EECE 4353 Image Processing

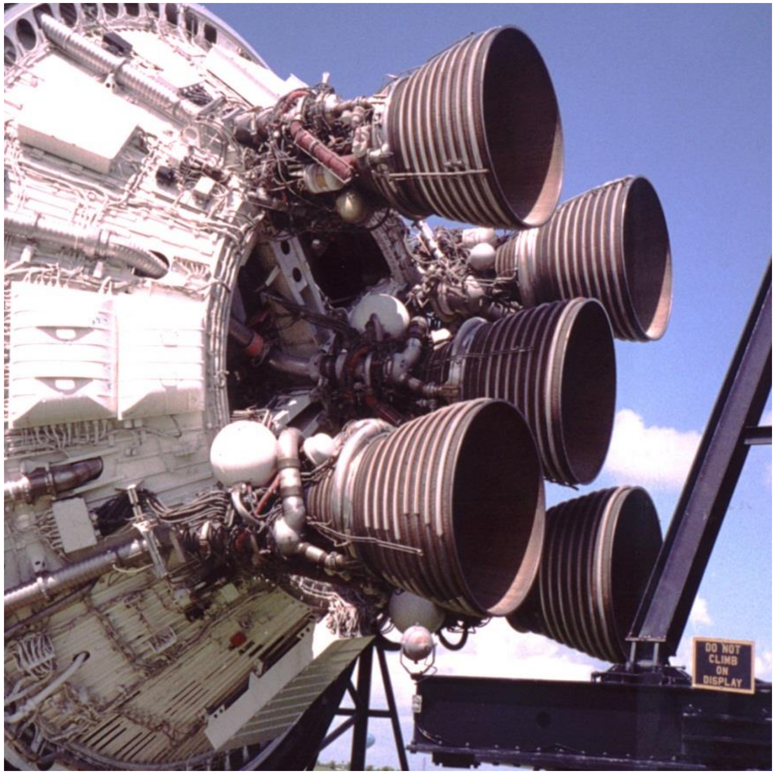
Lecture Notes: Reduction of Correlated Noise

Richard Alan Peters II

Department of Electrical Engineering and
Computer Science

Fall Semester 2016

Periodic Noise



original image

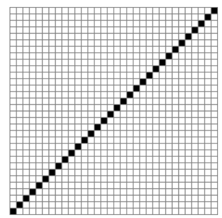
image + noise

Pratt & Whitney Rocketdyne J-2 rocket engines on Apollo 18's Saturn V second stage.

Noise Reduction through Directional Blurring

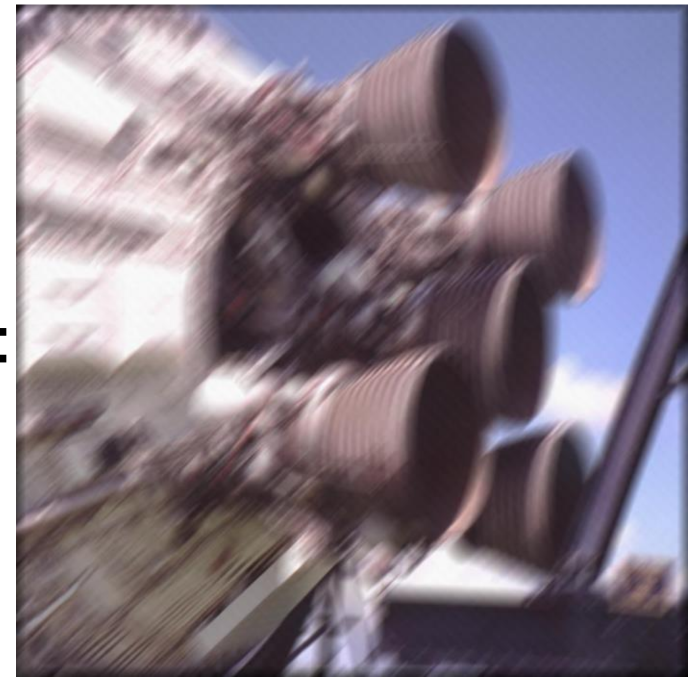
image + noise

*



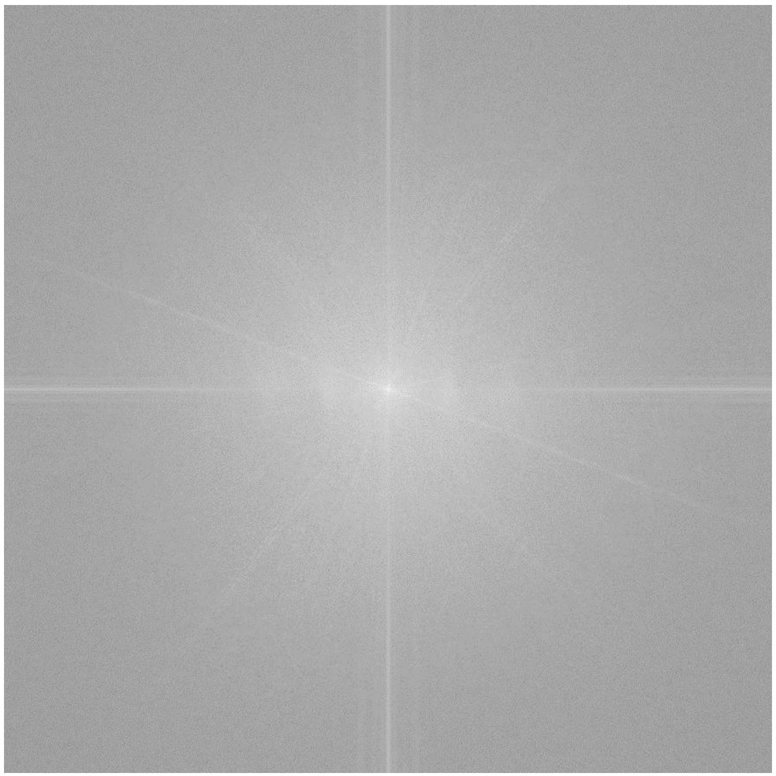
diagonal convolution mask

=



blurred image

Power Spectrum of Image with Periodic Noise



original image

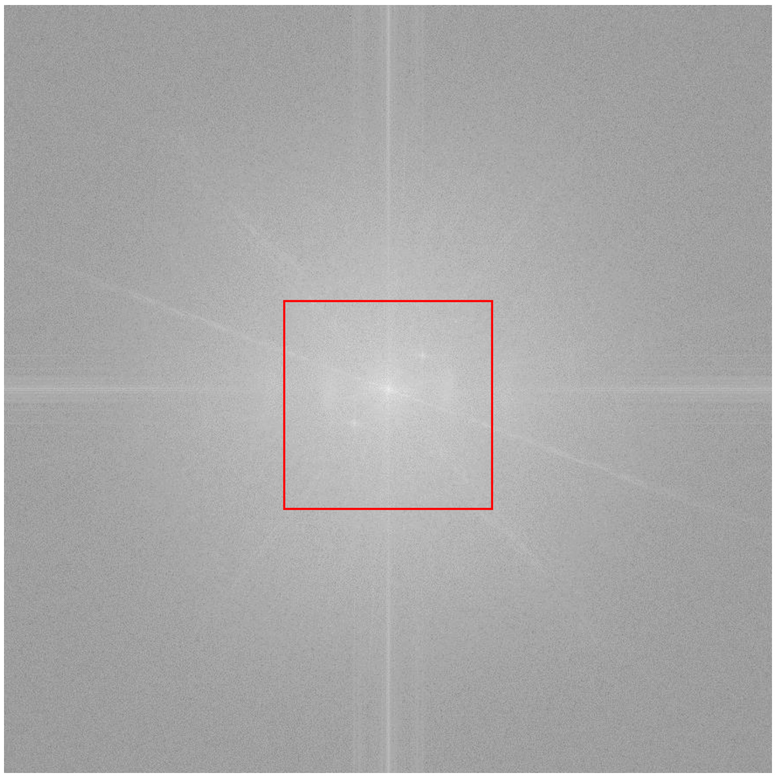
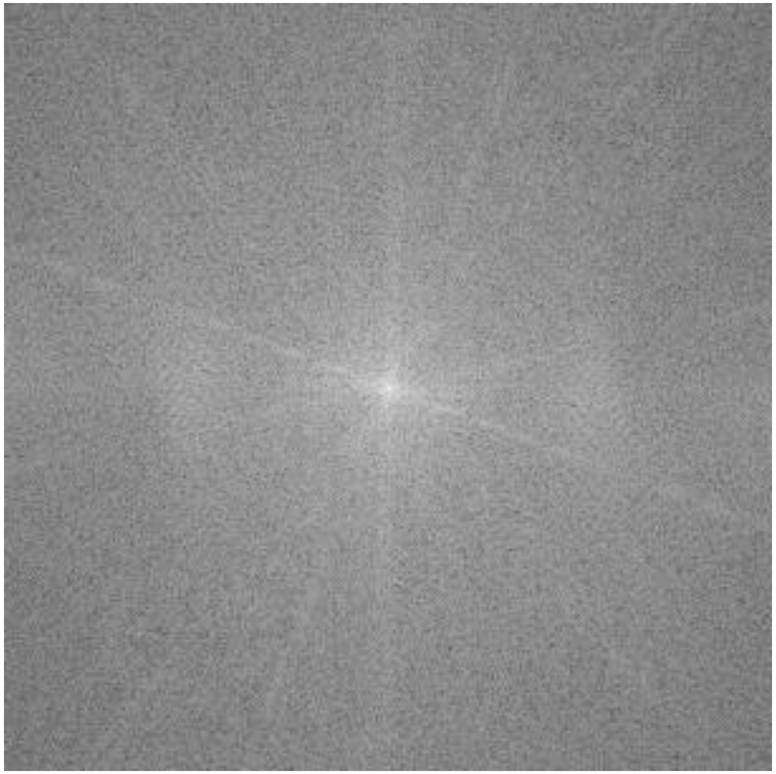


image + noise

Low Frequency Region



original image

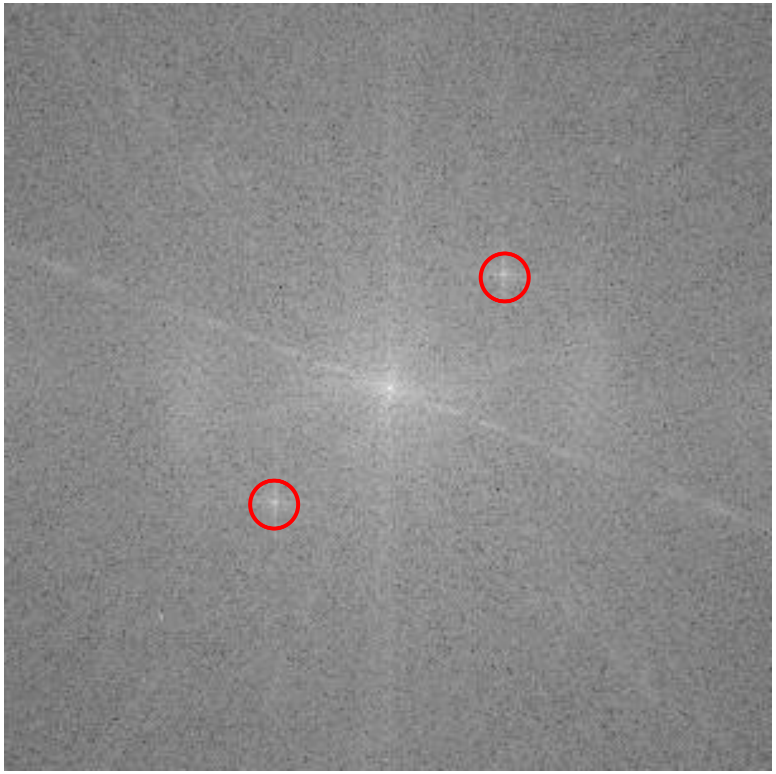
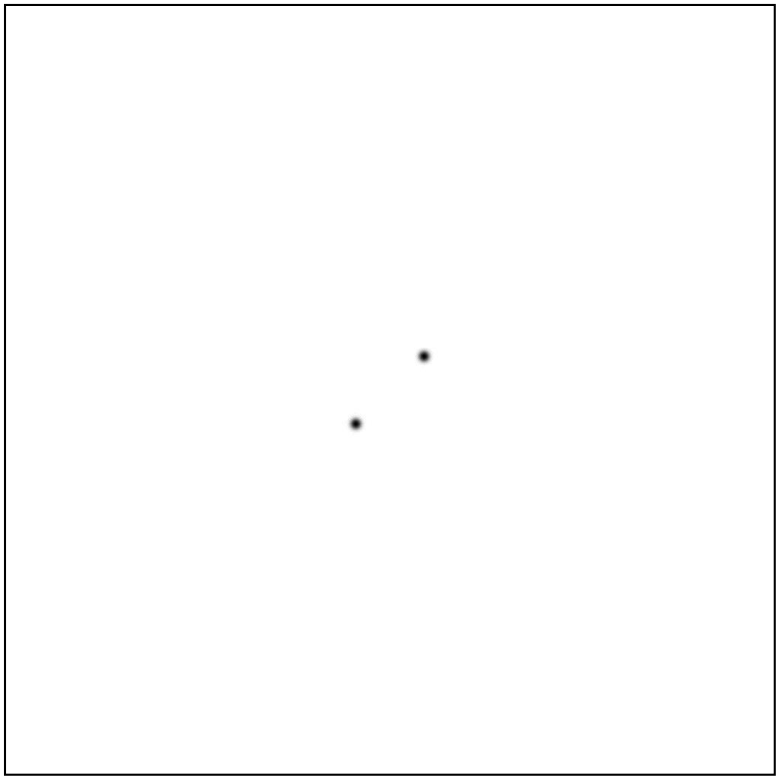
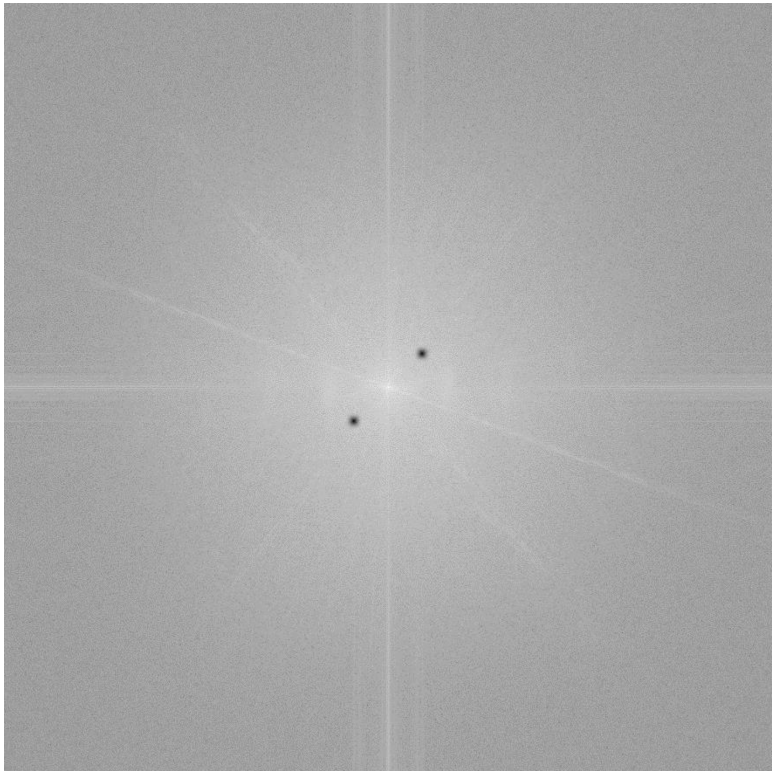


