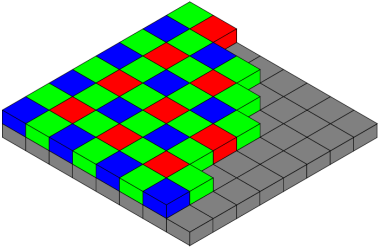
CFA Interpolation Detection

Leszek Świrski

October 15, 2009

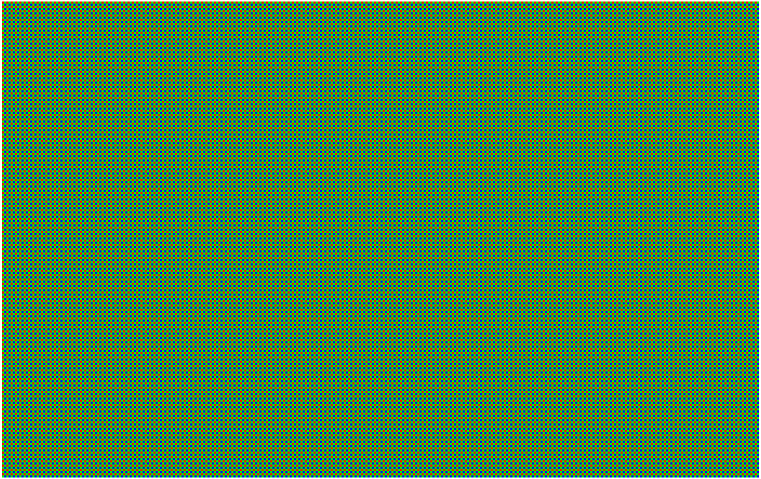
CFA?

- "Colour Filter Array"
- Photosensors have no wavelength specificity
- So filter RGB onto array of photosensors
- e.g. Bayer filter



