Digital Image Processing

Image Smoothing for Noise Reduction: Patch-Based Filtering

Suyash P. Awate

Image Denoising

- Bilateral filtering for gray and color images
  - IEEE Int. Conf. Computer Vision. 1998.
  - https://en.wikipedia.org/wiki/Bilateral_filter
- Bilateral Filtering: Theory and Applications
  - http://www.nowpublishers.com/article/Details/CGV-020
- Recent Trends in Denoising Tutorial (including algorithms for patch based filtering)
  - https://web.archive.org/web/20111118001653/http://www.stanford.edu:80/~slansel/tutorial/summary.htm
  - https://en.wikipedia.org/wiki/Non-local_means
- A Tour of Modern Image Filtering: New Insights and Methods, Both Practical and Theoretical
  - IEEE Signal Processing Magazine, Vol 30, Issue 1
  - http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6375938

Patch-Based Filtering

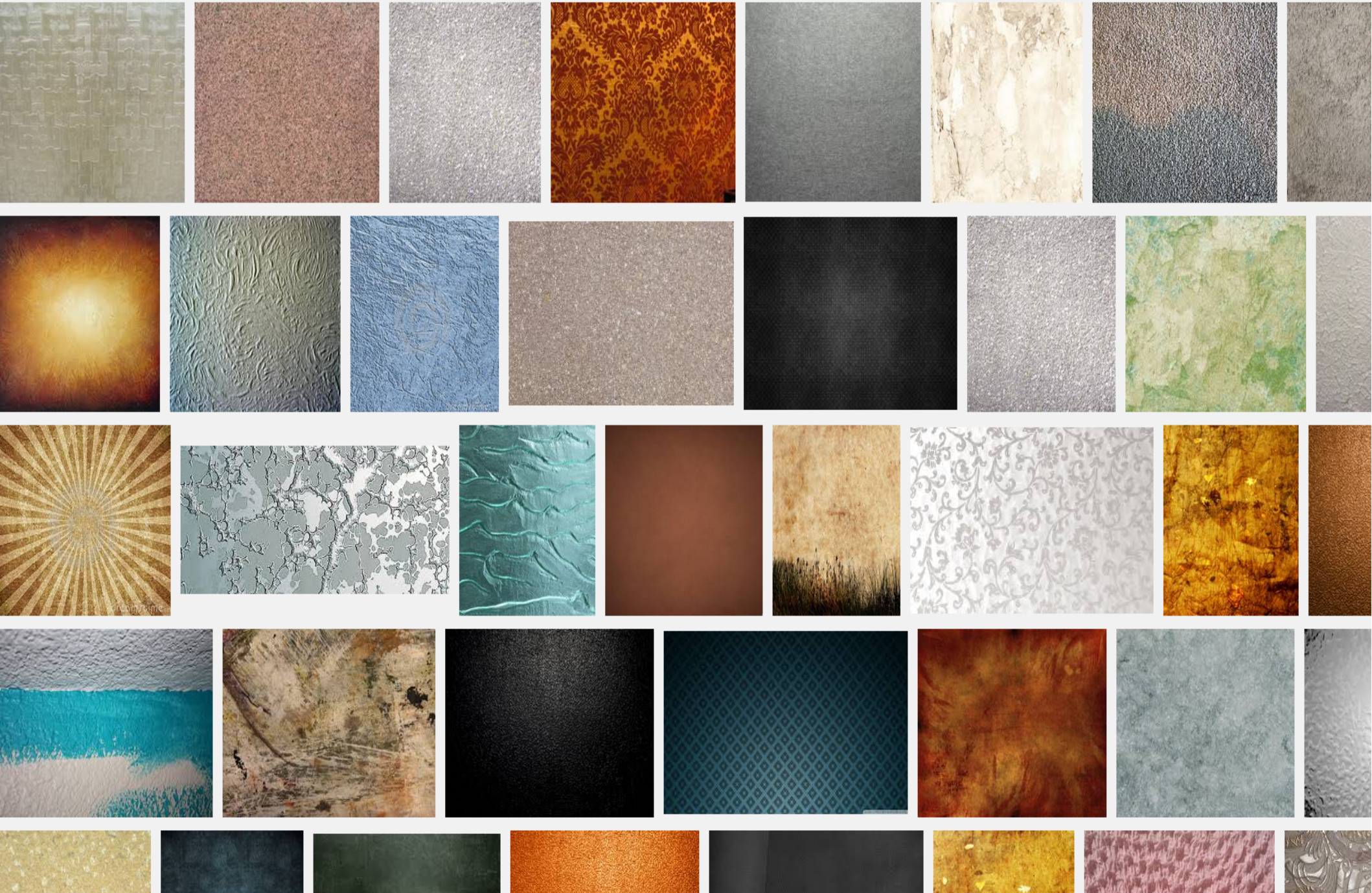
- Bilateral filter
  - Intensity-based similarity is solely based on intensity differences between pixels
  - Not best use of intensity information for **texture**
  - What is texture ?
    - Repeated spatial patterns of intensities
      - Can be **geometric**, e.g., checks on shirt
      - Can be **stochastic**, e.g. grass
    - Spatially homogeneous

Google textured

suyash

Web Images Videos Maps News More Search tools

SafeSear



Google

textureless

Web | Images | Maps | Videos | News | More | Search tools



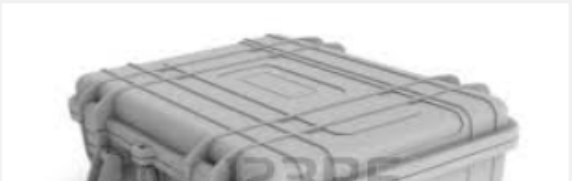


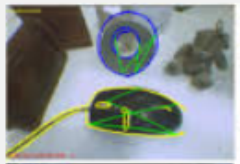





Technical Demonstration on Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes







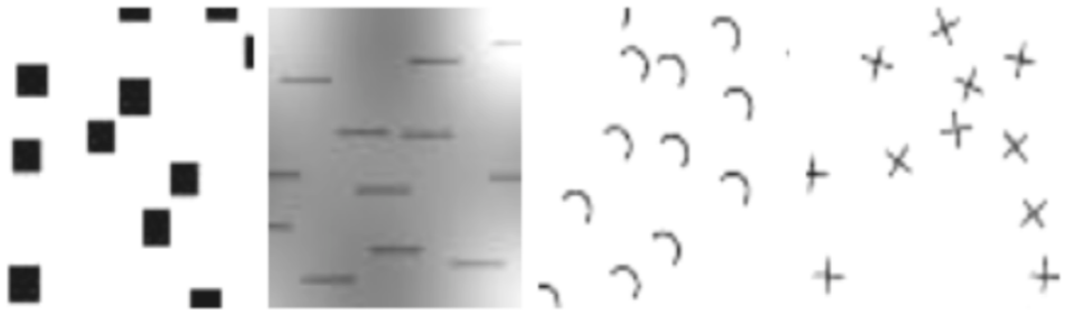

Artificial/periodic:

Artificial/periodic:

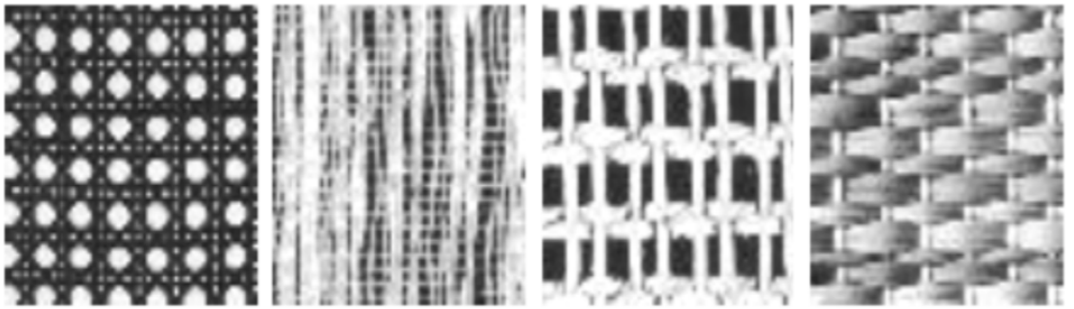
Artificial/non-periodic:

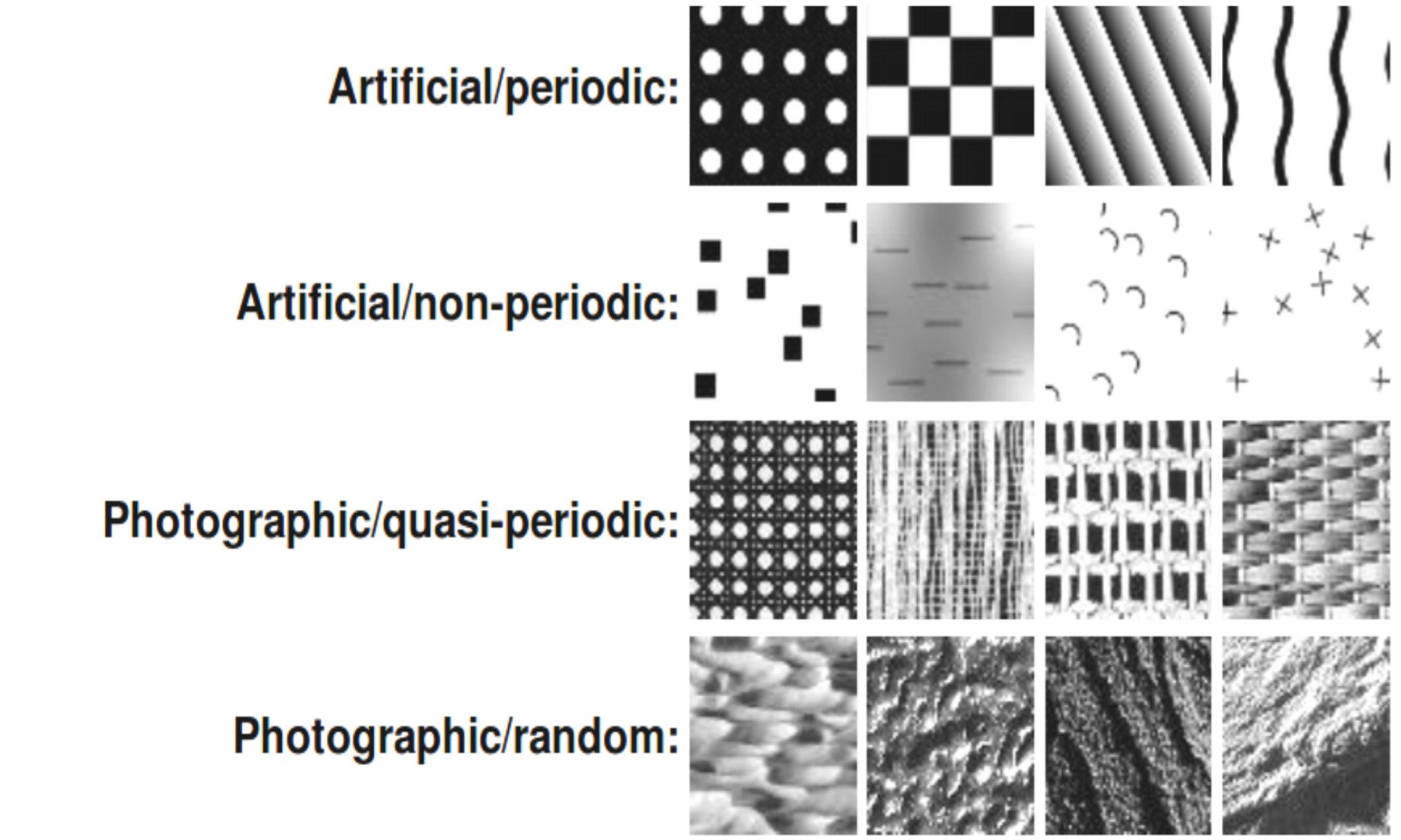
Artificial/periodic:

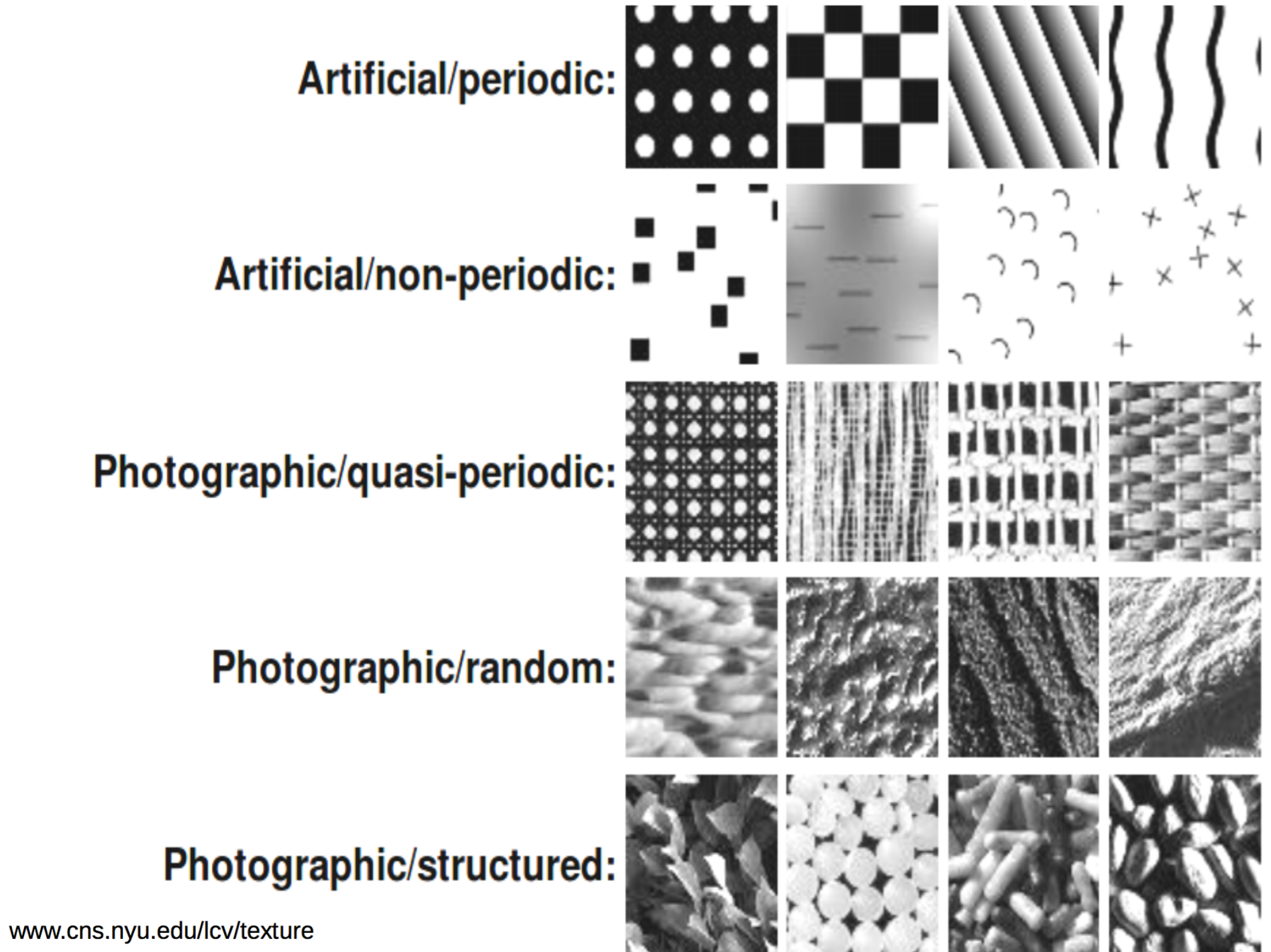
Artificial/non-periodic:



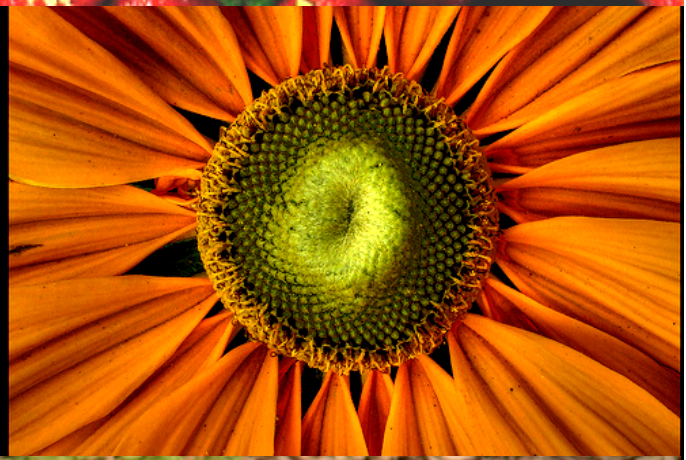
Photographic/quasi-periodic:




Artificial/periodic:
Artificial/non-periodic:
Photographic/quasi-periodic:
Photographic/random:

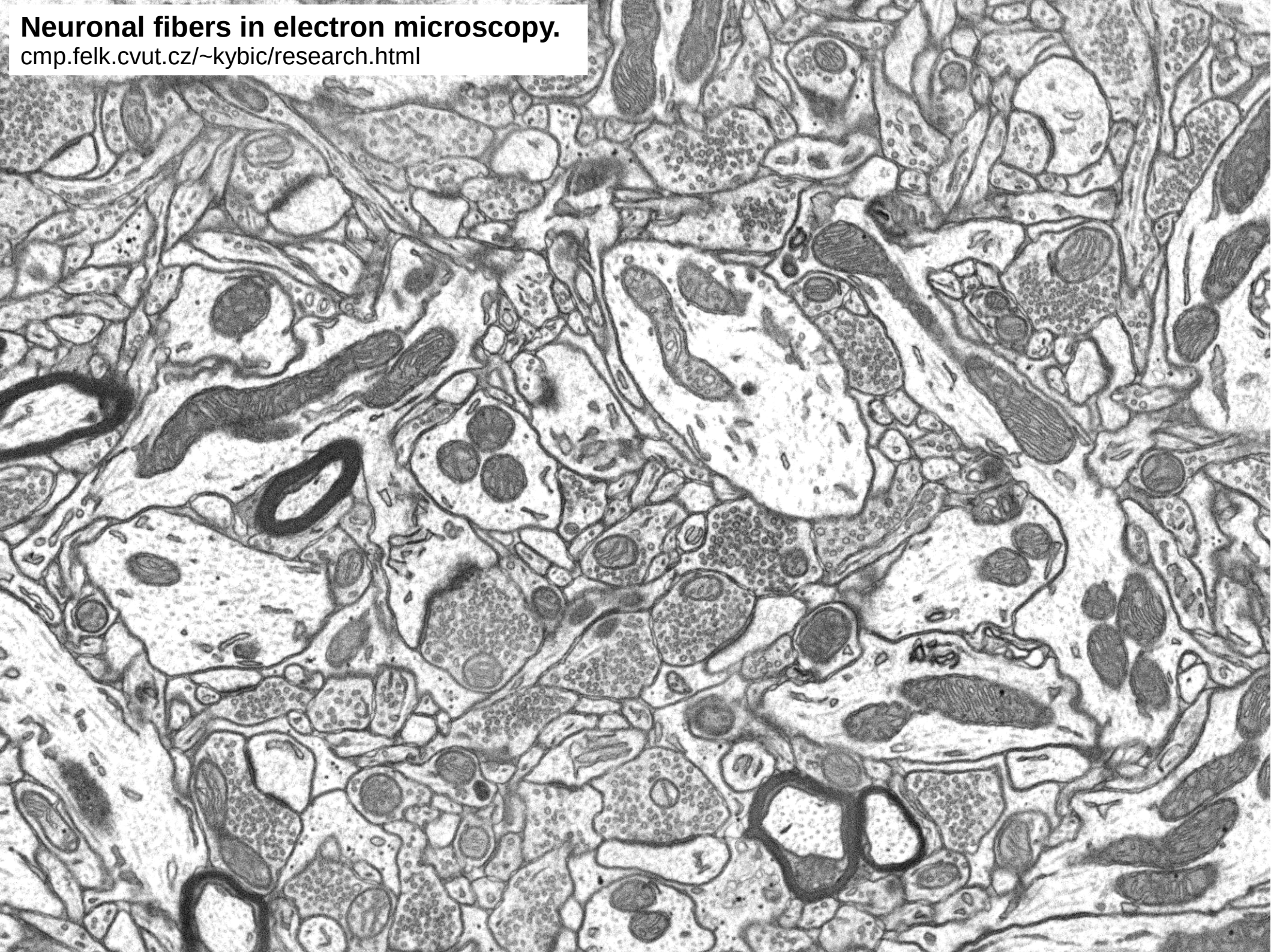

Artificial/periodic:
Artificial/non-periodic:
Photographic/quasi-periodic:
Photographic/random:
Photographic/structured:
www.cns.nyu.edu/lcv/texture



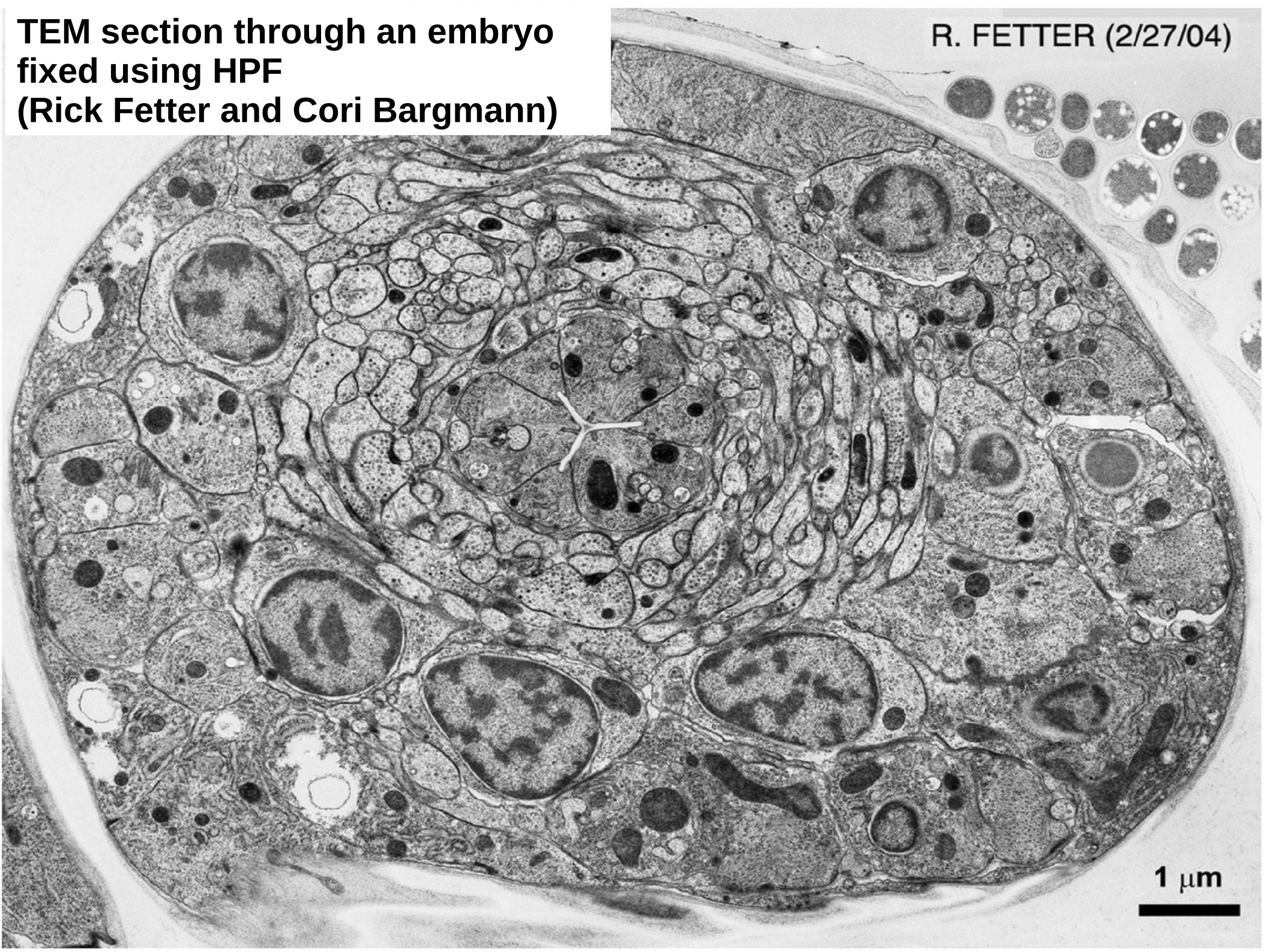





Neuronal fibers in electron microscopy.
cmp.felk.cvut.cz/~kybic/research.html



TEM section through an embryo fixed using HPF (Rick Fetter and Cori Bargmann)



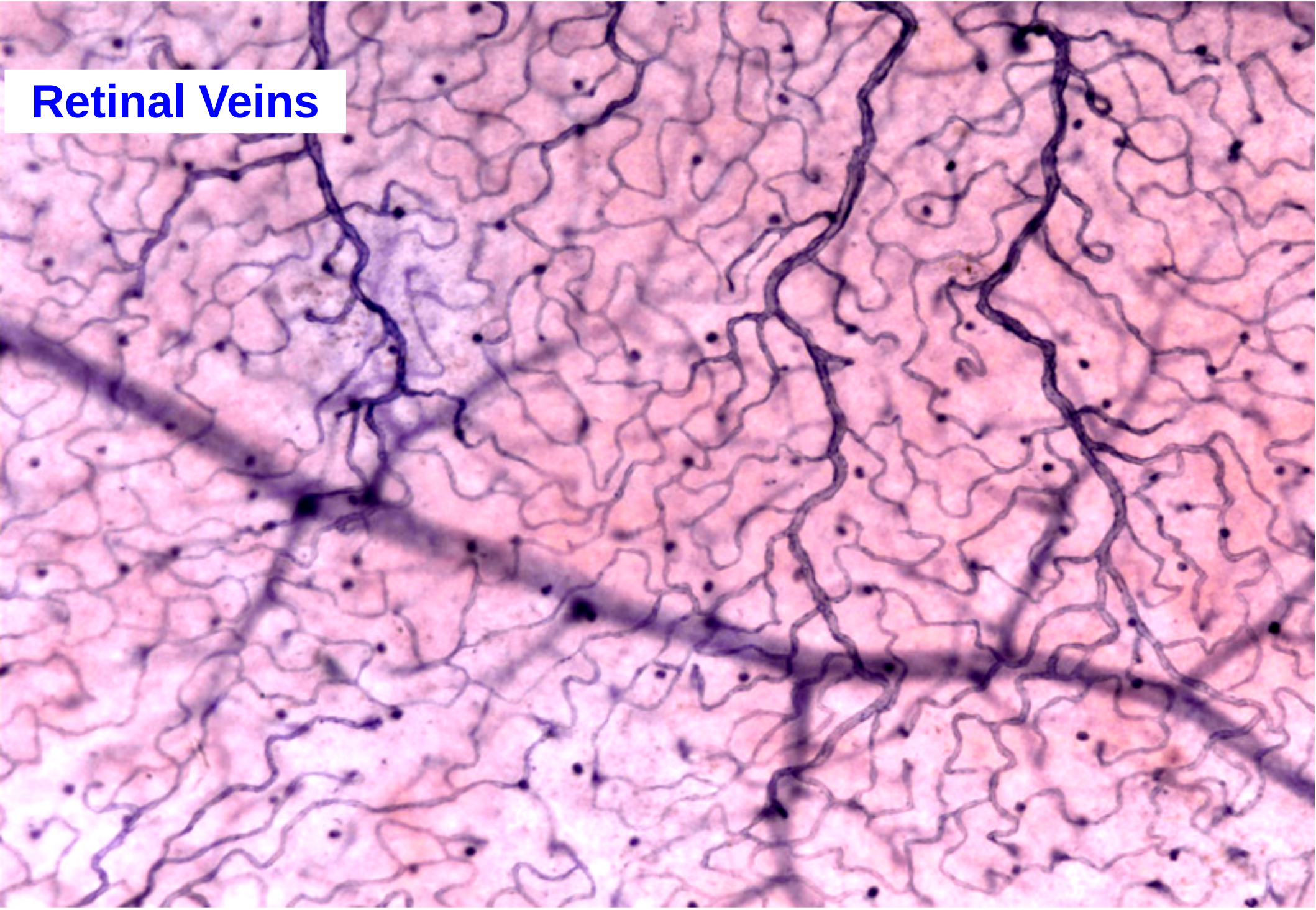
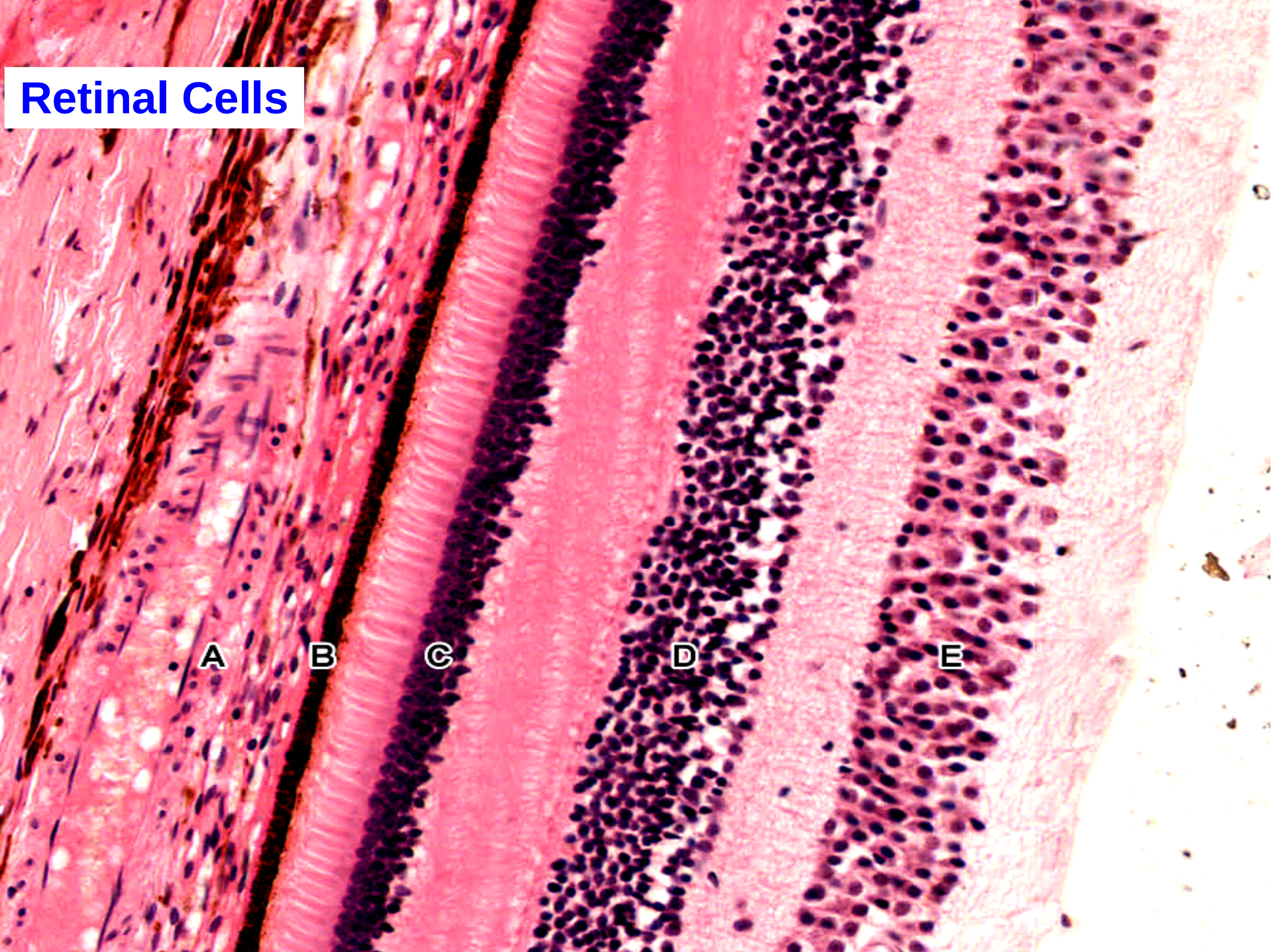
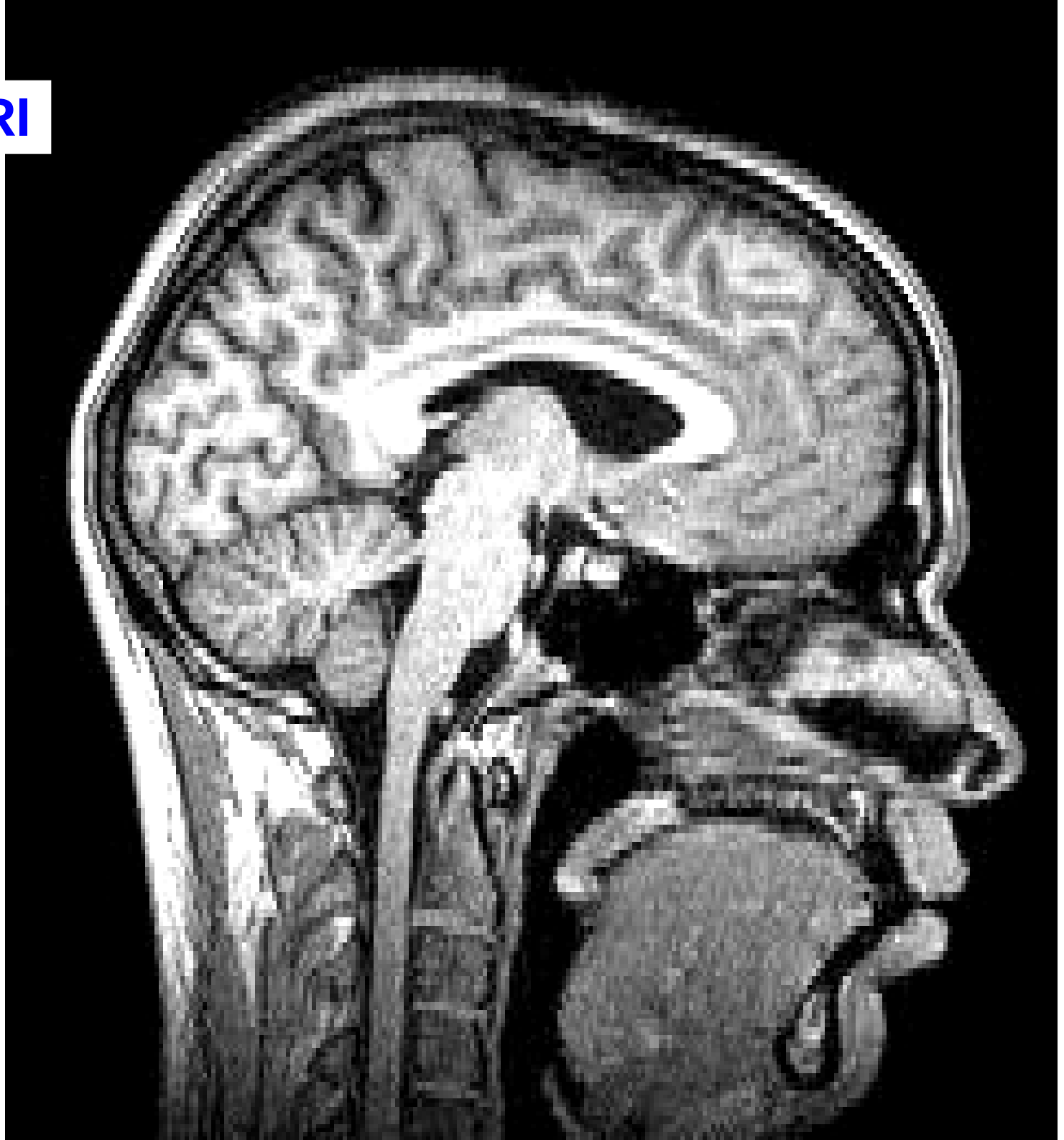


Fig. 18b. Flatmount view of a rat retina stained with NADPH-diaphorase at the level of focus of a major vein and venules. (Courtesy of Toby Holmes, Moran Eye Center).