image + noise

Noise Reduction through Notch Filtering

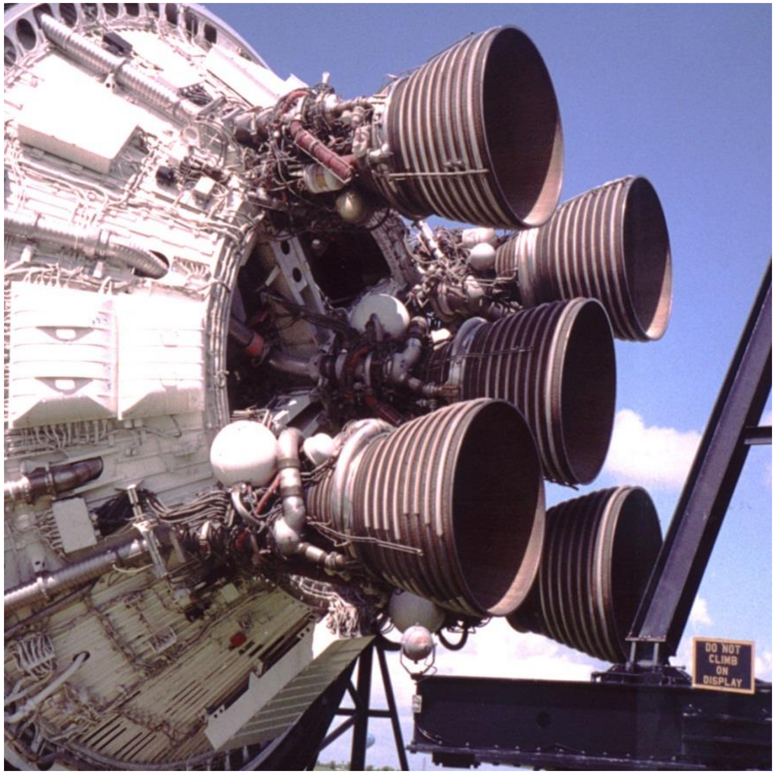


noise mask



masked power spectrum

Inverse of Masked Fourier Transform



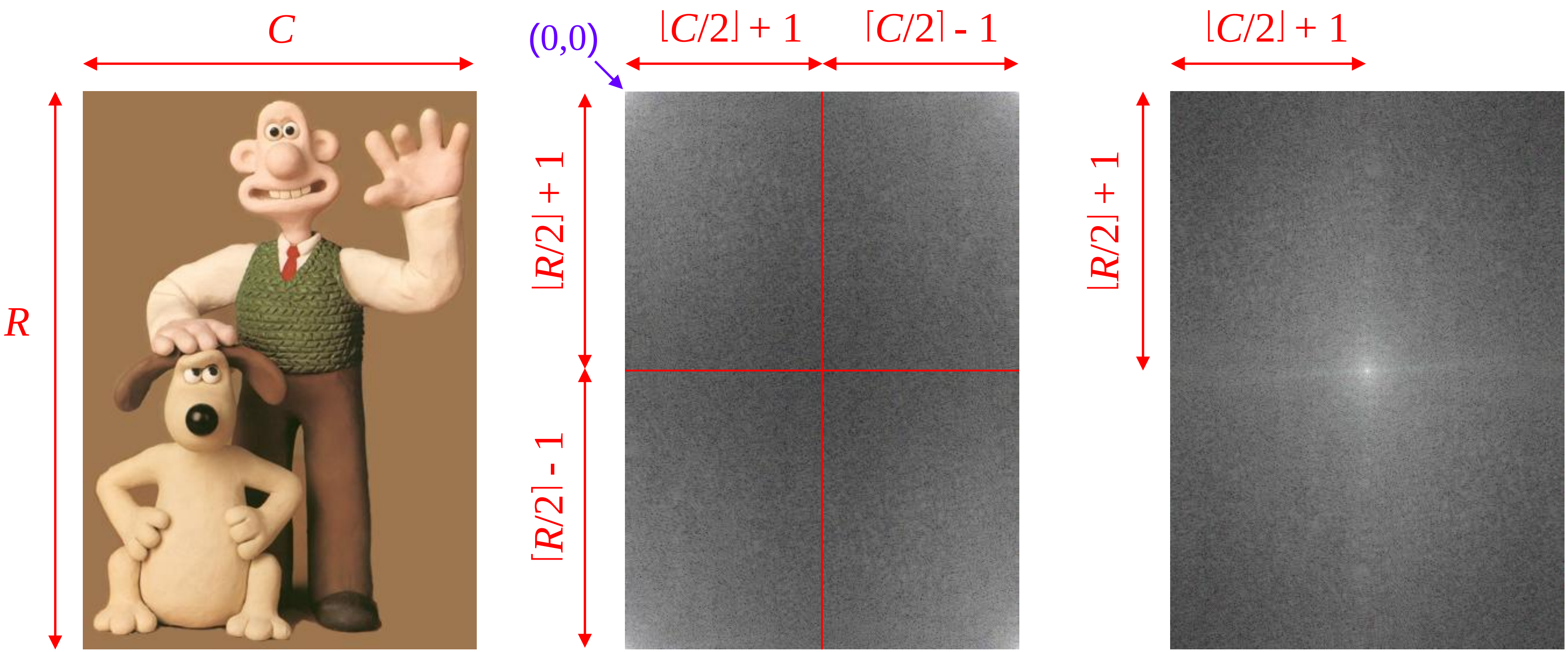
original image

noise reduced image

Notch filter reduction of periodic noise

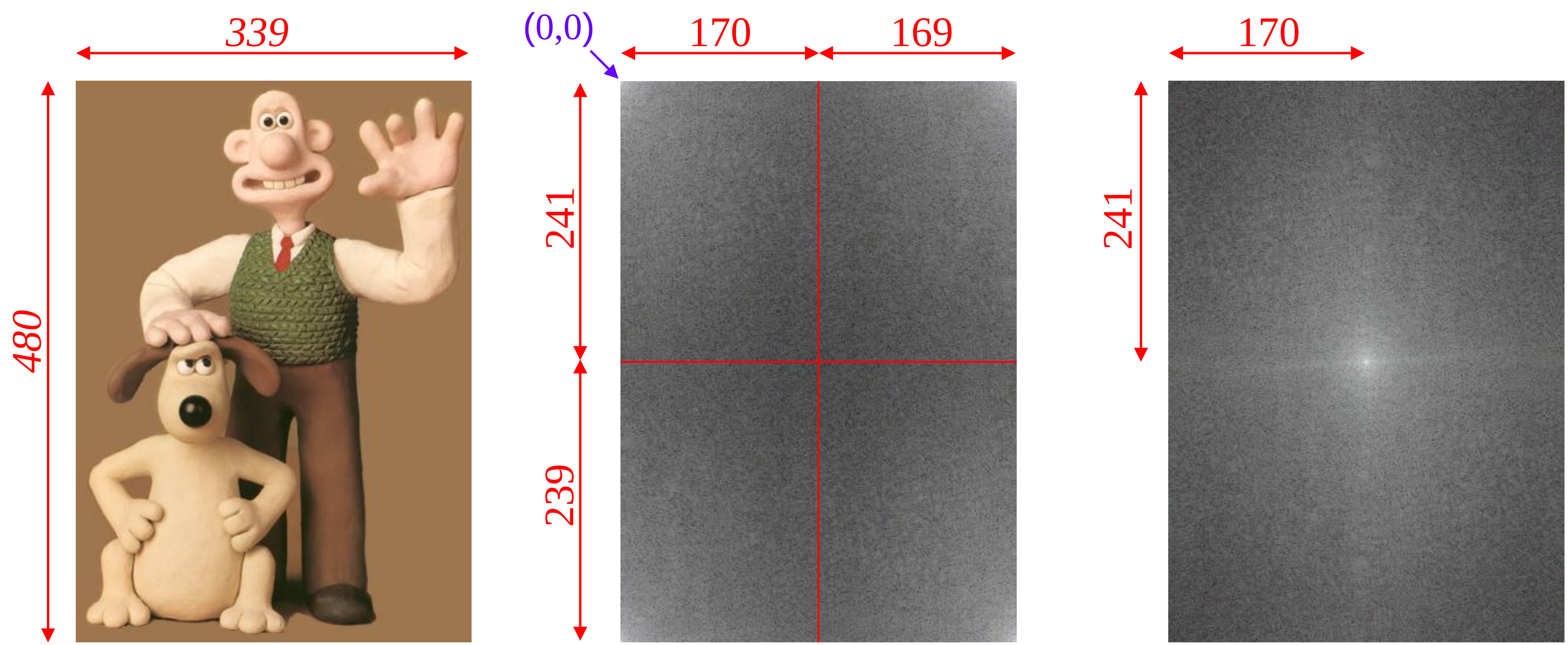
1. Take the FFT of the input image. **F** = `fft2(`**I**`)`.
2. Display the log power spectrum of the `fftshift`-ed **F**.
3. Find the locations, $\mathbf{x}_i$, of the spikes that correspond to the periodic distortion.
4. Create a 1-band image, **M**, of class `double` the same size as **I.**
5. Set all **M**'s pixels to 1.0.
6. Let $\mathfrak{N}(\mathbf{x}_i)$ be a neighborhood of $\mathbf{x}_i$ with area sufficient to cover the spike at $\mathbf{x}_i$.
7. For each $\mathbf{x}_i$ do: for each $\mathbf{y}_j \in \mathfrak{N}(\mathbf{x}_i)$, set $\mathbf{M}(\mathbf{y}_j) = 0$.
8. Blur **M** with a Gaussian whose σ is smaller than ½ the radius of $\mathfrak{N}(\mathbf{x}_i)$.
9. Take the `ifftshift` of **M**.
10. For each band, k, of the image let $\mathbf{G}_k = \mathbf{F}_k .{*}\mathbf{M}$.
11. Then the noise-reduced image is: **J** = `real(ifft2(`**G**`))`.

How to determine the frequency and period of a point in the log power spectrum of an image (ex.):



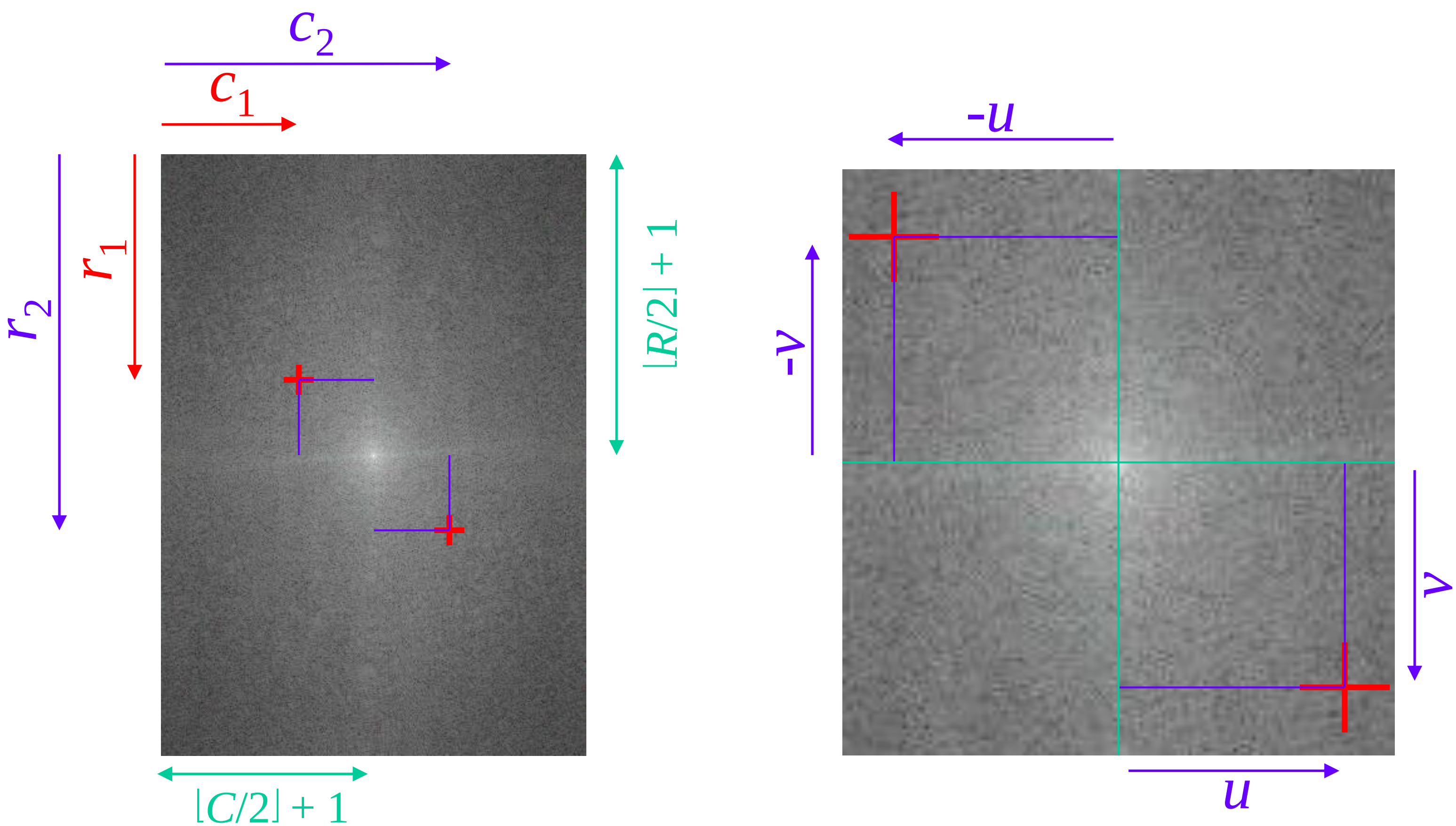
original image — quadrant partition of PS — loc of (0,0) in fftshift(PS)

How to determine the frequency and period of a point in the log power spectrum of an image (ex.):

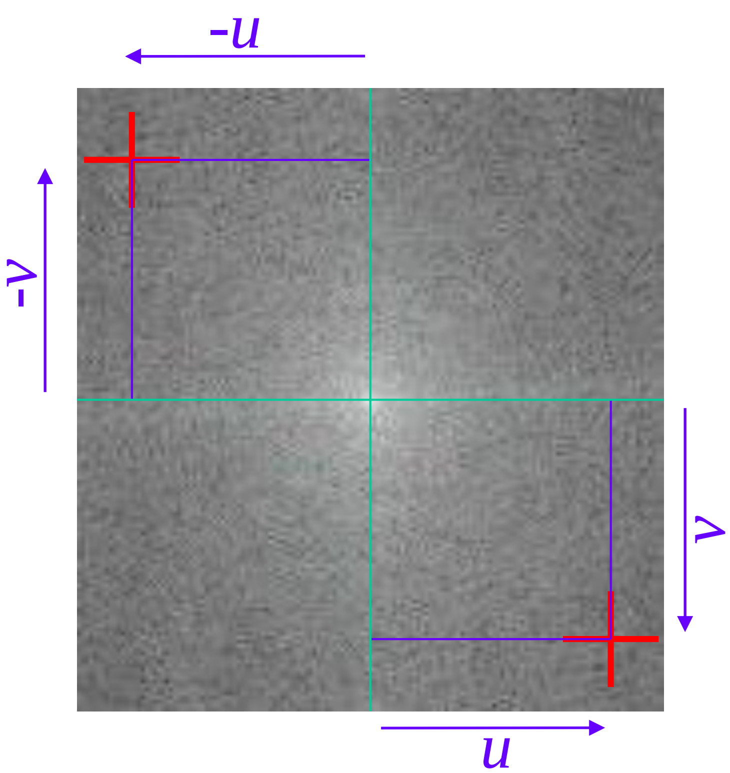


original image | quadrant partition of PS | loc of (0,0) in fftshift(PS)

How to determine the frequency and period of a point in the log power spectrum of an image:



How to determine the frequency and period of a point in the log power spectrum of an image:



$$v = r_2 - \lfloor R/2 \rfloor - 1$$
$$-v = \lfloor R/2 \rfloor + 1 - r_1$$
$$u = c_2 - \lfloor C/2 \rfloor - 1$$
$$-u = \lfloor C/2 \rfloor + 1 - c_1$$

$$\lambda_{\text{wf}} = \sqrt{\left(\tfrac{C}{u}\right)^2 + \left(\tfrac{R}{v}\right)^2}$$
$$\omega_{\text{wf}} = \frac{1}{\lambda_{\text{wf}}}$$
$$\theta_{\text{wf}} = \tan^{-1}\left(\frac{vC}{uR}\right)$$

Points on the Fourier Plane (of a Digital Image)

In the Fourier transform of an $R \times C$ digital image the wavelengths, λ_u and λ_v represent a fraction of the R and C values. That is,

$$\lambda_u = \frac{C}{u} \quad \text{and} \quad \lambda_v = \frac{R}{v} \quad \text{pixels.}$$