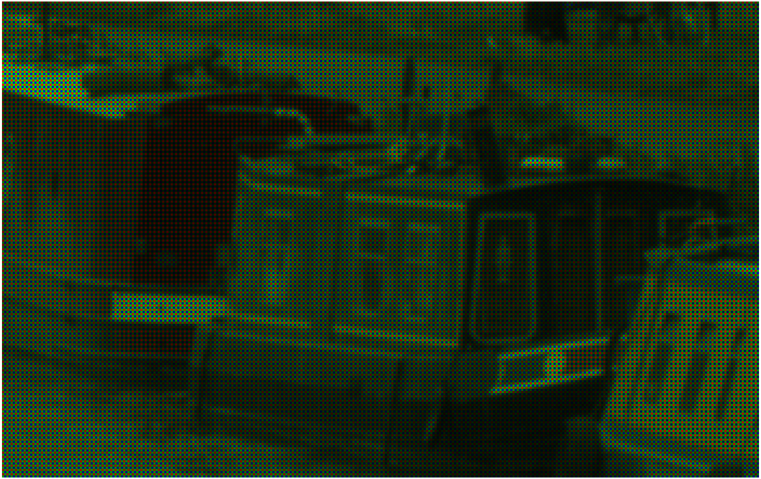
Target Image

Filters

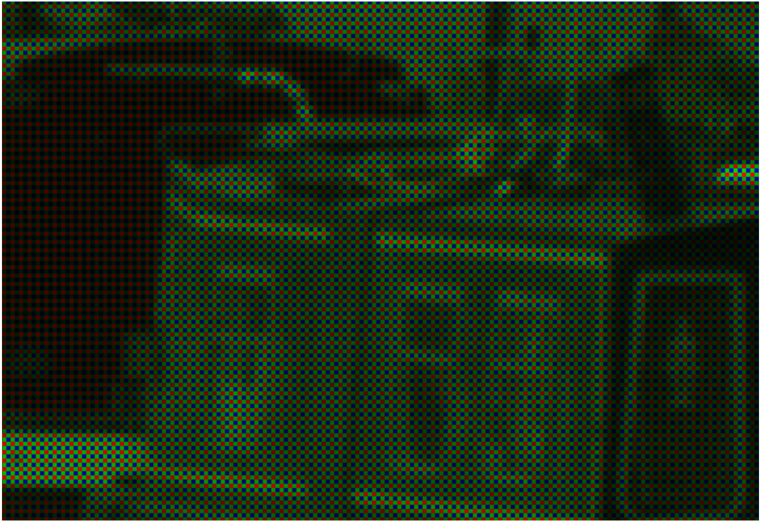


Sensor Data

Coloured Sensor Data



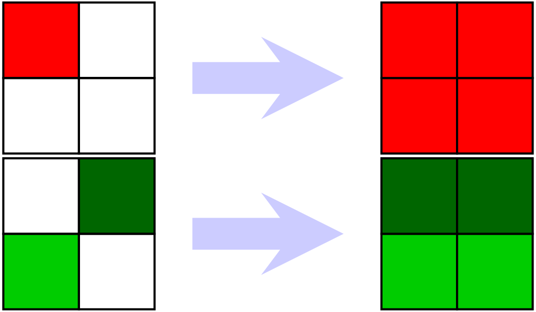
Coloured Sensor Data (Detail)



Why interpolate?

- Each pixel is only R, G or B
- Want full colour, full size image
- So ~~guess~~ interpolate!

Nearest Neighbour



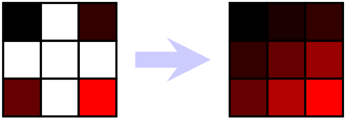
Nearest Neighbour

Nearest Neighbour

Bilinear (and other polynomials)

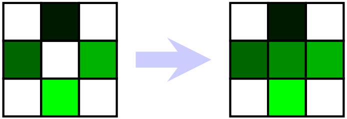
R  and B 

$$R'/B' = R/B * \frac{1}{4}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$



G 

$$G' = G * \frac{1}{4}\begin{bmatrix}0 & 1 & 0\\ 1 & 4 & 1\\ 0 & 1 & 0\end{bmatrix}$$



Bilinear (and other polynomials)

Bilinear (and other polynomials)

Smooth Hue Transition

Seperate luminance (G) and chrominance (R and B).
Interpolate G bilinearly

$$G' = G * \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

For R (and similarly for B), interpolate the ratio $R''_{ij} = \frac{R_{ij}}{G'_{ij}}$, and pointwise multiply by G

$$R'_{ij} = G_{ij} \times \left(R'' * \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)_{ij}$$

Smooth Hue Transition

Smooth Hue Transition

Median filter

- Bilinearly filter R, G, B to get R'', G'', B''.
- Calculate pairwise differences $(R'' - G''$, $R'' - B''$, $G'' - B'')$
- Median filter these to get M_{rg}, M_{rb}, M_{gb}
- Each resulting pixel is CFA pixel image plus/minus appropriate median.
  e.g. (1,0) is a green pixel in CFA, so

$$R'_{1,0} = G_{1,0} + (M_{rg})_{1,0}$$
$$G'_{1,0} = G_{1,0}$$
$$B'_{1,0} = G_{0,0} - (M_{gb})_{1,0}$$

Median filter

Median filter

Gradient-Based

- Want to preserve edges, so 'adaptively' interpolate G
- Approximate horizontal and vertical second derivatives of R and B and take absolutes, e.g.

$$\left|\frac{\partial^2 R}{\partial x^2}\right|_{i,j} \approx \left|\frac{R_{i,j-2} + R_{i,j+2}}{2} - R_{i,j}\right|$$

- Compare these H and V. If $H_{i,j} < V_{i,j}$, (i,j) is a horizontal edge, so interpolate horizontally.

$$G'_{i,j} = \begin{cases} \frac{G_{i,j-1}+G_{i,j+1}}{2} & H_{i,j} < Vi,j \\ \frac{G_{i-1,j}+G_{i+1,j}}{2} & H_{i,j} > Vi,j \\ \frac{G_{i,j-1}+G_{i,j+1}+G_{i-1,j}+G_{i+1,j}}{4} & H_{i,j} = Vi,j \end{cases}$$

Gradient-Based

Gradient-Based

And more...

- **Adaptive Colour Plane**
  Has adaptive interpolation for G using first order derivative of luminance and second order derivative of chrominance, and uses adaptive interpolation again for R and B.
- **Threshold-Based Variable Number of Gradients**
  Eight gradient samples taken from a 5×5 neighbourhood of each pixel, averages are calculated for each gradient, gradients of values less than a (dynamic!) threshold are averaged, and averages are added to/subtracted from the CFA values.

(Though both are similar in principle to the gradient-based)

Why detect?

