A HIGH QUALITY, FAST INVERSE HALFTONING ALGORITHM FOR ERROR DIFFUSED HALFTONES

Thomas D. Kite

Niranjan Damera-Venkata

Brian L. Evans

Alan C. Bovik

The University of Texas at Austin

ICIP-98 TA2.06

6 October 1998

OUTLINE

- Introduction to inverse halftoning
- Overview of proposed method
- Details of algorithm
- Results
- Complexity analysis
- Modeling and quality metrics
- Conclusions

INVERSE HALFTONING

- Attempt to recover grayscale images from halftones
- Applications
  - Digital copiers
  - Scanner software
  - Facsimile
- Need for inverse halftoning
  - Inability to manipulate halftones
  - Poor halftone compression ratios
- Efficiency requirements
  - Low computational complexity
  - Low memory requirement
  - Hardware implementation

PREVIOUS APPROACHES

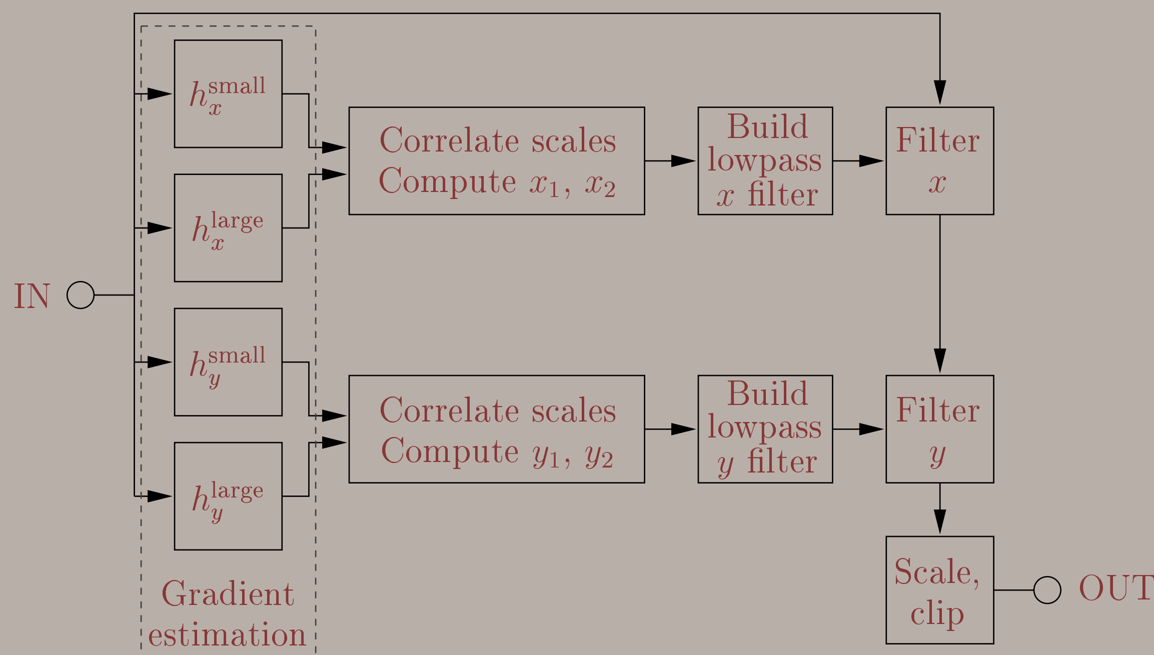
- Bayesian estimation [Schweizer & Stevenson 1993]
- Vector quantization [Ting & Riskin 1994]
- Projection onto convex sets [Hein & Zakhor 1995]
- Lowpass smoothing and nonlinear filtering [Wong 1995]
- Wavelet denoising [Xiong, Orchard & Ramchandran 1997]

- Most are iterative, slow, and memory-intensive
- Local integer operations preferred

OVERVIEW OF METHOD

- Anisotropic diffusion
  - Estimate image gradients
  - Compute diffusion coefficient
  - Smooth within areas, preserve edges
- Error diffused halftones
  - Highpass noise, SNR ≈ 3 dB
  - Tonal
- Solution
  - Specialized gradient estimator
  - Correlate estimate across scales
  - Separable—smooth parallel to edges
- Local operations
  - Low memory requirement
  - Low computational cost
  - Single pass

BLOCK DIAGRAM



- Estimate gradients at two scales in *x* and *y* directions
- Correlate gradients across scales
- Construct parametric smoothing filter in *x* and *y* directions
- Filter and clip