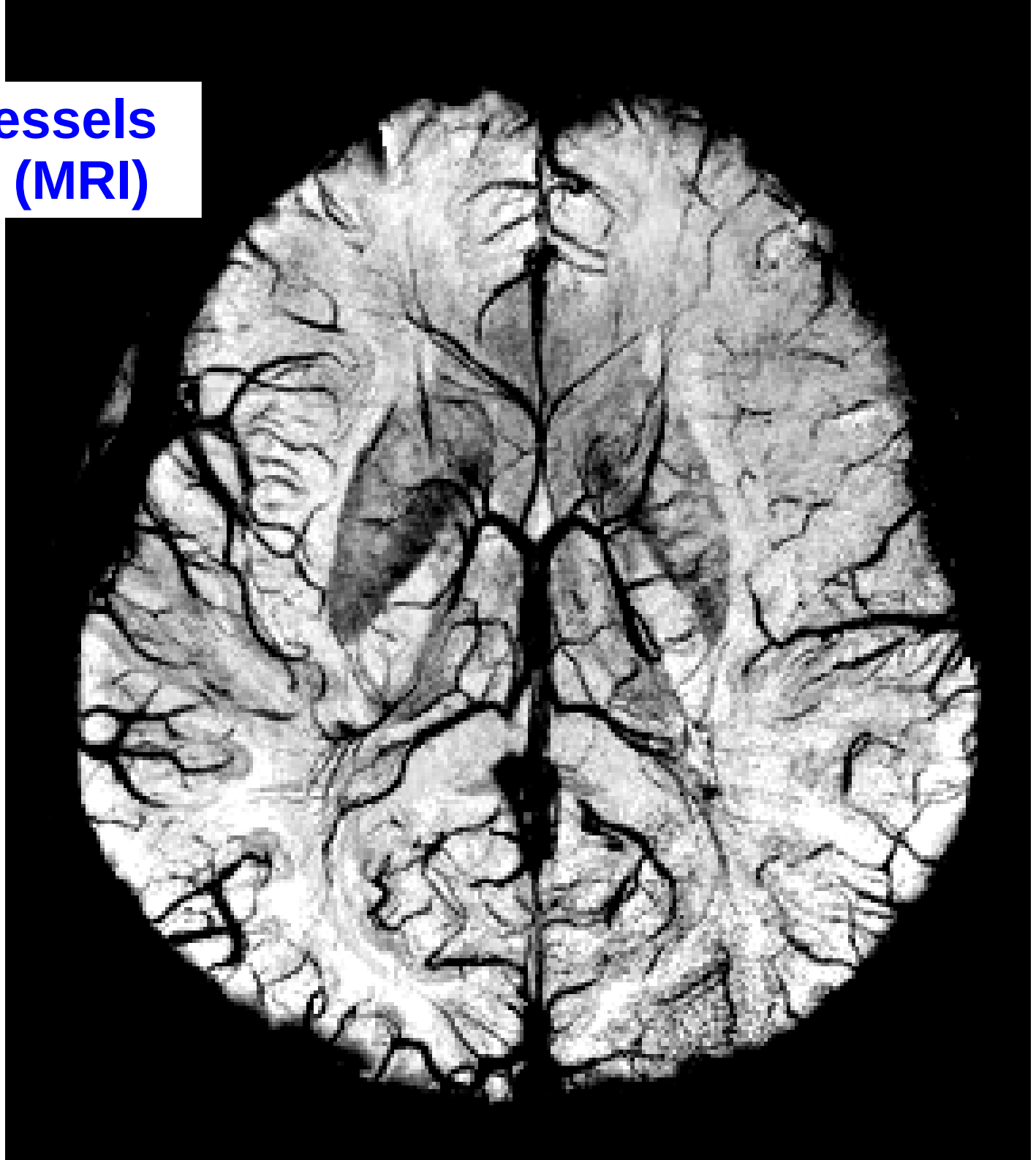
Retinal Cells

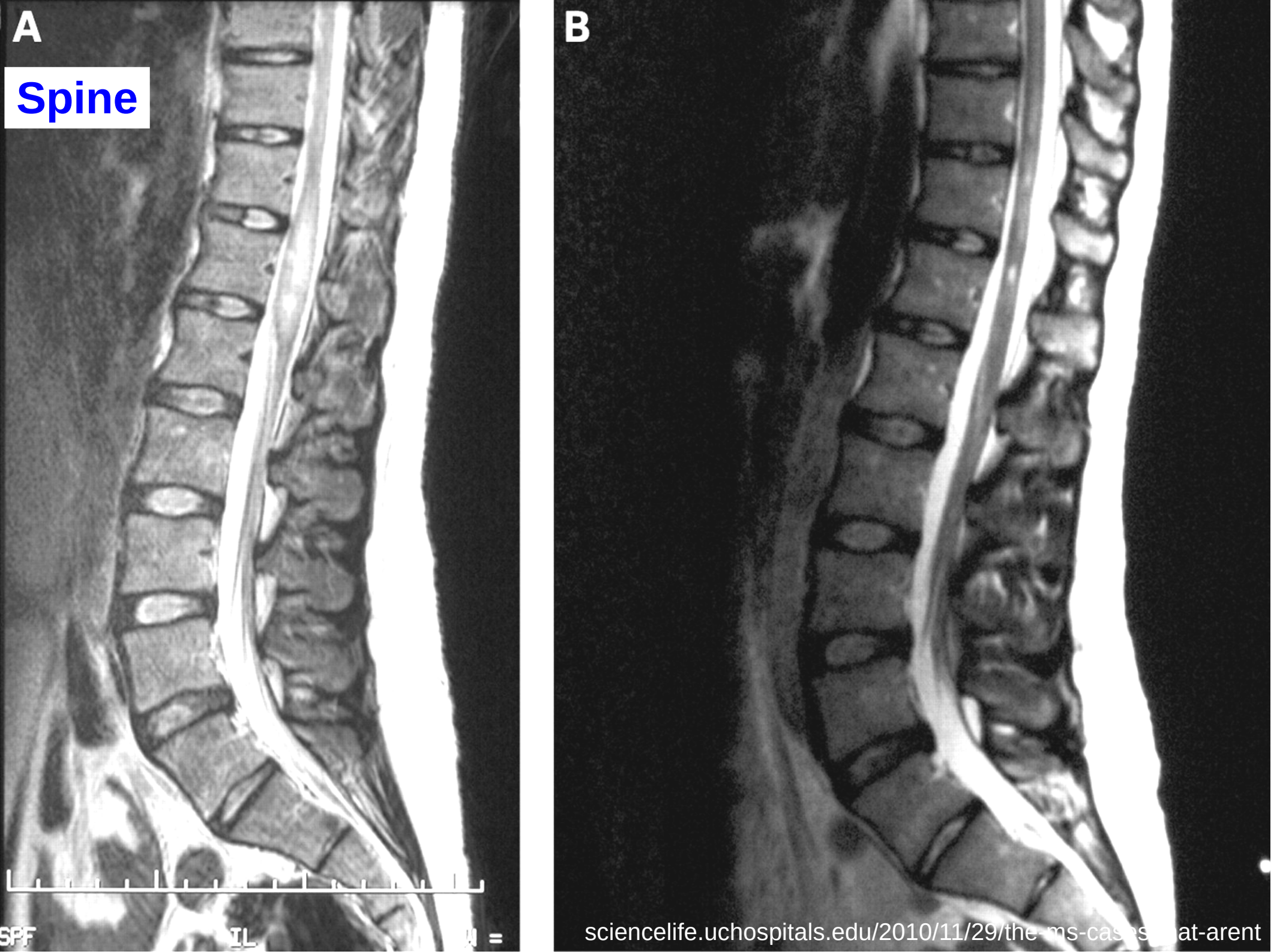


Head MRI



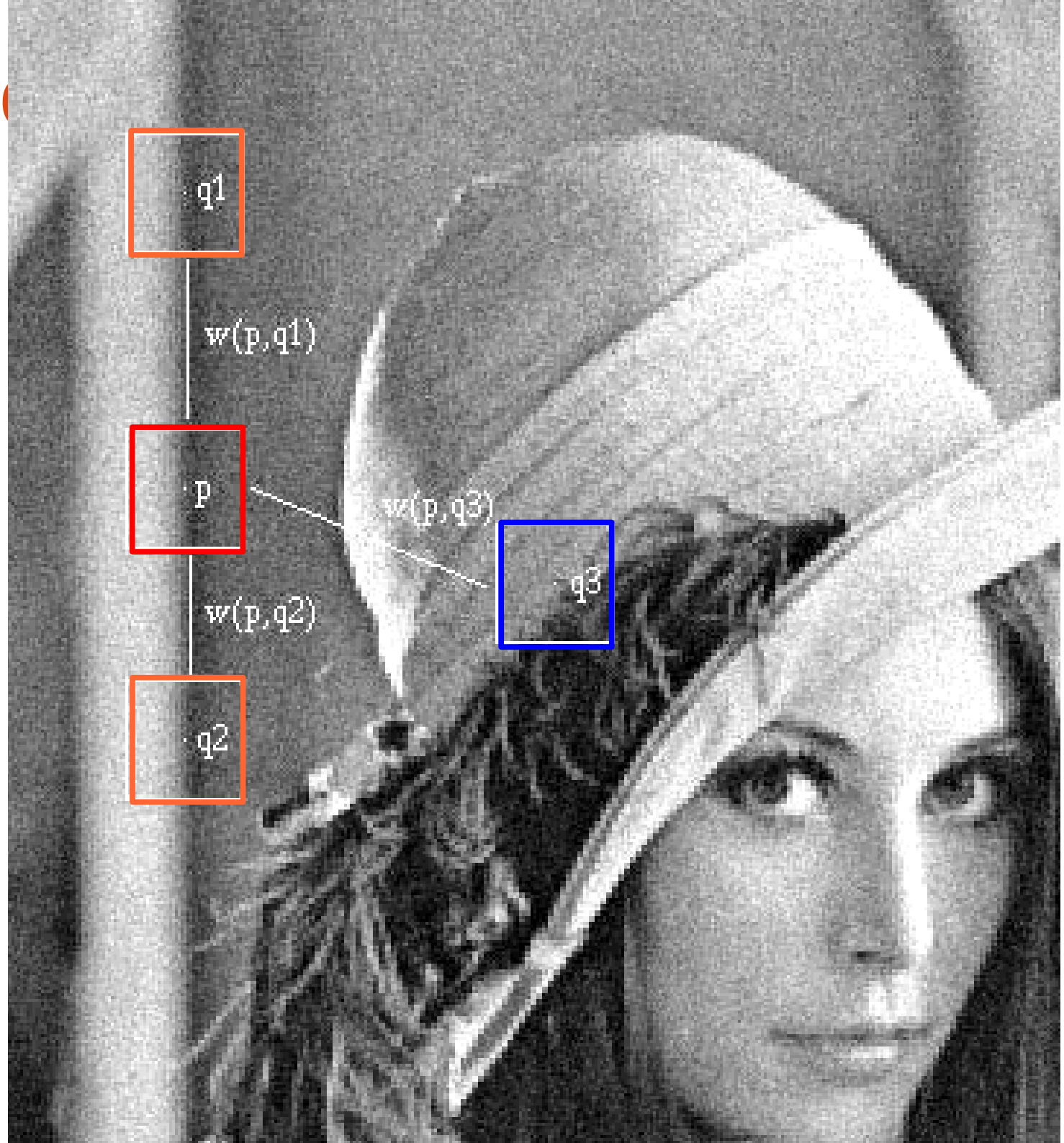
Blood Vessels in Brain (MRI)




A
Spine
B
sciencelife.uchospitals.edu/2010/11/29/the-ms-cas

Patch-Base

- For denoising pixel “p” :
  - Pixels “q1”, “q2” should have large weights
  - Pixel “q3” should have small weight



Patch-Based Filtering

- Filtered intensity
  = weighted average of other intensities
- How to design weights ?

$$\sum_{j \in \Lambda} \omega_h(j,k) f(j)$$

  - Based on patches around pixels
  - For denoising pixel “p”,
    a **pixel “q” will have a large weight if**
    patch around “q” is similar to patch around “p”
- Why ?

  (1) Helps give more weights to pixels that belong to the **same (part of) “object”**
    - Pixels on same (part of) object have similar patches around them

  (2) Helps counter larger **noise levels** (coming soon ...)

Patch-Based Filtering

- Filtered intensity
  = weighted average of other intensities

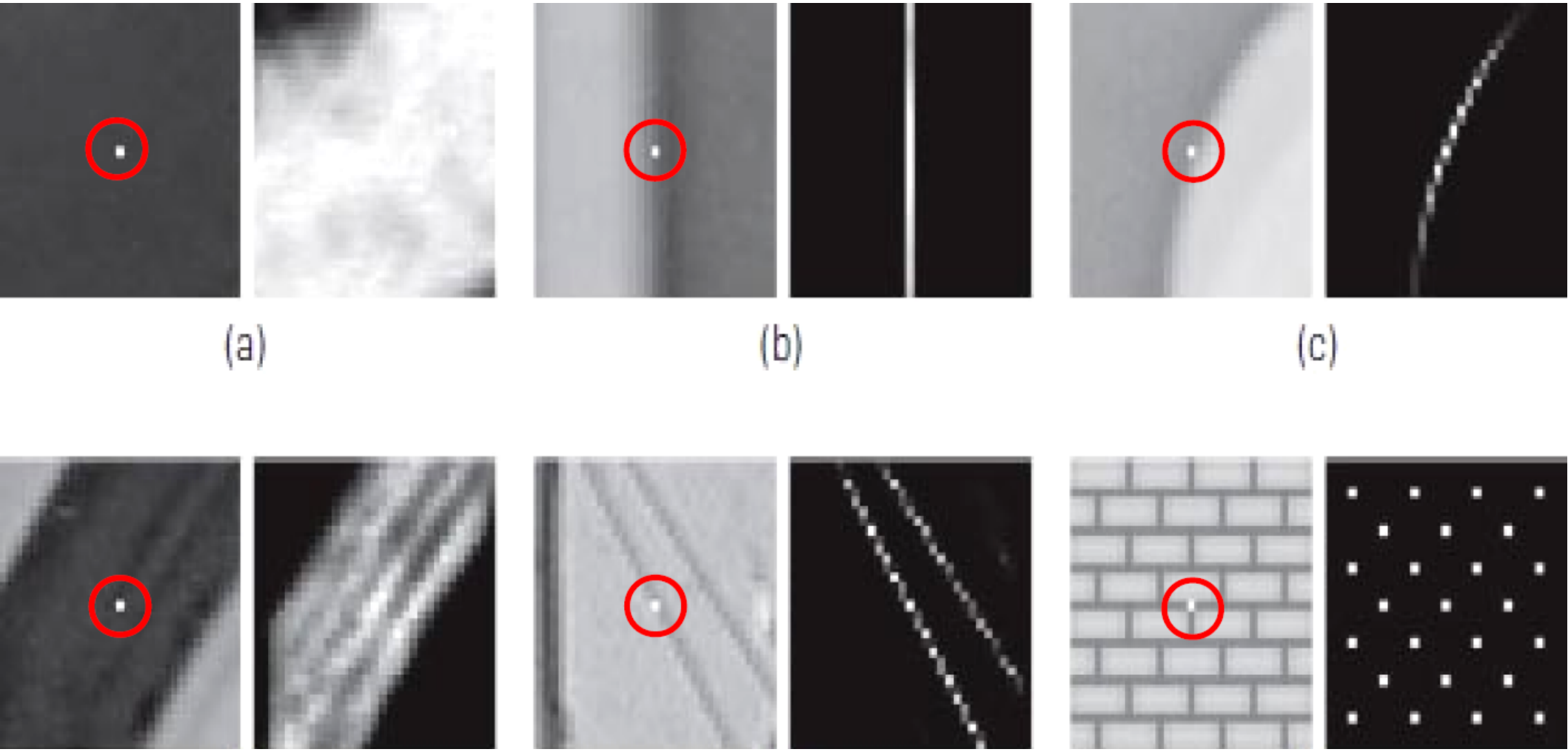
$$\sum_{j \in \Lambda} \omega_h(j,k) f(j)$$

- How to measure patch dissimilarity ?
  - Euclidean distance between patches
  - Weights = Gaussian on patch distance

$$\omega_h(j,k) = \frac{\exp\left(- \frac{\|f(\mathcal{R}_j) - f(\mathcal{R}_k)\|_{2,a}^2}{h^2} \right)}{\sum_{j \in \Lambda} \exp\left(- \frac{\|f(\mathcal{R}_j) - f(\mathcal{R}_k)\|_{2,a}^2}{h^2} \right)}$$

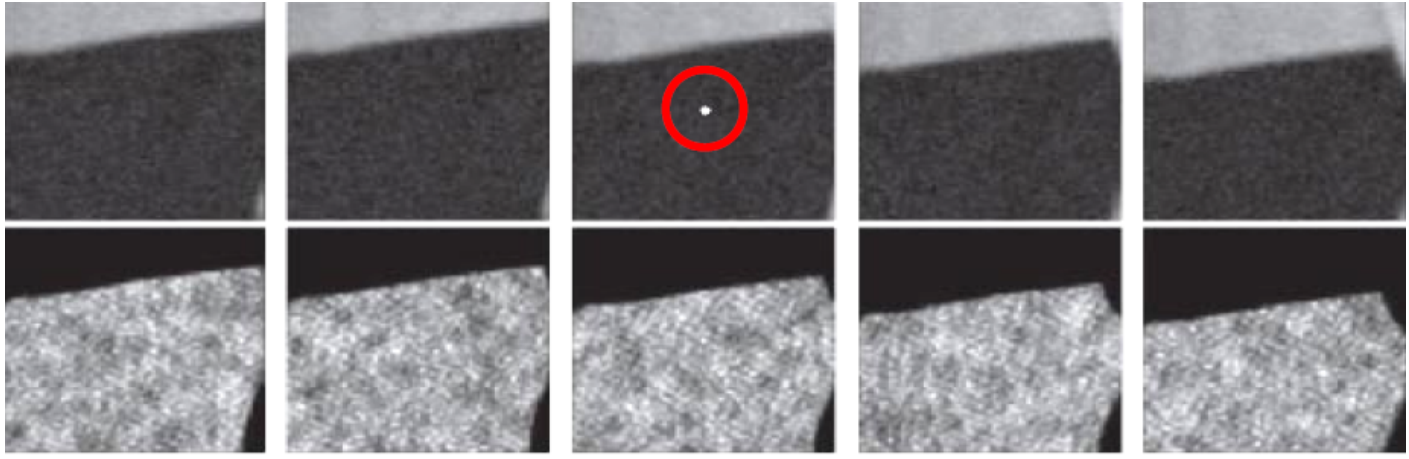
Patch-Based Filtering

- Pixel weights (right)
  for a chosen pixel in an image (left)

