CS 180 Project 1: Colorizing Prokudin-Gorskii Plates

Brief overview: I take a single tall grayscale scan with three equal-height plates (top→bottom = B, G, R). I split these channels, *align G→B and R→B using integer shifts*, and composite to RGB. For small JPGs I use single-scale exhaustive search; for large TIFs I use a coarse-to-fine pyramid. Only compressed JPG results are shown here, no TIFs.

Detailed Approach

High-level: I split the stacked image into `b`, `g`, and `r` channels, then found 2D integer translations that best align `g` and `r` to `b`. The search window for the single-scale version is `[−R, R] × [−R, R]`. For TIFs, I built a pyramid by 2× average-pooling. I aligned at the coarsest level and refined at each finer level using appropriately smaller windows.

- **Metrics:**
  - `ncc`: normalized cross-correlation on intensity.
  - `ssd`: I maximize `−SSD`.
  - `edges`/`edges5`: Sobel magnitude (light Gaussian blur for `edges5`), scored by `−SSD`; robust when exposures differ by channel.

    - `auto`: I evaluate `ncc` and `edges5` fairly by scoring both on a comparable scale and pick the better one.
- **Border policy:** during scoring I ignore an inner band (`--border`) to avoid black frames dominating. After shifting, I crop to the common overlap across B/G/R using the *applied* vectors. This removes wrap-around color edges. A small `--pre-crop` helps with thick frames.
- **What I tried when things failed:**
    1. Increase `--max-radius` and confirm via full score grids (debug) if the optimum was clipped.
    2. Switch metric to `edges5` on images with strong exposure differences (e.g., `emir.tif`).
    3. Increase `--border` (or asymmetric left/right) to down-weight thick black frames.
    4. Try a different anchor (`--anchor g`) and compare the score maps.
    5. Use the pyramid path for large shifts in TIFs.

Key Issues & Fixes

1) Shift-sign bug → ghosting

My search returns where the moving channel *is* relative to the reference (`(dy,dx)`). To align it, I **apply** `(-dy,-dx)`. Using the wrong sign produced double-ghosting; fixing the sign and reporting *applied* vectors resolved it.

2) `auto` metric fairness (Emir)

`emir.tif` has different per-channel exposures; intensity NCC can pick a wrong peak. My `auto` compares `ncc` and `edges5` in their own feature spaces but on a comparable numeric scale, so I don't bias toward NCC. For batch runs, I still force `edges5` on TIFs for safety.

3) Frames dominating scores

Thick black borders can dominate SSD/NCC. I ignore an inner band during scoring and crop the final overlap after shifting. This simple policy was more reliable for me than full automatic frame detection.

Results on Provided Images

single + ncc Offsets: G→B (5, 2), R→B (12, 3)

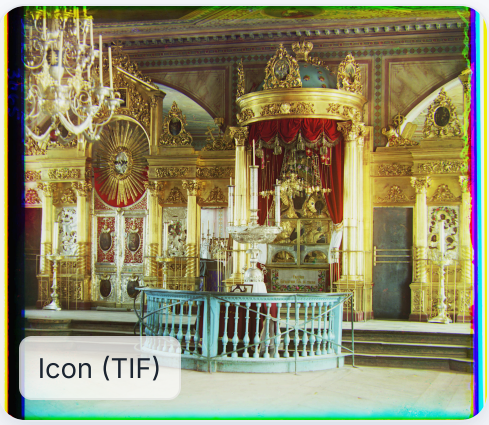
single + ncc Offsets: G→B (-3, 2), R→B (3, 2)

single + ncc Offsets: G→B (3, 3), R→B (6, 3)

pyramid + edges5 Offsets: G→B (25, 4), R→B (58, -4)

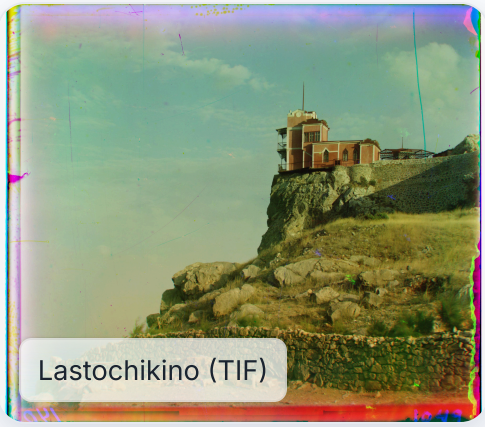
pyramid + edges5 Offsets: G→B (49, 23), R→B (107, 40)

pyramid + edges5 Offsets: G→B (60, 17), R→B (123, 14)



pyramid + edges5 Offsets: G→B (42, 17), R→B (90, 23)

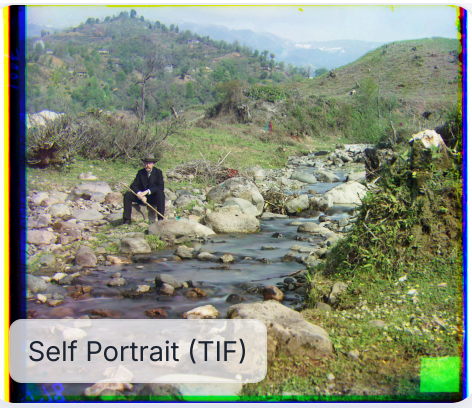
pyramid + edges5 Offsets: G→B (38, 22), R→B (77, 36)



pyramid + edges5 Offsets: G→B (-3, -2), R→B (76, -8)