The wavefront direction is given by

$$\theta_{\mathrm{wf}} = \tan^{-1}\left(\frac{vC}{uR}\right),$$

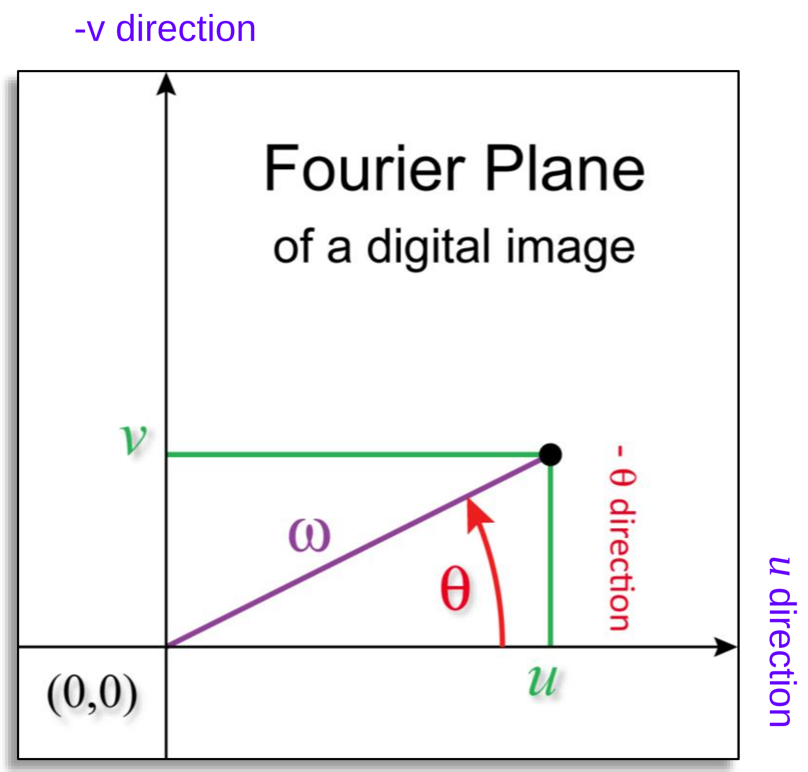
and the wavelength is

$$\lambda_{\mathrm{wf}} = \sqrt{\left(\frac{C}{u}\right)^2 + \left(\frac{R}{v}\right)^2}.$$

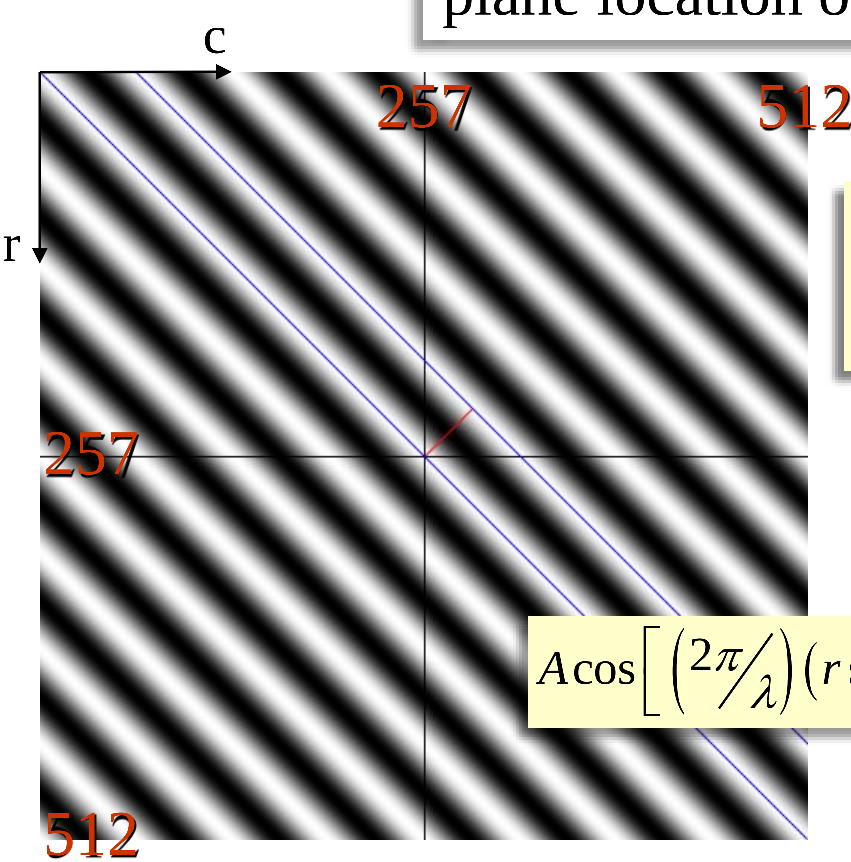
The frequencies represent fractions of R & C,

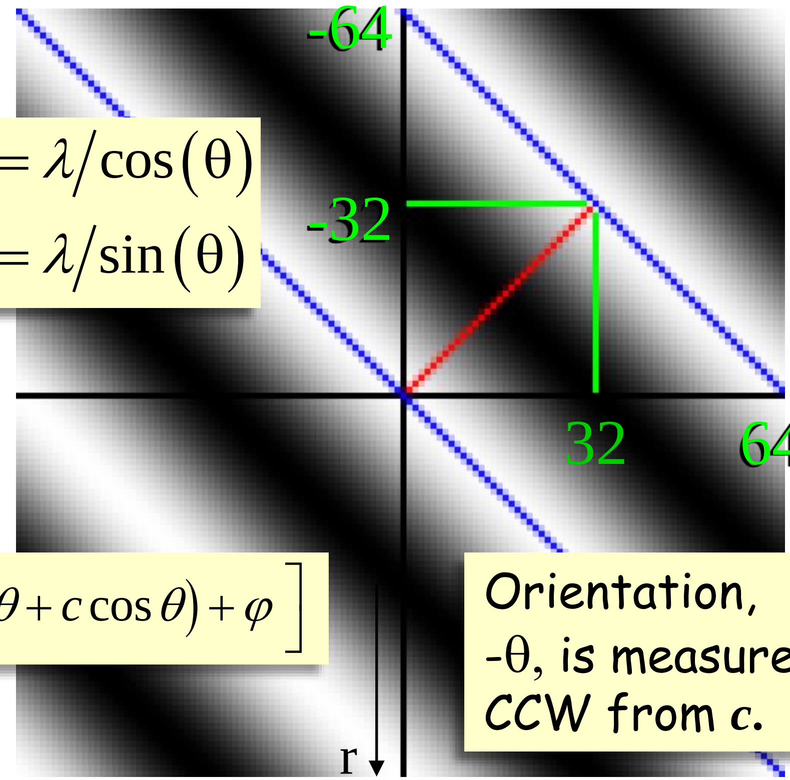
$$\omega_u = \frac{u}{C}, \quad \omega_v = \frac{v}{R}, \text{ and}$$

$$\omega_{\mathrm{wf}} = 1 \Bigg/ \sqrt{\left(\frac{C}{u}\right)^2 + \left(\frac{R}{v}\right)^2} \quad \text{cycles.}$$



How to determine the frequency plane location of a sinusoid:





$$\lambda_r = \lambda / \cos(\theta)$$

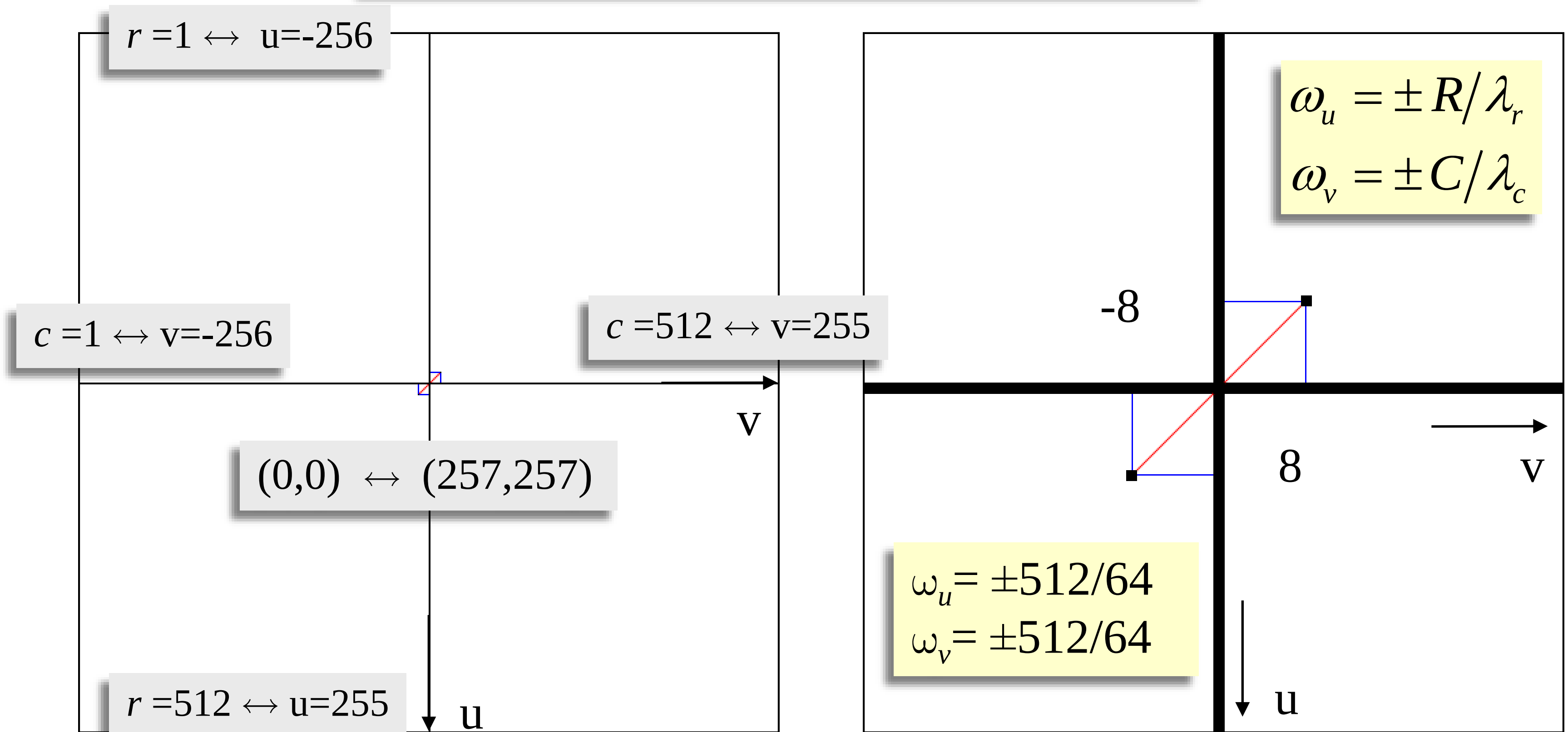
$$\lambda_c = \lambda / \sin(\theta)$$

$$A\cos\left[\left(2\pi/\lambda\right)\left(r\sin\theta + c\cos\theta\right) + \varphi\right]$$

Orientation, $-\theta$, is measured CCW from *c*.

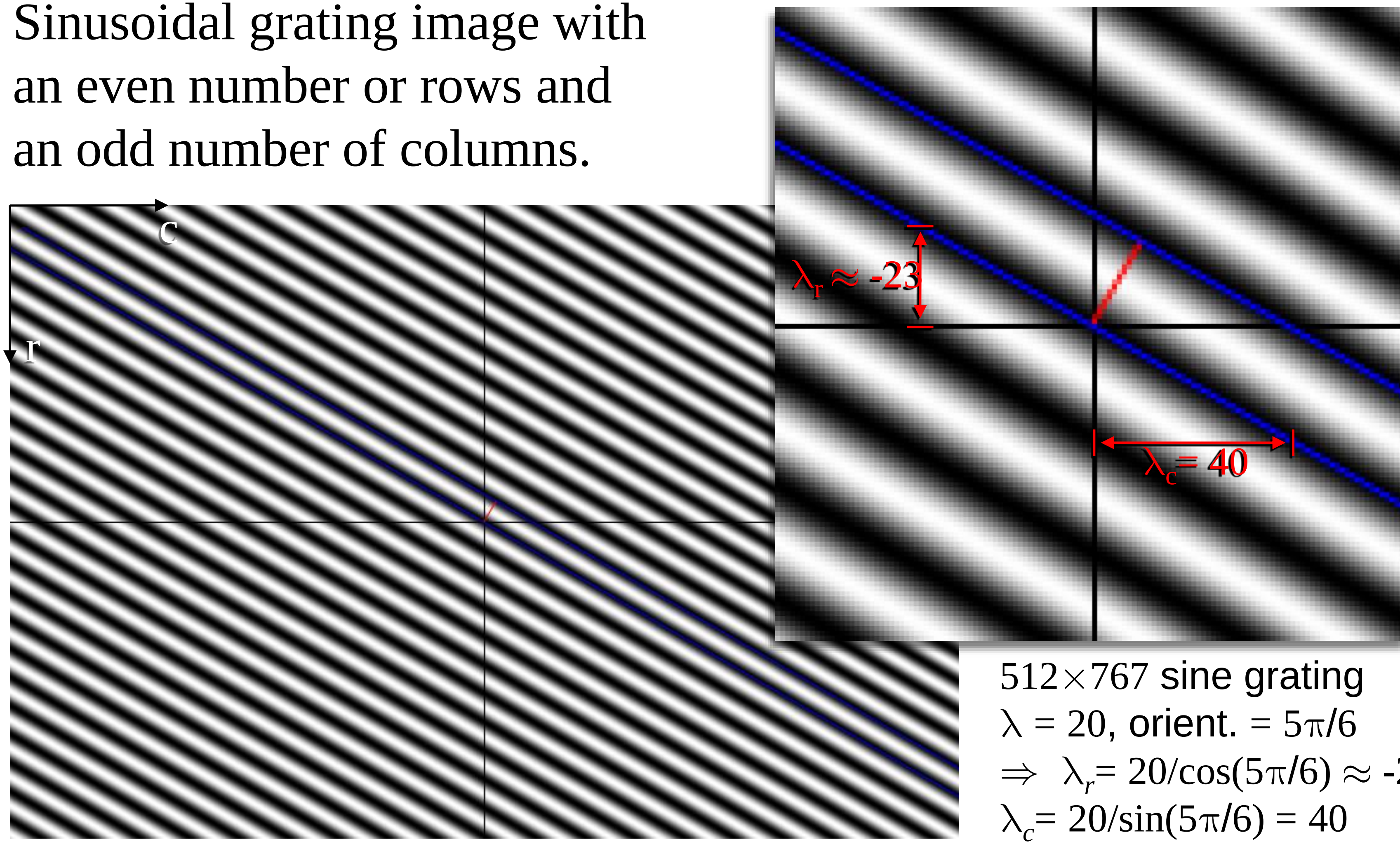
cosine grating $\lambda = 32\sqrt{2}$, $\theta = -\pi/4 \Rightarrow \lambda_r = 64 = 512/v$, $\lambda_c = 64 = 512/v$

How to determine the frequency plane location of a sinusoid:



cosine grating $\omega = 8\sqrt{2}$, orientation = $3\pi/4$ $\Rightarrow$ $\omega_u = \pm 8$, $\omega_v = \pm 8$

Sinusoidal grating image with an even number or rows and an odd number of columns.