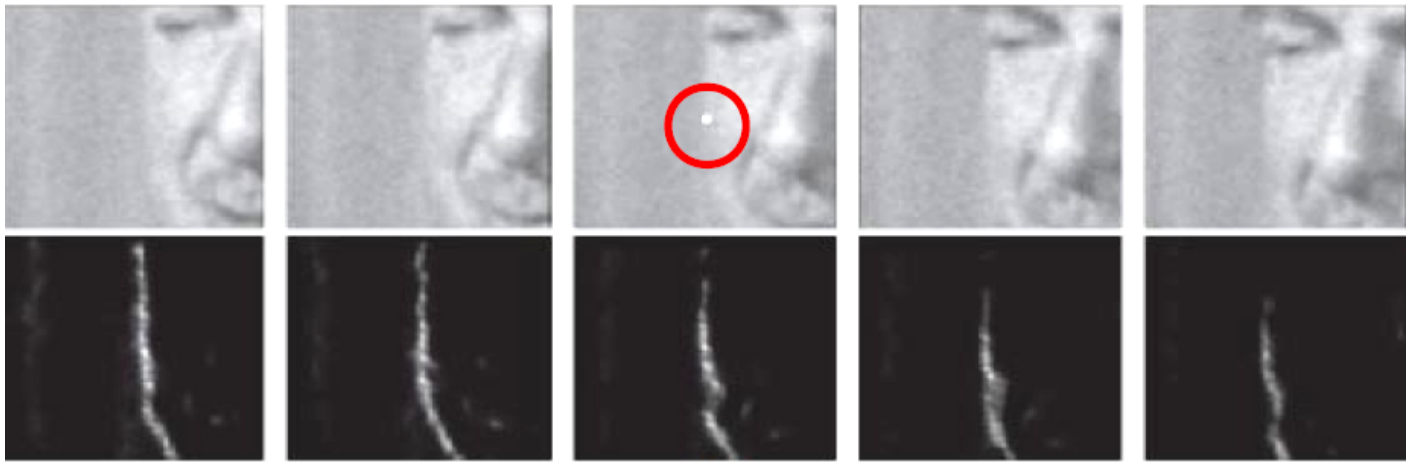


Patch-Based

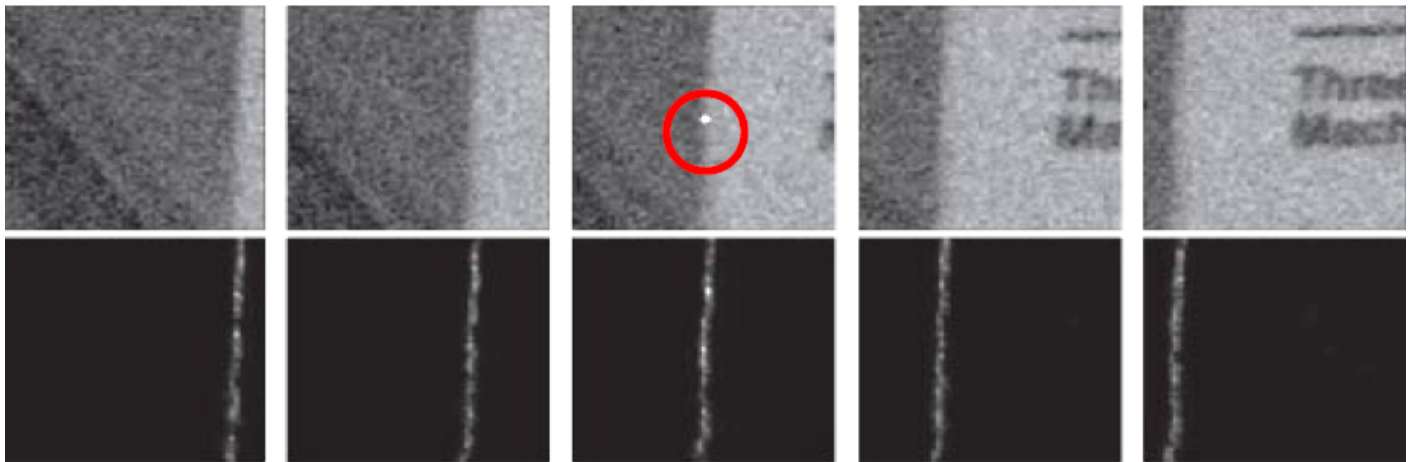
- Pixel weights (bottom) for a chosen pixel in a video (top)



(a)

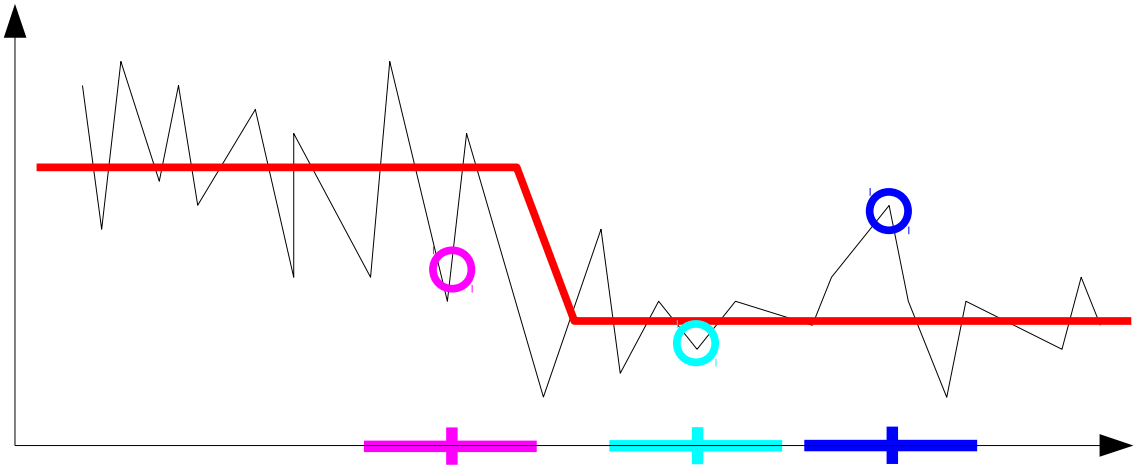


(b)



Patch-Based Filtering

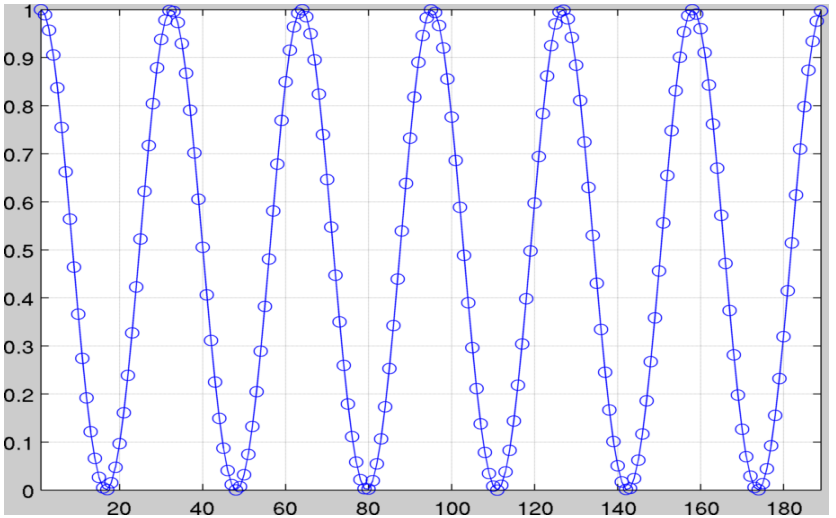
- Patch-based weights helpful in case of large noise levels
  - Red = noiseless image
  - Black = noisy image
  - **Blue** pixel intensity is closer to **magenta** pixel than **cyan** pixel
    - Undesirable
  - **Blue** intensity patch much more similar to **cyan** intensity patch than **magenta** intensity patch
    - Desirable



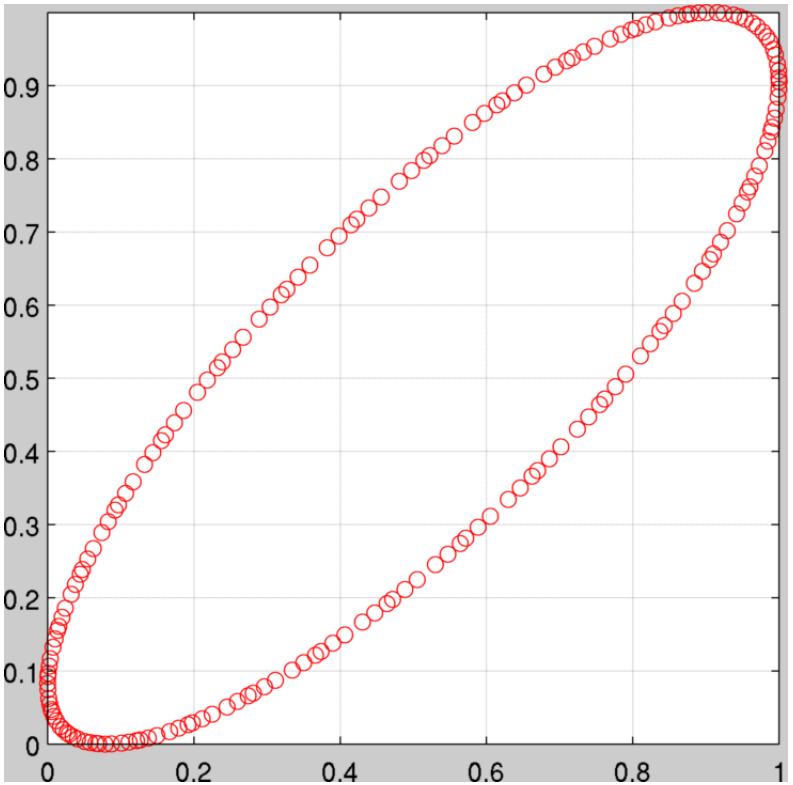
Patch-Based Filtering

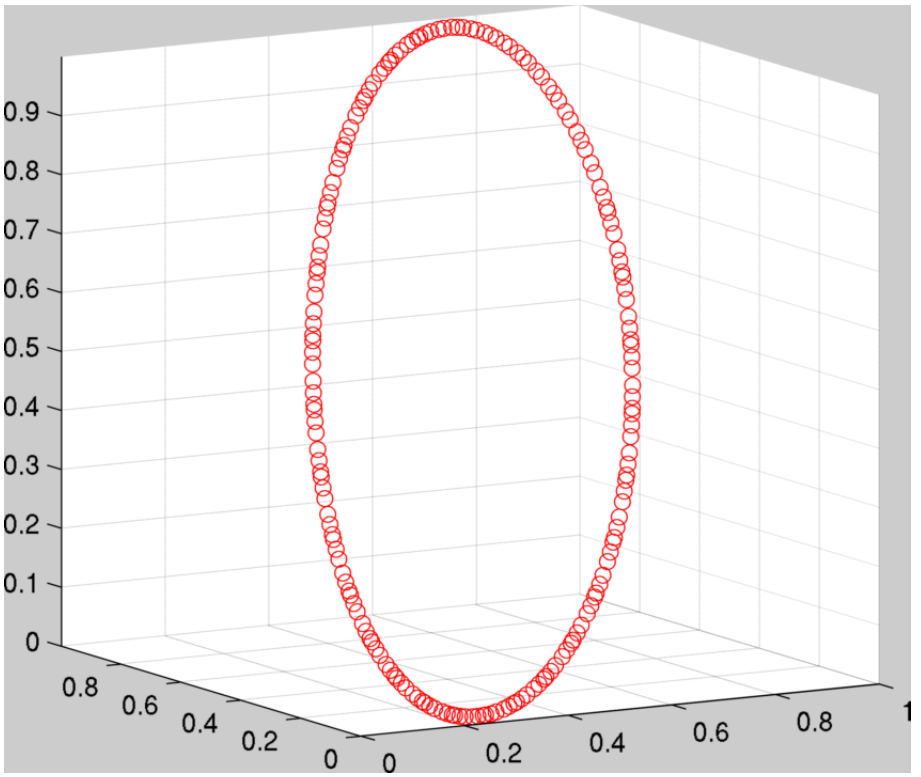
- “Typical” patch sizes = 5x5 ↔ 9x9 ↔ 15x15
- To compute filtered intensity at pixel “p”,
  don't compute weights for ALL other pixels
  - Too expensive
  - Instead, select a window around pixel “p”
  - “Typical” window sizes = 21x21 ↔ 101x101
- How to choose Gaussian standard deviation “h” ?
  - This is the important parameter
- Insights into the algorithm → see next ...

Patch-Based Filtering

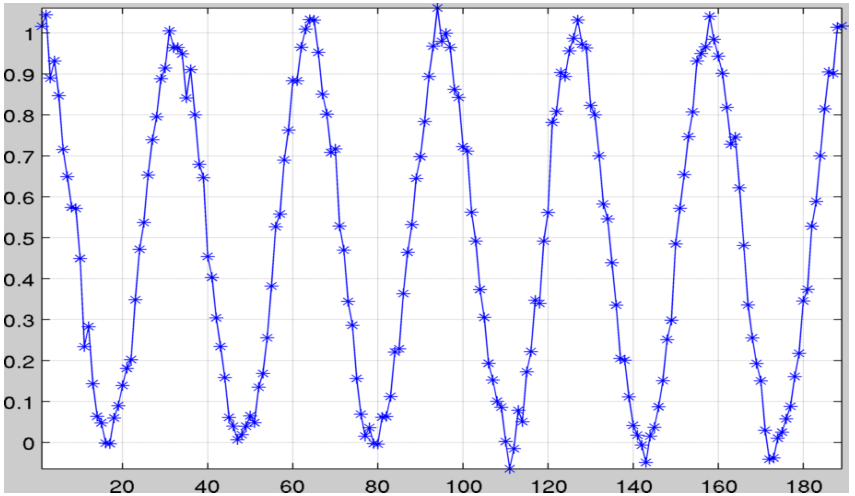


- Insights into patch distributions
  - Blue : digital image
  - Bottom left : patches with 2 pixels
  - Bottom right : patches with 3 pixels

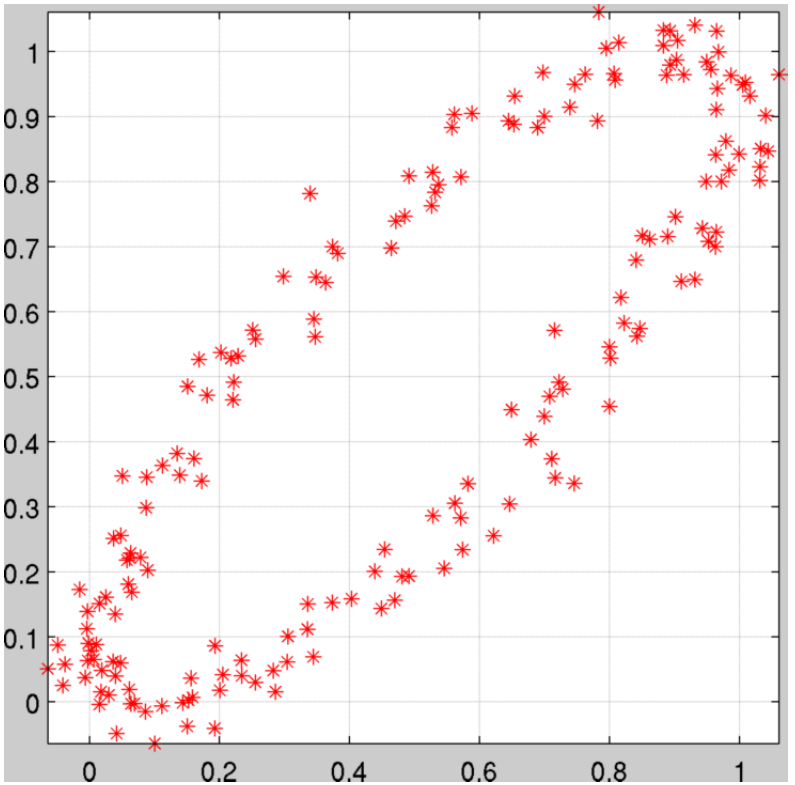


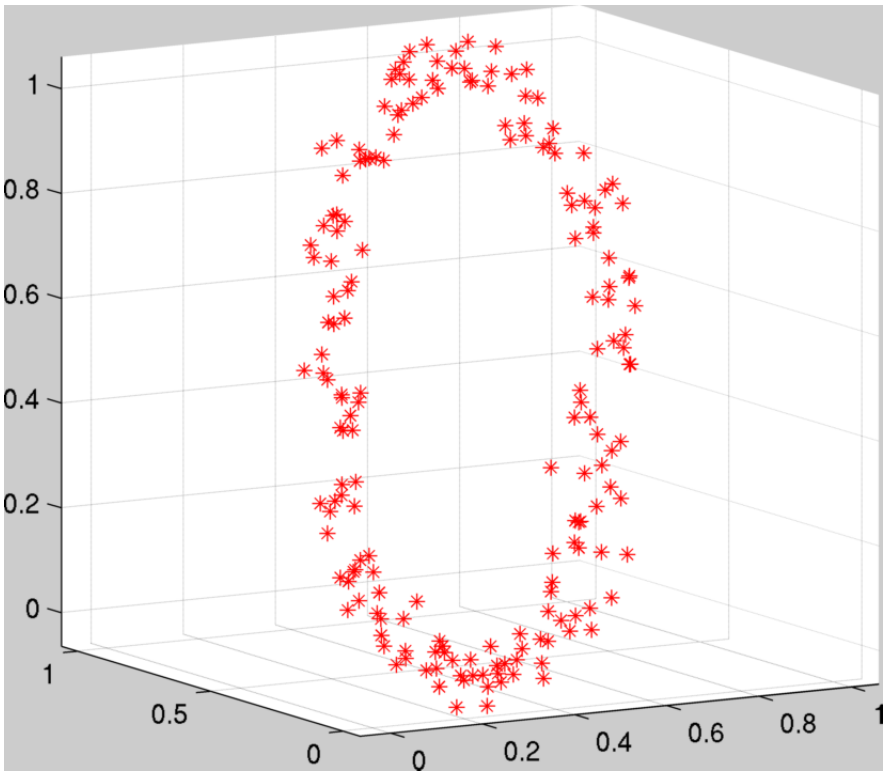


Patch-Based Filtering

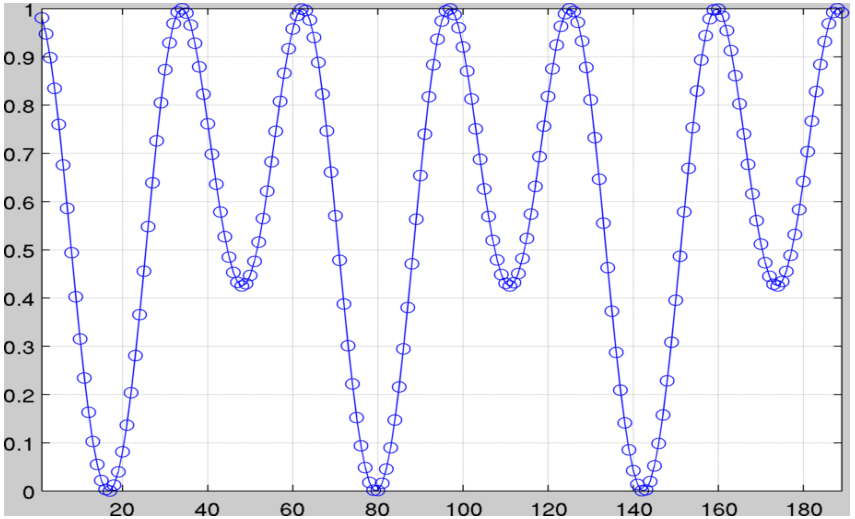


- Insights into patch distributions
  - Blue : digital image (noisy)
  - Bottom left : patches with 2 pixels
  - Bottom right : patches with 3 pixels