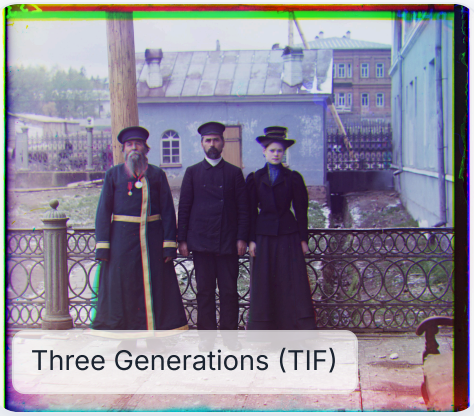
pyramid + edges5 Offsets: G→B (41, -17), R→B (93, -29)

pyramid + edges5 Offsets: G→B (81, 10), R→B (178, 13)



pyramid + edges5 Offsets: G→B (79, 30), R→B (176, 37)

pyramid + edges5 Offsets: G→B (49, -6), R→B (96, -25)



pyramid + edges5 Offsets: G→B (53, 13), R→B (111, 10)

Displacement Vectors (applied)

The table lists the *applied* G→B and R→B shifts (dy, dx). To keep the site lightweight, I only include compressed images from `output/` (no large TIFs are uploaded).

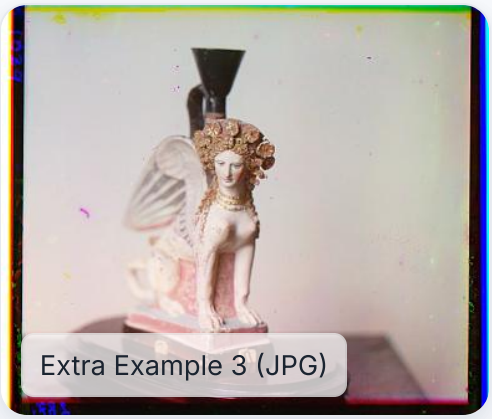
Image	Method	G→B (dy, dx)	R→B (dy, dx)
Cathedral (JPG)	single + ncc	(5, 2)	(12, 3)
Monastery (JPG)	single + ncc	(-3, 2)	(3, 2)
Tobolsk (JPG)	single + ncc	(3, 3)	(6, 3)
Church (TIF)	pyramid + edges5	(25, 4)	(58, -4)

Image	Method	G→B (dy, dx)	R→B (dy, dx)
Emir (TIF)	pyramid + edges5	(49, 23)	(107, 40)
Harvesters (TIF)	pyramid + edges5	(60, 17)	(123, 14)
Icon (TIF)	pyramid + edges5	(42, 17)	(90, 23)
Italil (TIF)	pyramid + edges5	(38, 22)	(77, 36)
Lastochikino (TIF)	pyramid + edges5	(-3, -2)	(76, -8)
Lugano (TIF)	pyramid + edges5	(41, -17)	(93, -29)
Melons (TIF)	pyramid + edges5	(81, 10)	(178, 13)
Self Portrait (TIF)	pyramid + edges5	(79, 30)	(176, 37)
Siren (TIF)	pyramid + edges5	(49, -6)	(96, -25)
Three Generations (TIF)	pyramid + edges5	(53, 13)	(111, 10)
Extra Example 1 (JPG)	single + ncc	(-3, 1)	(-4, 1)
Extra Example 2 (JPG)	single + ncc	(5, -1)	(10, -3)
Extra Example 3 (JPG)	single + ncc	(2, 3)	(11, 6)

Additional Examples (from the Prokudin-Gorskii collection)

single + ncc Offsets: G→B (-3, 1), R→B (-4, 1)

single + ncc Offsets: G→B (5, -1), R→B (10, -3)



single + ncc Offsets: G→B (2, 3), R→B (11, 6)

Failures & Difficult Images

- **Emir:** per-channel brightness differs significantly; raw-intensity NCC can mis-score. Using `edges5` (Sobel on a lightly blurred image) focuses on structure and aligns correctly for me.
- **Heavy frames:** when black borders dominate, metrics can chase them. Ignoring an inner band in scoring and cropping the final overlap removed the color bands I often saw at the edges.

How to Reproduce

Single image

- Small JPG (single-scale NCC): `python proj1_colorizer.py --input data/cathedral.jpg --method single --metric ncc --max-radius 15 --auto-crop`
- Large TIF (pyramid + edges): `python proj1_colorizer.py --input data/emir.tif --method pyramid --metric edges5 --auto-crop`

Batch everything

```
# TIFs: pyramid + edges5
for f in data/*.tif; do
  [ -e "$f" ] || continue
  python proj1_colorizer.py --input "$f" --method pyram
    --output "output/$(basename "${f%.*}")_color.jpg"
done

# JPGs: single-scale + NCC
for f in data/*.jpg; do
  [ -e "$f" ] || continue
  python proj1_colorizer.py --input "$f" --method singl
    --output "output/$(basename "${f%.*}")_color.jpg"
done
```

Notes & Takeaways

- Exhaustive search + clear visualizations helped me debug failures quickly.
- I found edge-based features safer than raw intensity when channels differ in exposure.

- My simplest border policy worked best: ignore during scoring, crop final overlap afterward.
- Future work I'd like to try: subpixel refinement, corner/feature-based matching, and learned border cropping.