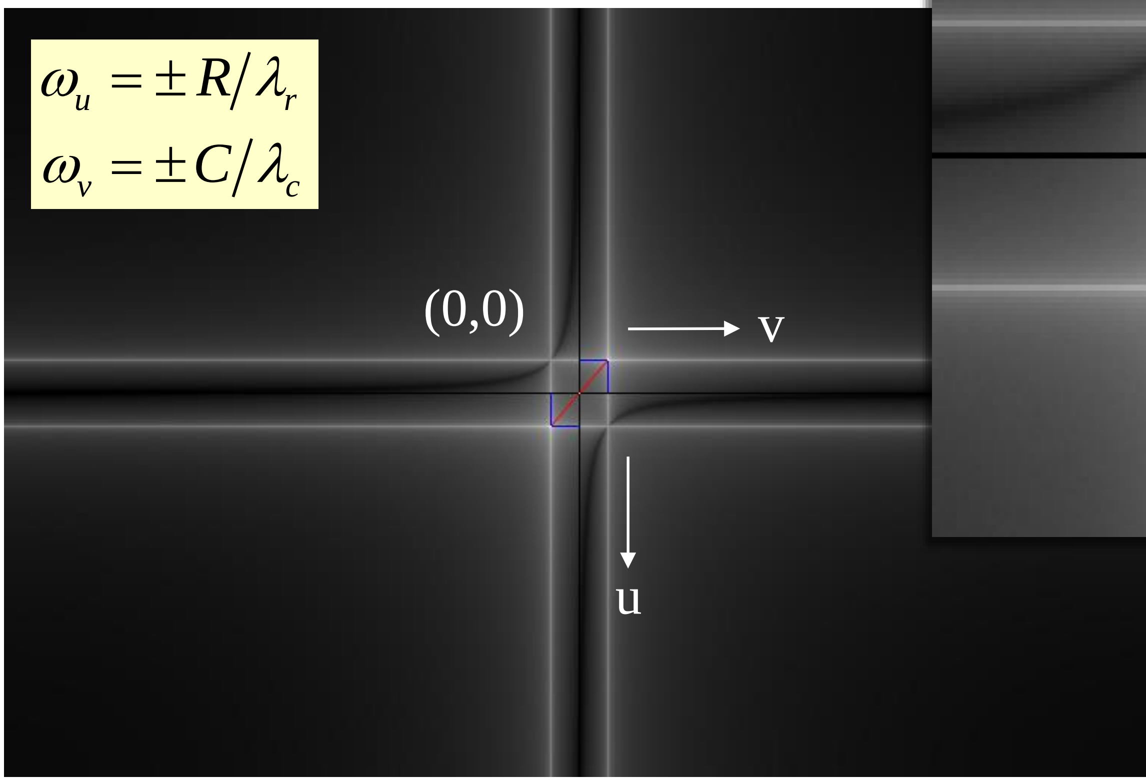
512×767 sine grating

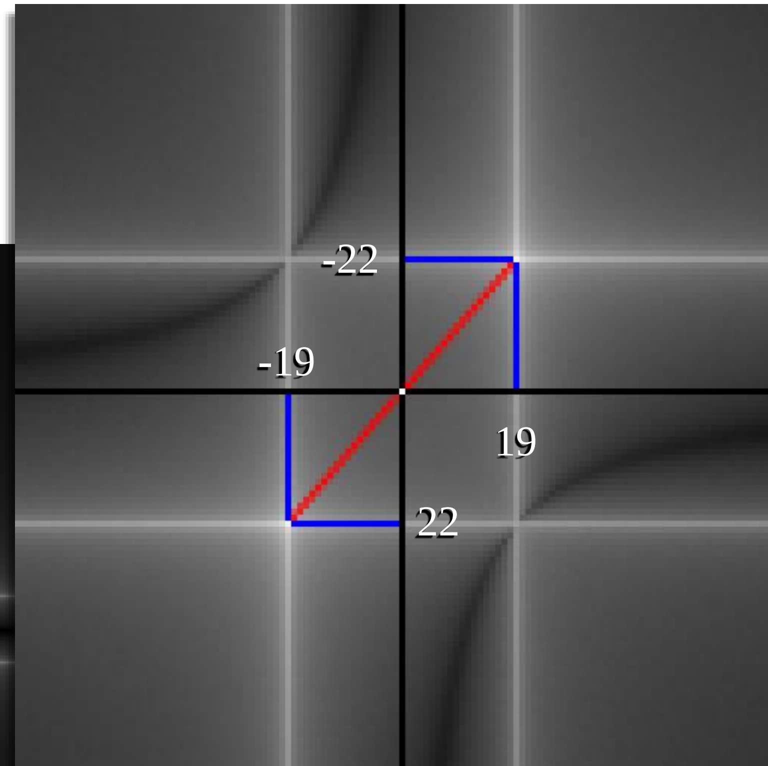
$\lambda = 20$, orient. $= 5\pi/6$

$\Rightarrow \; \lambda_r = 20/\cos(5\pi/6) \approx -23$,

$\lambda_c = 20/\sin(5\pi/6) = 40$

FT of Sinusoidal Grating Image





752×937 sine grating
λ = 20, orient. = $5\pi/6$
$\Rightarrow \omega_u = \pm 512/\lambda_r \approx \pm 22$,
$\omega_v = \pm 767/\lambda_c \approx \pm 19$

Processing Scanned Press-Printed Images

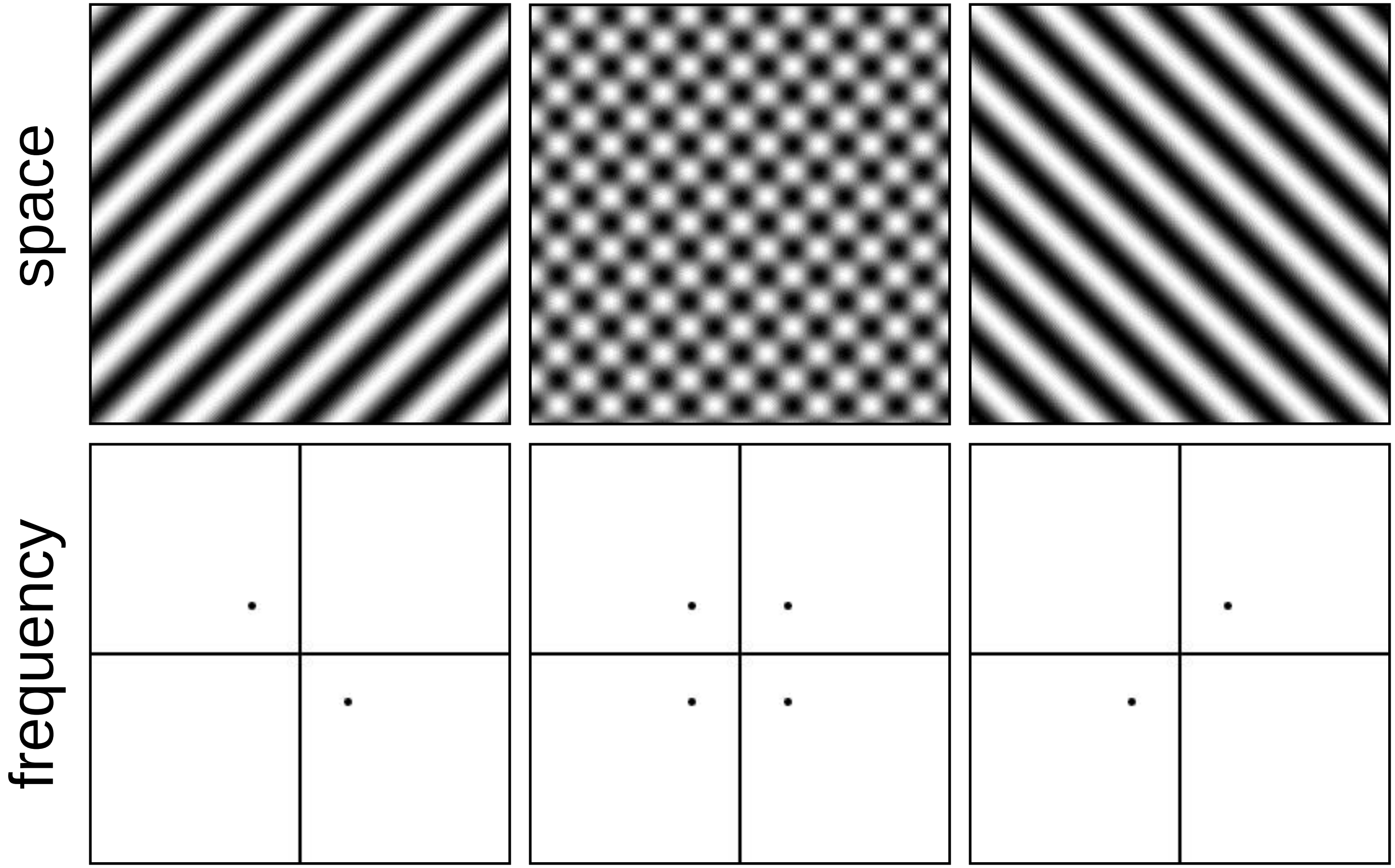
4-color printing:

1. A photograph or other color image is separated into four intensity band images: cyan, magenta, yellow, and black.
2. Each of these is multiplied by a halftone screen[1] – a dot mask with a unique orientation.
3. Each of the resulting four images shows a pattern of dots whose individual sizes indicate the amount of ink to be applied at each point.
4. The four images are printed, one atop the other, in the corresponding color.

[1]Merriam-Webster Dictionary: half·tone 2 (1): a photoengraving made from an image photographed through a screen and then etched so that the details of the image are reproduced in dots.

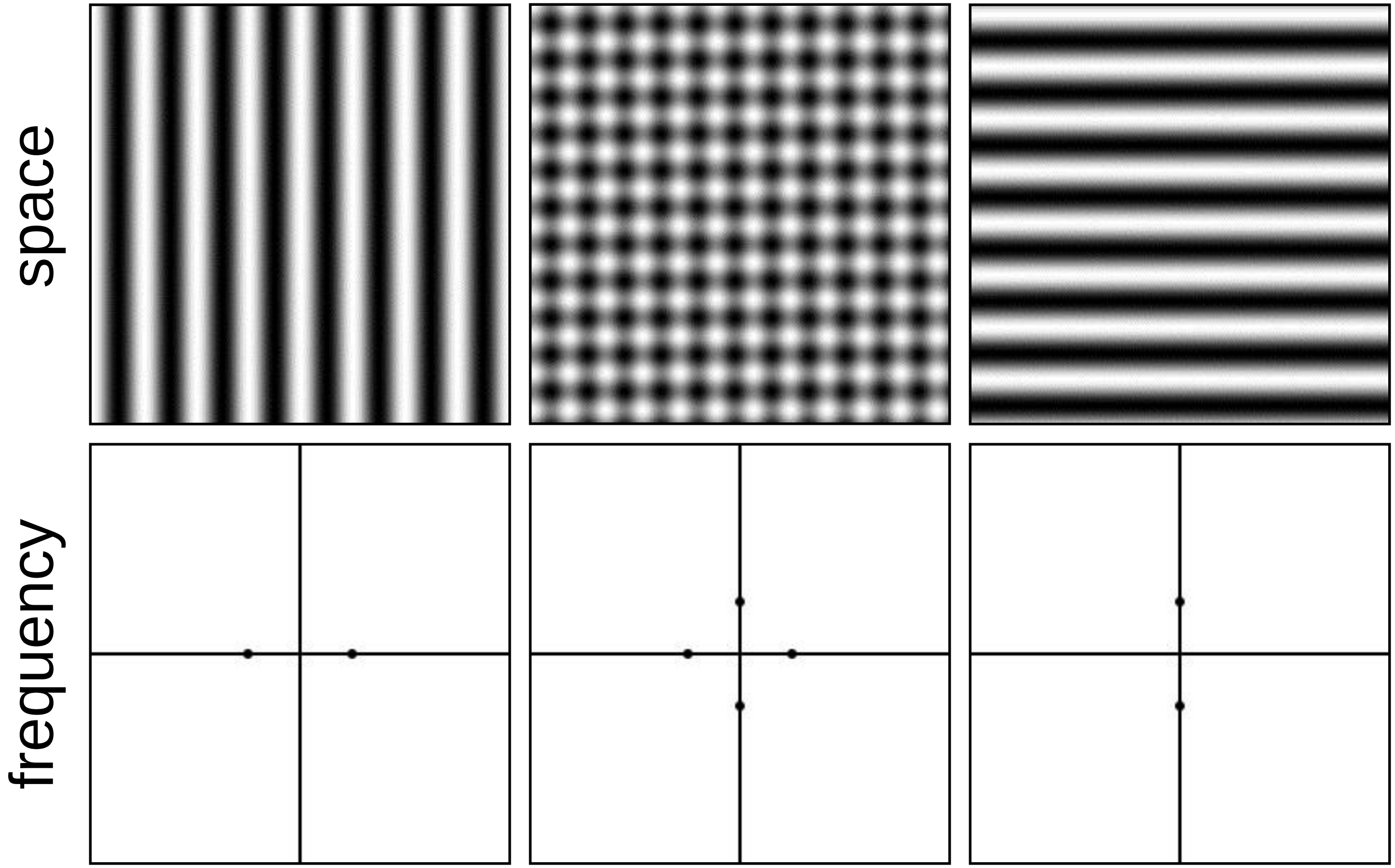
Halftone Screen (45°)

is the pointwise product of 2 sinusoidal gratings with perpendicular orientations.



Halftone Screens (90°)

The orientation of the screen is the average of the grating orientations.



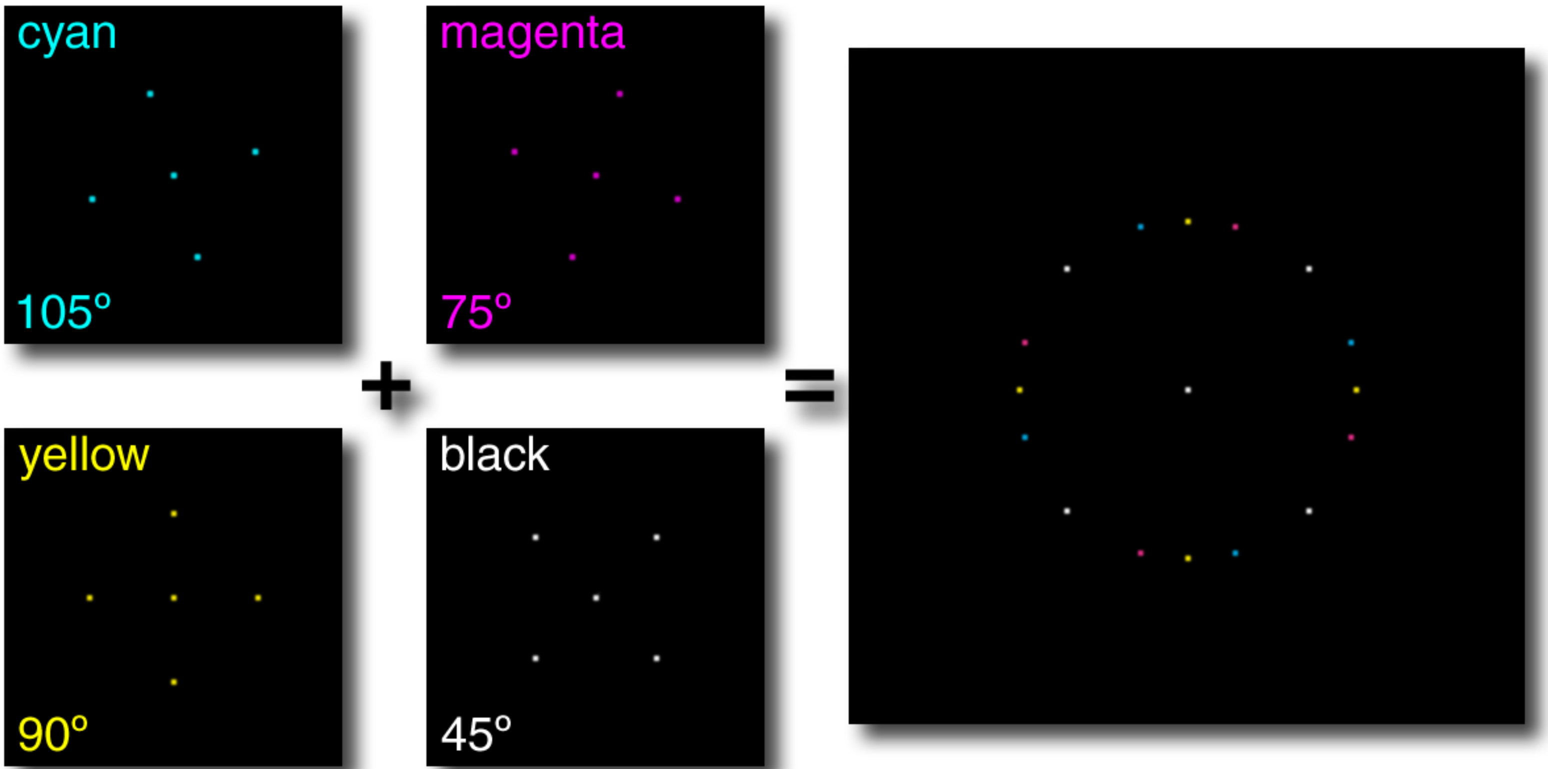
Standard Halftone Screen Angles

Cyan:	105°
Yellow:	90°
Magenta:	75°
Black:	45°

Image from Adobe Photoshop CS2 documentation.