- Image/Camera verification
- Identification of forged regions
- Recognition of PRCG (PhotoRealistic Computer Generated images)

General idea

- Interpolation creates correlation
- Most CFA interpolation is regular and approximately linear (especially for G)
- If can determine some regular correlation, image is interpolated

EM Algorithm

- ‘Expectation-Maximisation’
- Simulataneously estimate parameters of correlation (i.e. what interpolation is used) and which points are correlated to their neighbours
- Two-step iterative algorithm
- Creates a separable parameter space

A statistical Achilles' Heel

- Interpolation creates correlation
- Correlation decreases variance
- Variance can be measured!
- Periodic low variance is indicative of interpolation

A statistical Achilles' Heel

σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2
$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$
σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2
$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$
σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2
$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$
σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2	$\frac{\sigma^2}{4}$	σ^2

Gallagher and Chen

- First, high-pass filter:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Estimate variance using mean of absolutes along anti-diagonals

$$m(d) = \frac{\sum\limits_{x+y=d} |(h * i)_{x,y}|}{N_d}$$

Gallagher and Chen

- Want to find periodicity, so DFT to get $\left|M(e^{i\omega})\right|$
- Significant peaks demonstrate periodicity
- Green channel interpolates every other pixel, so expect a peak at $\omega = \pi$
- Quantify peak s as:

$$s = \frac{\left|M(e^{i\omega})\right|_{\omega=\pi}}{\text{median}_{\omega}\{\left|M(e^{i\omega})\right|\}}$$

Gallagher and Chen example 1

Original

Gallagher and Chen example 1

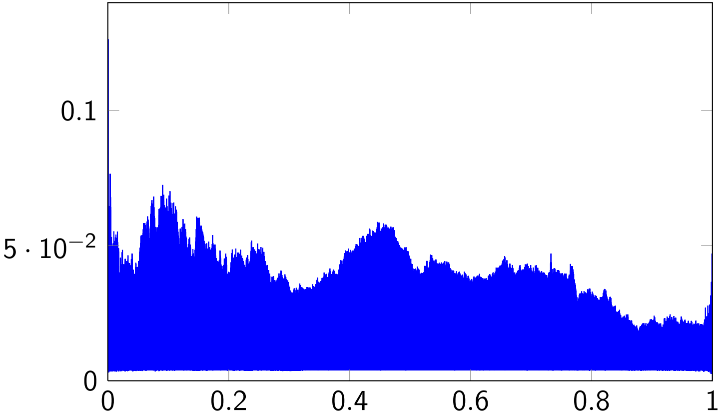
$i =$

Gallagher and Chen example 1

$(h * i) =$

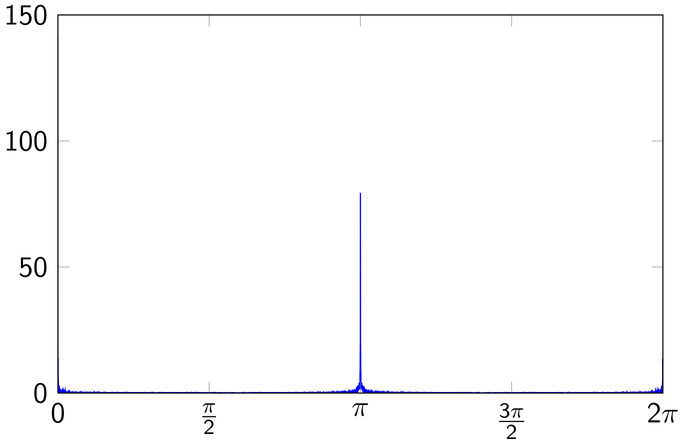
Gallagher and Chen example 1

$m(d) =$



Gallagher and Chen example 1

$$\left|M(e^{i\omega})\right| =$$



Gallagher and Chen example 1

$$
\begin{aligned}
s &= \frac{\left|M(e^{i\omega})\right|_{\omega=\pi}}{\mathrm{median}_{\omega}\{|M(e^{i\omega})|\}} \\
&= \frac{79.2337}{0.1191} \\
&= 665.0172
\end{aligned}
$$

