

Algorithms for Digital Color Cameras

John F. Hamilton, Jr.

Research Fellow
Eastman Kodak Company

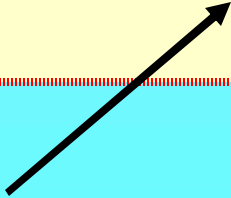


Problems in the World of Digital Color Imaging

It all starts with a discretely sampled image.

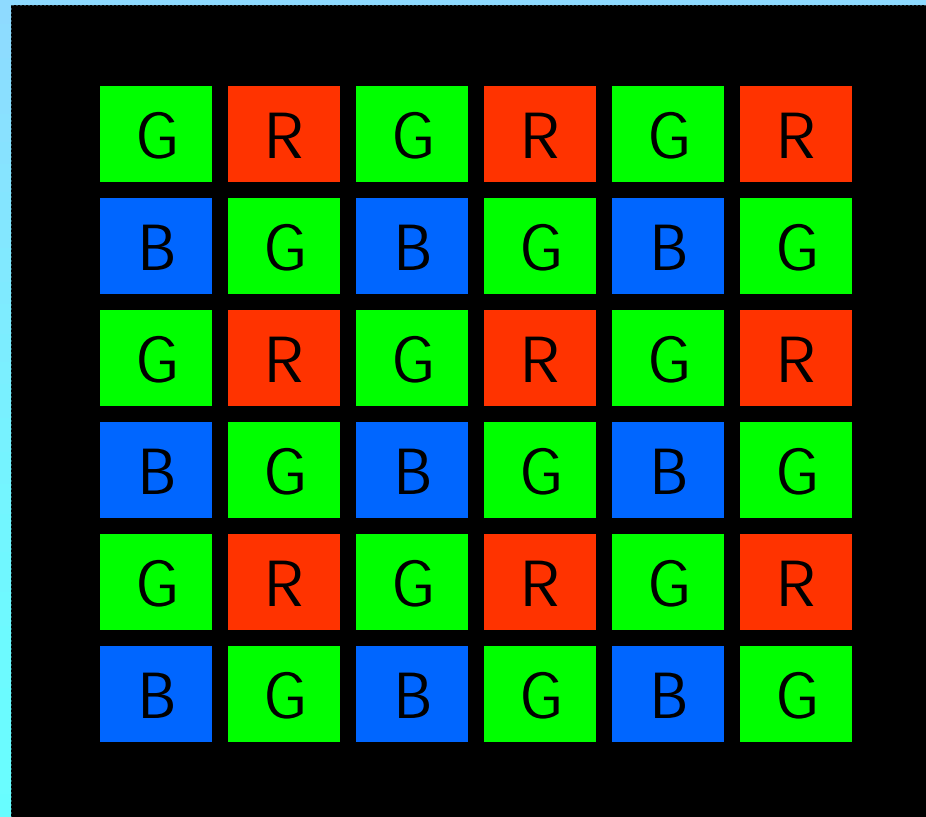
Most silicon sensors capture only one color value per pixel, but a full color image requires three colors (RGB) per pixel. Thus, two-thirds of every color image must be computed.

color interpolation



Color Capture

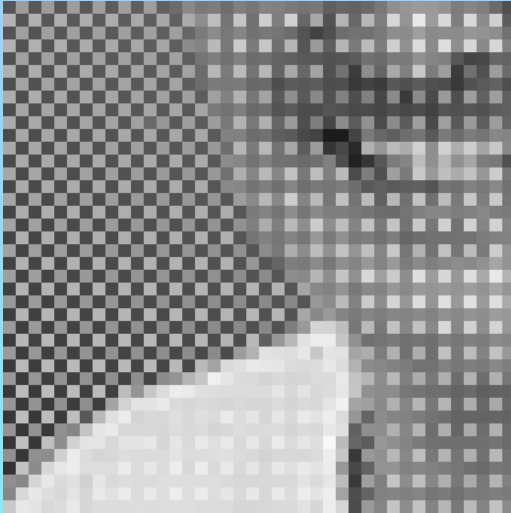
Most color sensors use the Bayer color filter array (CFA).



CFA Image Data



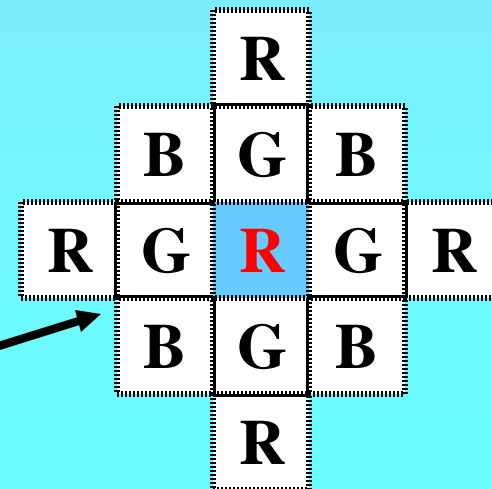
How to begin?



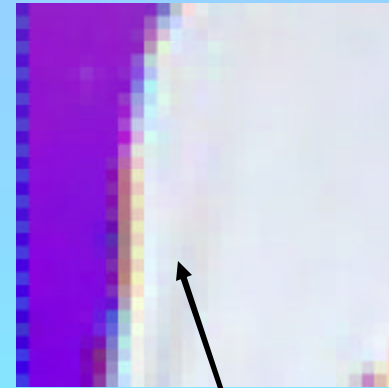
Each pixel has one color value.

Why not find the missing color values using bilinear interpolation?

Look at the code values in the vicinity of a pixel of interest.



Bilinear Interpolation

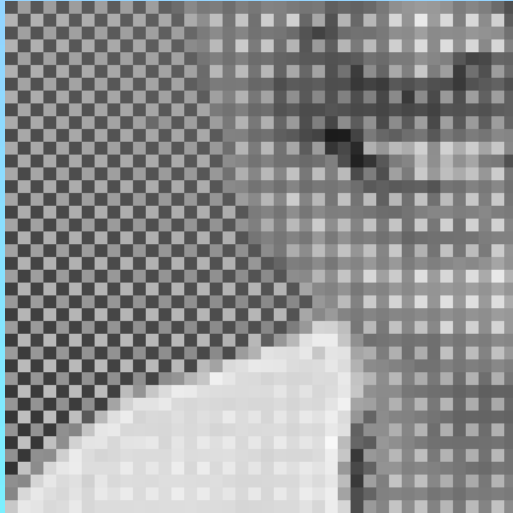


**What we
get is...**

**the zipper artifact and
the color edge artifact.**



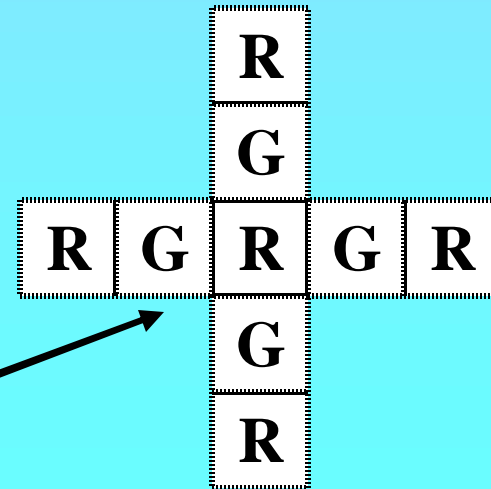
The Starting Point



50% of these pixels
captured green values.



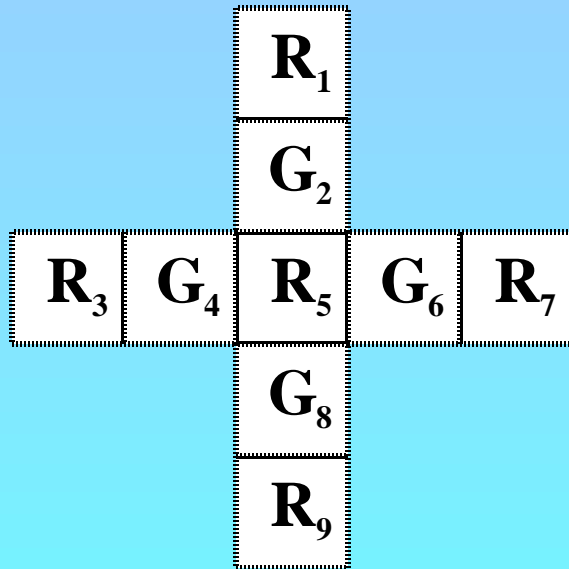
We start by finding green values
for the remaining 50%.



Because the central pixel is red,
we only need red and green values.