Each band has 2 perpendicular sinusoids + a DC component…

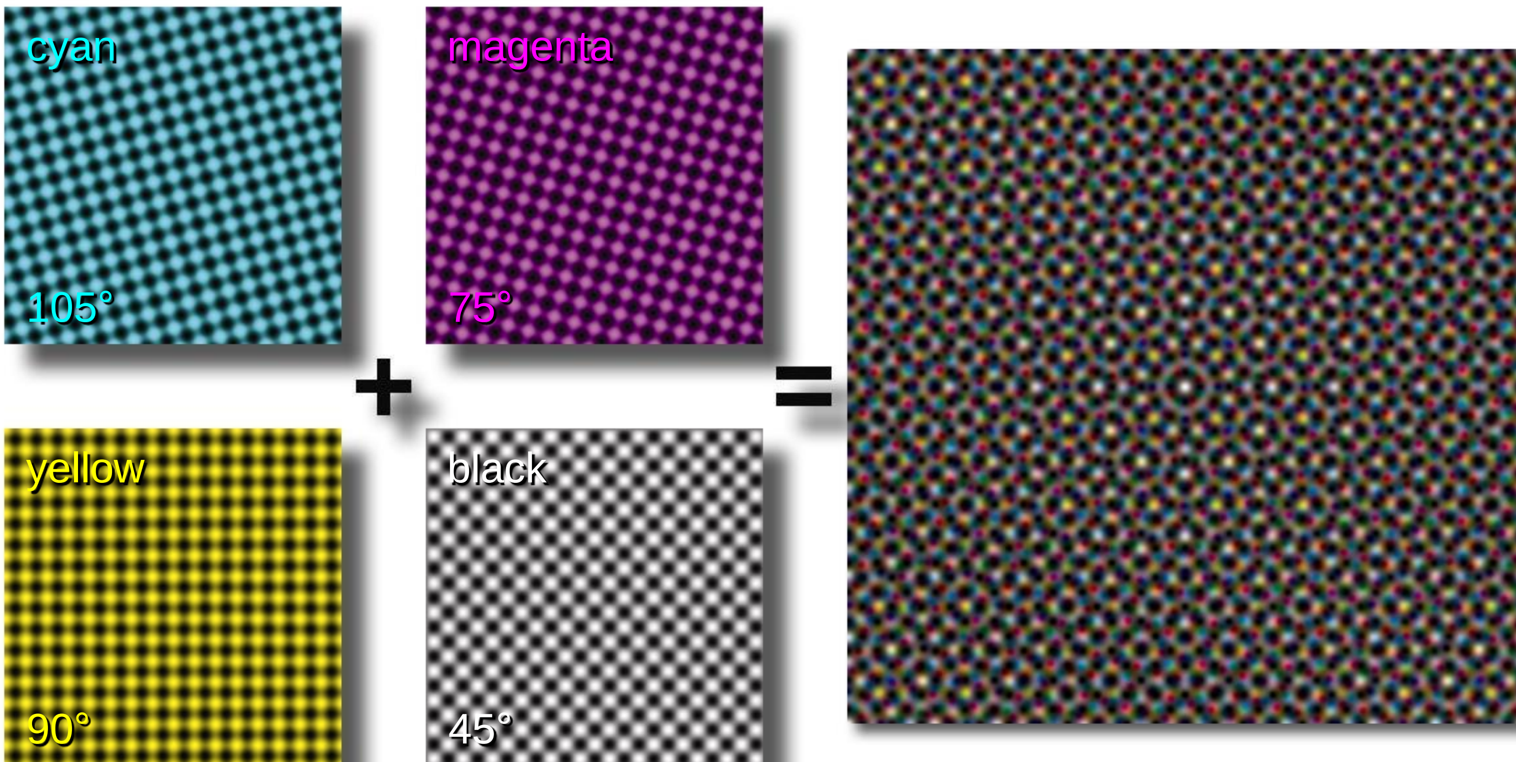
CMYK Standard Halftone Screens



Power Spectra

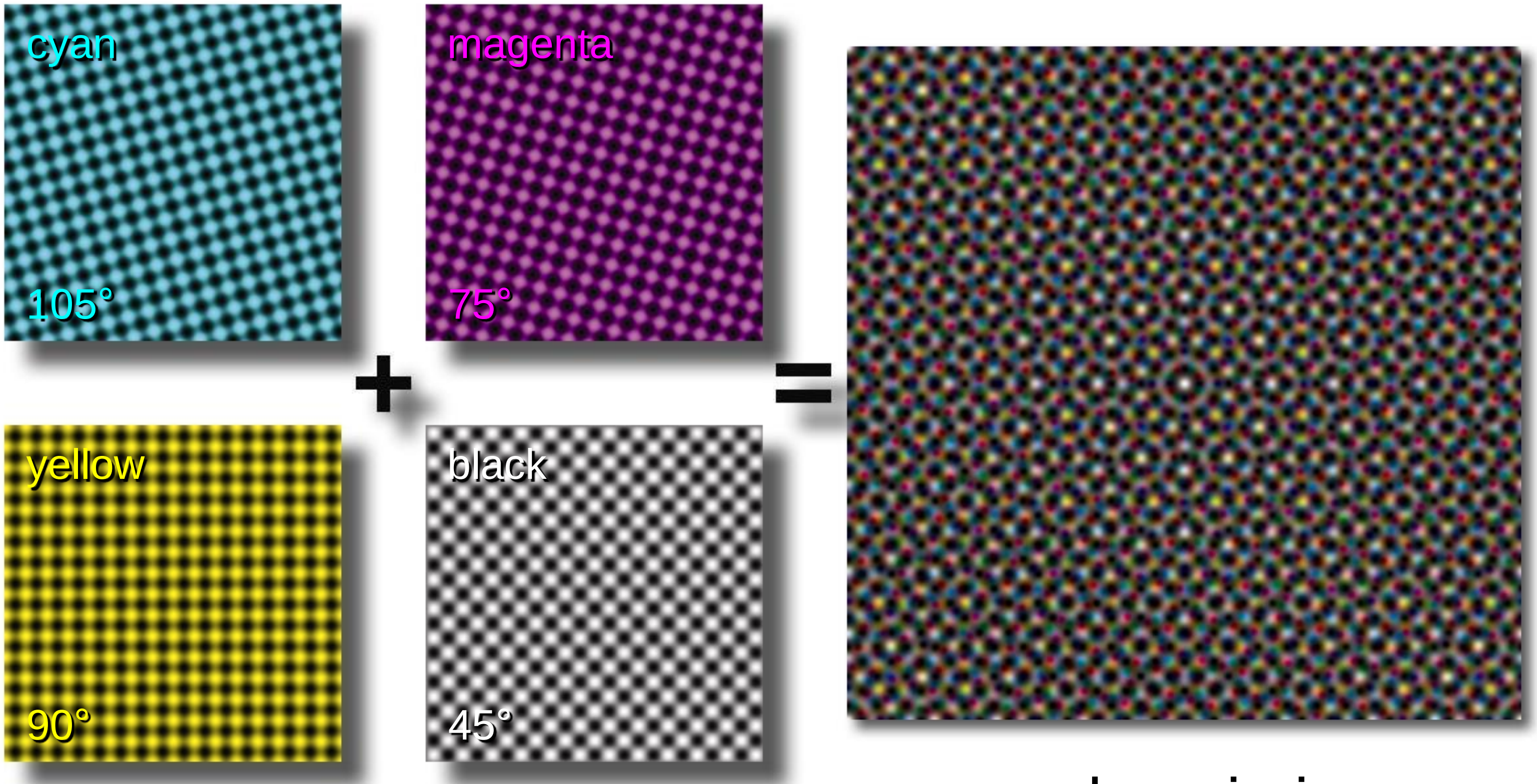
… which creates rectangular grids at 4 different angles.

CMYK Standard Halftone Screens



When the 4 are summed, the result is a “rosette” image.

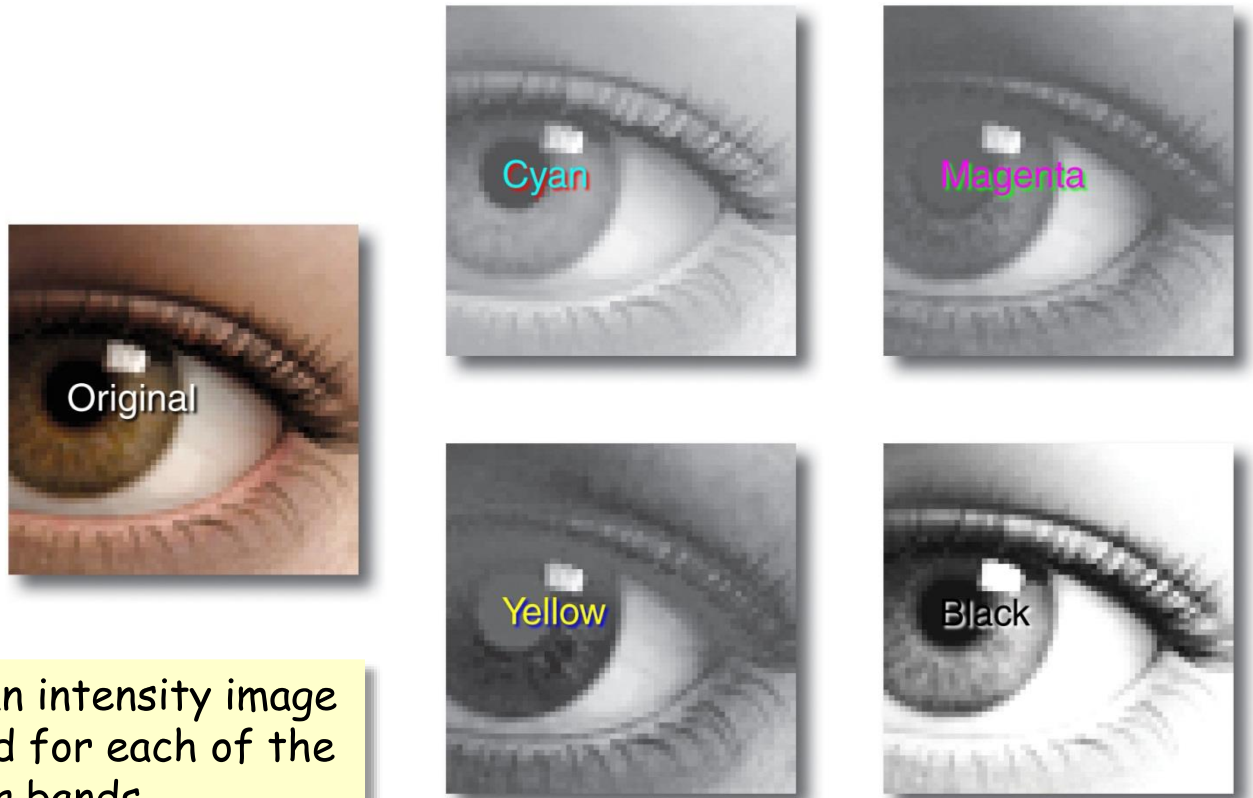
CMYK Standard Halftone Screens



space domain images

To print an image, it is separated into 4 color bands …

Example: Color Separation / Halftoning

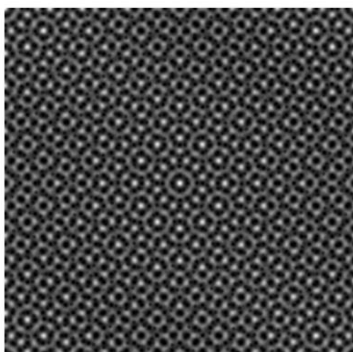


That is, an intensity image is created for each of the four color bands.

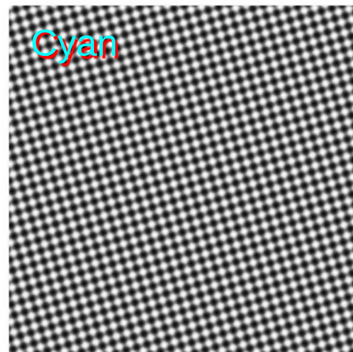
… each of which is multiplied by a corresponding screen.

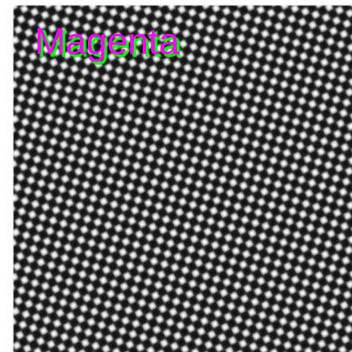
Color Separation / Halftoning

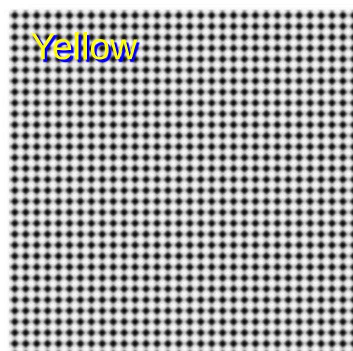
Each intensity image is multiplied by a corresponding screen, then

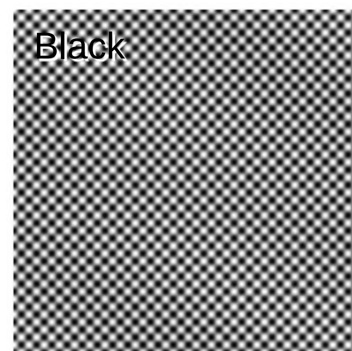


each screened image is printed in its own color on the same page.



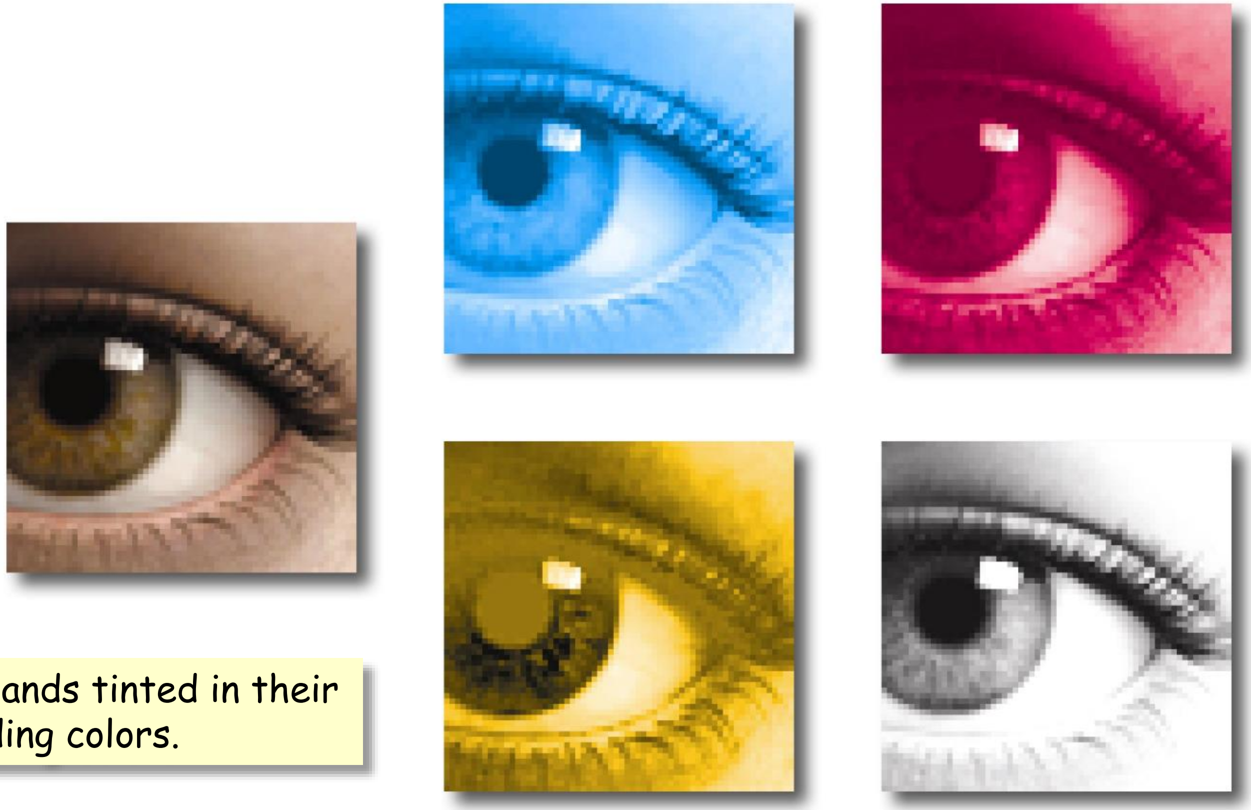






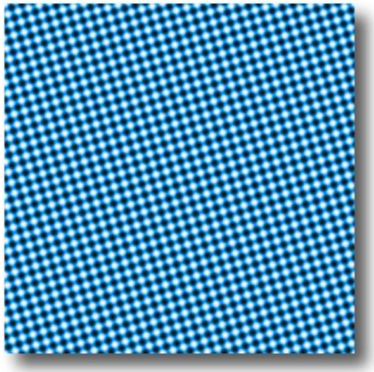
To print an image, it is separated into 4 color bands …

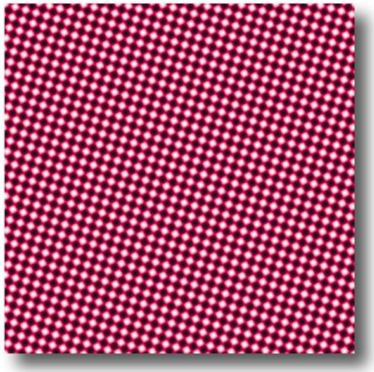
Example: Color Separation / Halftoning

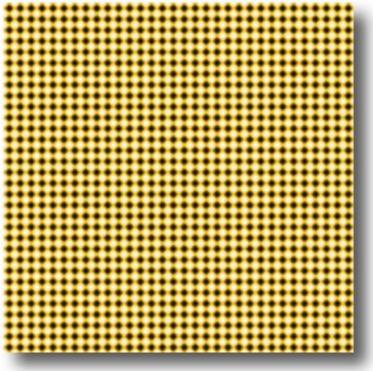


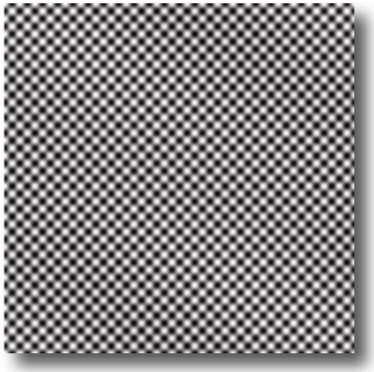
Here the bands tinted in their corresponding colors.