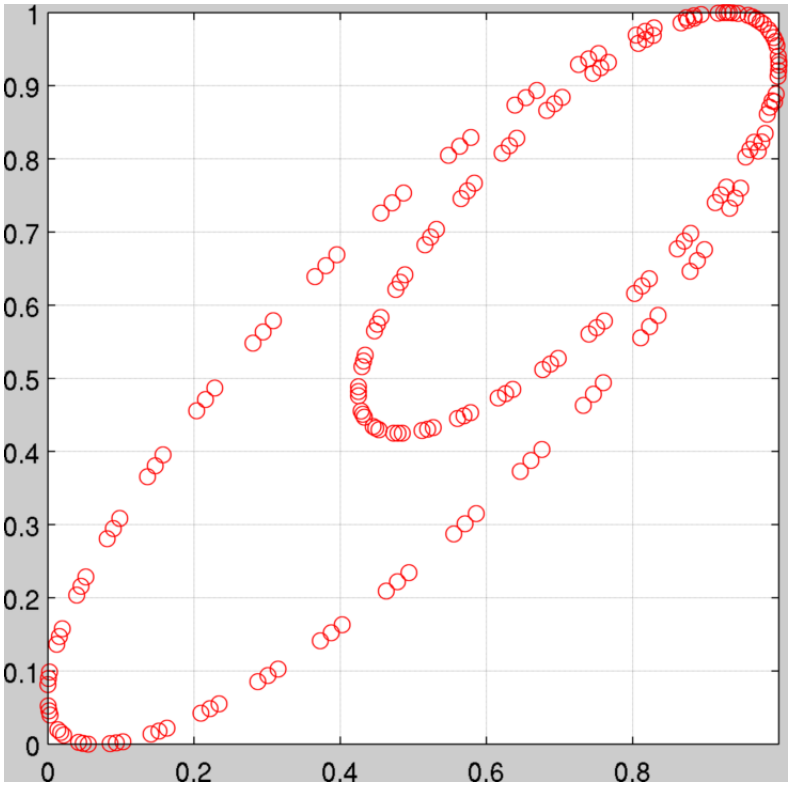


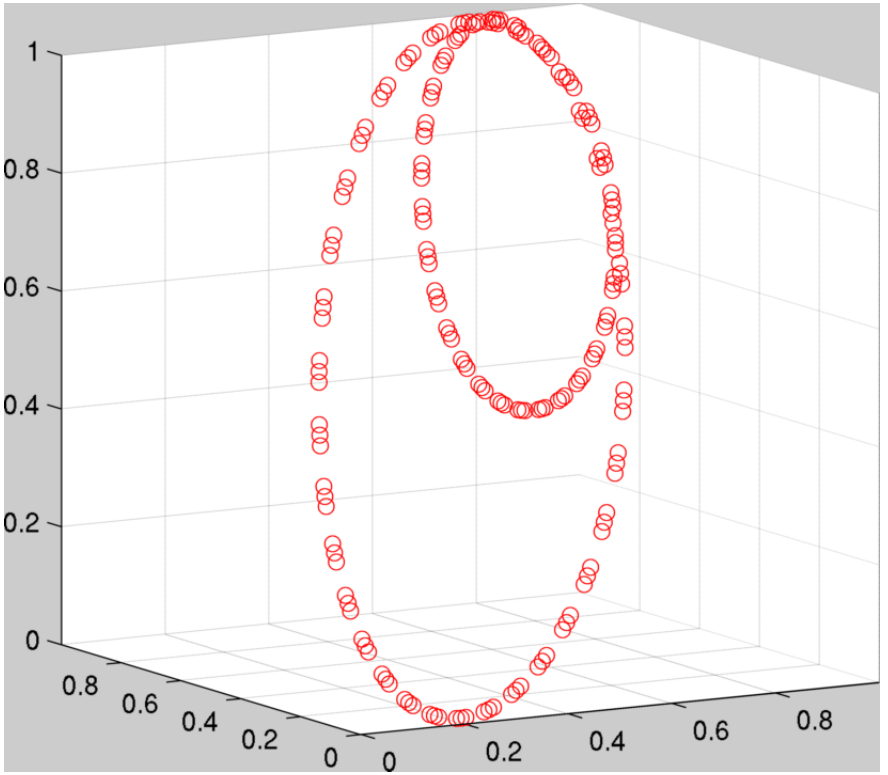


Patch-Based Filtering

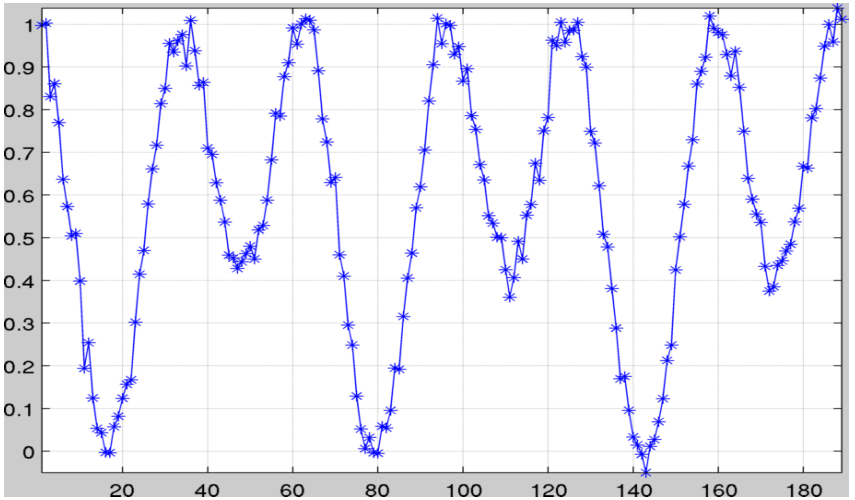


- Insights into patch distributions
  - Blue : digital image 2
  - Bottom left : patches with 2 pixels
  - Bottom right : patches with 3 pixels

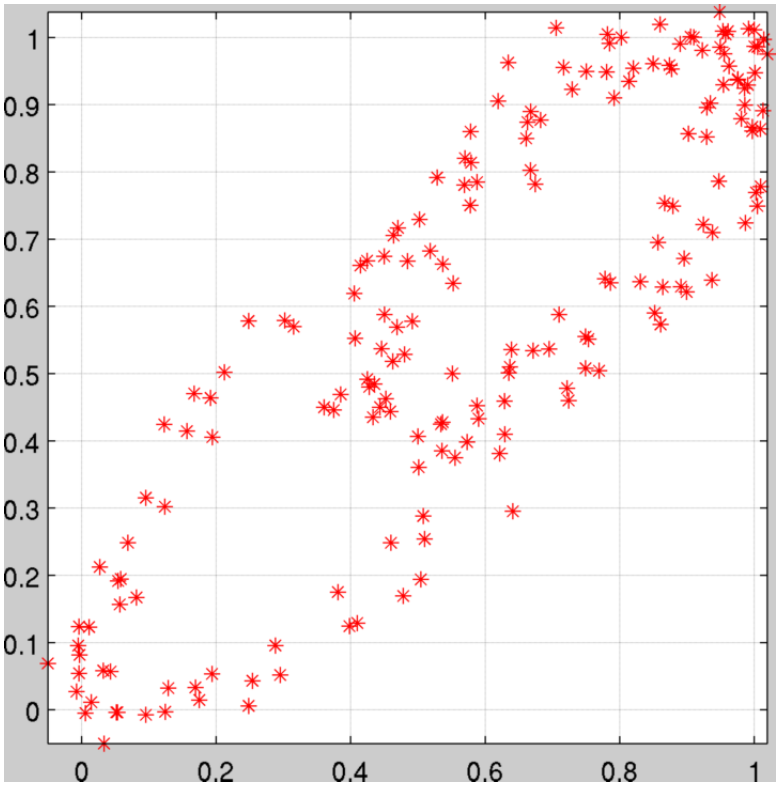


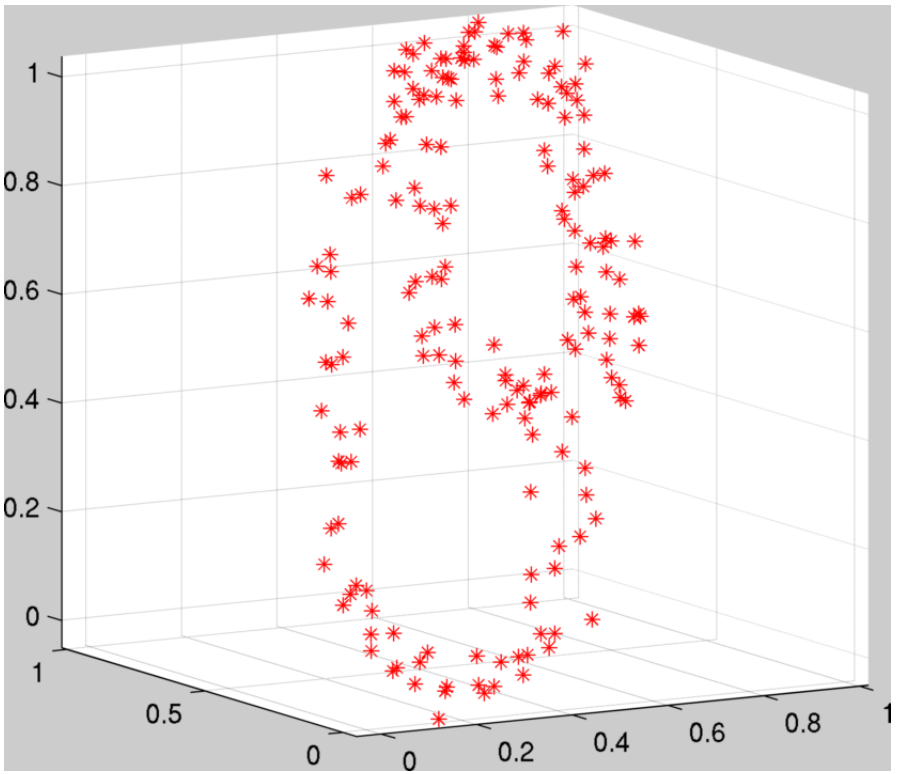


Patch-Based Filtering



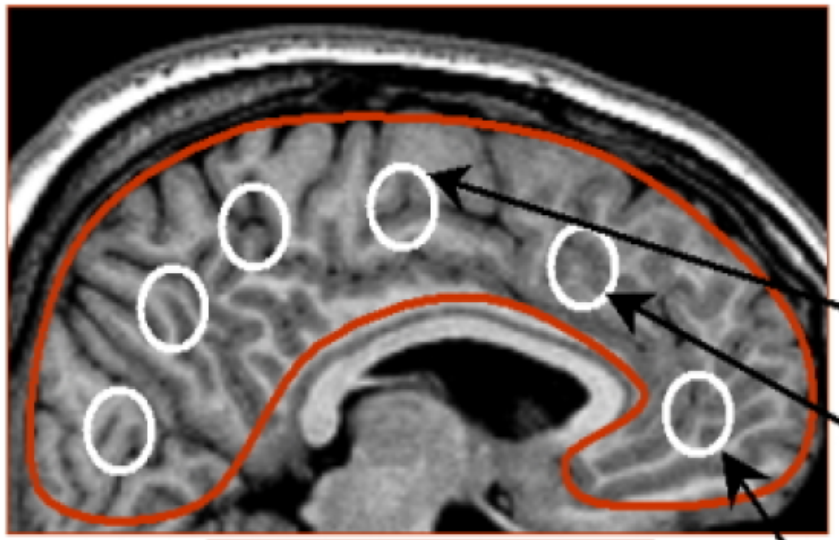
- Insights into patch distributions
  - Blue : digital image 2 (noisy)
  - Bottom left : patches with 2 pixels
  - Bottom right : patches with 3 pixels

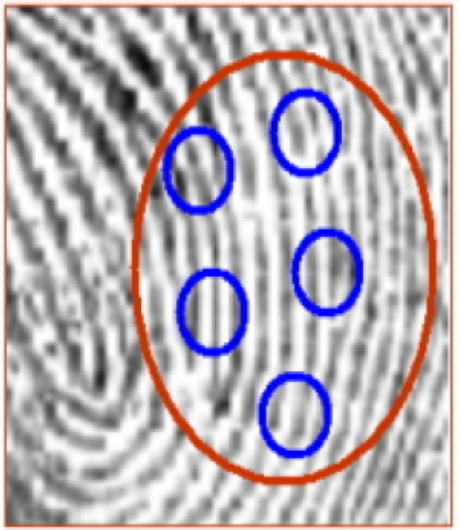


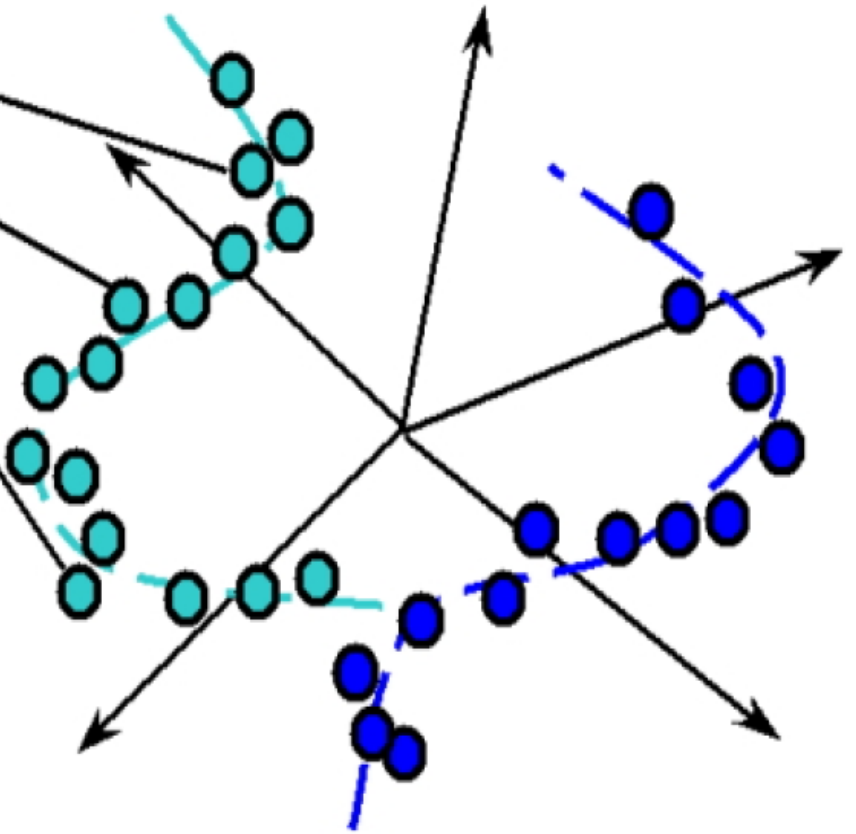


Patch-Based Filtering

- Insights into patch distributions
  - **Noise increases the scatter around the manifold**
  - **Update patches to reduce scatter → noise reduction**

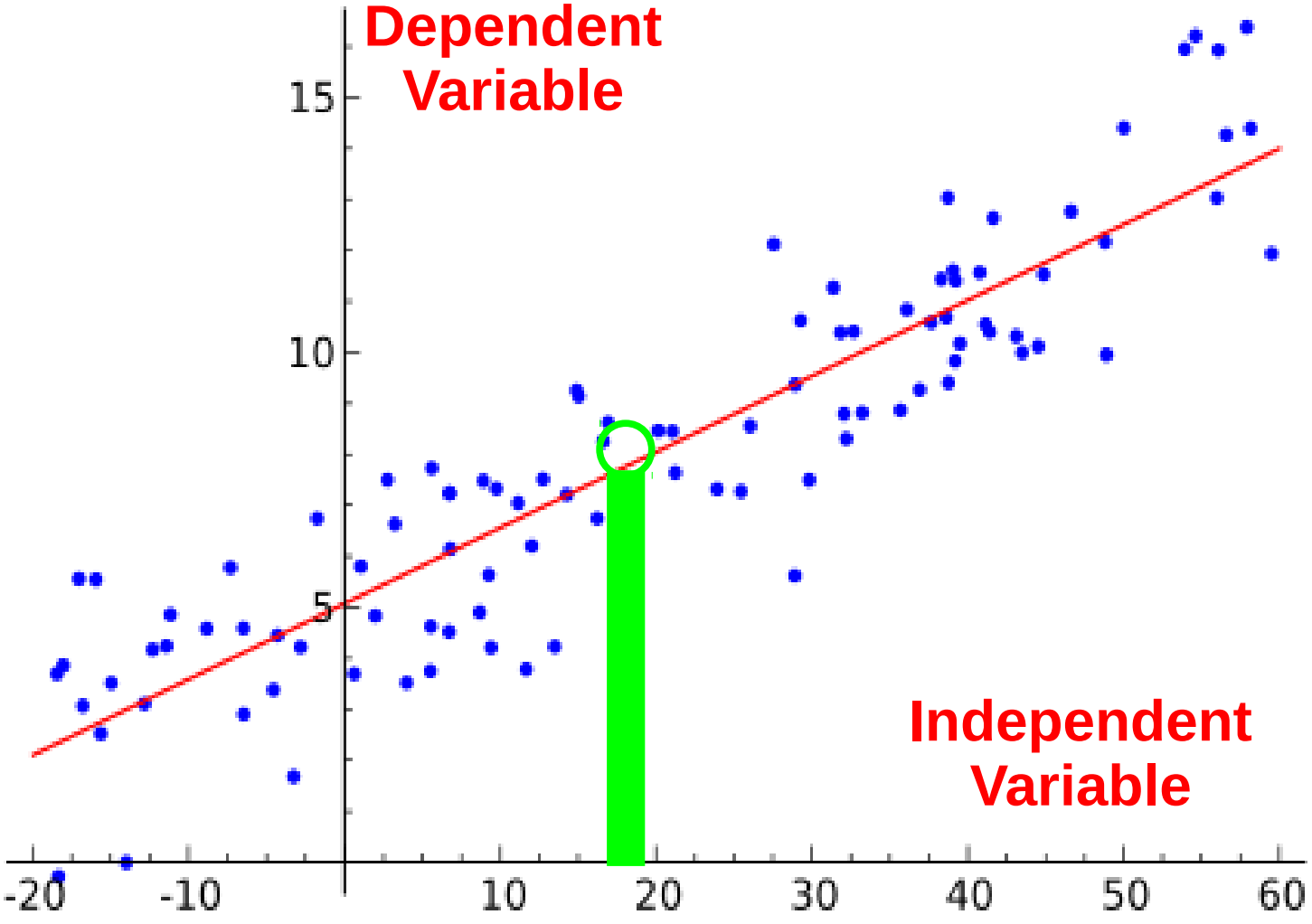






Patch-Based Filtering

- **Regression**
  - Model / estimate relationship between variables
  - Linear
  - **Given (xi,yi)**
    - **i = 1, ..., n**
  - **Given x0**
  - **Find y0**