Green Interpolation

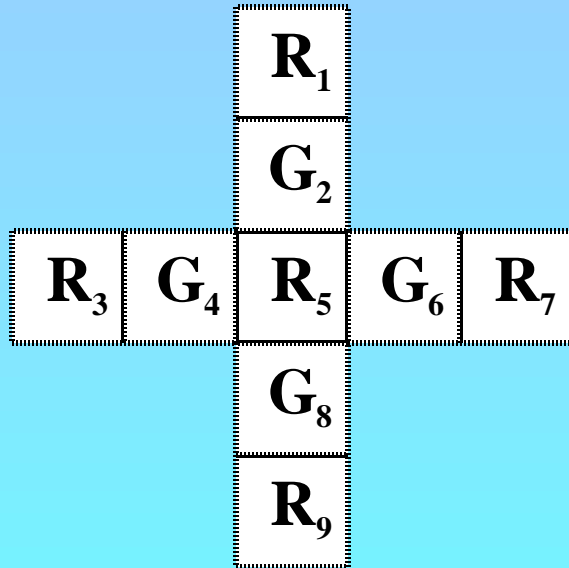


The main idea is to avoid interpolating across edges.

We compute two green estimates (one horizontal and one vertical).

We also compute two classifiers that tell us if either direction is likely to cross an edge.

Green Interpolation



The main idea is to avoid interpolating across edges.

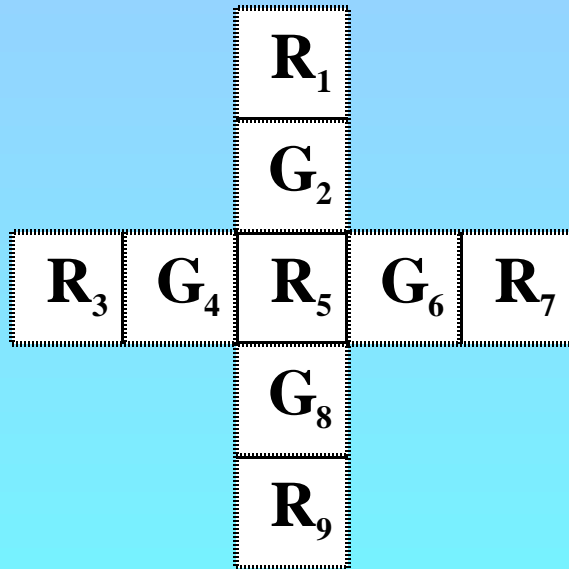
$$G_{5h} = (G_4 + G_6) / 2 + (-R_3 + 2R_5 - R_7) / 6$$

$$G_{5v} = (G_2 + G_8) / 2 + (-R_1 + 2R_5 - R_9) / 6$$

$$\text{horz} = \text{abs}(G_4 - G_6) + \text{abs}(-R_3 + 2R_5 - R_7)$$

$$\text{vert} = \text{abs}(G_2 - G_8) + \text{abs}(-R_1 + 2R_5 - R_9)$$

Estimator: Green Average



The green average is
our initial estimate.



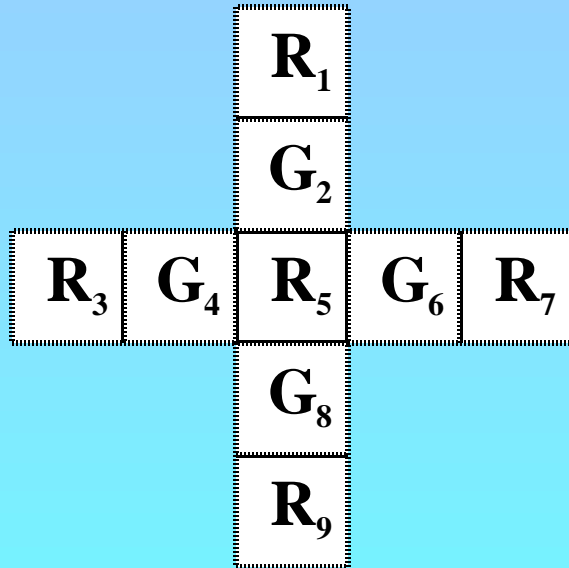
$$G_{5h} = (G_4 + G_6) / 2 + (-R_3 + 2R_5 - R_7) / 6$$

$$G_{5v} = (G_2 + G_8) / 2 + (-R_1 + 2R_5 - R_9) / 6$$

$$\text{horz} = \text{abs}(G_4 - G_6) + \text{abs}(-R_3 + 2R_5 - R_7)$$

$$\text{vert} = \text{abs}(G_2 - G_8) + \text{abs}(-R_1 + 2R_5 - R_9)$$

Estimator: Red Laplacian



The red second difference (Laplacian) is our correction term.



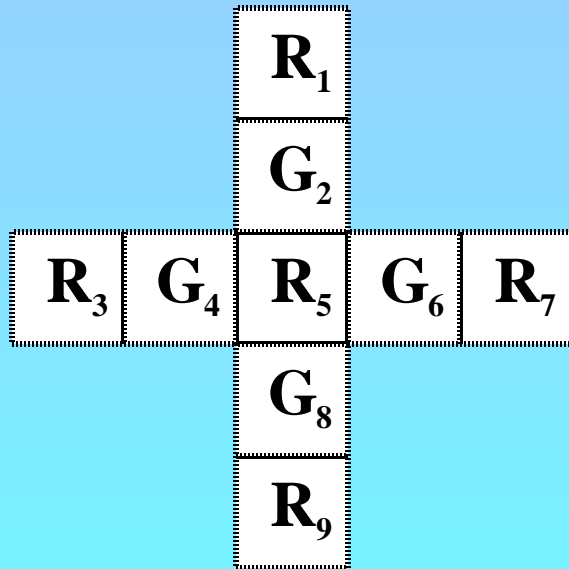
$$G_{5h} = (G_4 + G_6) / 2 + (-R_3 + 2R_5 - R_7) / 6$$

$$G_{5v} = (G_2 + G_8) / 2 + (-R_1 + 2R_5 - R_9) / 6$$

$$\text{horz} = \text{abs}(G_4 - G_6) + \text{abs}(-R_3 + 2R_5 - R_7)$$

$$\text{vert} = \text{abs}(G_2 - G_8) + \text{abs}(-R_1 + 2R_5 - R_9)$$

Classifier: Green Gradient



A non-zero green gradient indicates a possible edge.

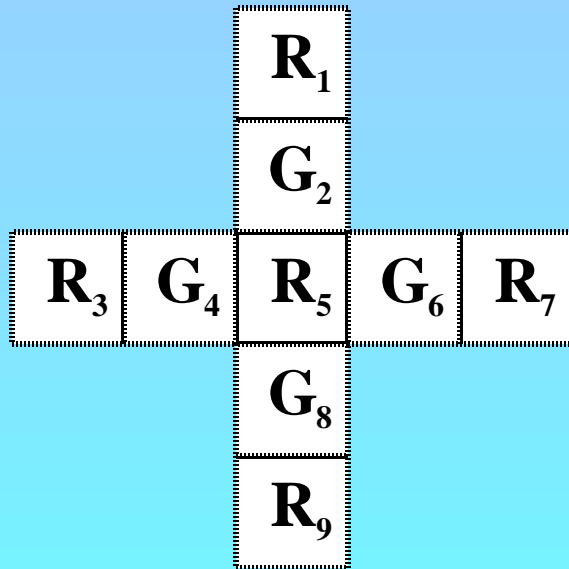
$$G_{5h} = (G_4 + G_6) / 2 + (-R_3 + 2R_5 - R_7) / 6$$

$$G_{5v} = (G_2 + G_8) / 2 + (-R_1 + 2R_5 - R_9) / 6$$

$$\text{horz} = \text{abs}(G_4 - G_6) + \text{abs}(-R_3 + 2R_5 - R_7)$$

$$\text{vert} = \text{abs}(G_2 - G_8) + \text{abs}(-R_1 + 2R_5 - R_9)$$

Classifier: Red Laplacian



A non-zero red Laplacian also indicates a possible edge.

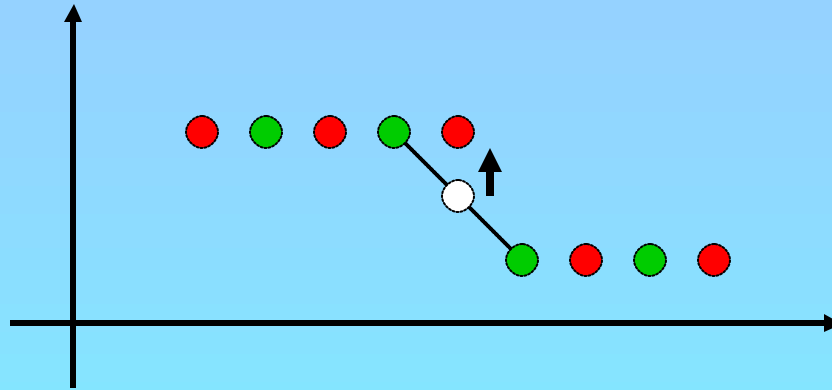
$$G_{5h} = (G_4 + G_6) / 2 + (-R_3 + 2R_5 - R_7) / 6$$