Example: Color Separation / Halftoning









Here the screens tinted in their corresponding colors.

…to get dot patterns for printing. The 4 are printed over each other to get the final result.

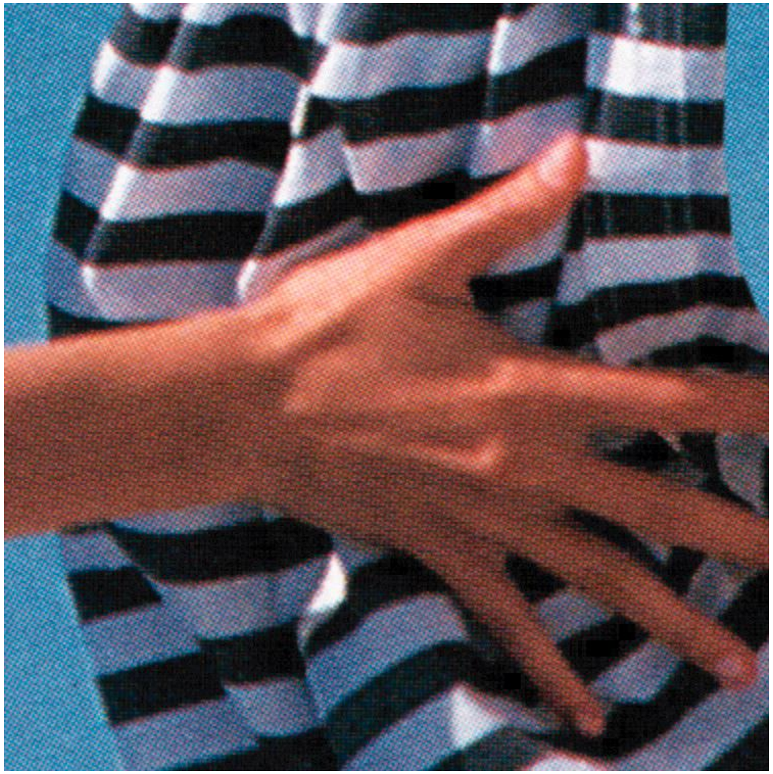
Example: Color Separation / Halftoning

Halftone Dots

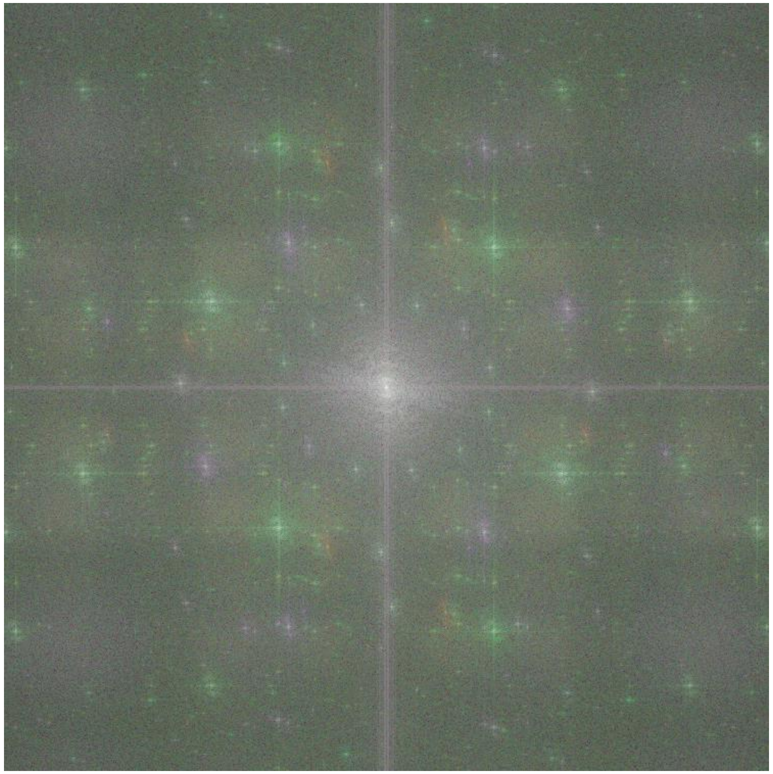
Image scanned (600 dpi) from a magazine

Detail: Circular patterns, the rosettes, are the result of the halftone dots.

Filtering Out Halftone Dot Distortion



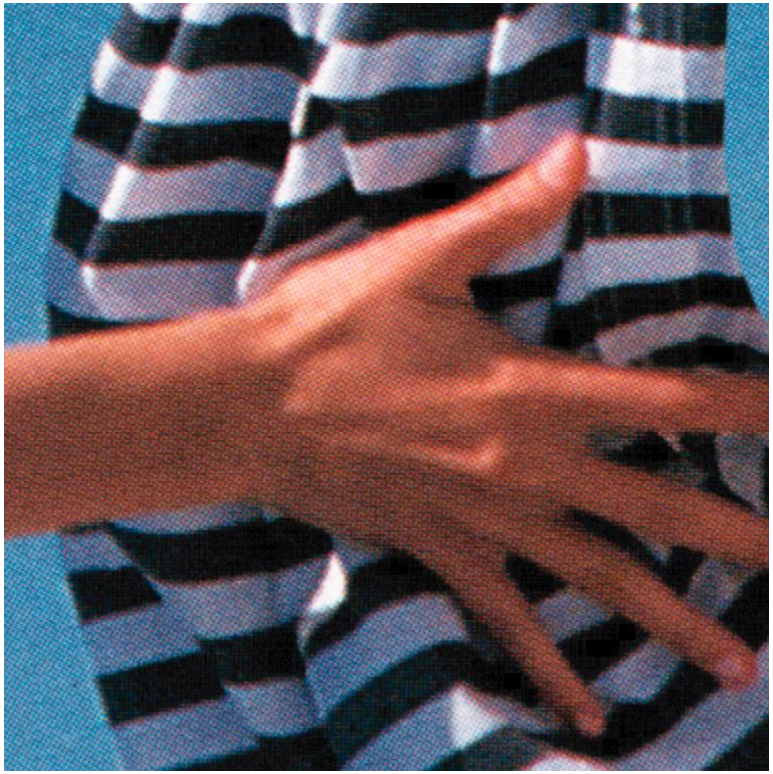
original



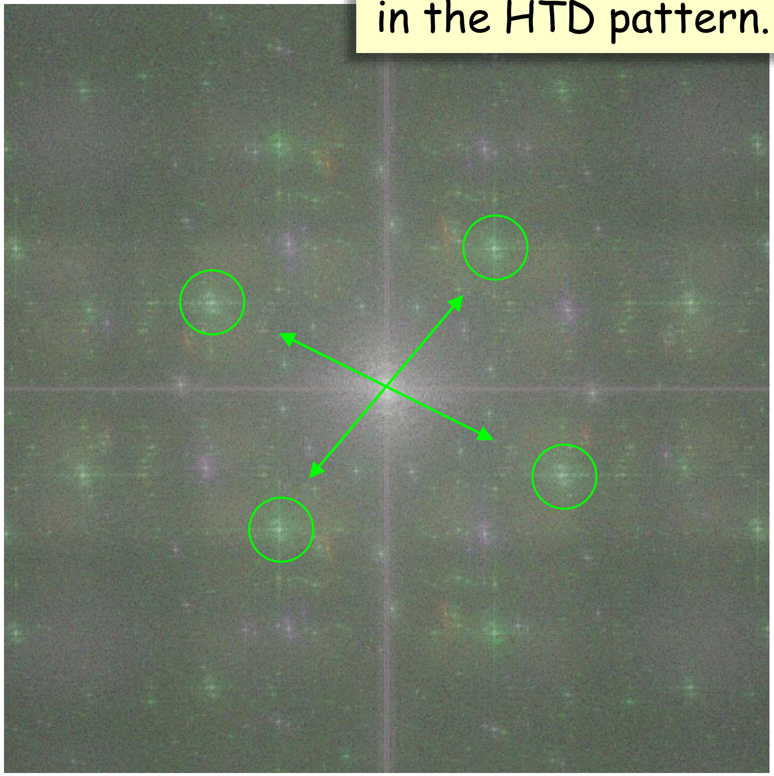
log power spectrum

Filtering Out Halftone Dot D

Each pair of peaks corresponds to a sinusoidal sub-pattern in the HTD pattern.



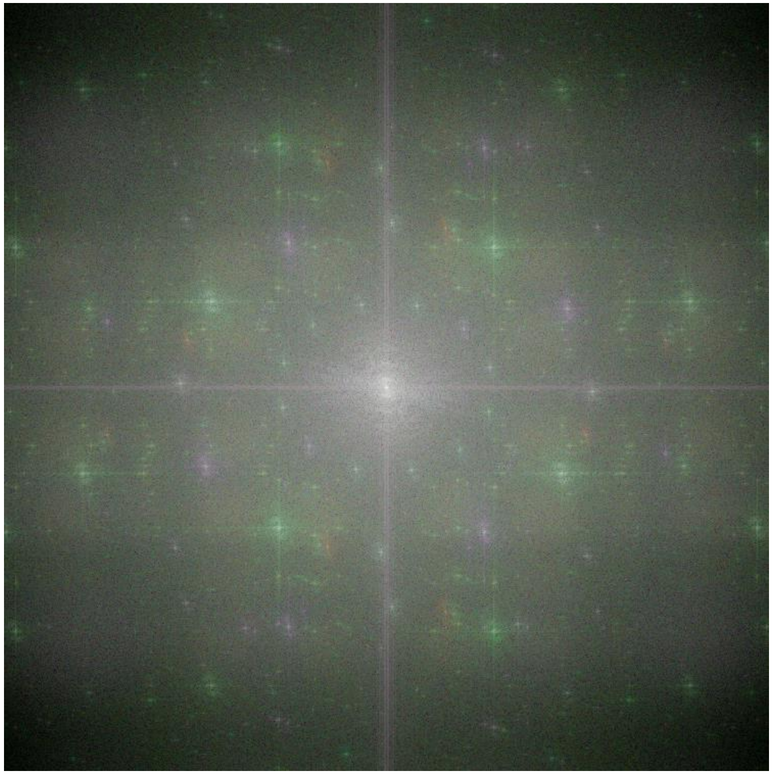
original



log power spectrum

Blurring with a Gaussian (σ = 1)

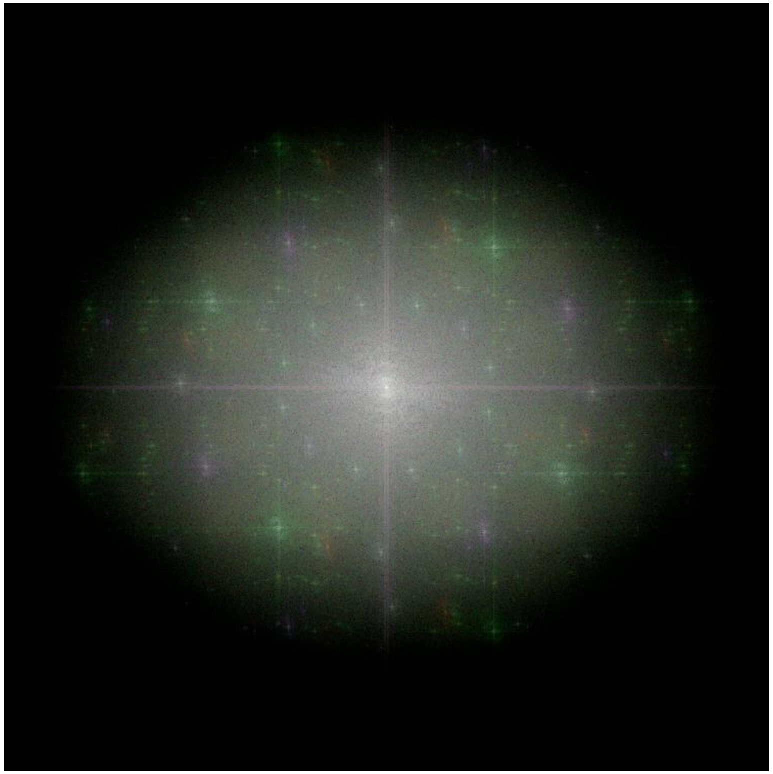
blurred image σ=1



log power spectrum σ=1

Blurring with a Gaussian (σ = 2)

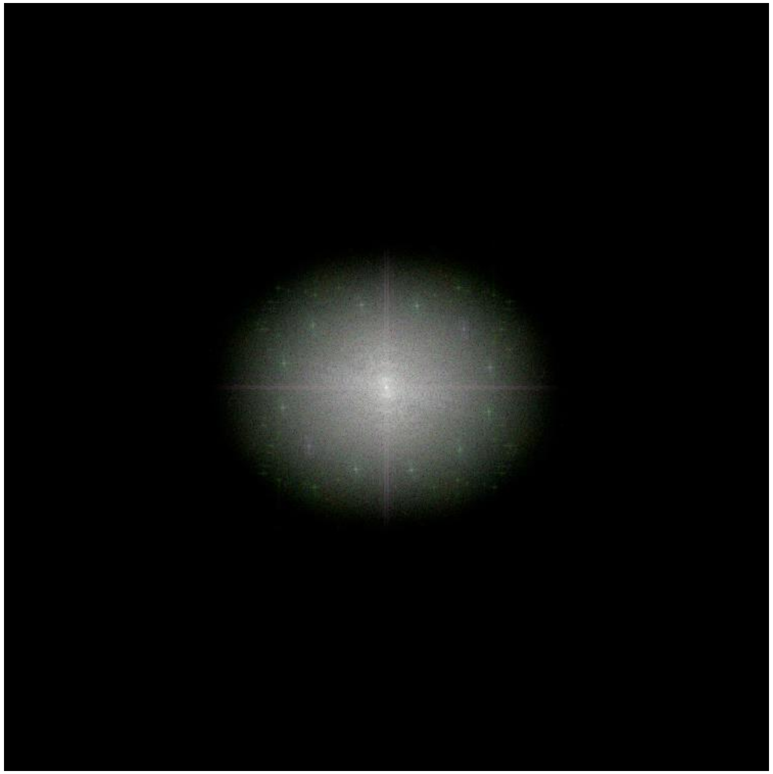
blurred image σ=2



log power spectrum σ=2

Blurring with a Gaussian ($\sigma = 4$)