Patch-Based Filtering

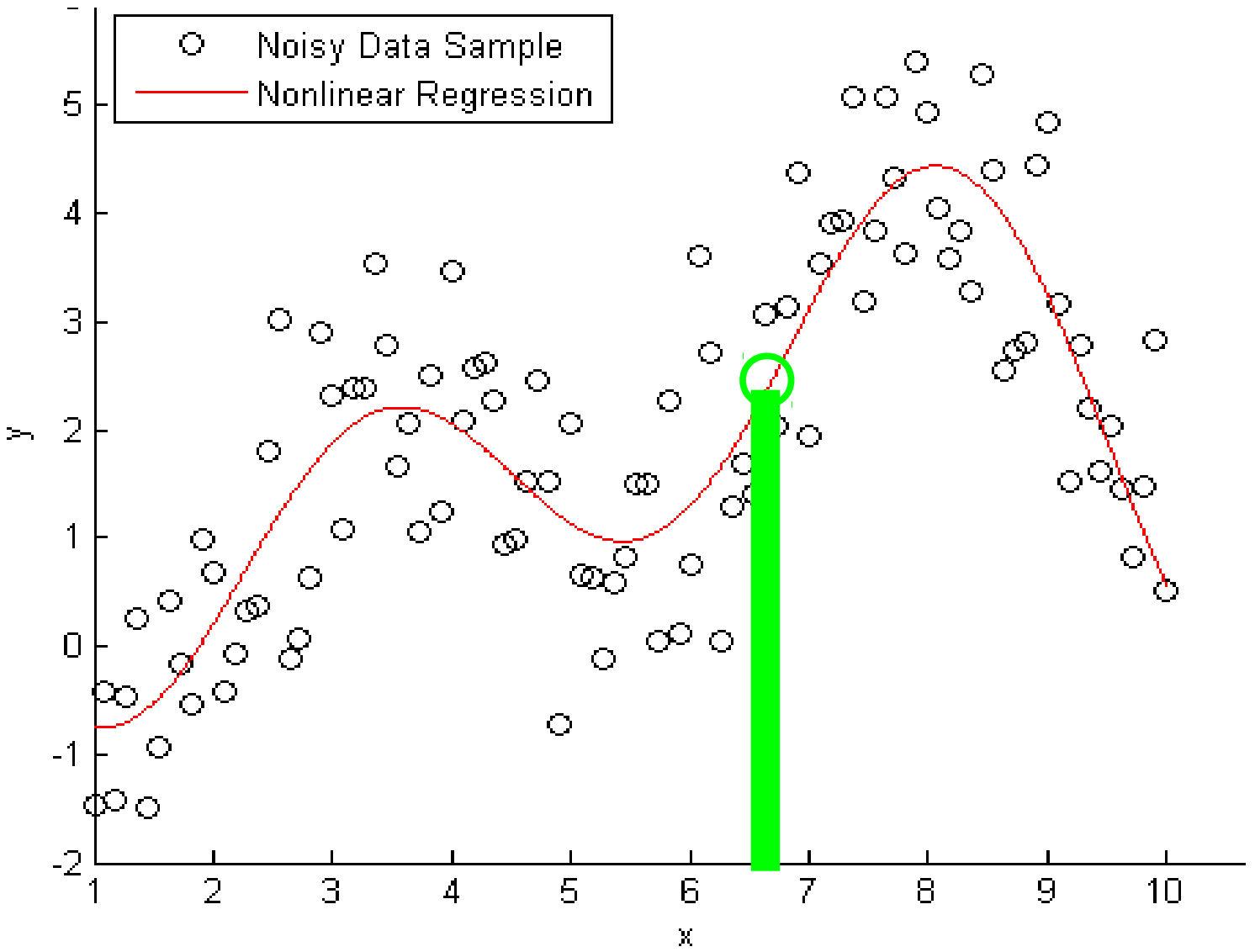
- Regression
  - Think of **filtering as a regression problem** !
  - At pixel “p”

    (1) Independent variable = intensities of neighbors of “p”

    (2) Dependent variable = intensity of “p”
  - This relationship is nonlinear !

Patch-Based Filtering

- Regression
  - **Nonlinear** regression
  - **Given (xi,yi)**
    - **i = 1, ..., n**
  - **Given x0**
  - **Find y0**



Patch-Based Filtering

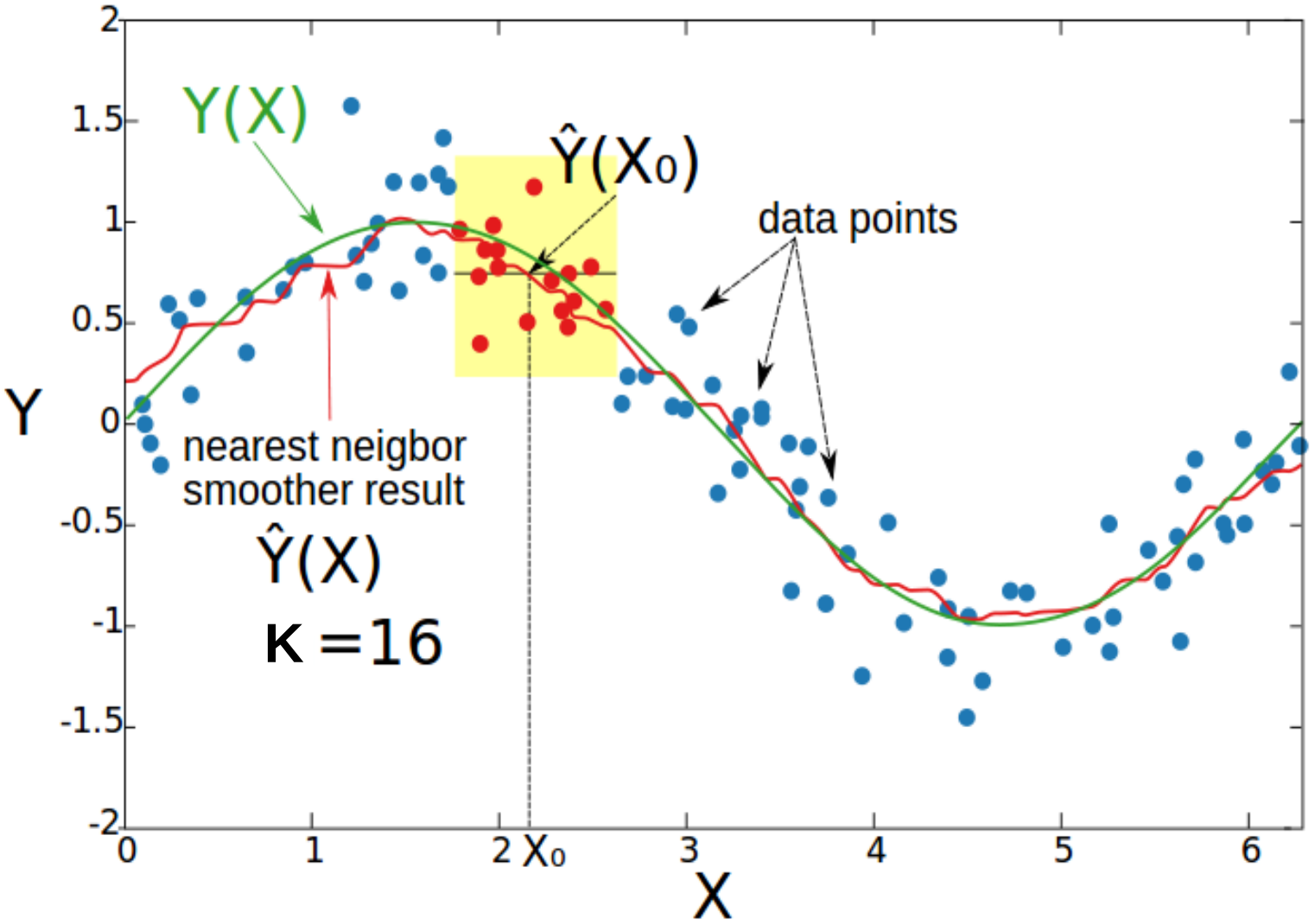
- Nonlinear regression
  - **Given (xi,yi), i=1,...,n. Given x0. Find y0.**
  - Principle 1
    - If we have a very large number of **pairs (xj,yj)**, where **xj = x0** (i.e., X coordinates match)
    - Then, we'd like **y0 → average of yj**

Patch-Based Filtering

- Nonlinear regression
  - **Given (xi,yi), i=1,...,n. Given x0. Find y0.**
  - Principle 1
    - If we have a very large number of **pairs (xj,yj)**, where **xj = x0** (i.e., X coordinates match)
    - Then, we'd like **y0 → average of yj**
  - Principle 2
    - In practice, we may have NO pair (xj,yj), where xj = x0
    - But, we may have pairs (xk,yk), where xk is close to x0
    - Then, we want y0 → **weighted average of yk**, where **weights are decreasing (non-increasing) with increasing distance || x0 – xk ||**

Patch-Based Filtering

- Nonlinear regression
  - **K Nearest Neighbor**
    - Given x0
    - Find K nearest neighbors {xj}
    - Average associated values {yj}
    - Weights = 0 or 1/K (binary)



Patch-Based Filtering

- Nonlinear regression
  - **Given (xi,yi), i=1,...,n. Given x0. Find y0.**
  - **Kernel regression**

$$\widehat{m}_h(x) = \frac{\sum_{i=1}^{n} K_h(x - x_i) y_i}{\sum_{i=1}^{n} K_h(x - x_i)}$$