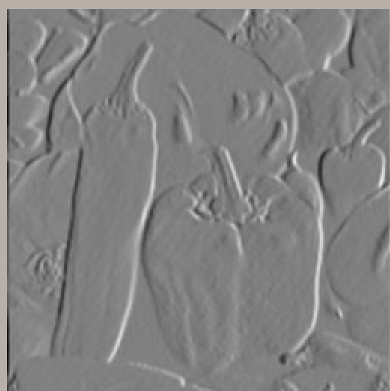
SMOOTHING FILTER DESIGN

- Filter requirements
  - Small, separable, FIR
  - Attenuation of highpass noise, tones
  - Parameter to control cutoff frequency
- Solution: 7-tap filter prototype
  - Unity gain at DC, zero at f_N
  - Constrained passband ripple, stopband attenuation
  - Two parameters (x_1, x_2) control cutoff
- Parameter elimination
  - Design 10 filters meeting constraints
  - Compute cubic fit of x_2 to x_1
- Result: continuous adjustment of cutoff from $0.07f_N$ to $0.50f_N$ by varying x_1

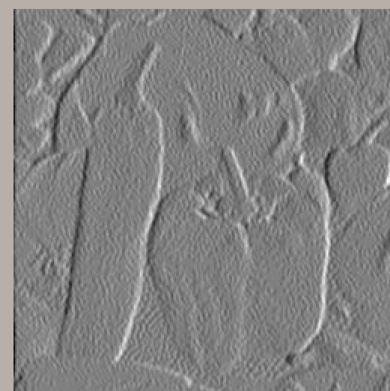
GRADIENT ESTIMATOR DESIGN

- Error diffused halftones
  - High power, highpass noise
  - Strong tones at (f_N, f_N), $(f_N, 0)$
- Solution: multiscale estimator
  - High stopband attenuation
  - Line zeros at band edges
  - Correlate estimates across two scales [Mallat & Zhong 1992]
- Correlation across scales
  - $e = |e_{\text{small}} \times e_{\text{large}} \times e_{\text{large}}|^{(1/3)}$
  - Control function *e* varies linearly with gradient
- Result: SNR of gradient estimate improved by 5 dB

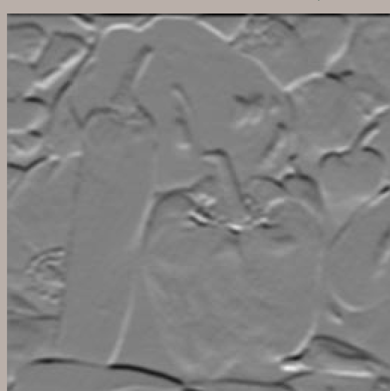
GRADIENT ESTIMATES



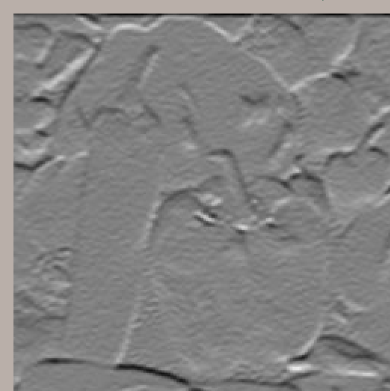
Small scale, *x*



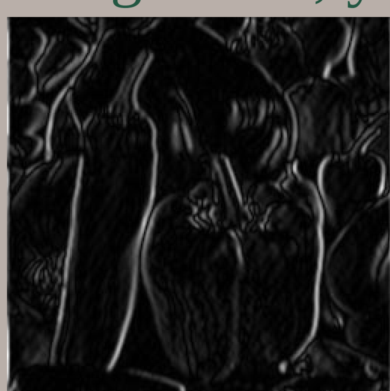
Small scale, x



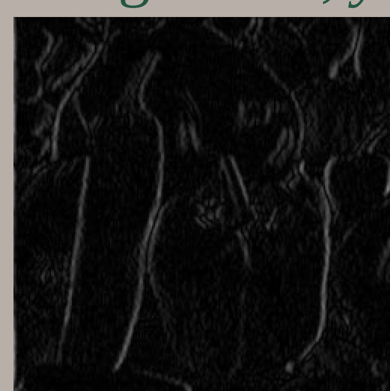
Large scale, y



Large scale, *y*



Control function, *x*



Control function, *x*

IMAGE CONSTRUCTION

- Compute smoothing filter parameters x_1, y_1 from x, y gradient estimates
- Compute (x_2, y_2) from (x_1, y_1) using cubic fit
- Construct x, y filters; quantize coefficients to 13-bit integers
- Filter 7 × 7 neighborhood in x and y directions separably
- Clip output to range 0-255