$$G_{5v} = (G_2 + G_8) / 2 + (-R_1 + 2R_5 - R_9) / 6$$

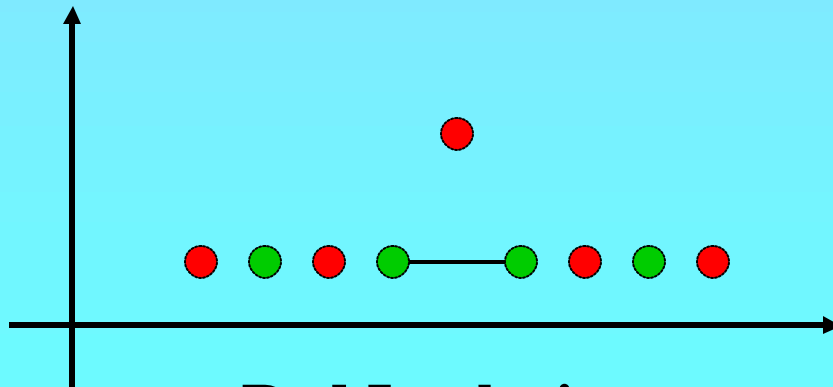
$$\text{horz} = \text{abs}(G_4 - G_6) + \text{abs}(-R_3 + 2R_5 - R_7)$$

$$\text{vert} = \text{abs}(G_2 - G_8) + \text{abs}(-R_1 + 2R_5 - R_9)$$

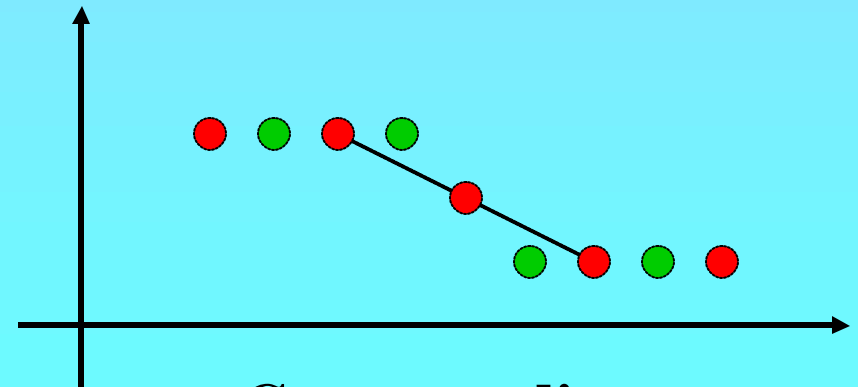
A Quick Overview



The red Laplacian helps correct the green average.

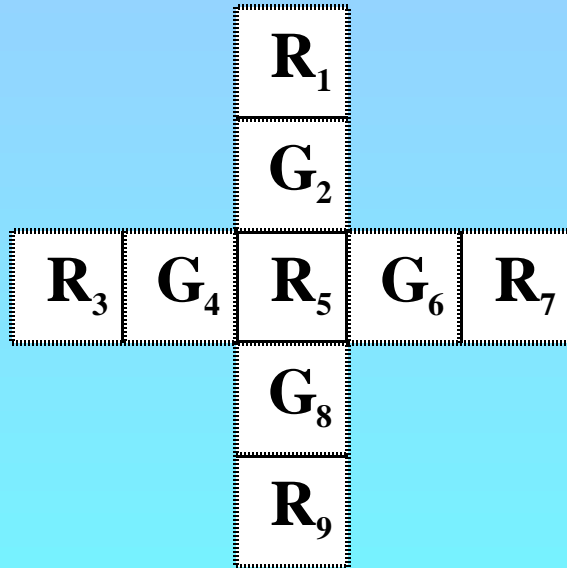


**Red Laplacian
detects the feature**



**Green gradient
detects the feature**

Adaptive Green Interpolation



```

If  vert < horz
      G5 = G5v
else
      G5 = G5h
end
  
```

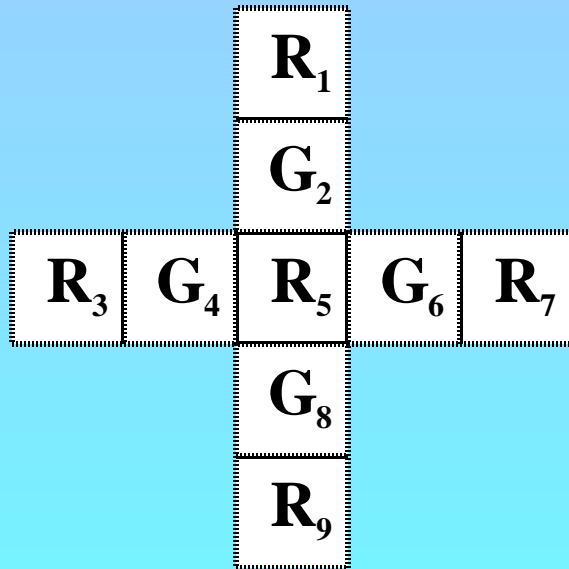
$$G_{5h} = (G_4 + G_6) / 2 + (-R_3 + 2R_5 - R_7) / 6$$

$$G_{5v} = (G_2 + G_8) / 2 + (-R_1 + 2R_5 - R_9) / 6$$

$$\text{horz} = \text{abs}(G_4 - G_6) + \text{abs}(-R_3 + 2R_5 - R_7)$$

$$\text{vert} = \text{abs}(G_2 - G_8) + \text{abs}(-R_1 + 2R_5 - R_9)$$

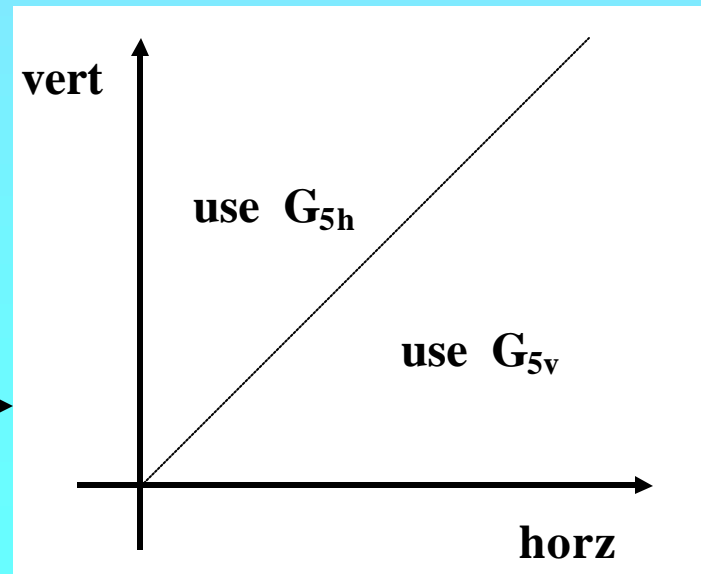
Adaptive Interpolation Policy



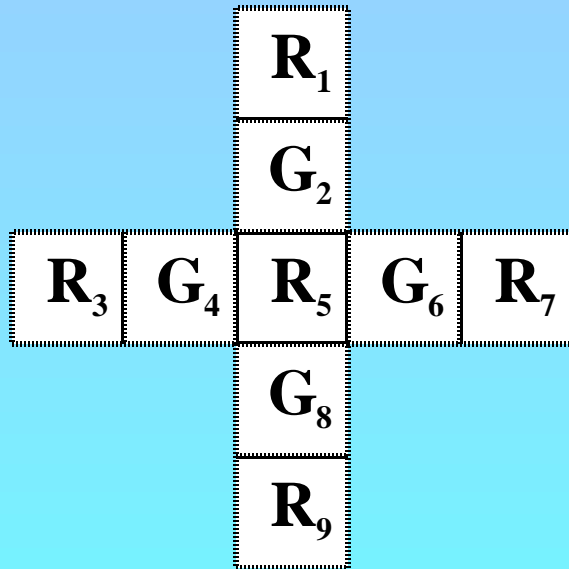
```

If  vert < horz
       $G_5 = G_{5v}$ 
else
       $G_5 = G_{5h}$ 
end
  
```

Interpolation Policy →

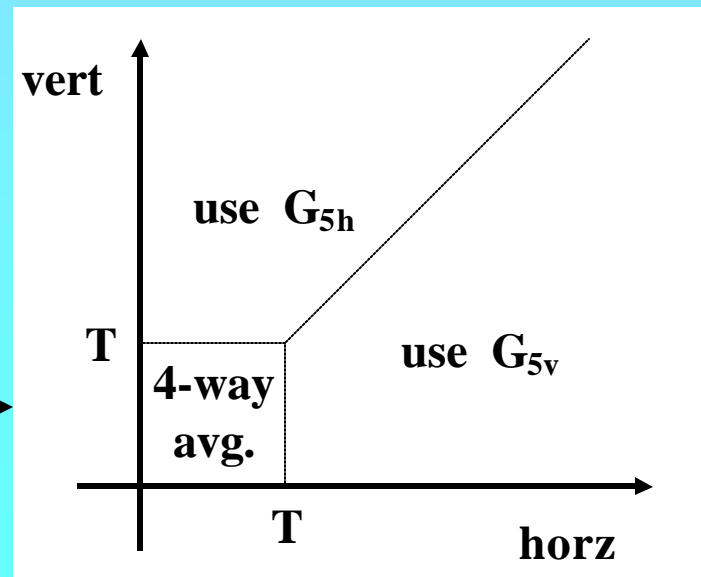


Adaptive Interpolation Policy



We can simultaneously reduce noise in flat fields while maintaining edge integrity.