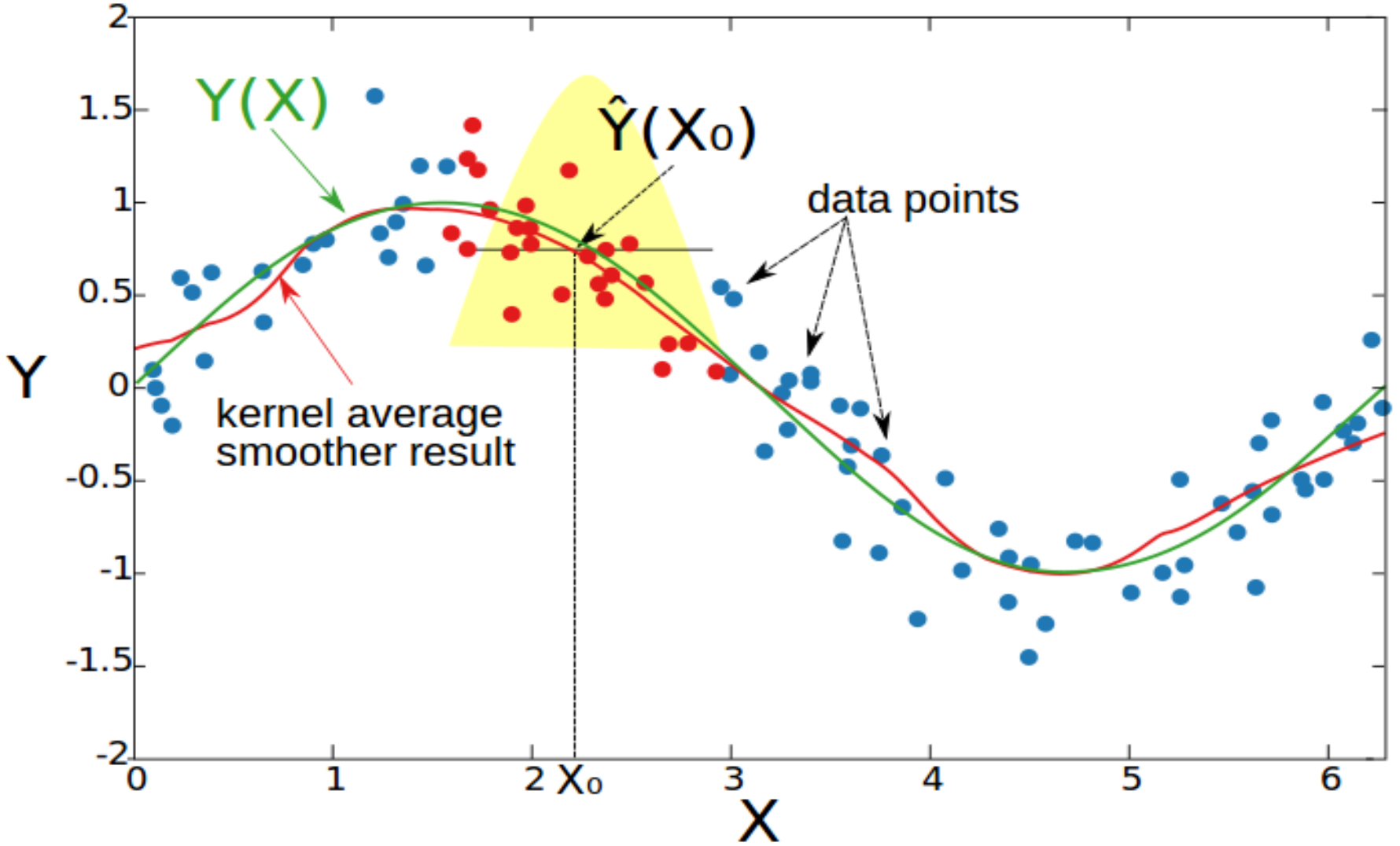
where K(.) is a kernel with bandwidth h

e.g., K(.) is a Gaussian, where h = standard deviation

  - Weights
    - Between 0 and 1 (NOT binary)
    - Sum to 1

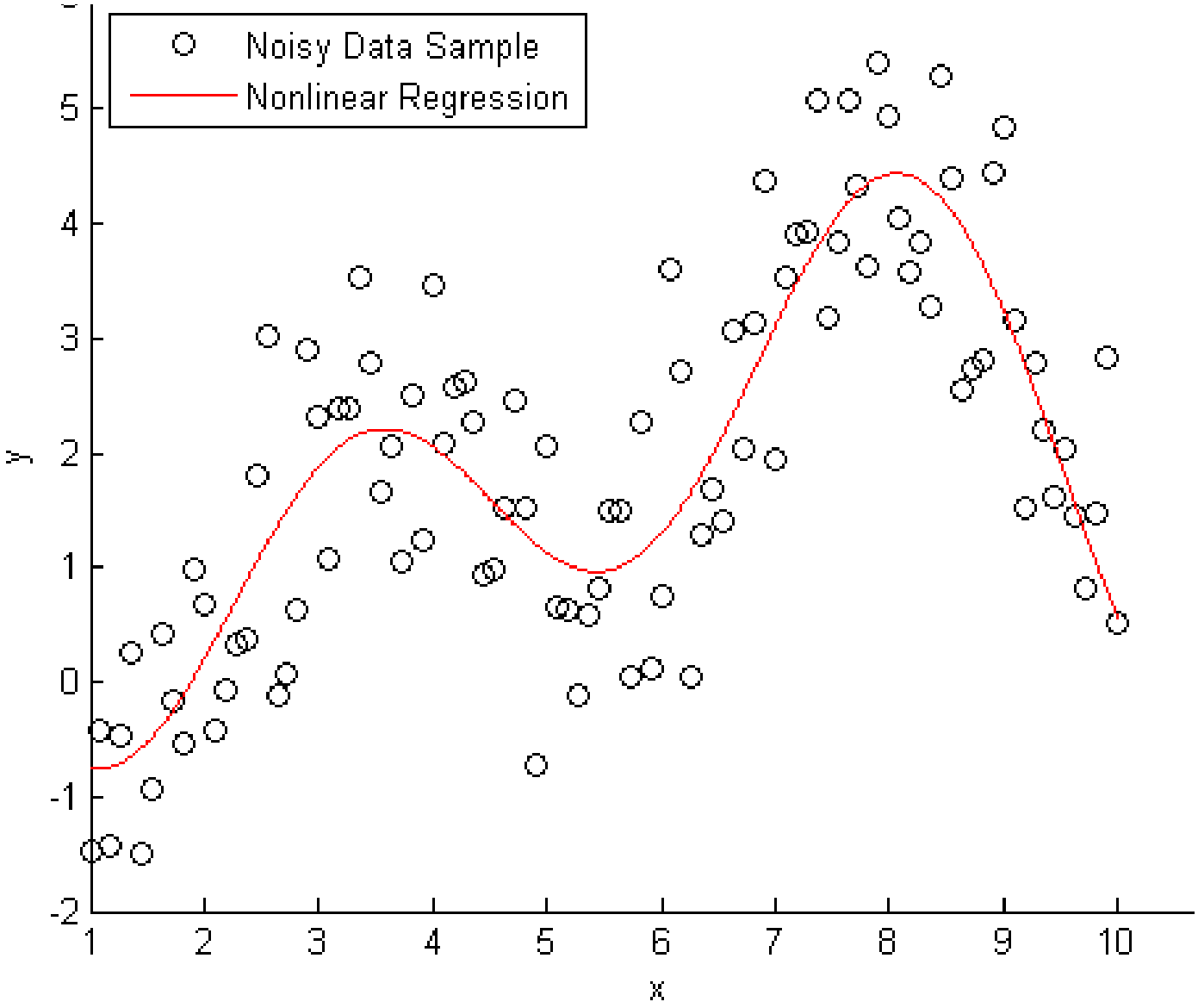
Patch-Based Filtering

- Nonlinear regression
  - Kernel regression → result is smoother than kNN



Patch-Based Filtering

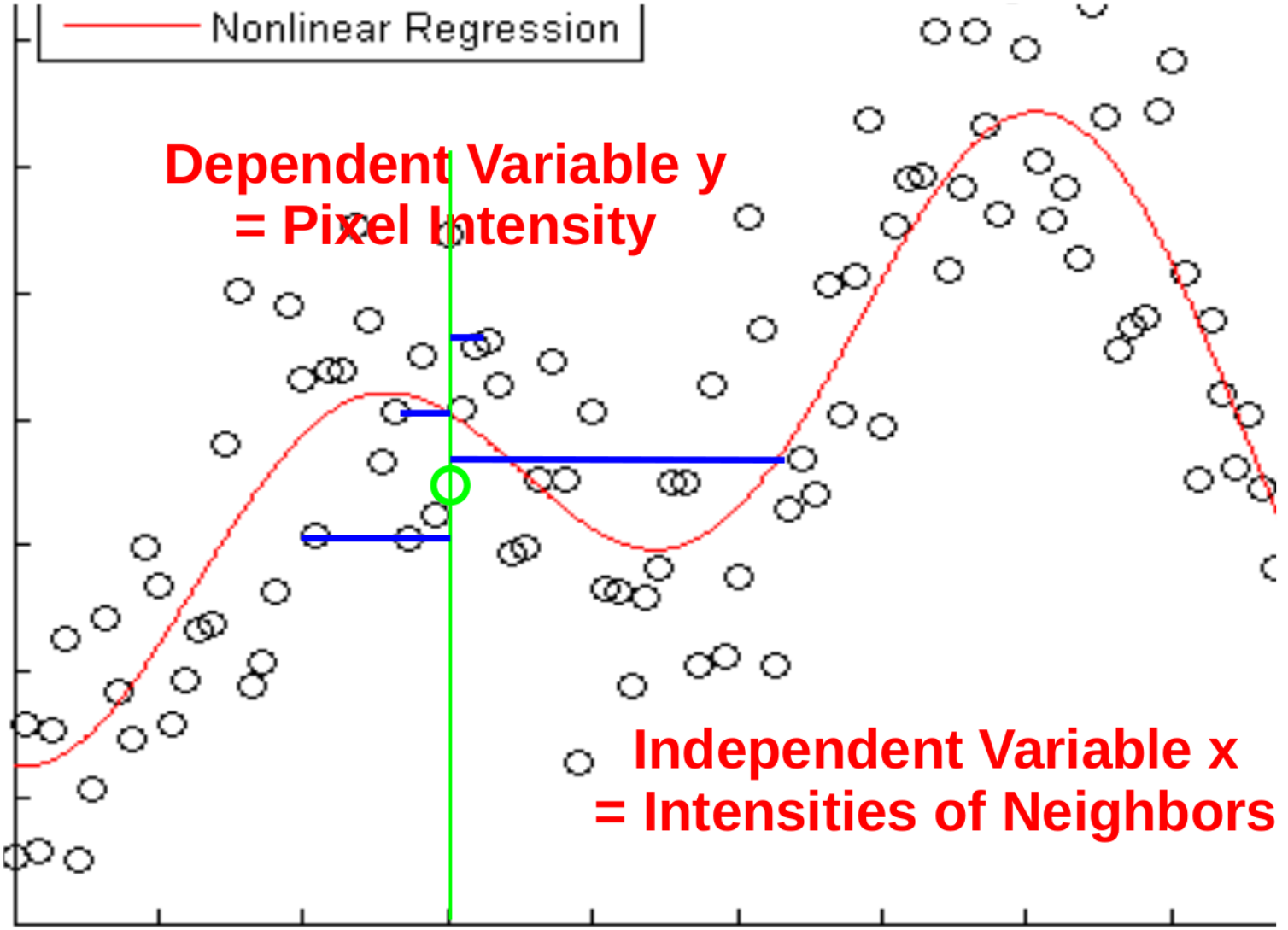
- Nonlinear regression
  - How to select standard deviation h ?
  - What if h → ∞ ?
  - What if h → 0 ?



Patch-Based Filter

$$\widehat{m}_h(x) = \frac{\sum_{i=1}^{n} K_h(x - x_i) y_i}{\sum_{i=1}^{n} K_h(x - x_i)}$$

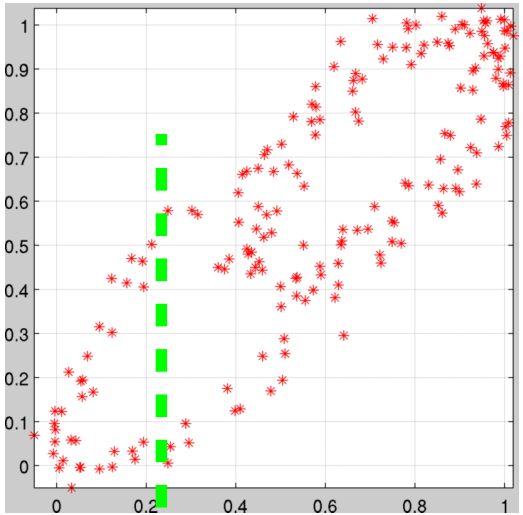
- **Image Filtering** + Nonlinea
  - Weights = **distances between *neighborhoods***
    - *Neighborhood* **doesn't** include center pixel !
  - How to update green patch to reduce scatter ?

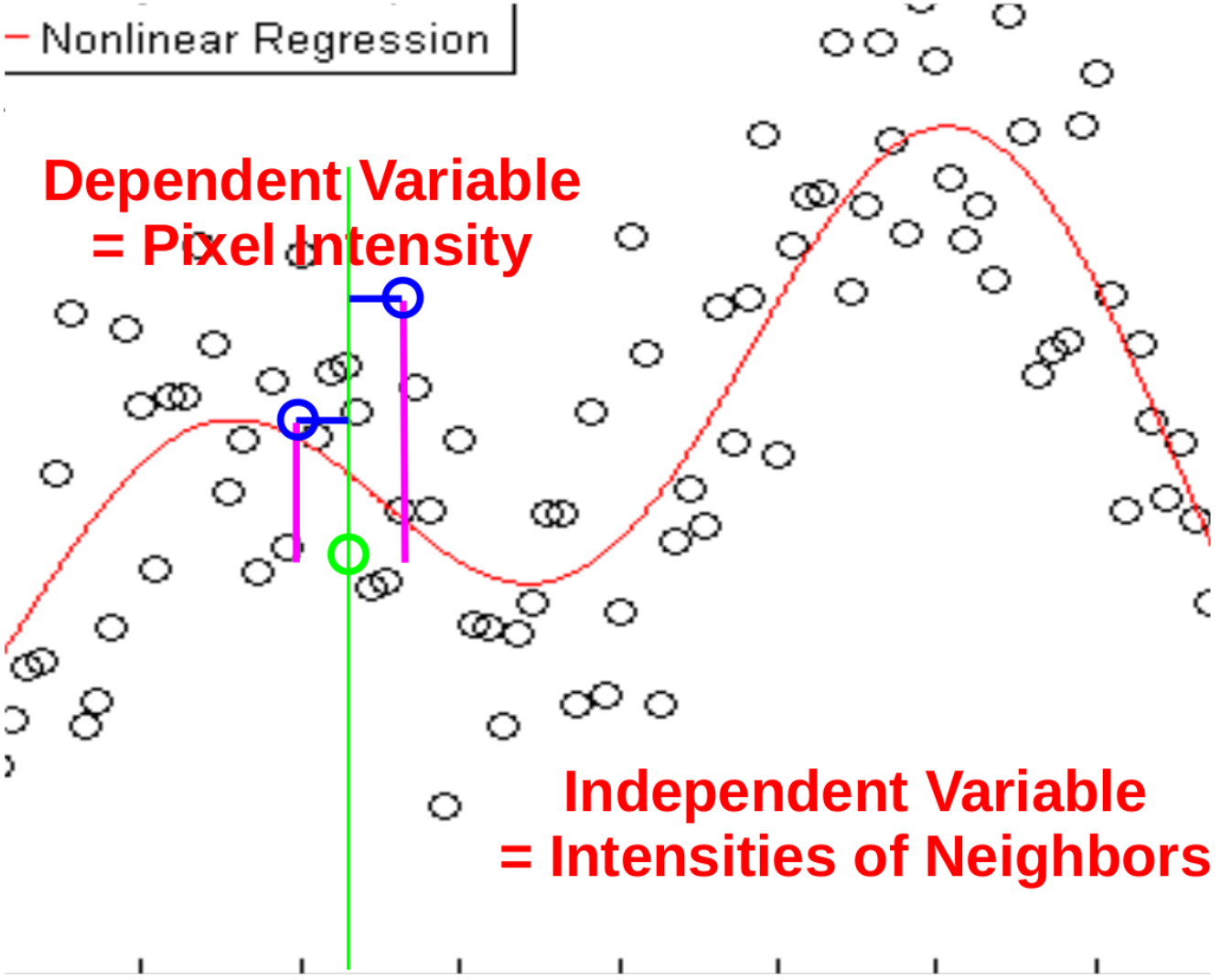


Patch-Based Filter

$$\widehat{m}_h(x) = \frac{\sum_{i=1}^{n} K_h(x - x_i) y_i}{\sum_{i=1}^{n} K_h(x - x_i)}$$

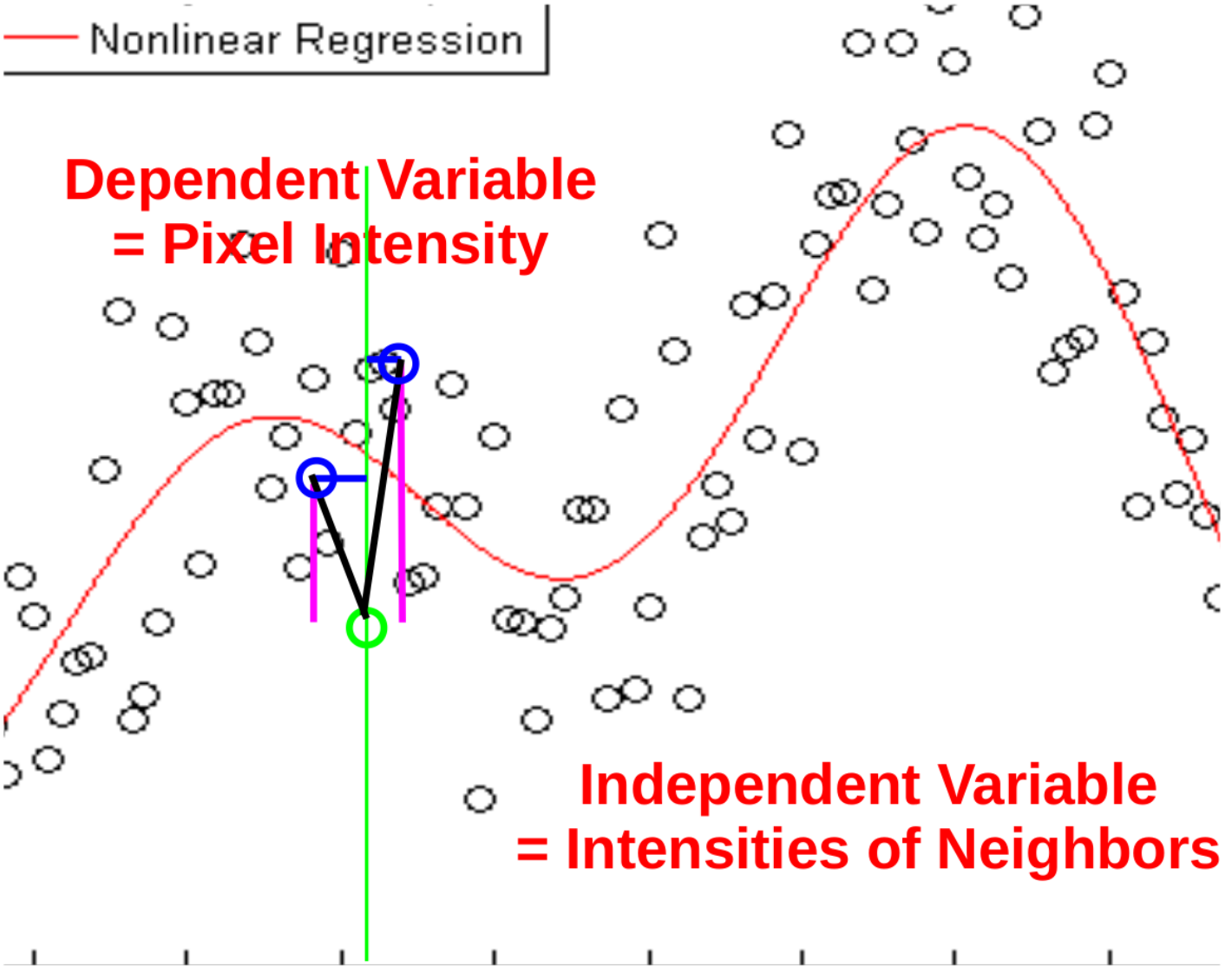
- Image Filtering + Nonlinear
  - How to update green patch to reduce scatter ?
  - **Why should blue neighborhoods get same weight ?**
    - One **blue** patch has **center pixel** intensity that is **closer** to center pixel intensity in green patch





Patch-Based Filter

- Image Filtering + Nonlinear regression
  - How to update green patch to reduce scatter ?
  - **Use distances between patches**
    - Patch (x,y) **includes center** pixel



Patch-Based Filtering

- Image Filtering + Nonlinear regression

  (1) Updates of adjacent pixels are interlinked

  - Updating center pixel of one patch updates neighborhoods of adjacent patches
    - Different neighborhood implies different update

  (2) Initial neighborhoods are noisy

  - Implies “sub-optimal” update

- How to reduce artifacts & improve reliability ?

$$\sum_{j \in \Lambda} \omega_h(j,k) f(j)$$

  - Take a **small step** towards weighted average
    - Update f(k) a fraction of the way to f*(k)
  - Perform **multiples passes** over the image (i.e., update image over >1 iterations)

Patch-Based Filtering

- How to select patches ?
  - For filtering pixel “p”
    - Select patches around all pixels “q” in the image
      - Computationally expensive. Quadratic complexity.
    - Select patches around pixels “q” within a window around pixel “p”
      - Linear complexity
      - Rectangular window or circular window
      - NO patches selected outside window
    - Select patches around pixels “q” randomly distributed according to a (spatial isotropic) Gaussian centered at pixel “p”
      - Fix number of patches
      - Linear complexity
      - Most patches selected close to “p”
      - Some patches selected far from “p”

Patch-Based Filtering

- Patch shape and Boundary conditions
  - Rotational invariance
  - Cropped / weighted patch at pixel “p” → distances between cropped / weighted patches at all other “q”

(a) (b) (c)

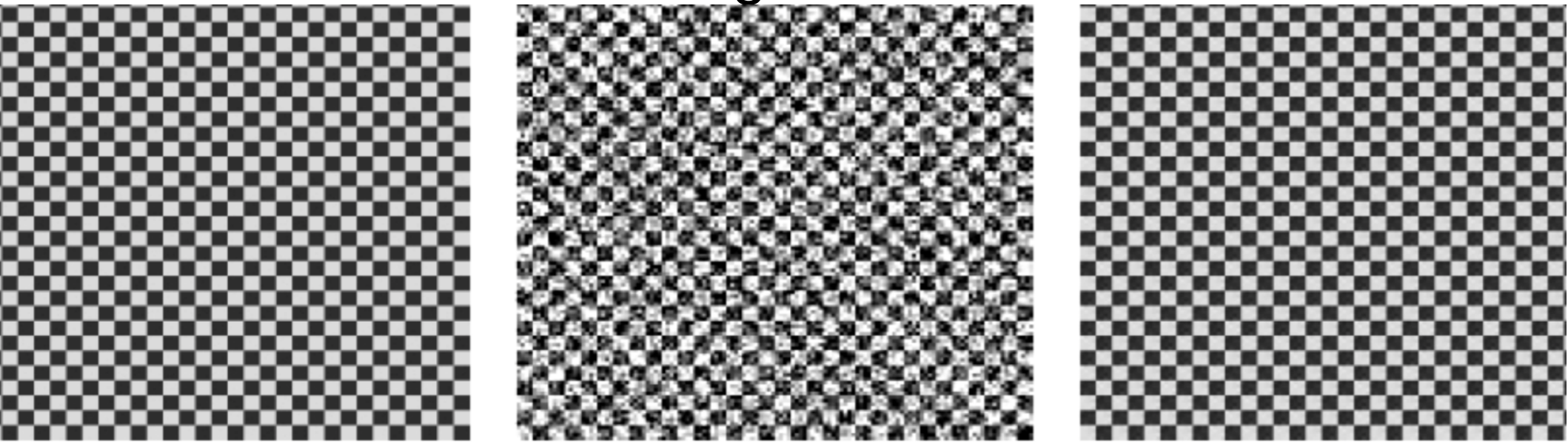
Figure 3.3. Neighborhood shapes. (a) Preserving rotational invariance via a neighborhood mask consisting of a flat central circular plateau with cubic splines on the sides. (b) The discrete sampling of the mask (black $\equiv$ 1, white $\equiv$ 0) for a 9×9 pixels neighborhood. (c) Anisotropic neighborhoods at boundaries.

