Image Compression Based on Spatial Redundancy Removal and Image Inpainting

Presented by Alexander Cullmann

Paper by Vahid Bastani, Mohammad Sadegh Helfroush, Keyvan Kasiri

Journal of Zhejiang University-SCIENCE C (Computers & Electronics), Vol. 11, No. 2, 92–100, 2010.

Outline

Image Compression

- eliminate redundancy
- even better compression: drop some unnecessary information
- JPEG use 8x8 blocks, cosine transformation and quantization
- 8x8 blocks are source of blocking artifacts (getting more and more visible in higher compression)

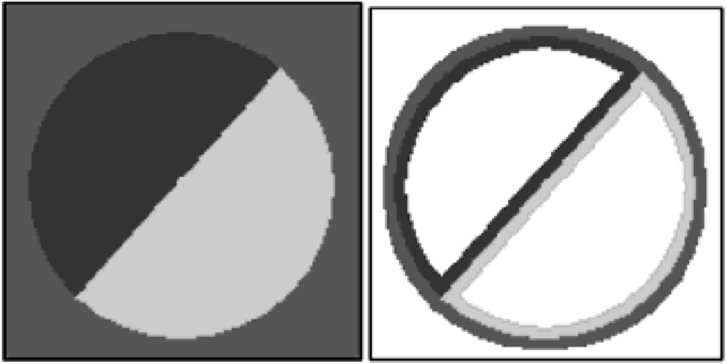
How to do better?

Where is the Redundancy?

- "normal" pictures consist of separate regions
- pixels in neighborhood are likely to be (almost) equal (high correlation)
- a lot of information is located at edges
- boundary of a region specifies not only shape but change of pixel values

$\Rightarrow$ boundary pixel are enough information to recalculate an image

Example: Boundary is Enough



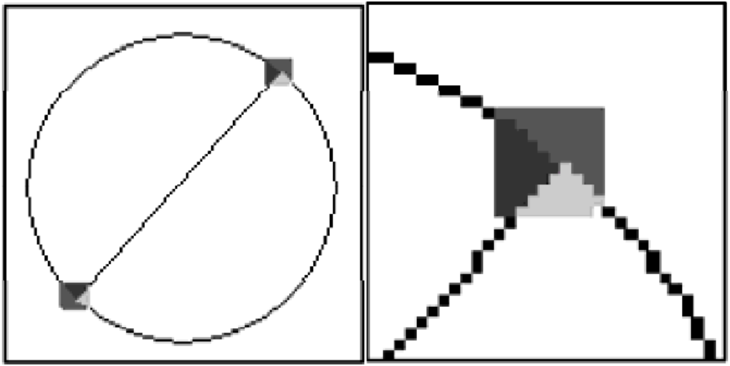
Left: Image with tree regions; **Right**: Extracted edges of the same image

More Redundancy Along Edges

- no significant changes along edges
- pixel values at endpoints of edge sufficient to recover values at entire edge
- those endpoints are called 'source points'

$\Rightarrow$ Source Points + Shape of edges are enough information to recalculate an image

Example: Source Points and Shape are Enough

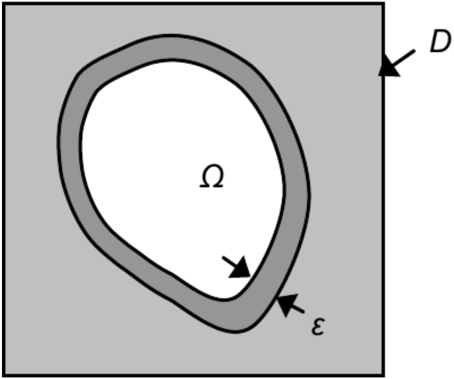


Left: Source points and boundaries; **Right**: Zoomed to source point

Image Inpainting

- Goal: fill in missing / damaged regions in a visually plausible, non detectable way
- in general: resulting inpainted image not necessarily similar to the original
- but similarity possible if “missing” parts are chosen wise