Threshold T is set to distinguish between image detail and noise. →



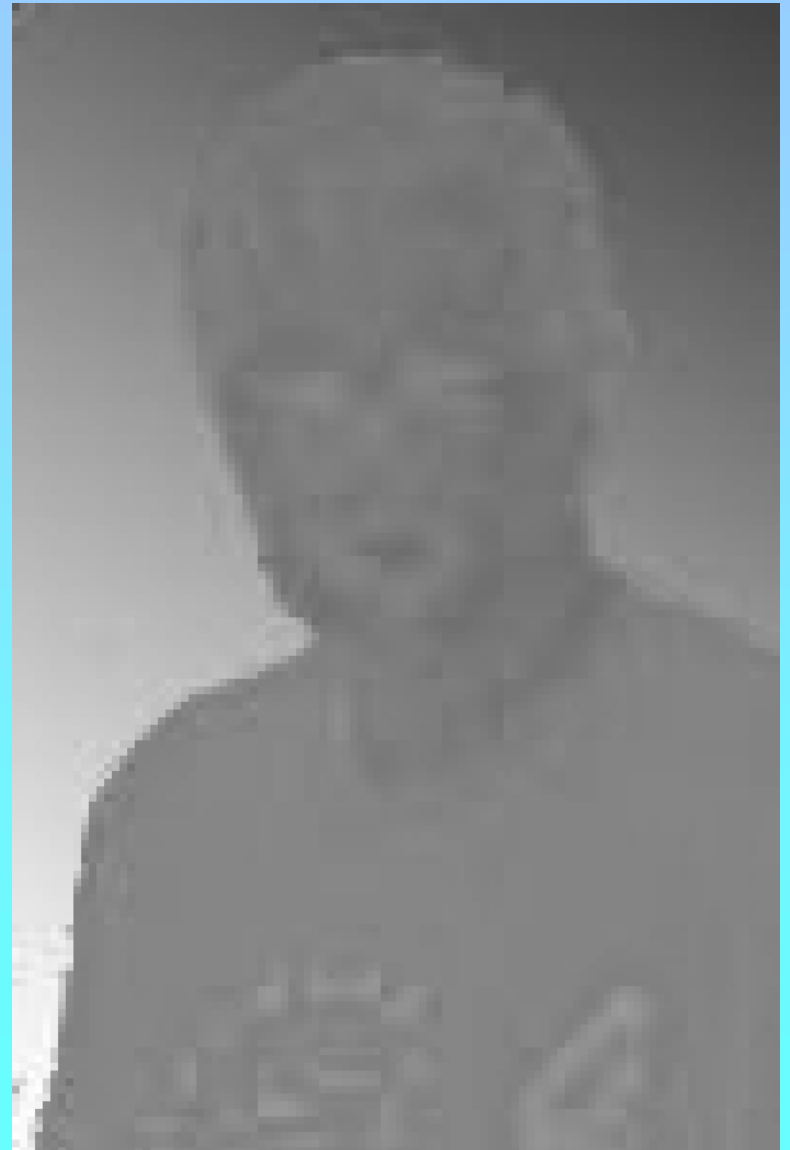
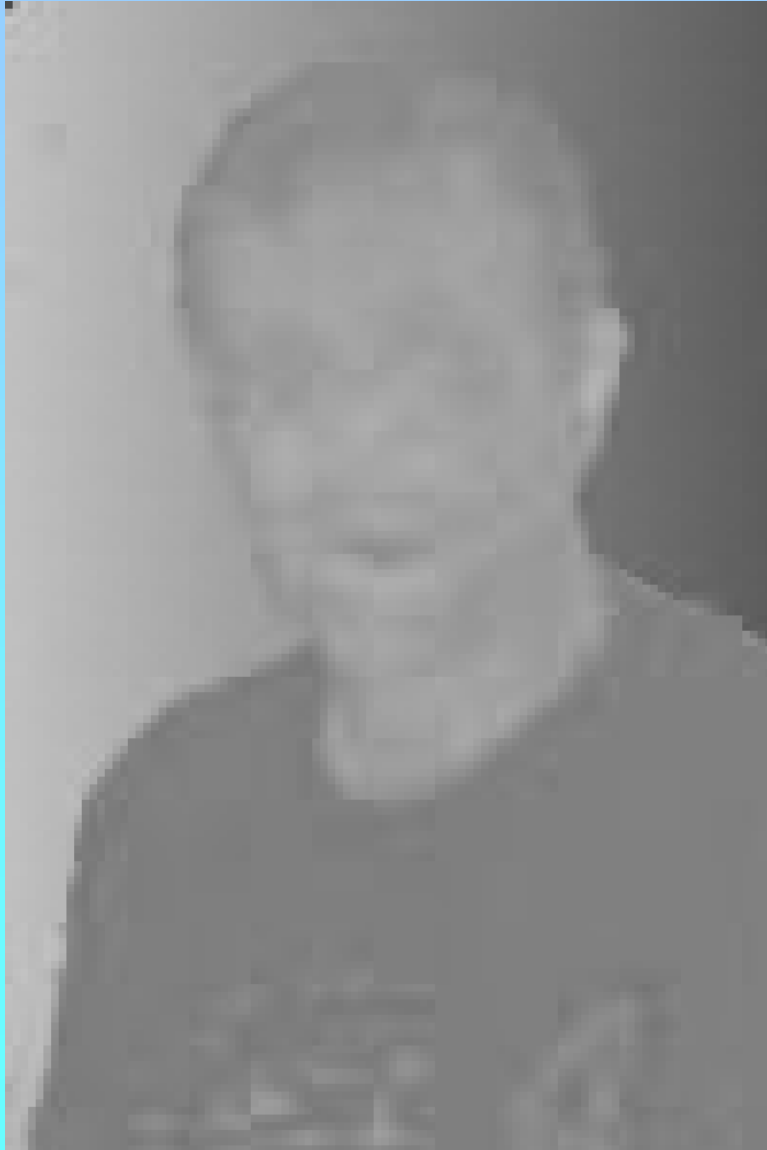
Red and Blue Interpolation

Once every pixel has a green color value, compute R-G and B-G color differences.

Interpolate the color differences using bilinear interpolation.

So, WHY is it okay to use bilinear interpolation NOW??

R-G and B-G



Color Differences

Color difference records tend to have:

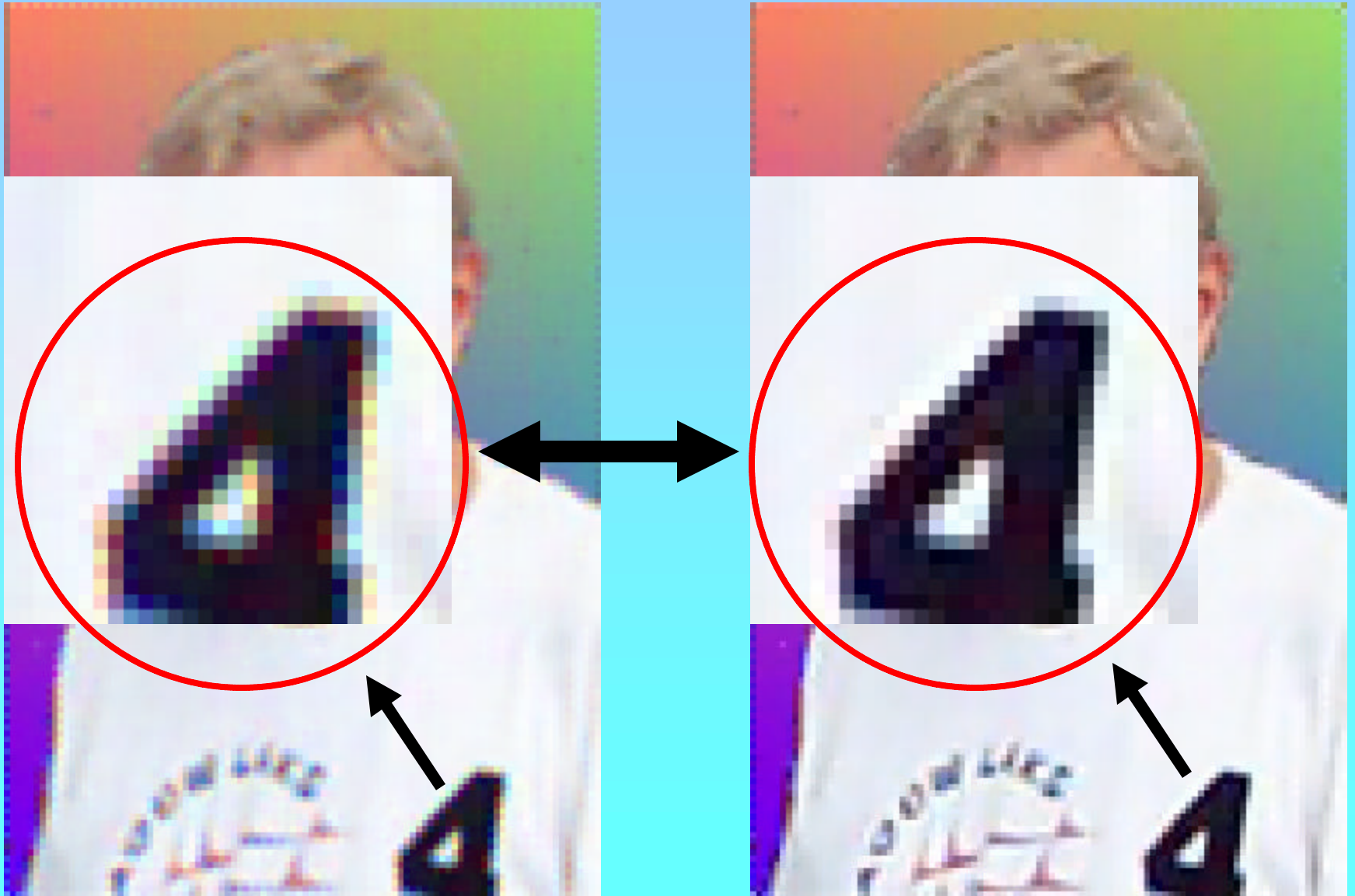
- (1) low contrast, and**
- (2) little high spatial frequency detail**

