The Effects of L1 Image Transform on Image Compression

Presented by Corey Wingo, Sravan Kumar Guduru, and Kartikey Gupta

Local Smooth Filters

- Removes High-Frequency Noise
- Operates locally
- Edge-preserving
- e.g., bilateral filter

L1 Piecewise Flattening

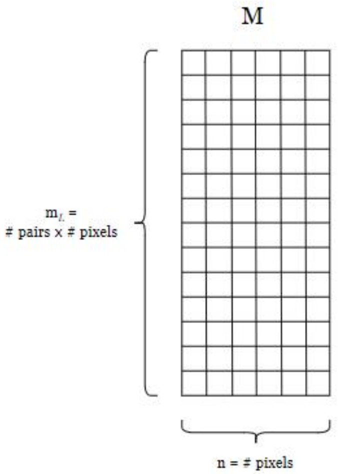
- Removes Low-Frequency Noise
- Operates globally
- Edge-preserving

Bilateral Original L1 Flattening

L1 Piecewise Flattening

- (1) If two neighboring pixels have similar colors, pull those colors closer together.
- (2) If a color discontinuity exists across two neighboring pixels, keep those colors apart.

Original Image



$$\mathbf{L}_{3m_l \times 3n} = \begin{pmatrix} \mathbf{M} & & \\ & \mathbf{M} & \\ & & \mathbf{M} \end{pmatrix}$$

Compute L matrix

Algorithm 1 Split-Bregman method for piecewise image flattening

procedure IMAGEFLATTEN(ϵ, λ)
  Initial: $\mathbf{z}^0 = \mathbf{z}^{in}; d_1^0, b_1^0, d_2^0, b_2^0 = 0;$
  while $||\mathbf{z}^k - \mathbf{z}^{k-1}||_2^2 > \epsilon$ **do**
    $\mathbf{A} = \beta \mathbf{I}_{3n \times 3n} + \lambda \mathbf{L}^T \mathbf{L} + \lambda \alpha^2 \mathbf{G}^T \mathbf{G}$
    $\mathbf{v} = \beta \mathbf{z}^{in} + \lambda \mathbf{L}^T (d_1^k - b_1^k) + \lambda \alpha \mathbf{G}^T (d_2^k - b_2^k)$
    Update $\mathbf{z}^{k+1}$ by solving $\mathbf{A}\mathbf{z}^{k+1} = \mathbf{v}$
    $d_1^{k+1} = \mathbf{Shrink}(\mathbf{L}\mathbf{z}^{k+1} + b_1^k, \frac{1}{\lambda})$
    $d_2^{k+1} = \mathbf{Shrink}(\alpha \mathbf{G}\mathbf{z}^{k+1} + b_2^k, \frac{1}{\lambda})$
    $b_1^{k+1} = b_1^k + \mathbf{L}\mathbf{z}^{k+1} - d_1^{k+1}$
    $b_2^{k+1} = b_2^k + \alpha \mathbf{G}\mathbf{z}^{k+1} - d_2^{k+1}$
    $k = k + 1$
  return $\mathbf{z}^k$
procedure SHRINK($\mathbf{y}, \gamma$)
  return $\frac{\mathbf{y}}{\|\mathbf{y}\|} * \max(\|\mathbf{y}\| - \gamma, 0)$

Solve Energy Optimization

Flattened Image

Implementation

- Python + CuPy (CUDA-based)
- ~400 lines
- Heavy reliance on sparse matrices

Hardware

- AMD Ryzen 5 5600X 6-Core 12-Thread
- 8GB Nvidia RTX 3070
- 32GB DDR4-3600 RAM

Dataset

- (150 x 150)-pixel image = ~180 seconds and uses ~7 GB of VRAM
- Resized PNG images for lossless feature
- Converted to JPEG, WebP after flattening


118x180


160x120


180x120

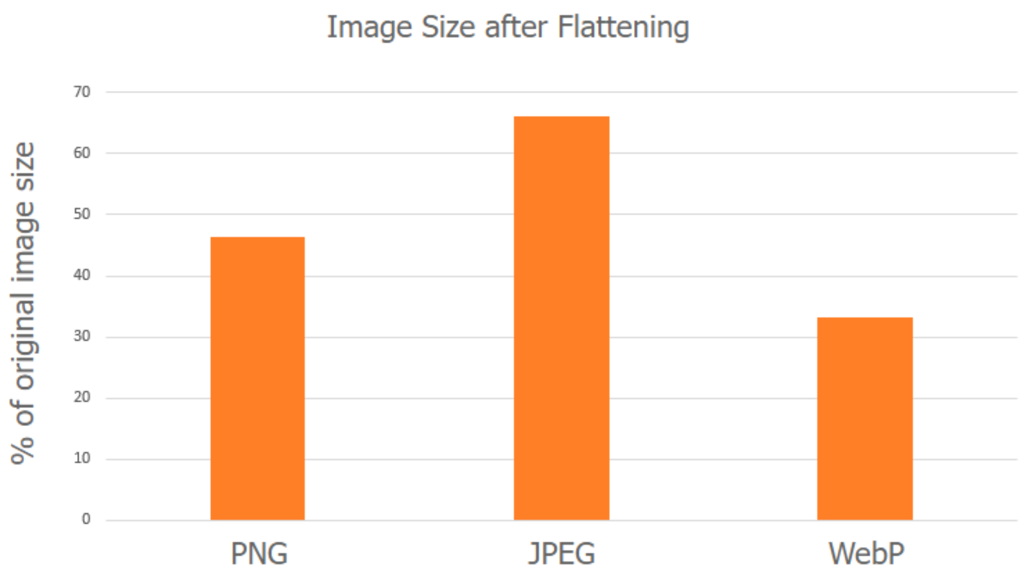

136x180


200x112

. . .

Results

- All flattened images resulted in better compression
- More colors/edges = less flattening = less compression
- Algorithm is very heavy on space allocation



References

[1] Sai Bi, Xiaoguang Han, and Yizhou Yu. An l1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition. ACM Trans. Graph., 34(4), jul 2015.

[2] https://github.com/sai-bi/L1Flattening