Inpainting a Region

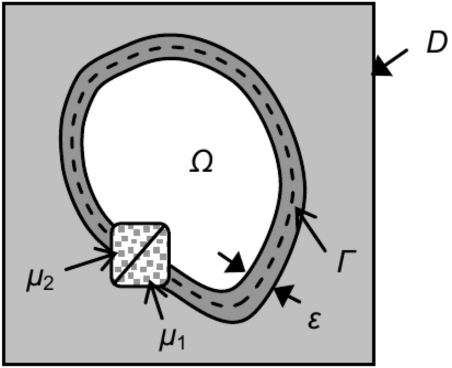


ε: the boundary of the region
Ω: the region to recover
D: image domain

inpainting as boundary value problem:

$$\begin{cases} \Delta u & = 0 \text{ in } \Omega \\ u & = u_0\big|_{\varepsilon} \end{cases}$$

From Source Pixel to Boundary



ε: the boundary (to recover)

Ω: the region to (finally) recover

D: image domain

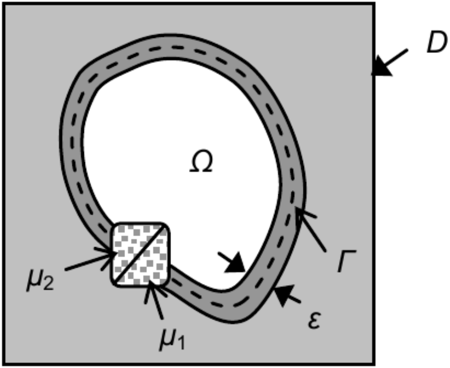
μ_1 and μ_2: source points

Γ: boundary indicating the edge

$\Gamma = \{(x_t, y_t) | x_t = f(t), y_t = g(t)\}$

$$\begin{cases} \frac{d^2u}{dl^2} = 0 \\ u|_{\mu_1} = u_0|_{\mu_1}, u|_{\mu_2} = u_0|_{\mu_2} \end{cases}$$

Both Steps in One Equation



let

$$\lambda(x,y) = \begin{cases} 1, & (x,y) \in \varepsilon \\ 0, & (x,y) \in \Omega \end{cases}$$

then

$$\begin{cases} \lambda \frac{d^2u}{dl^2} + (1-\lambda)\Delta u = 0 \\ u|_{\mu_1} = u_0|_{\mu_1}, u|_{\mu_2} = u_0|_{\mu_2} \end{cases}$$

Modification In The Numerical Approach

central difference of Laplace equation:

$4u_c - u_N - u_E - u_S - u_W = 0$

$u_c = \frac{1}{8}(c_N \cdot u_N + c_E \cdot u_E + c_S \cdot u_S + c_W \cdot u_W$
$+ c_{NW} \cdot u_{NW} + c_{NE} \cdot u_{NE} + c_{SW} \cdot u_{SW} + c_{SE} \cdot u_{SE})$

NW	N	NE
W	C	E
SW	S	SE

with coefficients as follows:

case $\lambda = 0$ (inside the region):

$$\begin{cases} c_N = c_E = c_S = c_W = 2, \\ c_{NW} = c_{NE} = c_{SW} = c_{SE} = 0 \end{cases}$$

case $\lambda = 1$ (on the curve):

$$\begin{cases} c_{t-1} = c_{t+1} = 4, \\ c_{else} = 0 \end{cases}$$

Modification In The Numerical Approach

central difference of Laplace equation:

$4u_c - u_N - u_E - u_S - u_W = 0$

$$u_c = \frac{1}{8}(c_N \cdot u_N + c_E \cdot u_E + c_S \cdot u_S + c_W \cdot u_W$$
$$+c_{NW} \cdot u_{NW} + c_{NE} \cdot u_{NE} + c_{SW} \cdot u_{SW} + c_{SE} \cdot u_{SE})$$

NW	N	NE
W	C	E
SW	S	SE

with coefficients as follows:

case $\lambda = 0$ (inside the region):

$$\begin{cases} c_N = c_E = c_S = c_W = 2, \\ c_{NW} = c_{NE} = c_{SW} = c_{SE} = 0 \end{cases}$$

case $\lambda = 1$ (on the curve):

$$\begin{cases} c_{t-1} = c_{t+1} = 4, \\ c_{else} = 0 \end{cases}$$

Modification In The Numerical Approach

central difference of Laplace equation:

$4u_c - u_N - u_E - u_S - u_W = 0$

$u_c = \frac{1}{8}(c_N \cdot u_N + c_E \cdot u_E + c_S \cdot u_S + c_W \cdot u_W$
$+ c_{NW} \cdot u_{NW} + c_{NE} \cdot u_{NE} + c_{SW} \cdot u_{SW} + c_{SE} \cdot u_{SE})$

NW	N	NE
W	C	E
SW	S	SE

with coefficients as follows:

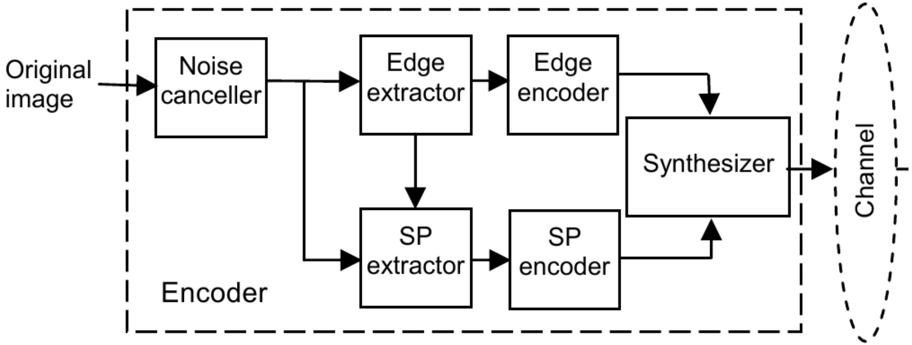
case $\lambda = 0$ (inside the region):

$$\begin{cases} c_N = c_E = c_S = c_W = 2, \\ c_{NW} = c_{NE} = c_{SW} = c_{SE} = 0 \end{cases}$$

case $\lambda = 1$ (on the curve):

$$\begin{cases} c_{t-1} = c_{t+1} = 4, \\ c_{else} = 0 \end{cases}$$

Image Encoder - Block Diagram



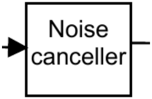
Noise Canceler

- Perona Malik filter: $\frac{\delta u}{\delta t} = div(g(\|\nabla u\|)\nabla u)$
- vanishes near eges
- increases to 1 away from egdes

$\Rightarrow$ smoothes without blurring edges

$\Rightarrow$ removes noise

$\Rightarrow$ increases efficiency

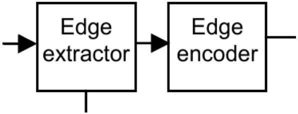


Edge Extractor and Encoder

- specifies boundary of different regions
- should detect real transitions

$\Rightarrow$ Sobel

- encoding with lossless encoder
- as binary image



Source Point Extractor and Encoder

- for each edge: SP are the points by which the edges may be recovered
- SP includes at least two pixels on both sides of edge
- indicate variation in the direction perpendicular to edge
- stored row wise in an array

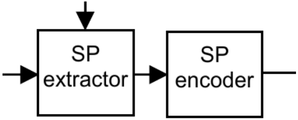
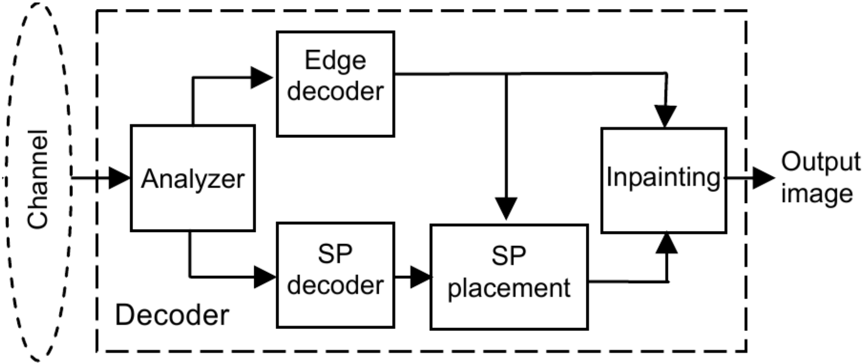
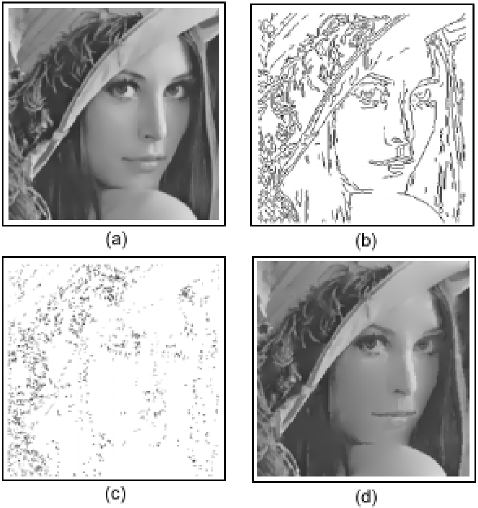