**Thus, color difference records
are generally well conditioned
for bilinear interpolation.**

Non-Adaptive vs Adaptive

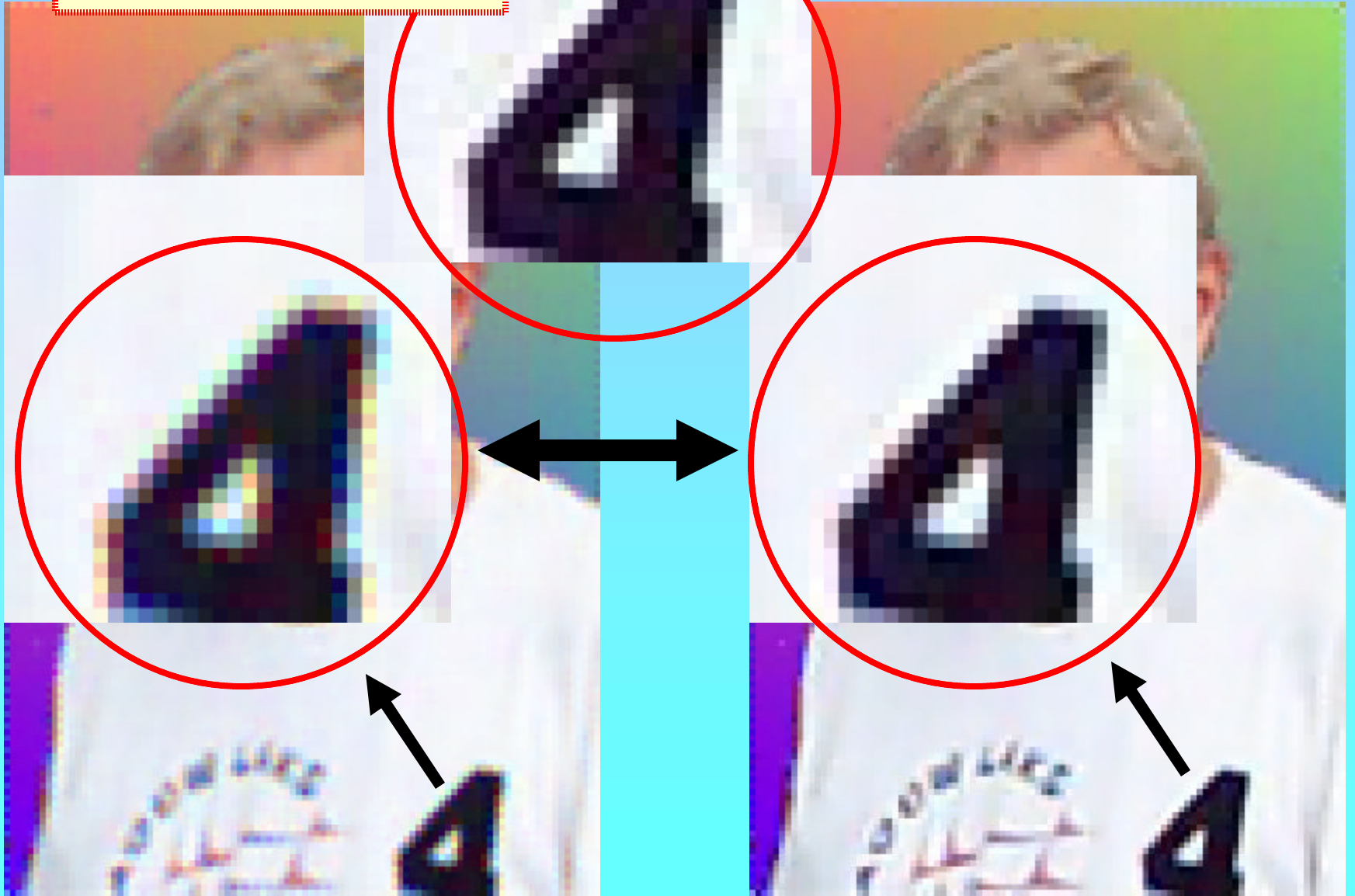


Non-Adaptive vs Adaptive



Original

Adaptive



Another Problem in the World of Digital Color Imaging

Sampled images are subject to aliasing.

When the spatial frequency content of an image exceeds the Nyquist limit for a given sensor, aliasing occurs.

Aliasing in a Fabric Pattern

A common source of aliasing artifacts is fabric.

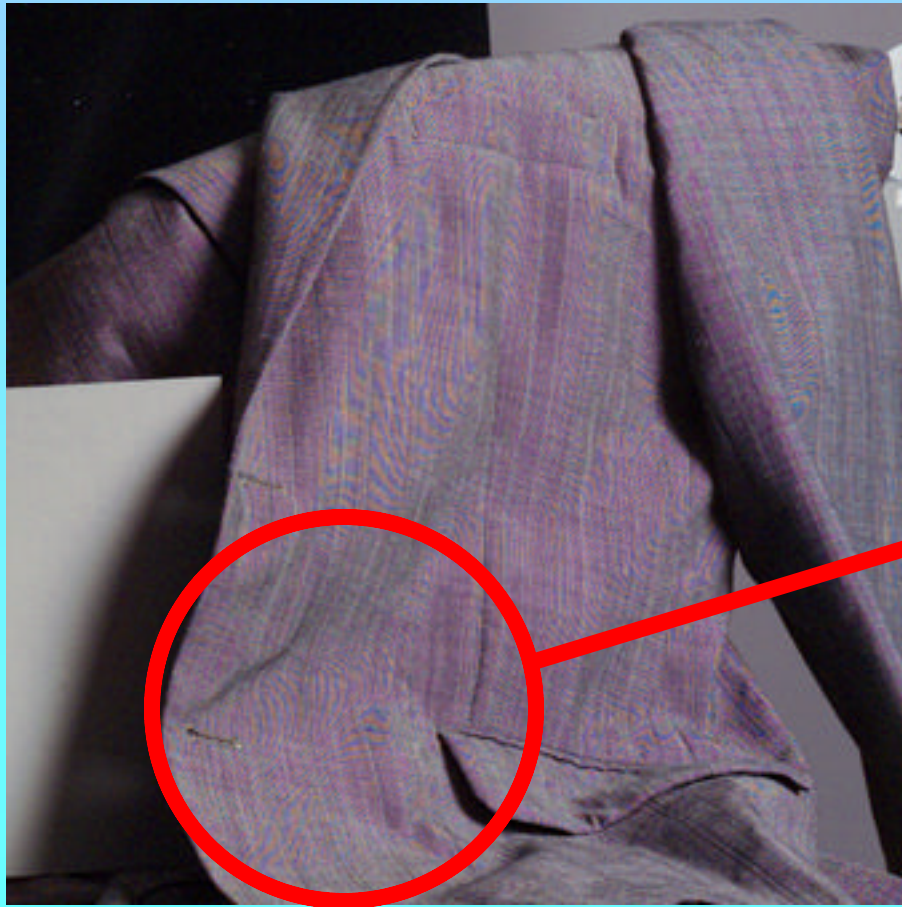


Different threads in a weave pattern can produce a high-contrast, high spatial frequency input signal.

Because the color channels alias differently, the artifact becomes color aliasing and is even more objectionable.