Gallagher and Chen example 2

Original

Gallagher and Chen example 2

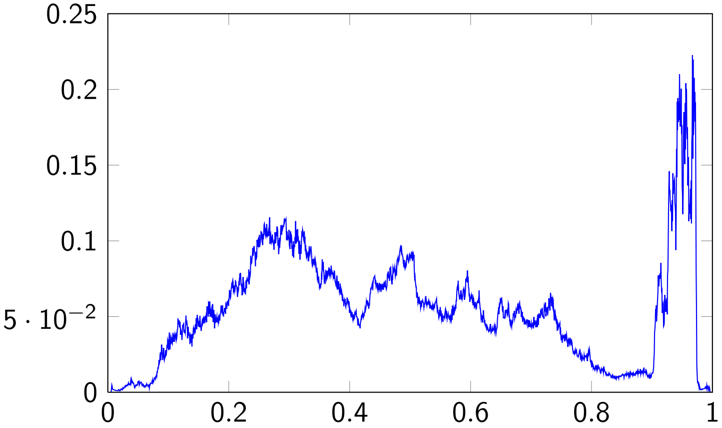
$i =$

Gallagher and Chen example 2

$(h * i) =$

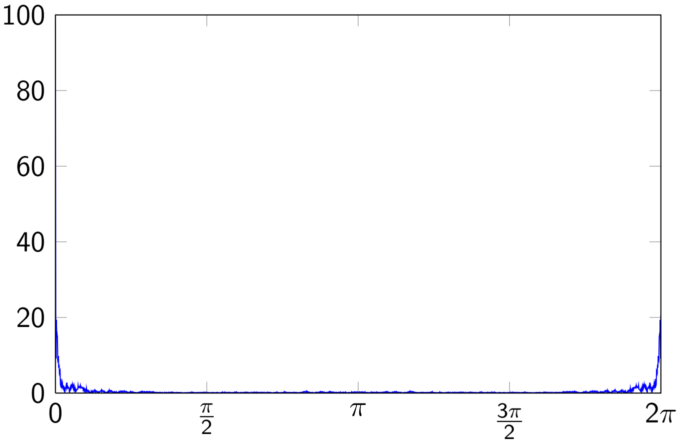
Gallagher and Chen example 2

$m(d) =$



Gallagher and Chen example 2

$$\left|M(e^{i\omega})\right| =$$



Gallagher and Chen example 2

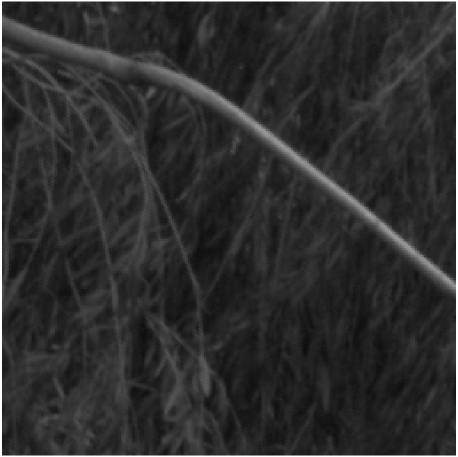
$$
\begin{aligned}
s &= \frac{\left|M(e^{i\omega})\right|_{\omega=\pi}}{\text{median}_{\omega}\{|M(e^{i\omega})|\}} \\
&= \frac{0.2546}{0.1688} \\
&= 1.5081
\end{aligned}
$$

