Decrease (e.g., 0.2)** for stricter adherence to the pixel data (better for artistic printing).

Advanced Settings (Adv Script Only)

- `POSTERIZE_COLORS` : Max colors to use when flattening color images (default: 8).
- `LEVEL_ADJUSTMENT` : Contrast stretch settings (default: "0%,80%").

🛠️ Dependencies & Installation

All tools must be installed and accessible via your system PATH.

1. Python 3.x

- **All OS:** Download from Python.org or use your package manager.

2. ImageMagick (for cleaning)

- **Windows:** Download the **DLL version** from ImageMagick.org.
  - **IMPORTANT:** During installation, check the box **"Install legacy utilities (e.g. convert)"** and **"Add to PATH"**.
- **macOS:**

```
brew install imagemagick
```

- **Linux:**

```
sudo apt install imagemagick
```

3. Potrace (for vectorizing)

This is the engine that handles the tracing.

- **Windows:**
  1. Download the **Windows (64-bit)** zip from SourceForge.
  2. Extract the zip. You will see `potrace.exe`.
  3. Move the extracted folder to a permanent location (e.g., `C:\Program Files\potrace`).
  4. Add that folder to your System **PATH** environment variable.
  5. Verify by opening a terminal and typing `potrace --version`.
- **macOS:**

```
brew install potrace
```

- **Linux:**

```
sudo apt install potrace
```

4. Inkscape (for exporting PNG/PDF)

- **Windows:** Download the MSI/EXE from Inkscape.org.
- **macOS:**

```
brew install --cask inkscape
```

- **Linux:**

```
sudo apt install inkscape
```

Happy coloring, scaling, and making! 🎨