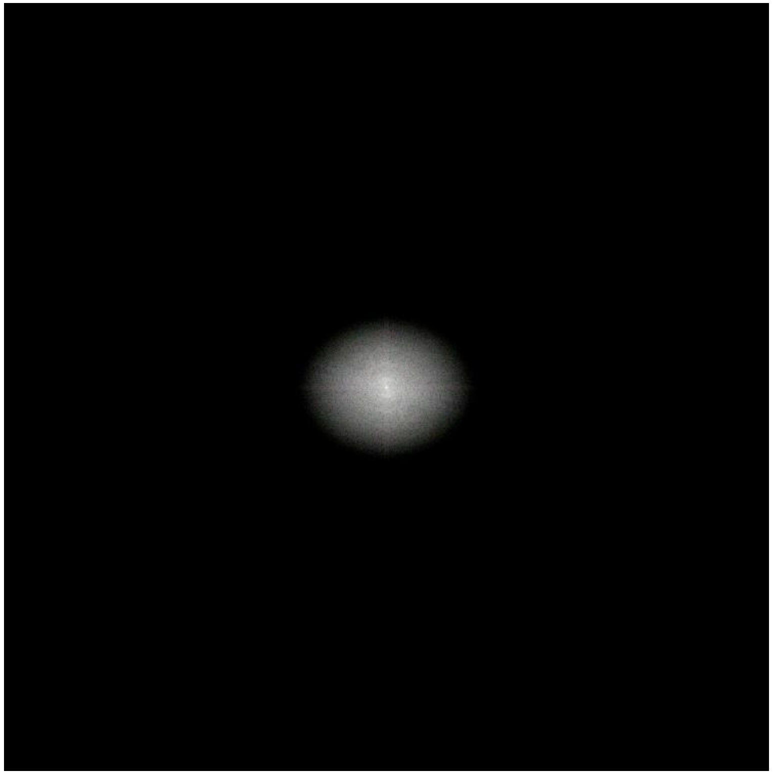
blurred image σ=4



log power spectrum σ=4

Blurring with a Gaussian ($\sigma = 8$)

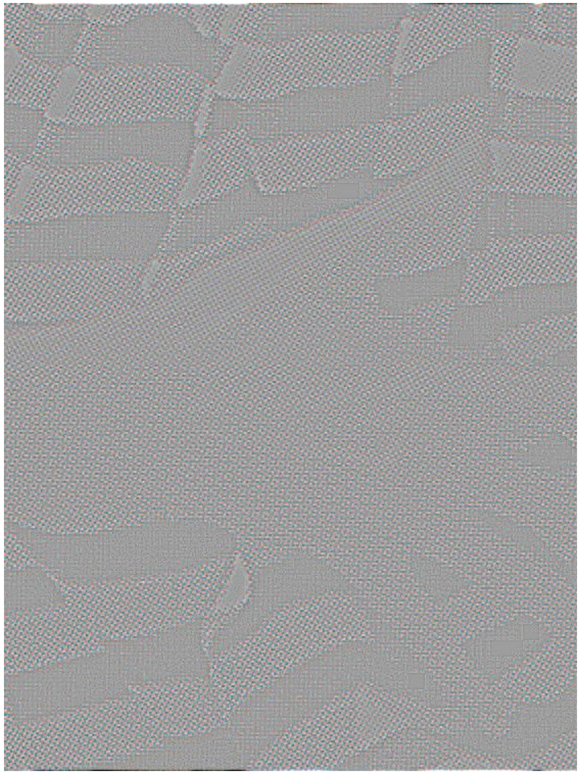
blurred image σ=8



log power spectrum σ=8

Blurring with a Gaussian ($\sigma = 1$)

original



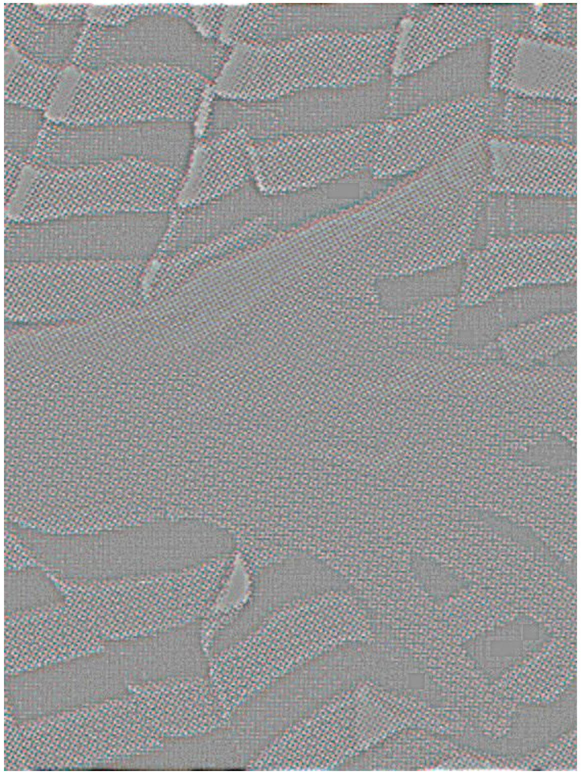
difference

blurred $\sigma = 1$

middle gray = 0, normalized

Blurring with a Gaussian ($\sigma = 2$)

blurred $\sigma = 2$



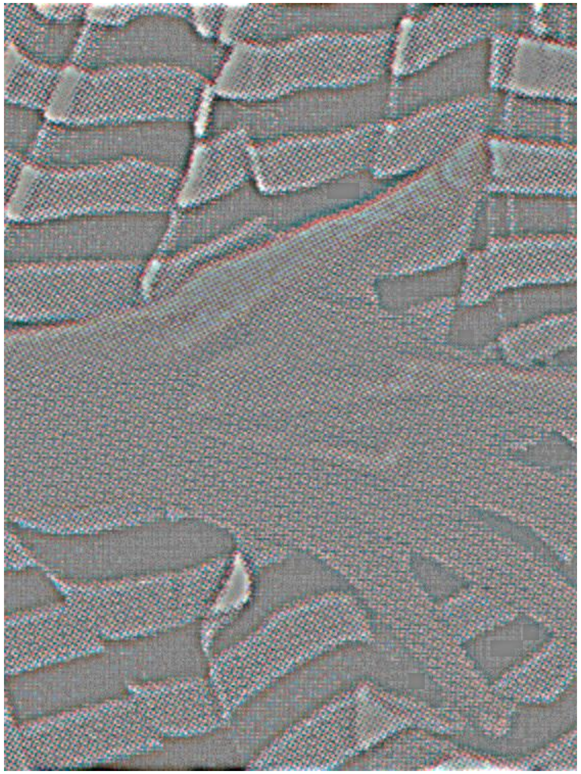
difference

original

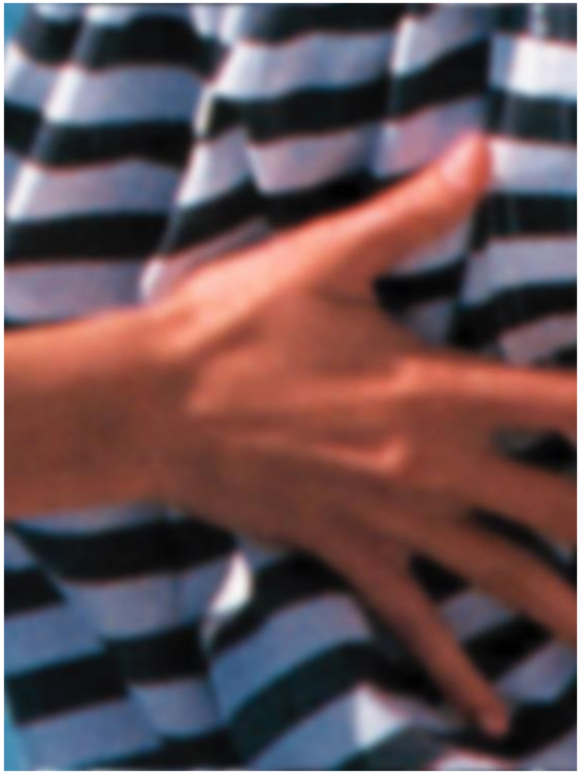
middle gray = 0, normalized

Blurring with a Gaussian ($\sigma = 4$)

original



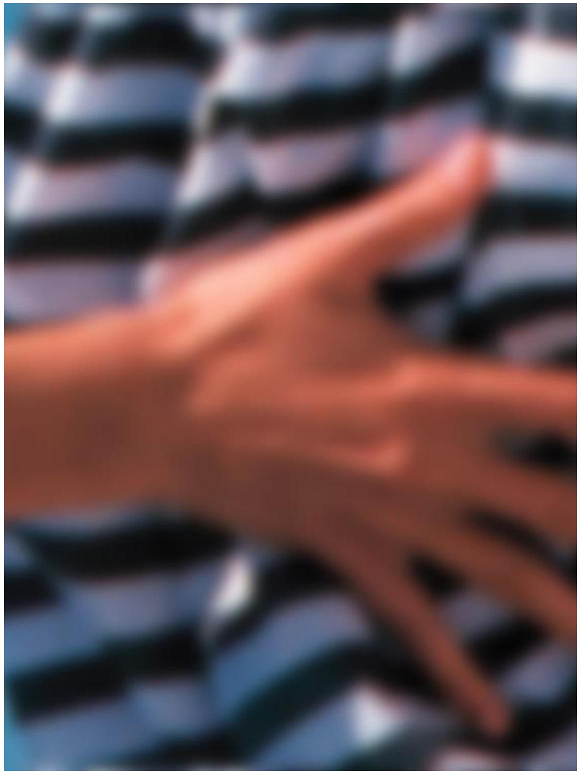
difference



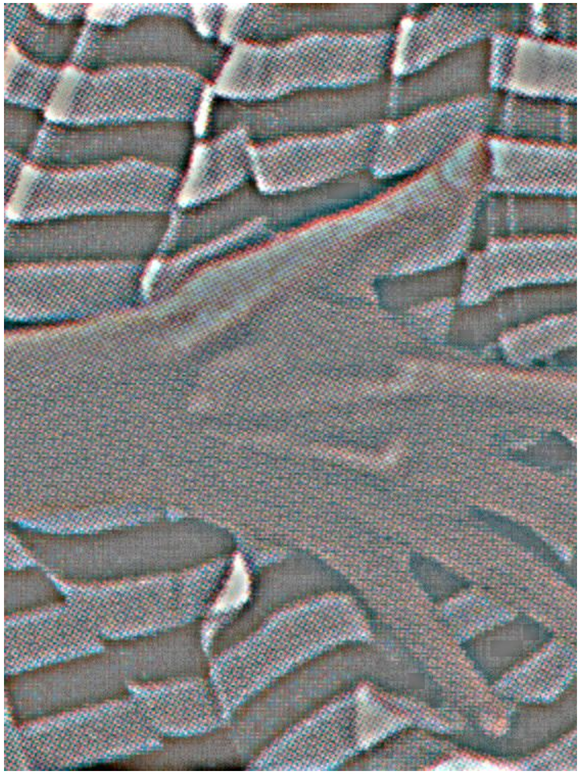
blurred $\sigma = 4$

middle gray = 0, normalized

Blurring with a Gaussian ($\sigma = 8$)



blurred $\sigma = 8$



difference

original

middle gray = 0, normalized

Problem with Blurring to Reduce HTD Distortion

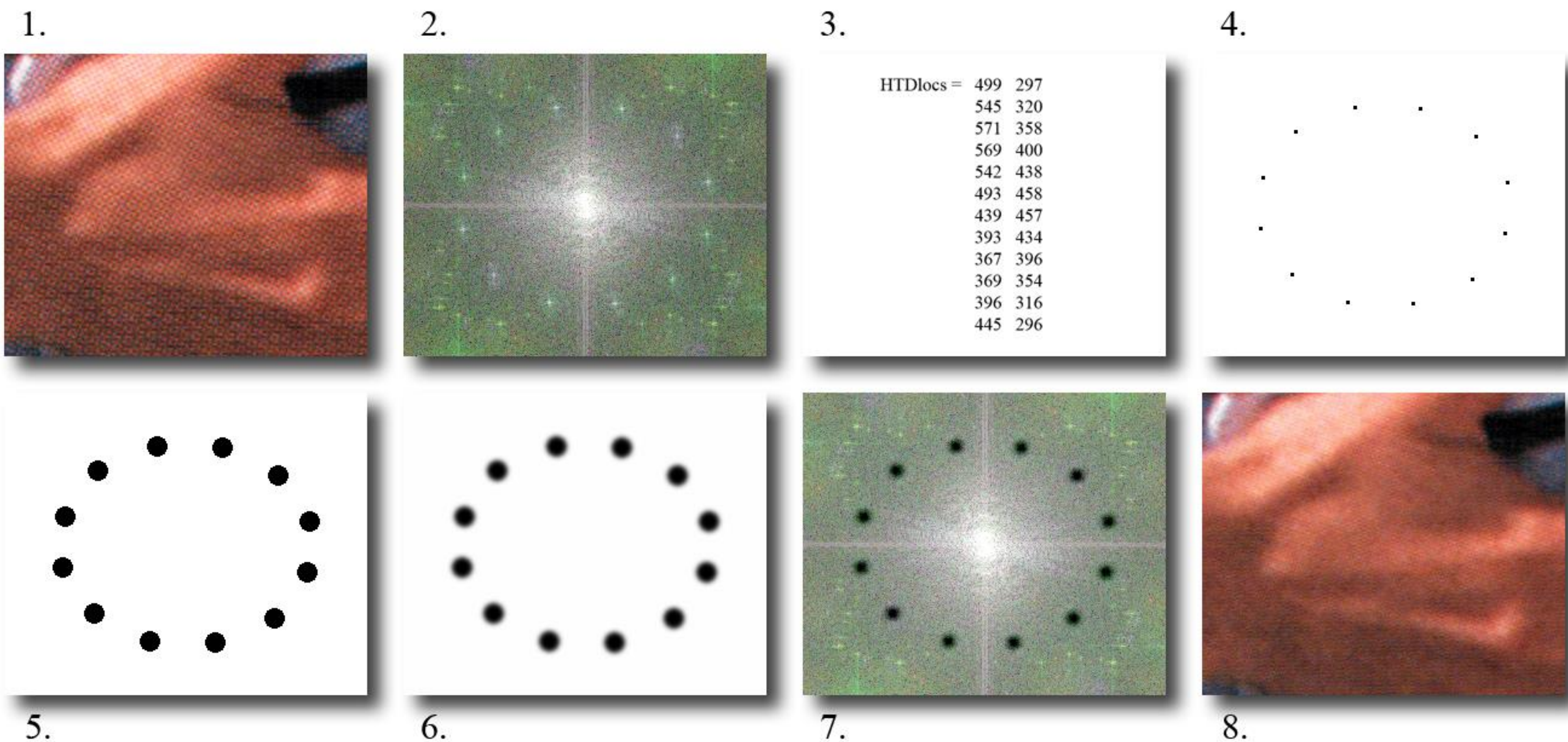
$\Longrightarrow$ It blurs everything.

Better to remove the HTD frequency components selectively:

1. Read in the image.
2. Compute the log power spectrum of the image.
3. Find the locations of the HTD spectrum peaks.
4. Mark these on a mask.
5. Enlarge the points to regions that cover most of the energy.
6. Blur the mask for used as a notch filter.
7. Multiply the Fourier transform of the image by the mask.
8. Take the inverse Fourier transform of the result.

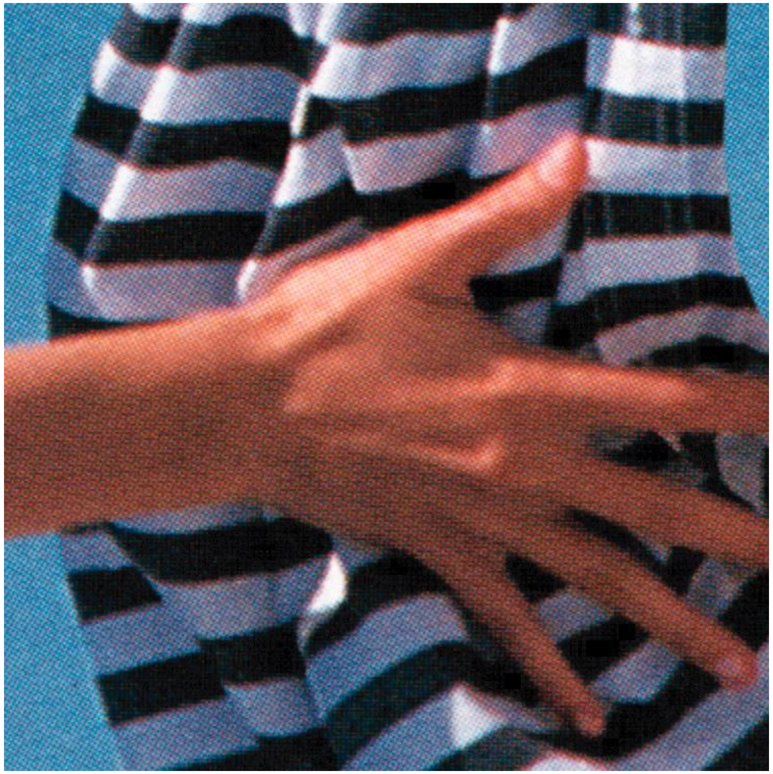
Remove HTD Distortion Selectively

… through notch filtering.

