Colorizing Mars

Documented proposed solution or strategy

This section documents the strategy for coloring high-resolution black-and-white Mars imagery using low-resolution color data. The approach combines high-performance geospatial libraries (**Rasterio**) with computer vision libraries (**OpenCV**, **scikit-image**) to handle alignment, color separation, and filtering. The core idea is to use the ***L*a*b**** color space to separate luminance (detail) from chrominance (color) and apply **histogram matching** for color consistency.

Approach Details:

- **Read Input Files:**
  - Read the high-resolution B&W image (JPEG or GeoTIFF) using `cv2.imread` for simple images or `rasterio` for GeoTIFFs.
  - Read the low-resolution color GeoTIFF using `rasterio`.
- **Align Data:**
  - The low-resolution image data, read as a NumPy array.
  - Convert the low-resolution image data from `(Channel, Height, Width)` to `(Height, Width, Channel)` to match the high-resolution image.
  - Resize the low-res image to the high-res dimensions using `cv2.resize` with interpolation (`INTER_CUBIC` or `INTER_LINEAR`).
  - This step allows both images to be processed pixel-to-pixel.
- **Convert to L*a*b*:**
  - Convert the resized low-res RGB data to the L*a*b* color space using cv2.cvtColor(..., cv2.COLOR_RGB2LAB).
  - LAB separates **brightness (L*)** from **color channels (a*, b*)**, making it easier to manipulate color without affecting details.
    - L* → Lu*minance (lightness / brightness / detail)*
    - ***a**** → *Chrominance channel (green → red)*
    - ***b**** → *Chrominance channel (blue → yellow)*
- **Histogram Matching:**
  - Use `skimage.exposure.match_histograms` on the luminance channel.
  - This updates the pixel intensities of the high-res image to match the statistical distribution of the low-res luminance channel.
  - Ensures brightness in the high-res image follows the low-res reference.

Syntax: *skimage.exposure.match_histograms(image, reference, *, channel_axis=None, multichannel=False)*

- **Refine Chroma Channels:**
  - Smooth or sharpen the a* and b* channels using edge-preserving filters:
    - `cv2.bilateralFilter`: Smooths colors based on proximity and color similarity.
    - `cv2.ximgproc.guidedFilter`: Uses the high-res luminance channel as a guide to preserve sharp edges in the color data.
- **Reconstruct**
  - Combine the matched L* channel with the filtered a* and b* channels into a final L*a*b* array.
  - Convert this back to RGB (`cv2.cvtColor(..., cv2.COLOR_Lab2RGB)`).
- **Georeference & Save:**
  - Use `rasterio` to save the output image as a GeoTIFF ,by using the low-res GeoTIFF metadata, bounds, and CRS to save the high-resolution colorized image as a GeoTIFF using Rasterio.
  - This preserves spatial referencing for GIS applications or Cesium.

Concluding Note:

To my knowledge, in previous projects I have used OpenCV and CNNs (Neural Networks) for image resolution enhancement. For these Mars images, I experimented with OpenCV and Rasterio. The challenge was that one image was JPEG and the other a GeoTIFF. I initially used OpenCV combined with Rasterio to maintain georeferencing, and through research and experimentation, I learned how to use OpenCV L*a*b* color space to separate brightness and color, histogram matching for reference-based adjustments, and bilateral and guided filters for edge-preserving color refinement.Since OpenCV is implemented in C++, this entire workflow can also be performed in C++ for maximum performance.