Gallagher and Chen example 3

Original

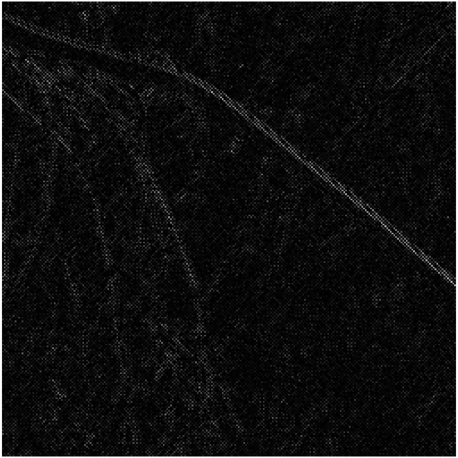
Gallagher and Chen example 3

$i =$



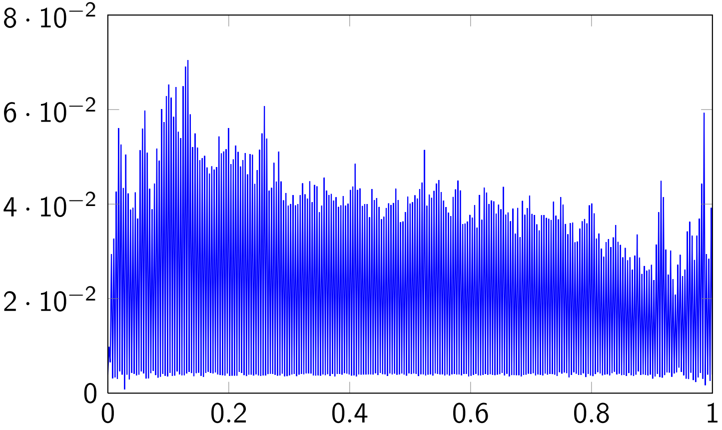
Gallagher and Chen example 3

$(h * i) =$



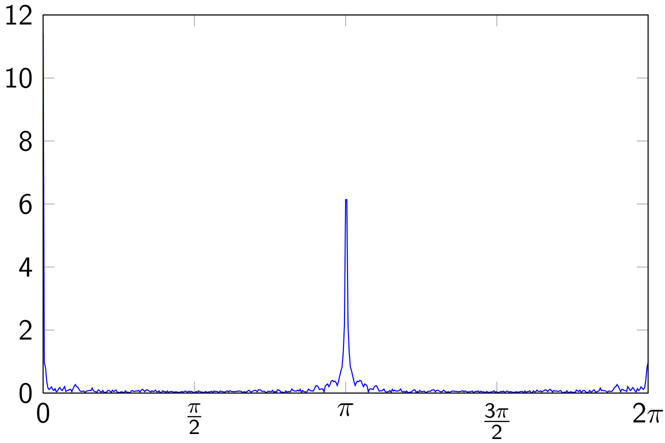
Gallagher and Chen example 3

$m(d) =$



Gallagher and Chen example 3

$$\left|M(e^{i\omega})\right| =$$



Gallagher and Chen example 3

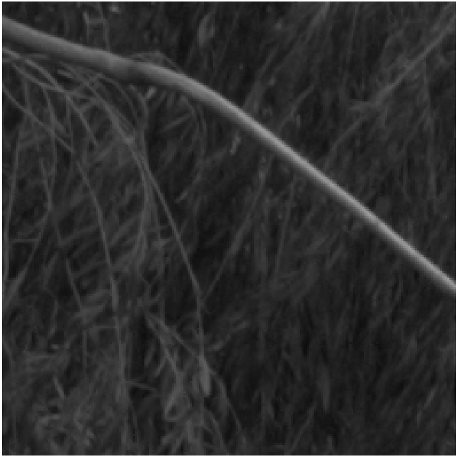
$$
\begin{aligned}
s &= \frac{\left|M(e^{i\omega})\right|_{\omega=\pi}}{\text{median}_{\omega}\{|M(e^{i\omega})|\}} \\
&= \frac{6.1377}{0.0574} \\
&= 106.8595
\end{aligned}
$$