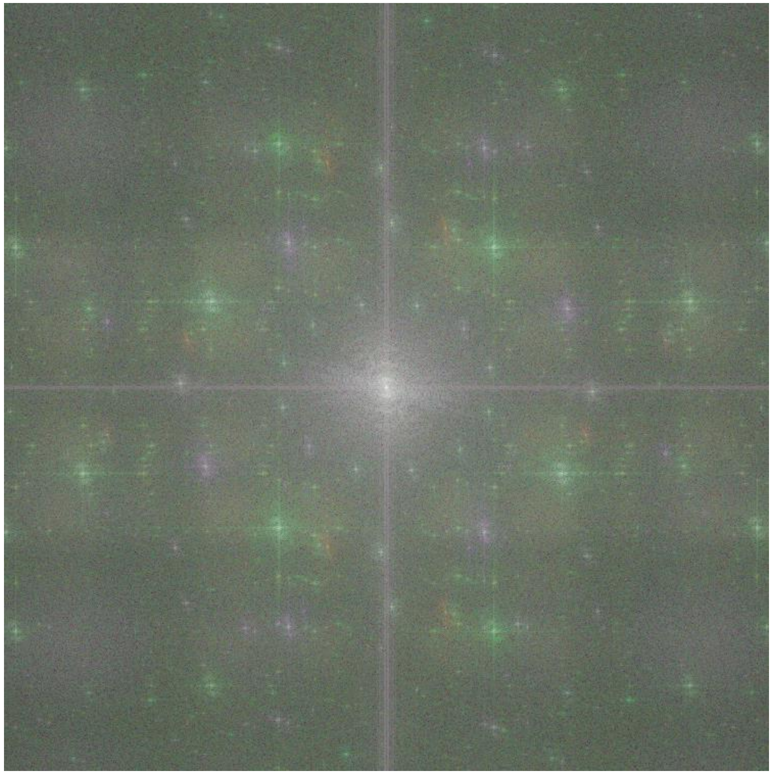


1. Read in image; 2. Compute power spectrum; 3. Locate HTD frequency components; 4. Mark locs on a mask; 5. Enlarge points to regions; 6. Blur the mask; 7. Multiply FT of image by mask; 8. Take inverse FT of result;

Notch Filtering of Halftone Dot Distortion



original

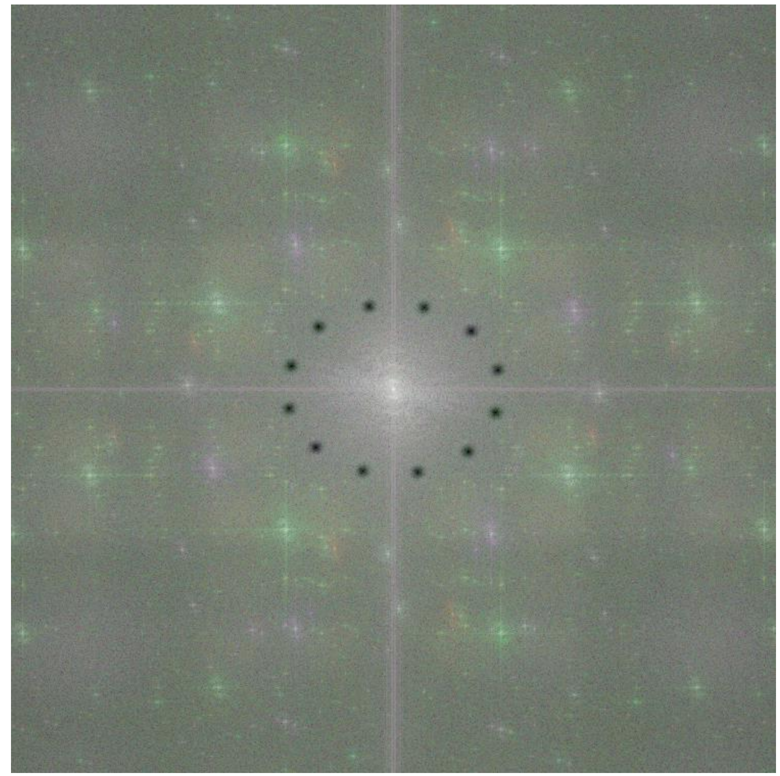


log power spectrum

Notch Filtering

Since not much distortion was removed, these must be subharmonics of the true dot frequencies

frequency masked 1

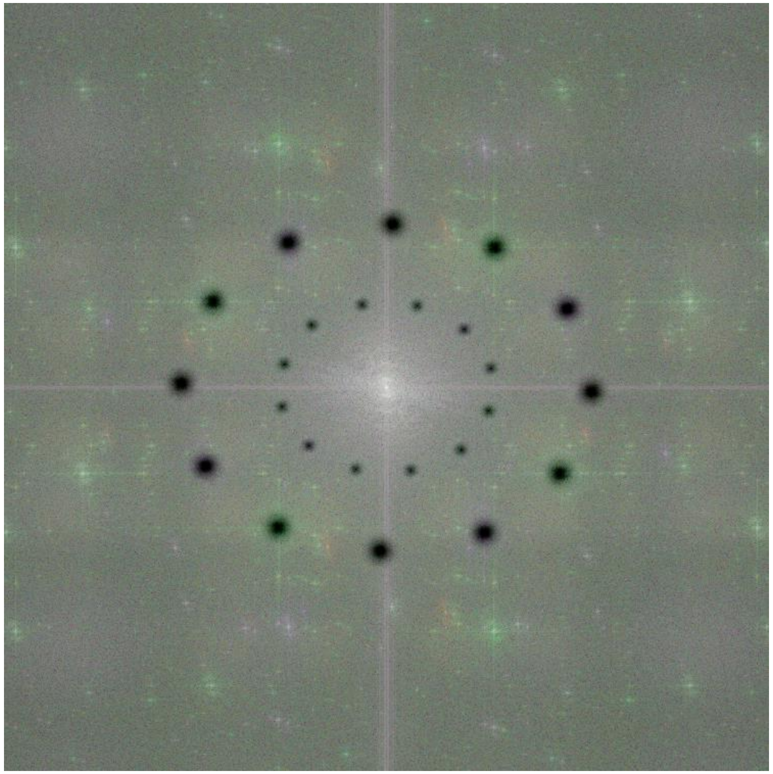


log power spectrum

Notch Filtering

The outer ring are the actual HTD frequencies. Can we do any better?

frequency masked 2

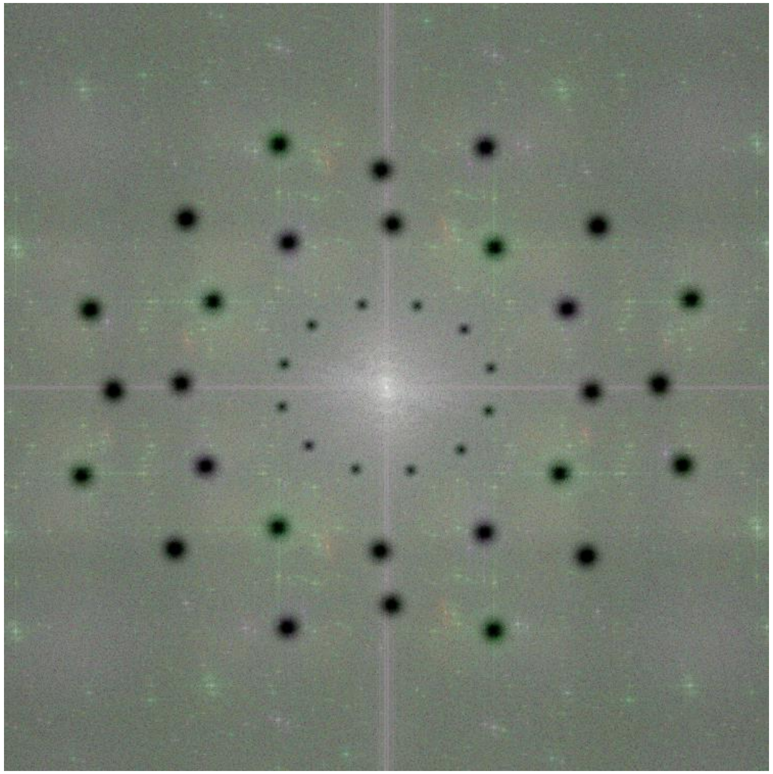


log power spectrum

Notch Filtering

Not much. The harmonics contribute little energy to the image.

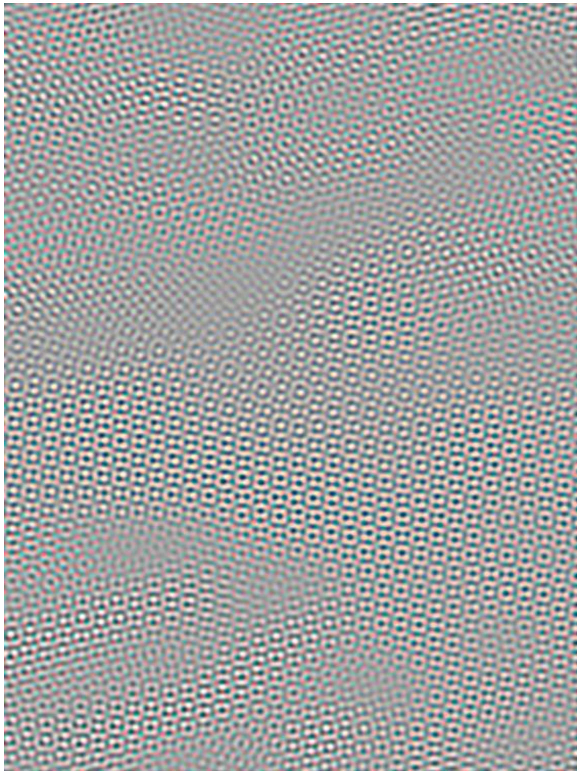
frequency masked 3



log power spectrum

Notch Filter Difference Images

original



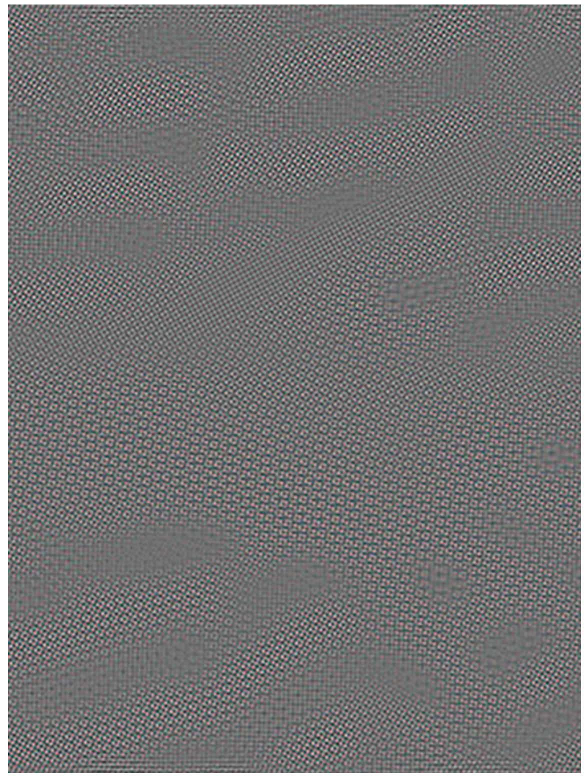
difference

frequency masked 1

middle gray = 0, normalized

Notch Filter Difference Images

frequency masked 2



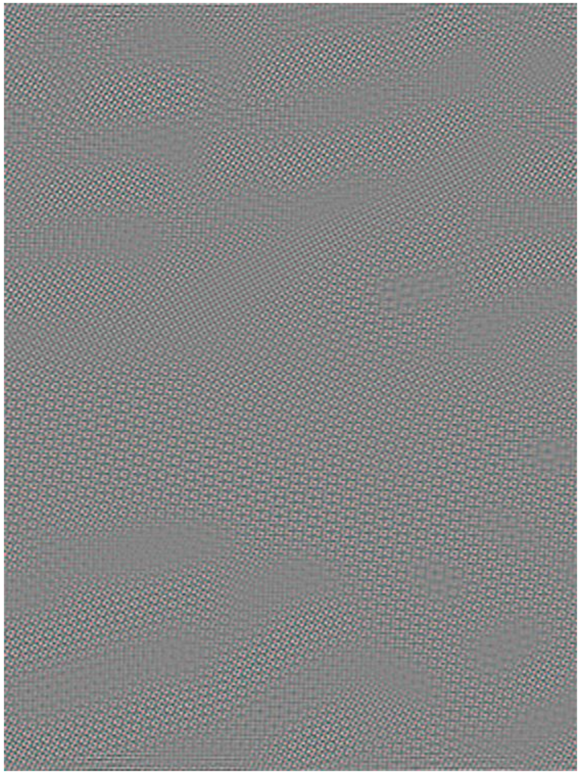
difference

original

middle gray = 0, normalized

Notch Filter Difference Images

original



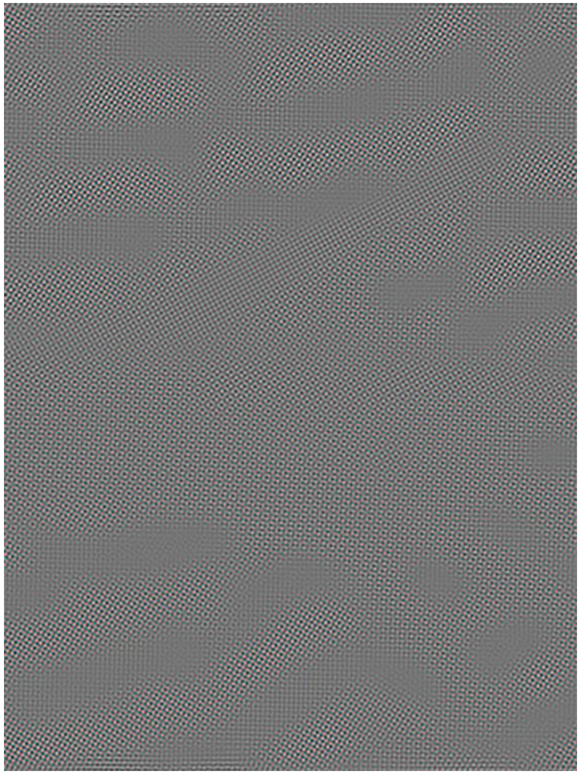
difference

frequency masked 3

middle gray = 0, normalized

Notch Filter Difference Images

frequency masked 1



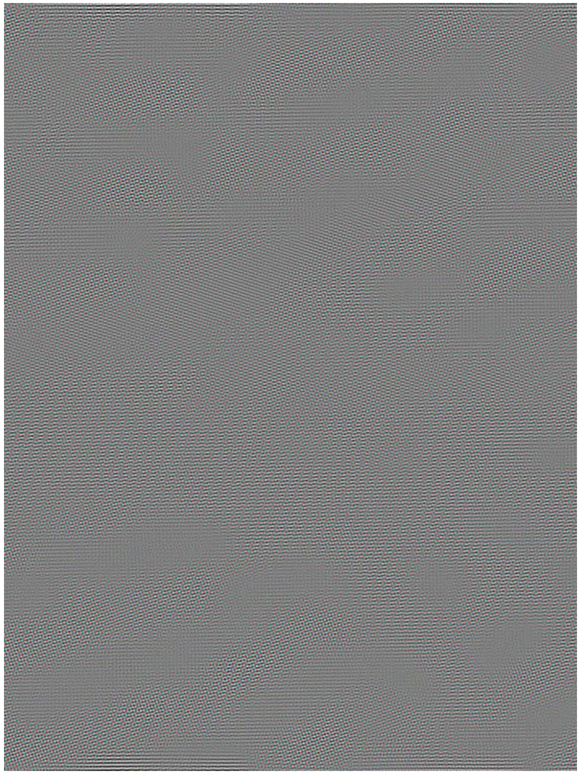
difference

frequency masked 2

middle gray = 0, normalized

Notch Filter Difference Images

frequency masked 2



difference

frequency masked 3

middle gray = 0, normalized