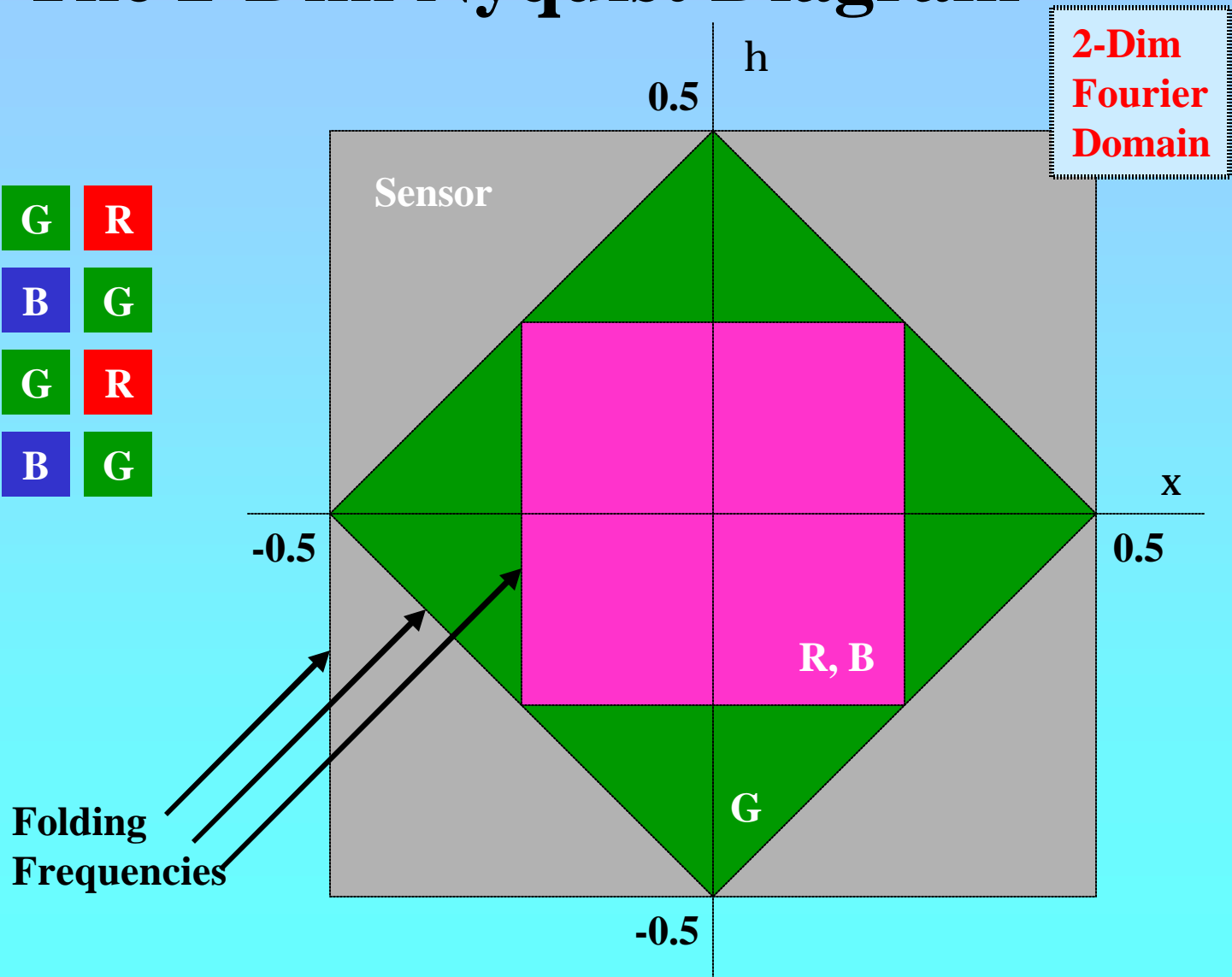
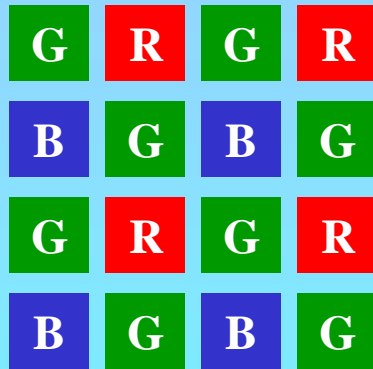
Aliasing in a Fabric Pattern

Aliasing can cause colors to appear where none were before.

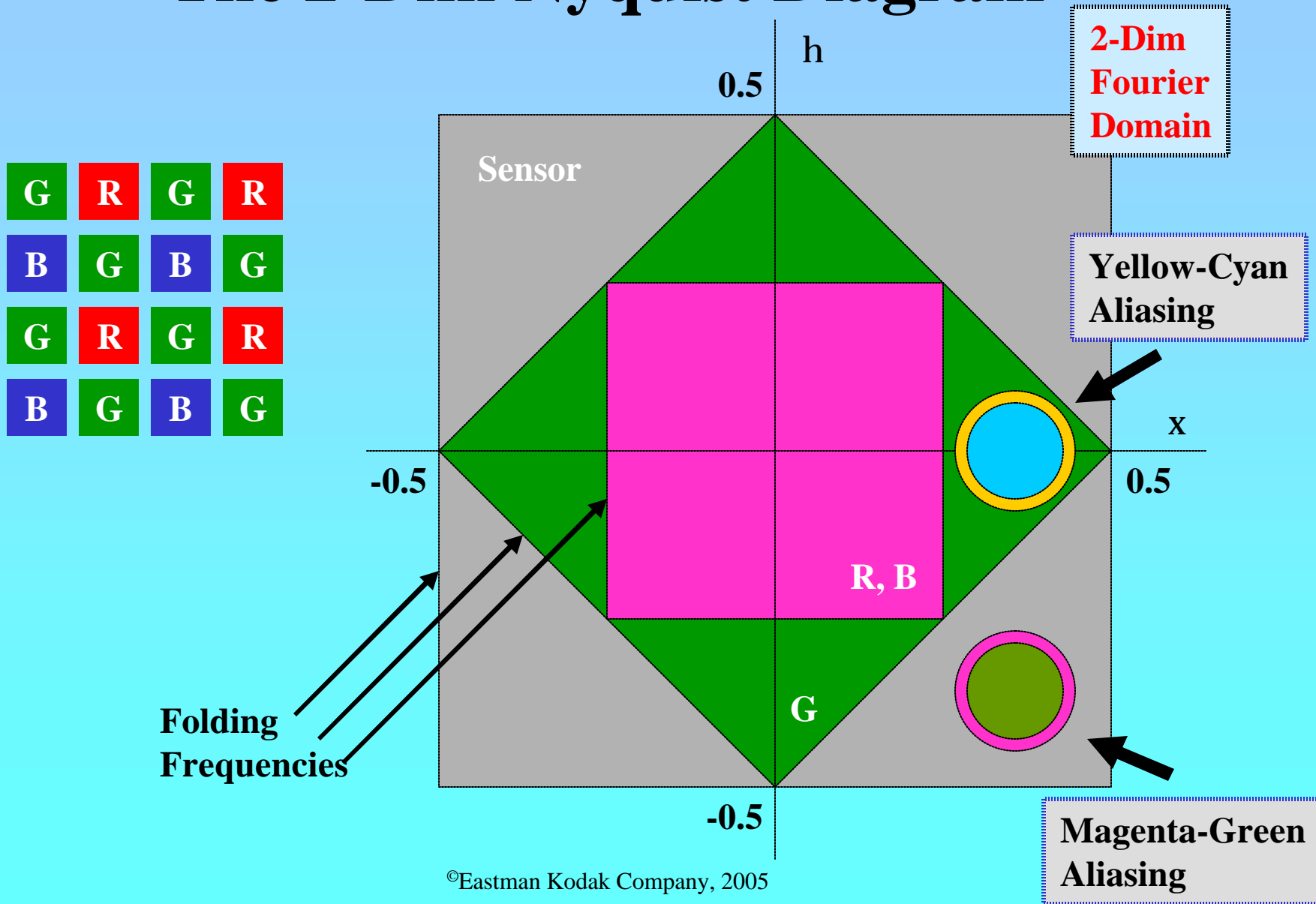


ZOOM

The 2-Dim Nyquist Diagram

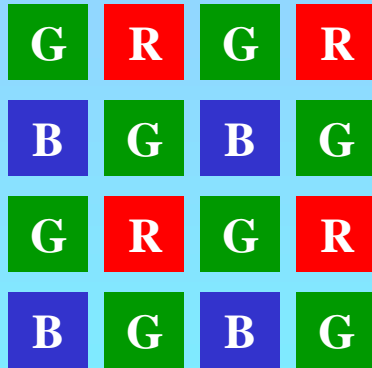


The 2-Dim Nyquist Diagram

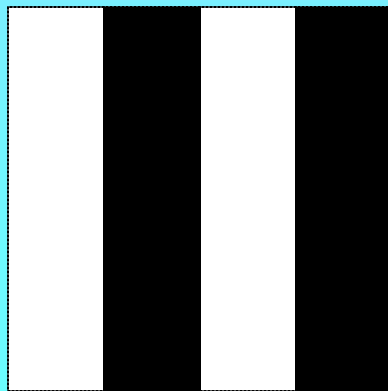
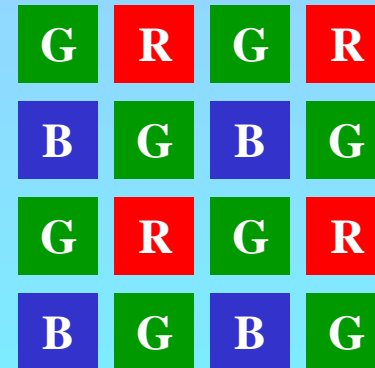