Gallagher and Chen example 4

Original

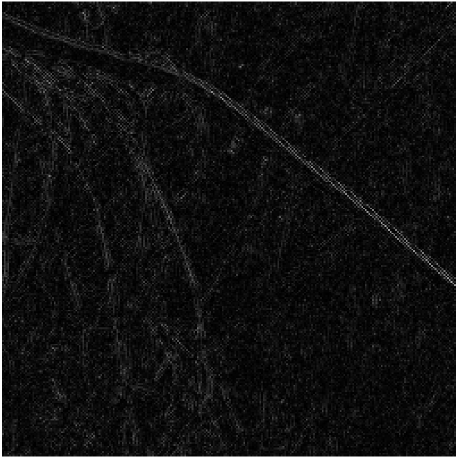
Gallagher and Chen example 4

$i =$



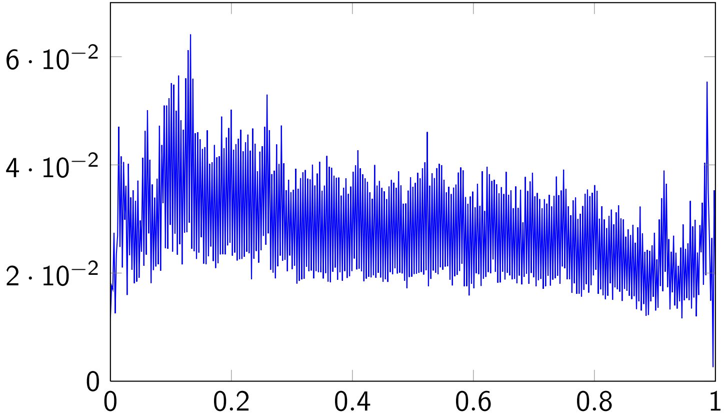
Gallagher and Chen example 4

$(h * i) =$



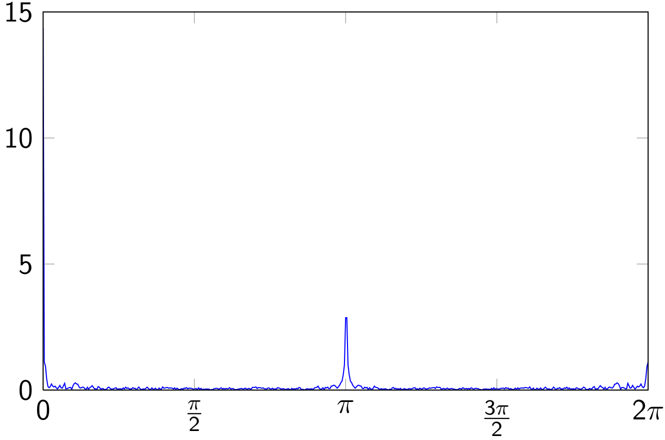
Gallagher and Chen example 4

$m(d) =$



Gallagher and Chen example 4

$$\left|M(e^{i\omega})\right| =$$



Gallagher and Chen example 4

$$
\begin{aligned}
s &= \frac{\left|M(e^{i\omega})\right|_{\omega=\pi}}{\text{median}_{\omega}\{|M(e^{i\omega})|\}} \\
&= \frac{2.8713}{0.0637} \\
&= 45.1039
\end{aligned}
$$

Problems (that aren't)

- Non-linear interpolation makes this less simple
- JPEG introduces compression
- The demonstrated method doesn't give the interpolation parameters
- Can't you fake CFA interpolation?

Problems (that aren't)

- Non-linear interpolation makes this less simple
  But non-linear interpolation is still locally sufficiently linear, so it's close enough
- JPEG introduces compression
- The demonstrated method doesn't give the interpolation parameters
- Can't you fake CFA interpolation?

Problems (that aren't)

- Non-linear interpolation makes this less simple
  But non-linear interpolation is still locally sufficiently linear, so it's close enough
- JPEG introduces compression
  But high-quality JPEG leaves enough information
- The demonstrated method doesn't give the interpolation parameters
- Can't you fake CFA interpolation?

Problems (that aren't)

- Non-linear interpolation makes this less simple
  But non-linear interpolation is still locally sufficiently linear, so it's close enough
- JPEG introduces compression
  But high-quality JPEG leaves enough information
- The demonstrated method doesn't give the interpolation parameters
  No, but we don't need them just for detection
- Can't you fake CFA interpolation?

Problems (that aren't)

- Non-linear interpolation makes this less simple
  But non-linear interpolation is still locally sufficiently linear, so it's close enough
- JPEG introduces compression
  But high-quality JPEG leaves enough information
- The demonstrated method doesn't give the interpolation parameters
  No, but we don't need them just for detection
- Can't you fake CFA interpolation?
  Sure, but adding such a method of detection makes forgery harder

Identifying Forged Regions

- Can identify forged regions by running the above algorithm locally on each pixel
- For each pixel, estimate local (within radius n) variance

$$m(x, y) = \frac{\sum_{i=-n}^{n} |(h * i)_{x+i,y+i}|}{2n+1}$$

- For each pixel, calculate local DFT, and from it the peak s_{xy}
- Low values of s_{xy} means pixel (x, y) part of forged region

Tamper hiding

- Tampering with a photo destroys CFA correlations
- Restoring CFA-like correlations hides tampering
- Want an image of similar size and quality

Naïve method

- Can simply sample image into CFA image, and reinterpolate
- Problem: Throws away $\frac{2}{3}$ of pixel data!
- Similar information loss to linear kernel smoothing (e.g. Gaussian)

‘Ideal’ method

- Can represent linear interpolation as matrix operation

$$\mathbf{y} = \mathbf{H}\mathbf{x}$$

- Manipulated image adds an additive residual signal

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \epsilon$$

- For given **H**, want to find **x** which minimises $\|\epsilon\|$

'Ideal' method

- This is a least squares problem, with solution

$$\mathbf{x} = \underbrace{(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T}_{\text{Moore-Penrose Pseudoinverse}}\mathbf{y}$$

- Can therefore synthesise a least error CFA pattern

$$\mathbf{y}_{\text{CFA}} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$

- But calculating $(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ efficiently is tricky...