INVERSE HALFTONE RESULTS I

Original image

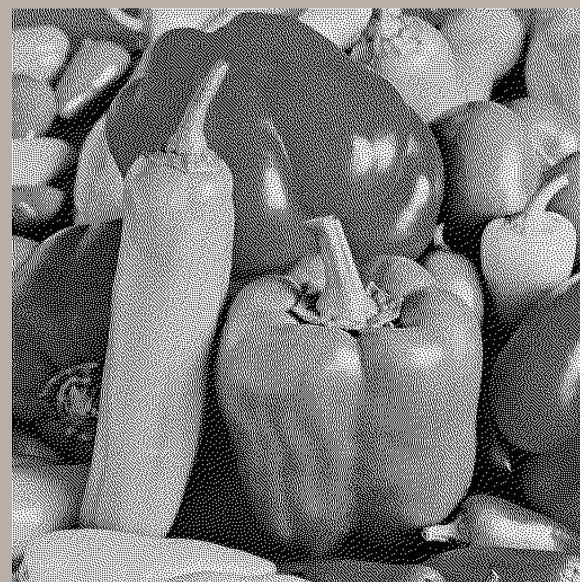
Halftone

Proposed method

Wavelet method

INVERSE HALFTONE RESULTS II

Original image



Halftone

Proposed method

Wavelet method

IMPLEMENTATION

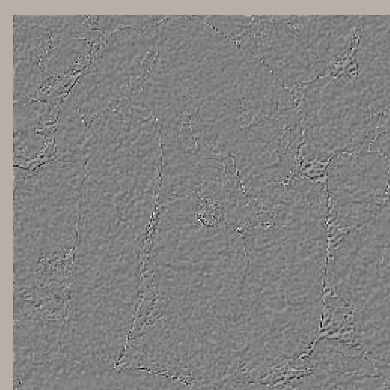
- Optimization for efficiency
  - Gradient estimators: integer additions
  - Gradient correlation: low overhead Newton-Raphson cube root
  - Smoothing filter: applied separably, integer coefficients
- Operations per pixel
  - 303 increments (++)
  - 30–226 integer additions (128 ave.)
  - 7 integer multiplications
  - 34 floating-point additions
  - 19 floating-point multiplications
  - 5 floating-point divisions
- Memory: 7 image rows
- Execution time: 2.9 s (512 × 512 image, 167 MHz Sun Ultra-2)

INVERSE HALFTONING MODEL

- Forward/inverse halftoning system blurs image, adds noise
- Model inverse halftoning
  - Compute unsharpened halftone
  - Inverse halftone; save filter parameters at each pixel
  - Filter original using saved filters
- Residual has low linear correlation with original (mostly noise)

Inverse halftone

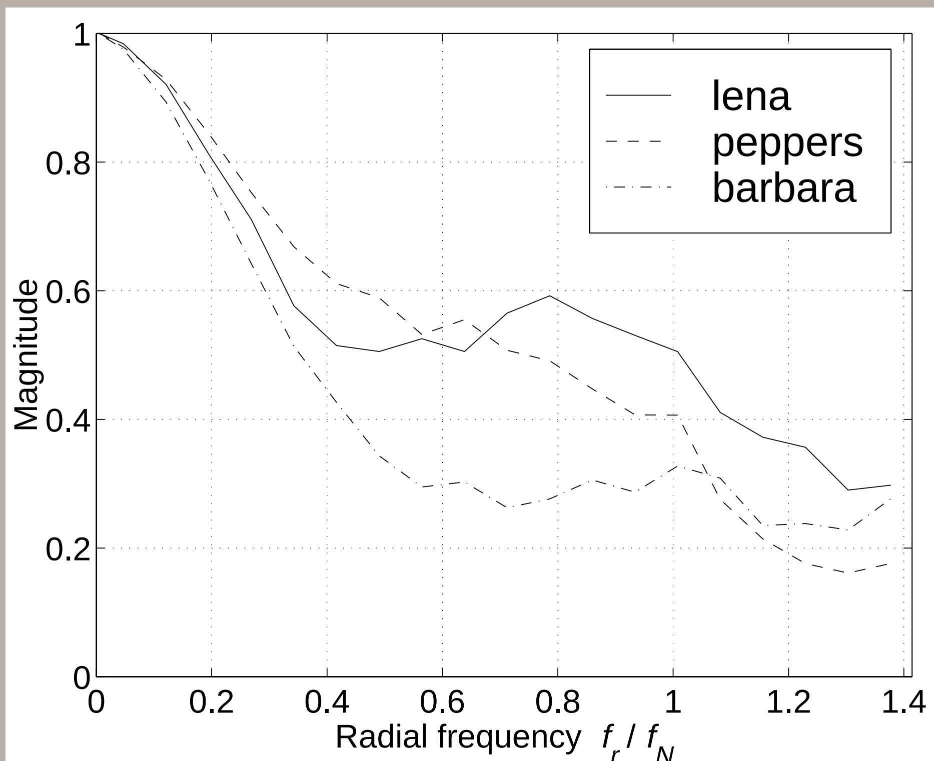
Modeled



Residual (×4)

INVERSE HALFTONE QUALITY

- Compute weighted SNR of inverse halftone relative to model
- Compute effective transfer function of system
  - Divide FFT of model by FFT of original image point-for-point
  - Radially average over annuli



CONCLUSIONS

- Fast inverse halftoning method for error diffused halftones
  - High quality
  - Low computational requirement
  - Low memory requirement
  - Suitable for hardware and embedded software
- Quality metrics for inverse halftones
  - Separate degradations into frequency distortion (blurring), noise injection
  - Quantify blur with effective transfer function
  - Quantify noise with weighted SNR
  - Enables optimization of general inverse halftoning methods