Image Decoder - Block Diagram



Example: Encoding and Decoding



(a) Original Image (8 bpp)
(b) Edges
(c) Source Points
(d) Recovered Image (0.6 bpp)

Experiments: The Setting

- noise removal: Jacobi iterative method, 30 iterations
- edge detection: Sobel, threshold set manually
- binary edge image encoded with JBIG algorithm
- Source Points encoded by entropy coding
- gray level images

- Pentium Celeron 1.8 GHz, 512 MB RAM, Matlab R2007b
- encoder: 4 s
- decoder: 40 s

Different Image Quality Indices

PSNR

- peak signal-to-noise ratio
- ratio of the squared image intensity dynamic range to the mean squared difference of the original and distored image
- widely used
- does not reflect human perception
- the higher the better

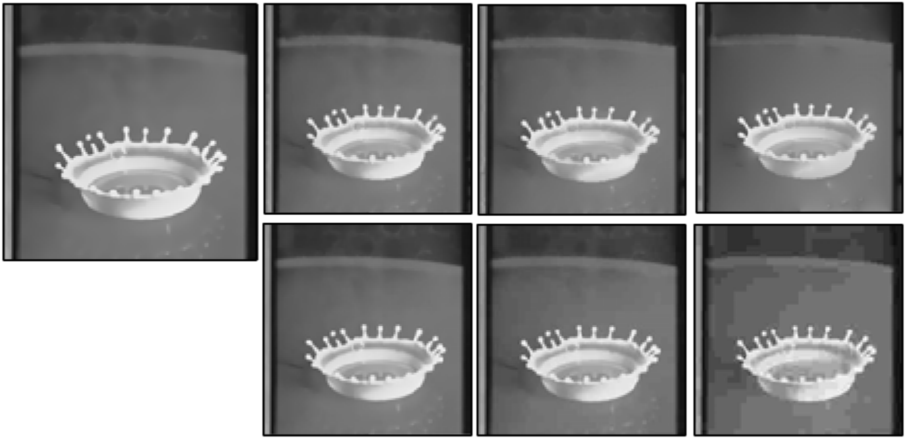
SSIM

- structural similarity
- ratio of four times covariance times mean to the sum of squared variances times sum of squared means
- based on the human perception
- the nearer to 1, the better

Comparison with JPEG

		PSNR (dB)		SSIM	
		Proposed	JPEG	Proposed	JPEG
Splash	0.8 bpp	32.83	41.87	0.9748	*0.9859*
	0.4 bpp	30.07	36.00	*0.9631*	0.9579
	0.2 bpp	28.48	30.16	*0.9509*	0.8780
Peppers	0.8 bpp	28.30	35.40	0.9266	*0.9544*
	0.4 bpp	23.90	31.00	0.8513	*0.8890*
	0.2 bpp	20.61	36.31	*0.7832*	0.7593

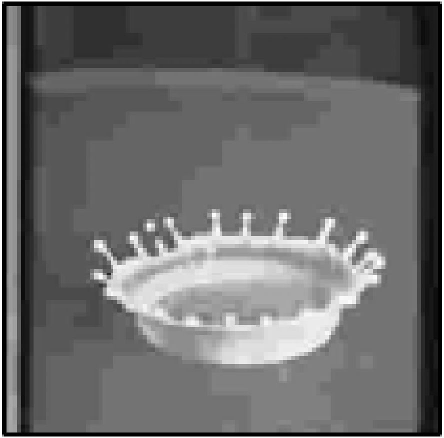
Example: Splash



Left: Original image, 8 bpp; **Right:**top row: proposed algorithm, bottom row: JPEG