The 2-Dim Nyquist Diagram

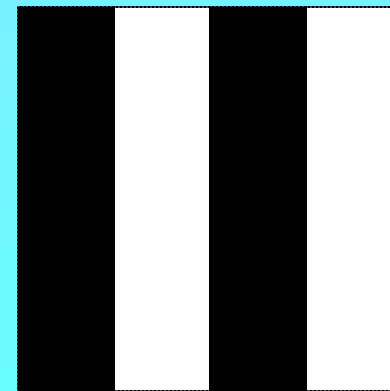
Dark Cyan



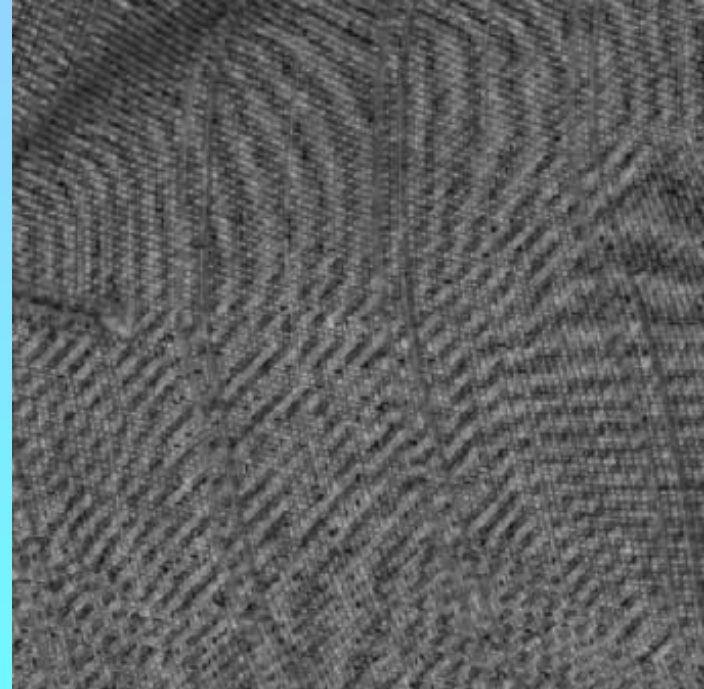
Dark Yellow



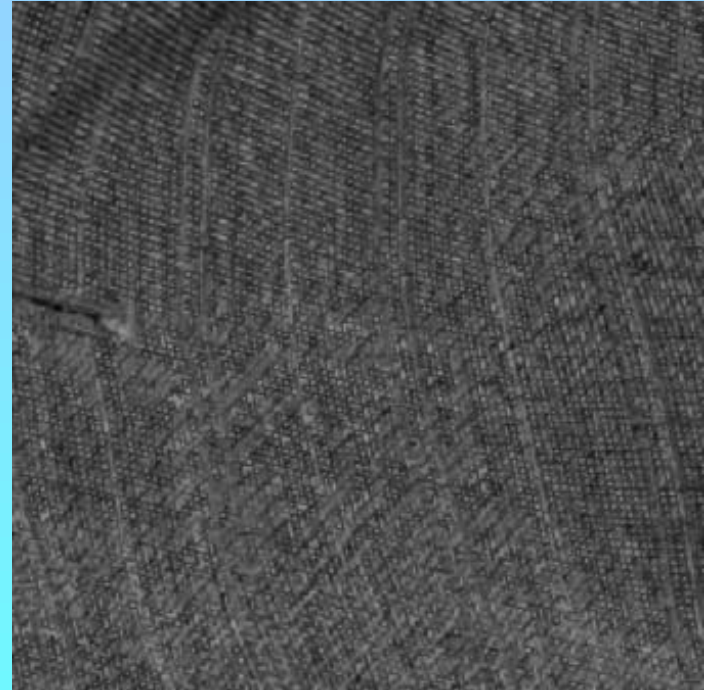
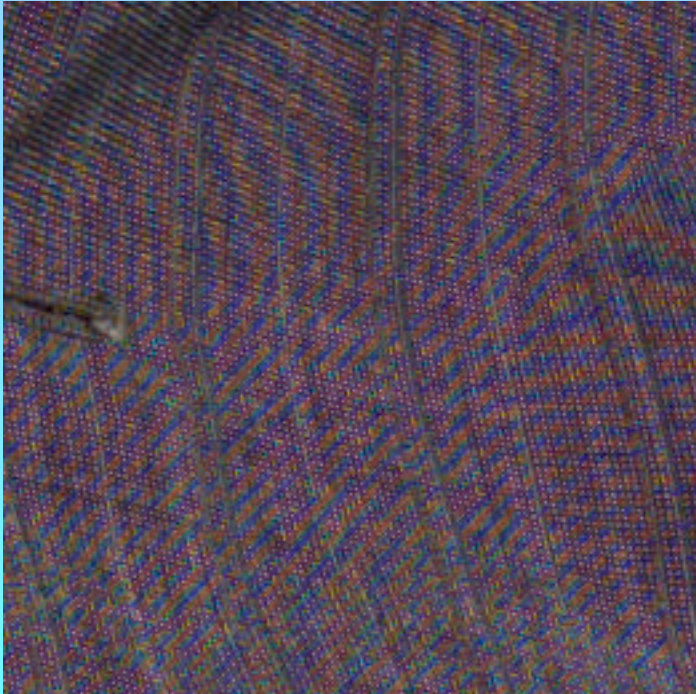
Slight shift in registration



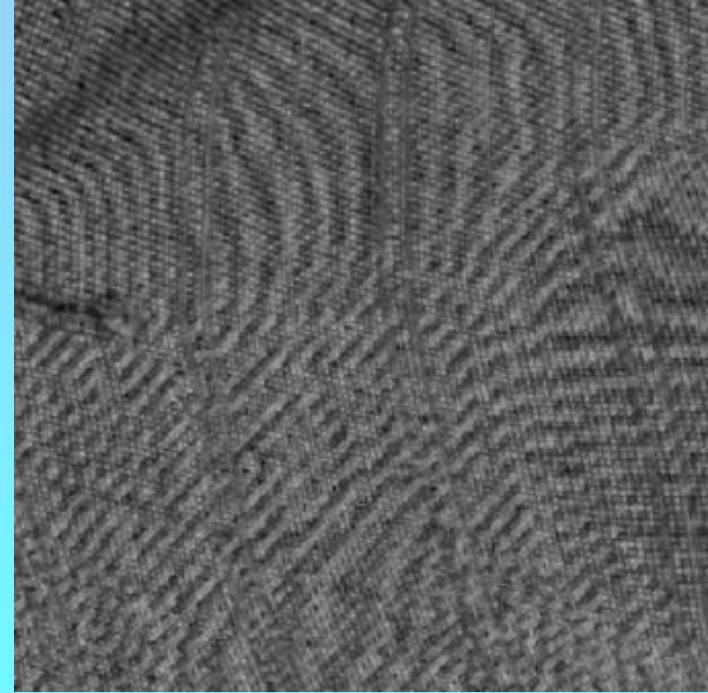
Red Channel



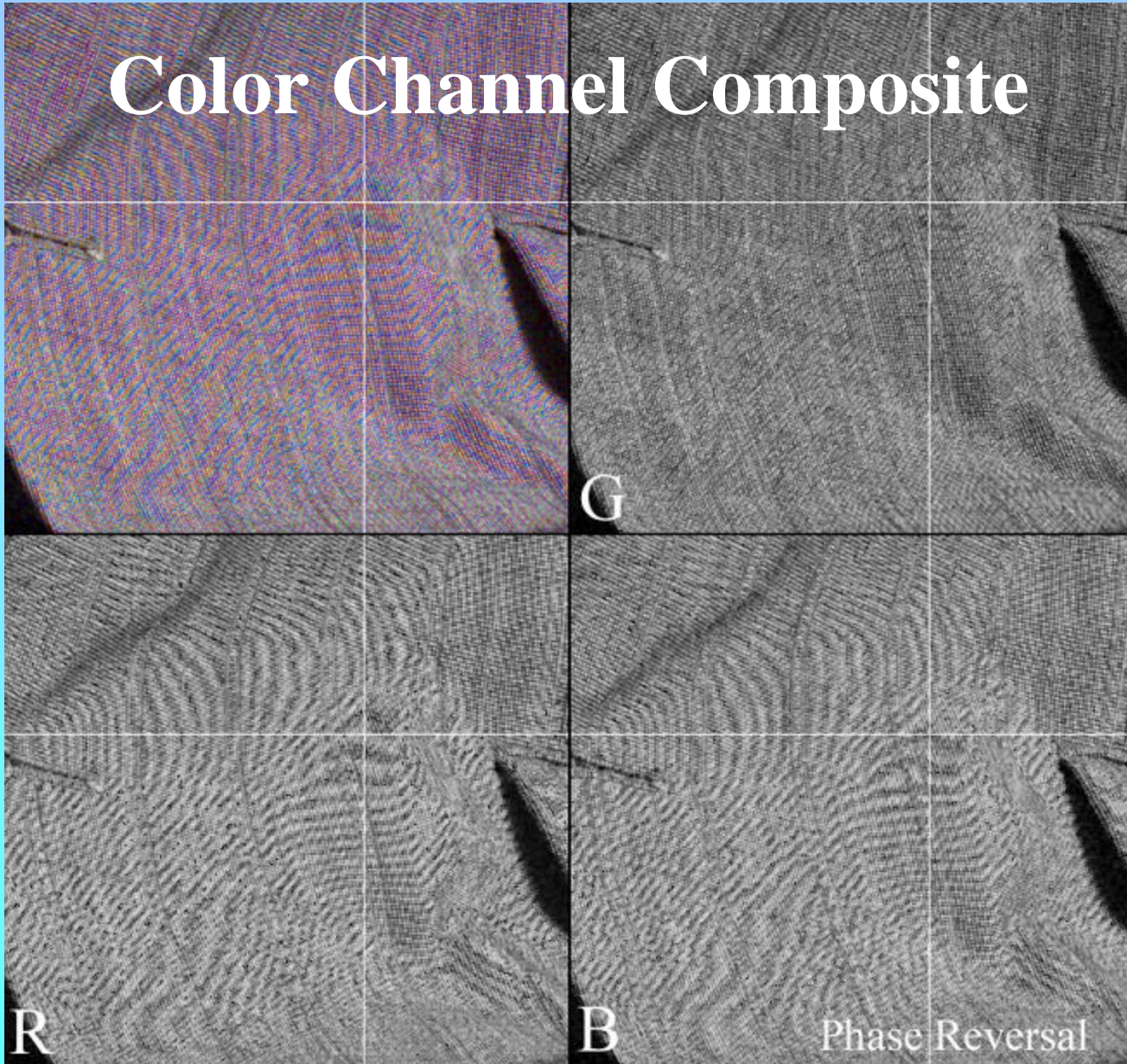
Green Channel



Blue Channel



Color Channel Composite



How do we fix this?

In many cases, the aliasing is of the yellow-cyan type.

Also, in many cases, the green channel still carries high frequencies while the red and blue channels have aliased back down to low frequencies.

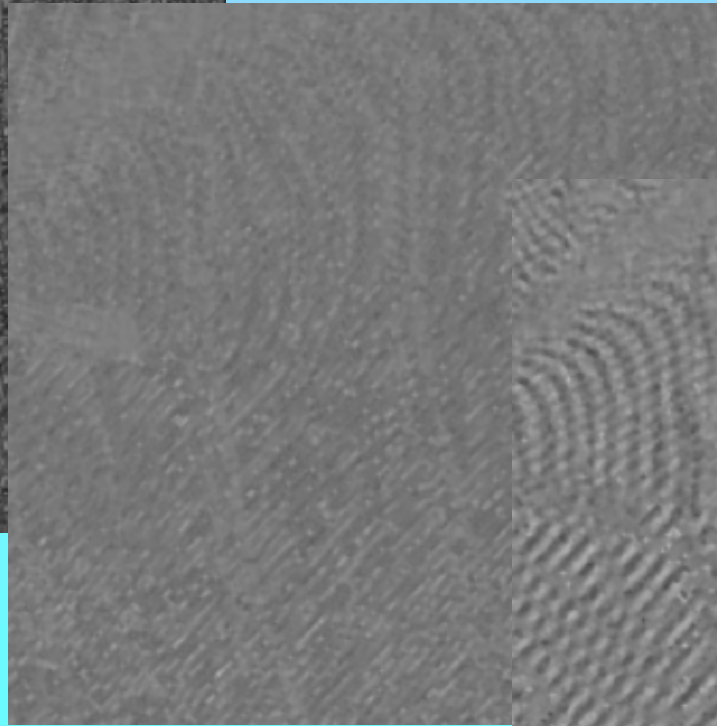
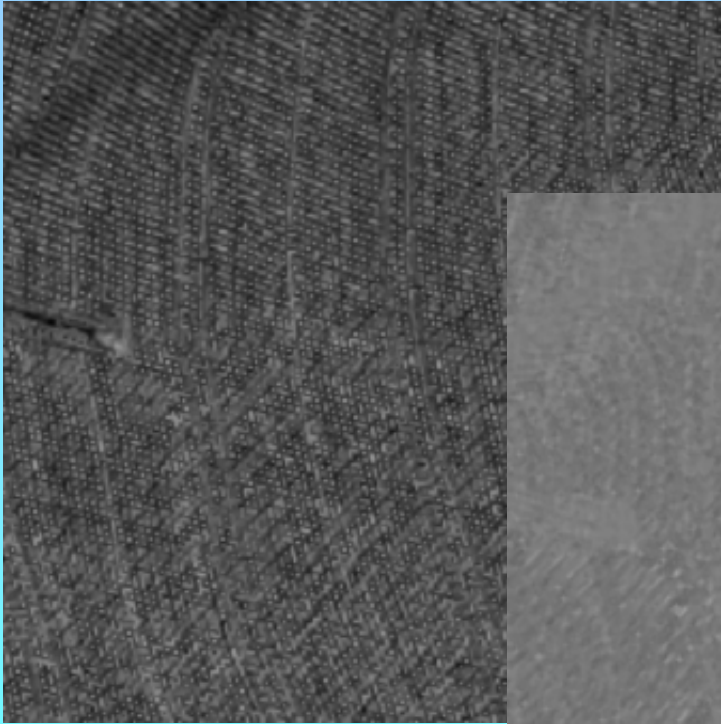
The Basic Idea:

Use (valid) green data to help identify the presence of red and blue aliasing.

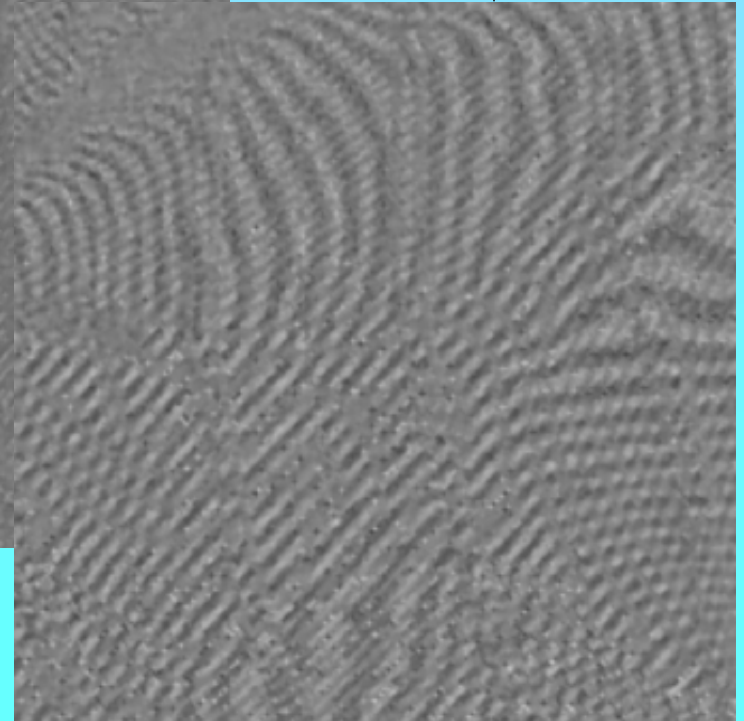
How do we proceed?

- **Convert RGB into a color difference space.**
 - G (luminance)
 - $(2G - R - B)/4$ (chrominance 1)
 - $(B - R)/2$ (chrominance 2)
- **Reduce image size by 3X down-sampling.**
- **Compute a texture image for each channel, indicating the level of low-amplitude, high spatial frequency image content.**

Luminance and Chrominance



**Most of the
aliasing artifact
now resides here.**



Next

- **Combine texture images into one.**
- **Partition image into three types of regions**
 - **lower texture** (likely to be noise)
 - **moderate texture** (likely to hold aliasing)
 - **higher texture** (likely to be scene detail)
- **Focus on regions of moderate texture**
 - **gain coherence with morphological operations**
 - **call these regions “selected”**

Finally

- **Noise clean the two chrominance channels**
 - aggressive cleaning within selected regions
 - normal cleaning within non-selected regions
- **Up-sample cleaned chrominances (3X) to original size using bilinear interpolation**
- **Cleaned chrominances are combined with the original (untouched) luminance and converted back to an RGB image.**

The Results



Before



After

Consequences

- **Our previous aliasing reduction algorithm was only viable as a post-processing step.**
- **The current algorithm was an order of magnitude faster than the first one**
 - **region-based rather than edge-based**
 - **also recall the 3X reduction in size**
- **Thus, the current algorithm became viable as an in-camera processing step.**