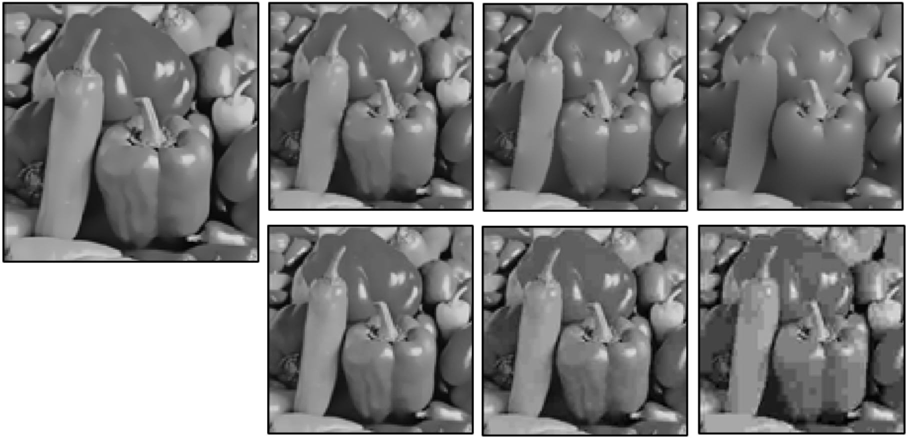
left to right: 0.8 bpp, 0.4 bpp, 0.2 bpp

Example: Splash



Left: proposed algorithm, 0.2 bpp **Right:** JPEG, 0.2 bpp

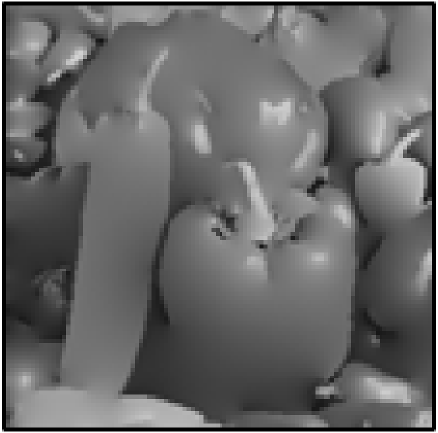
Example: Peppers

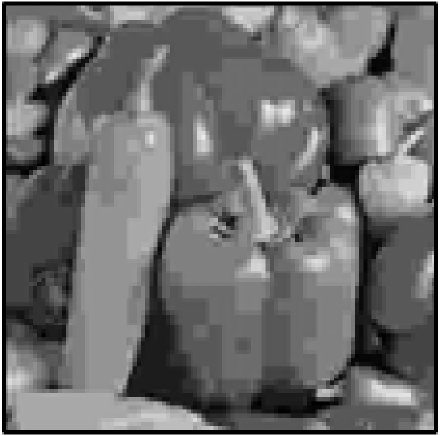


Left: Original image, 8 bpp; **Right:**top row: proposed algorithm, bottom row: JPEG

left to right: 0.8 bpp, 0.4 bpp, 0.2 bpp

Example: Peppers





Left: proposed algorithm, 0.2 bpp **Right:** JPEG, 0.2 bpp

Conclusion

- new method for image compression
- high correlated regions skipped during encoding
- recovered using image inpainting
- good for high compression (1:40 !)
- details are lost, but image looks much better than JPEG

differential element along with the curve

Let n be a tangent vector to the curve:

$$n = \left(\frac{dx_t}{dt}, \frac{dy_t}{dt} \right) \Bigg/ \sqrt{\left(\frac{dx_t}{dt} \right)^2, \left(\frac{dy_t}{dt} \right)^2}$$

then the first derivative along the curve Γ

$$\frac{du}{dl} = \left(\frac{\delta u}{\delta x} \frac{dx_t}{dt} + \frac{\delta u}{\delta y} \frac{dy_t}{dt} \right) \Bigg/ \sqrt{\left(\frac{dx_t}{dt} \right)^2, \left(\frac{dy_t}{dt} \right)^2}$$