Patch-Based Filtering

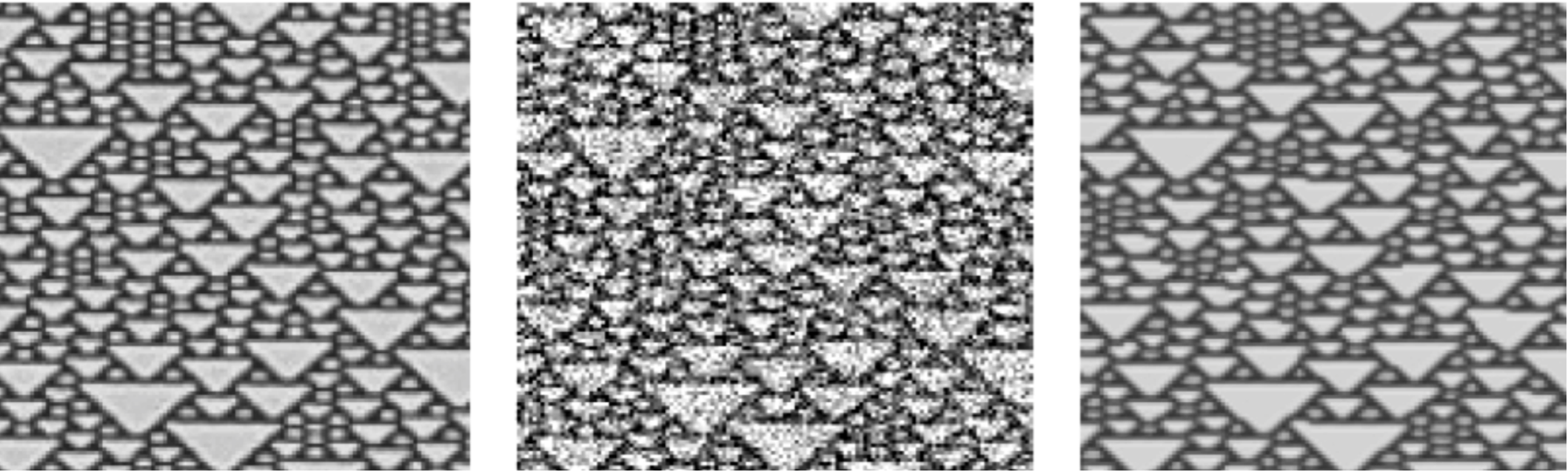
- Computational cost
  - Image M x N
  - Patch P x Q
  - Window for searching patches R x S
  - Number of operations = MN PQ RS
- There are ways to make this fast, while preserving accuracy
  - Non-trivial

Patch-Based Filtering

- Results on textured images

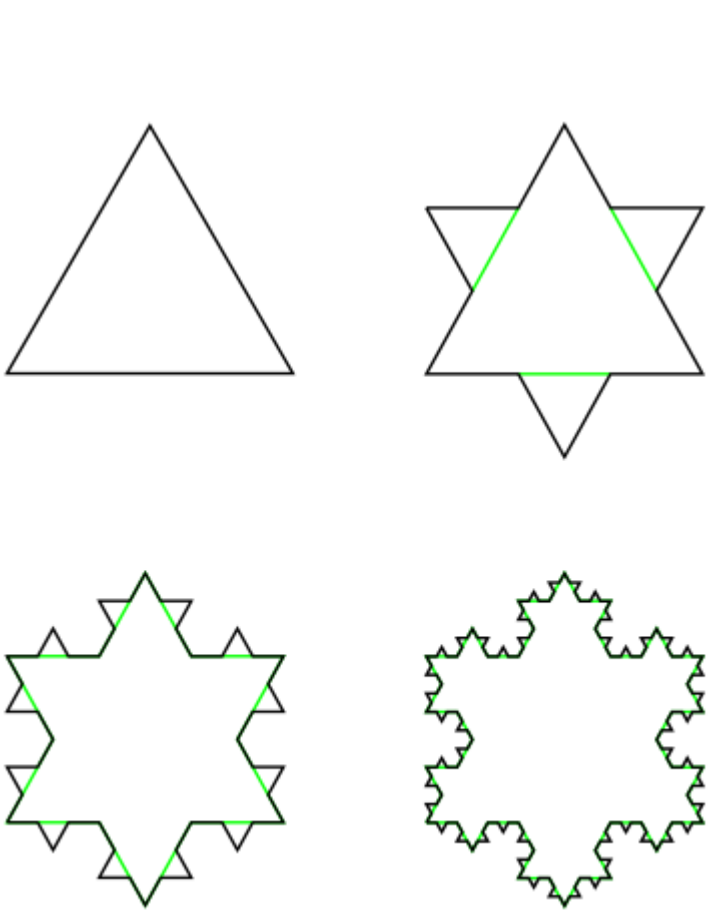


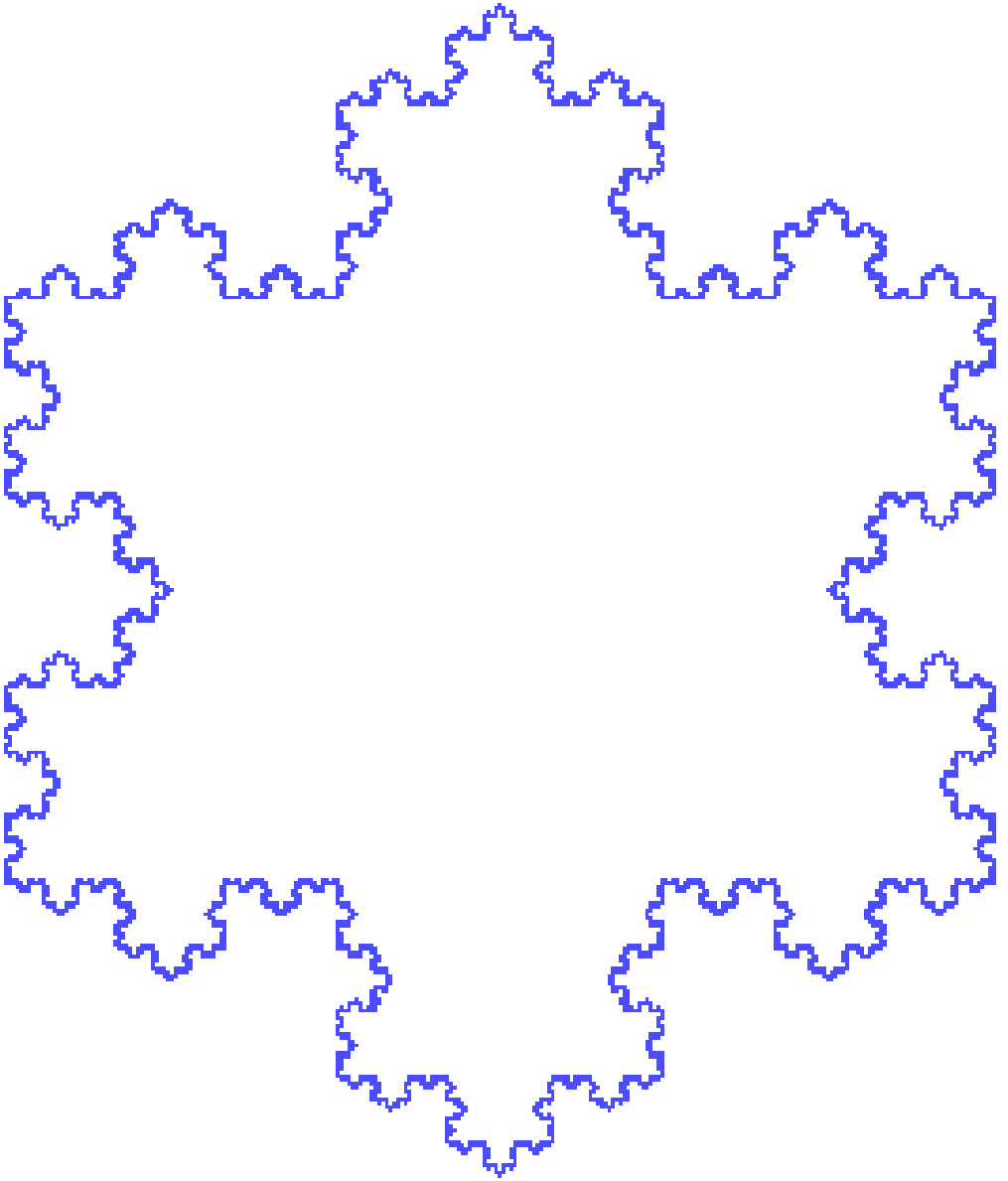
(a) (b) (c)



Patch-Based Filtering

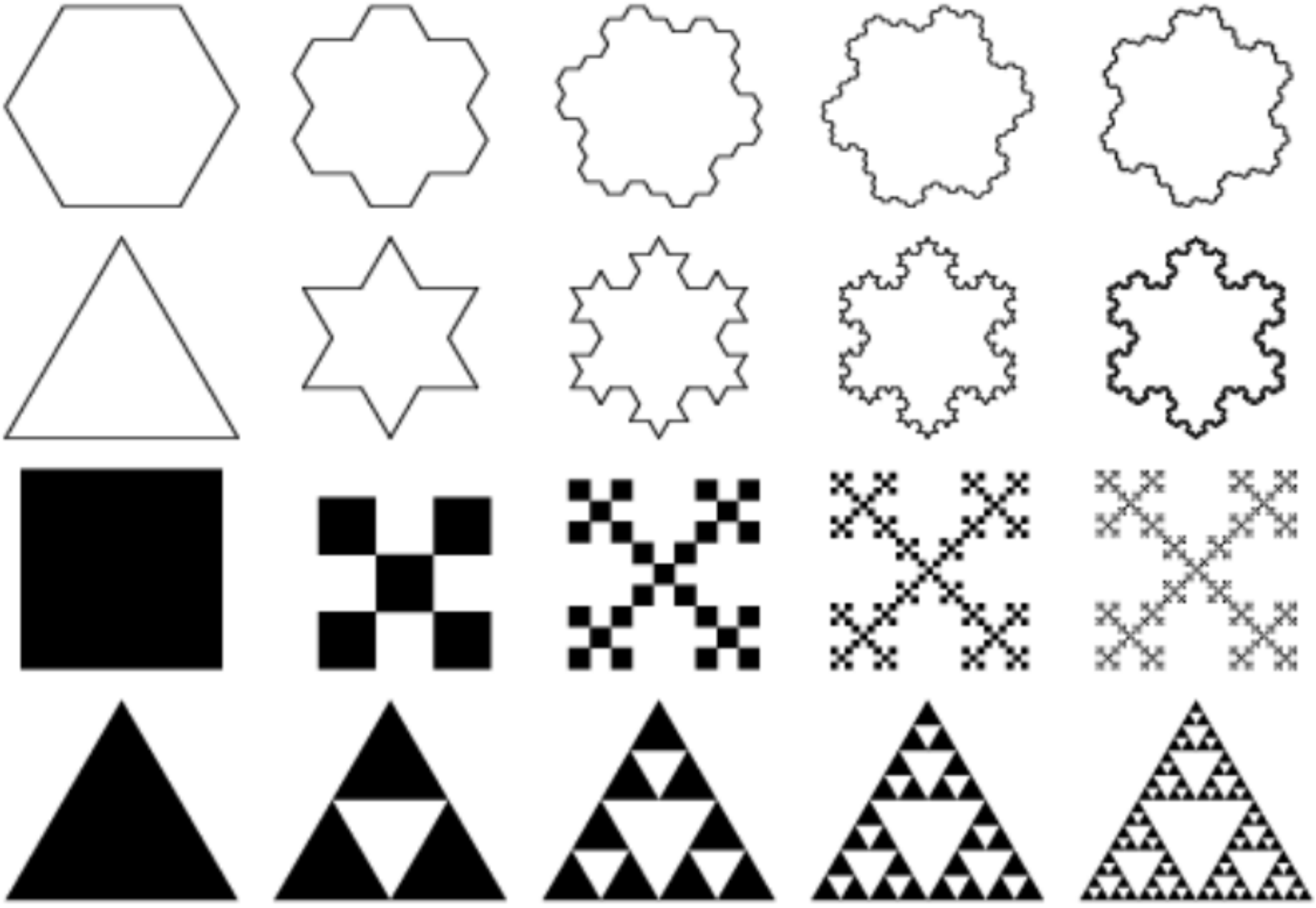
- Fractal = repeating pattern (texture) at all scales
  - Koch snowflake





Patch-Based Filtering

- Fractal = repeating pattern (texture) at all scales

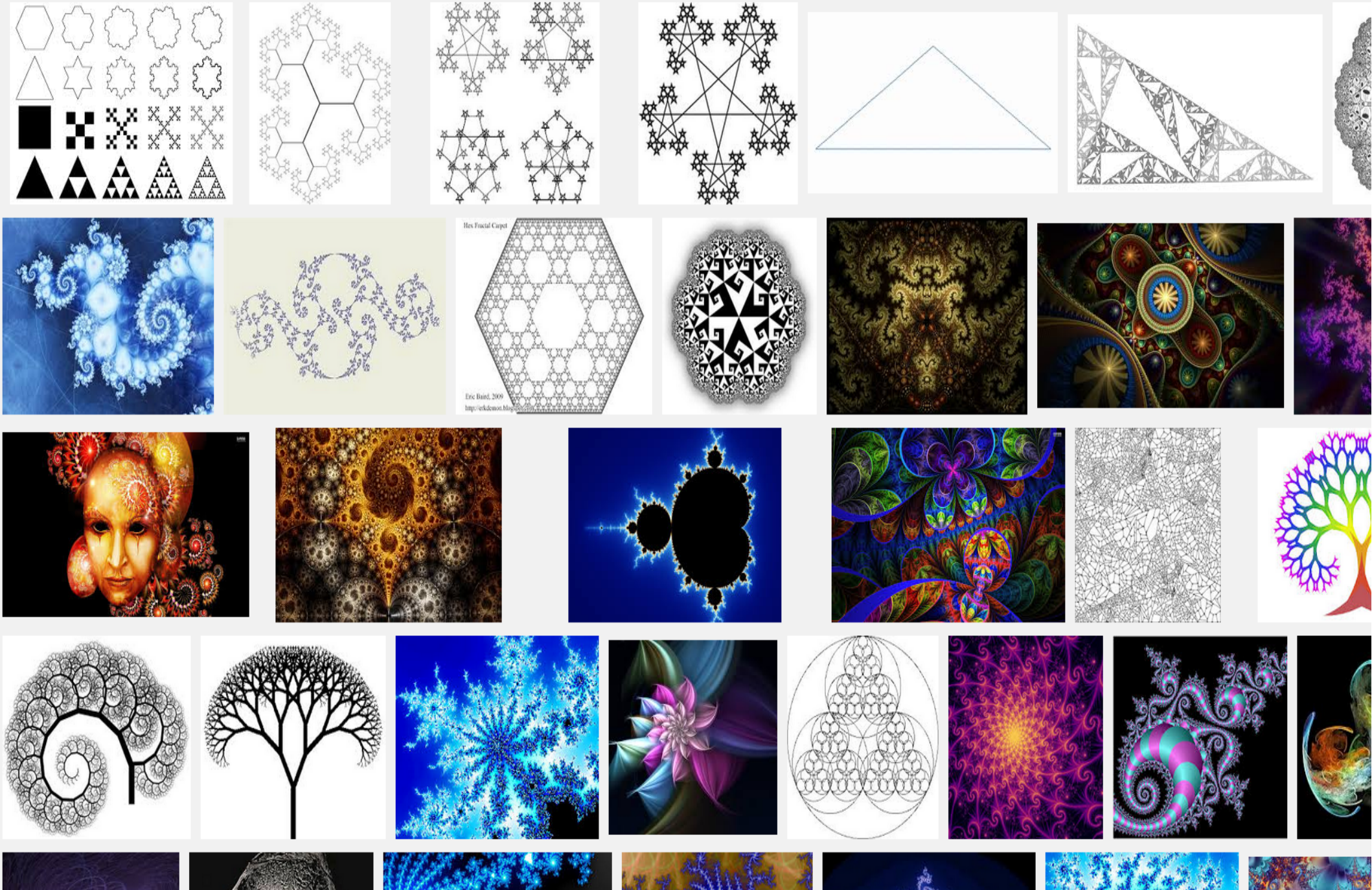


Google fractal

suyash

Web Images Videos Books News More Search tools

SafeSearc



Patch-Based Filtering

- Fractals in nature (snowflakes)

Patch-Based Filtering

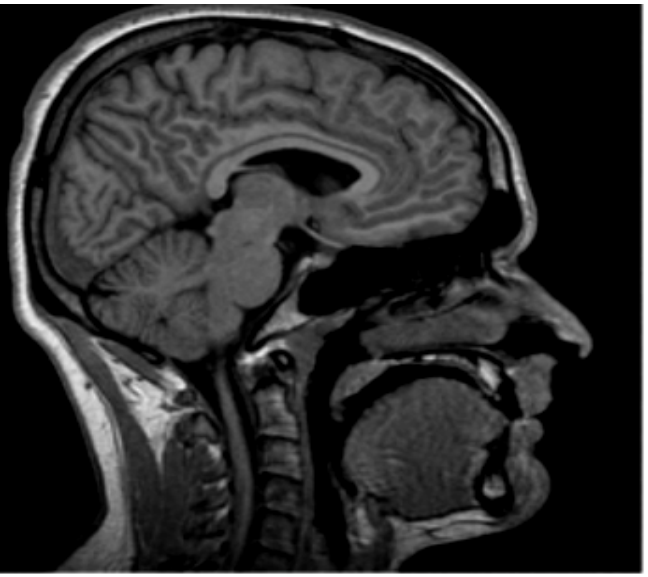
- Fractals in nature
  - High-voltage breakdown within 4″ block of acrylic
  - Lightning

Patch-Based Filtering

- Fractals in nature (Romanesque cauliflower)

Patch-Based Filtering

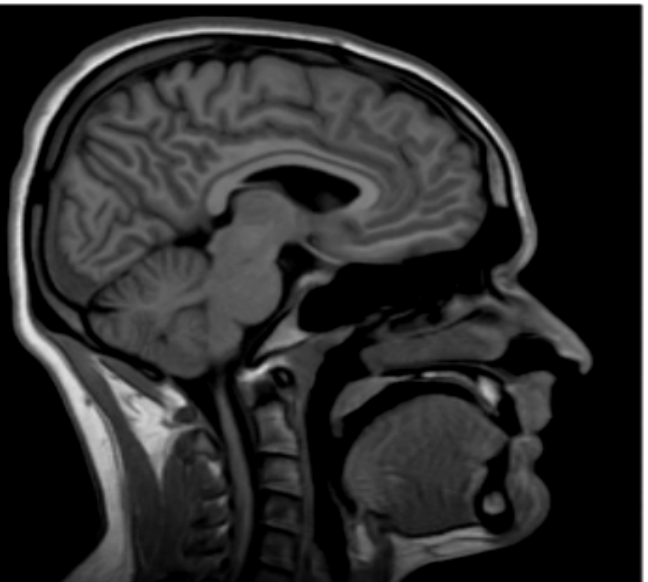
- Results on textured images



(a2)

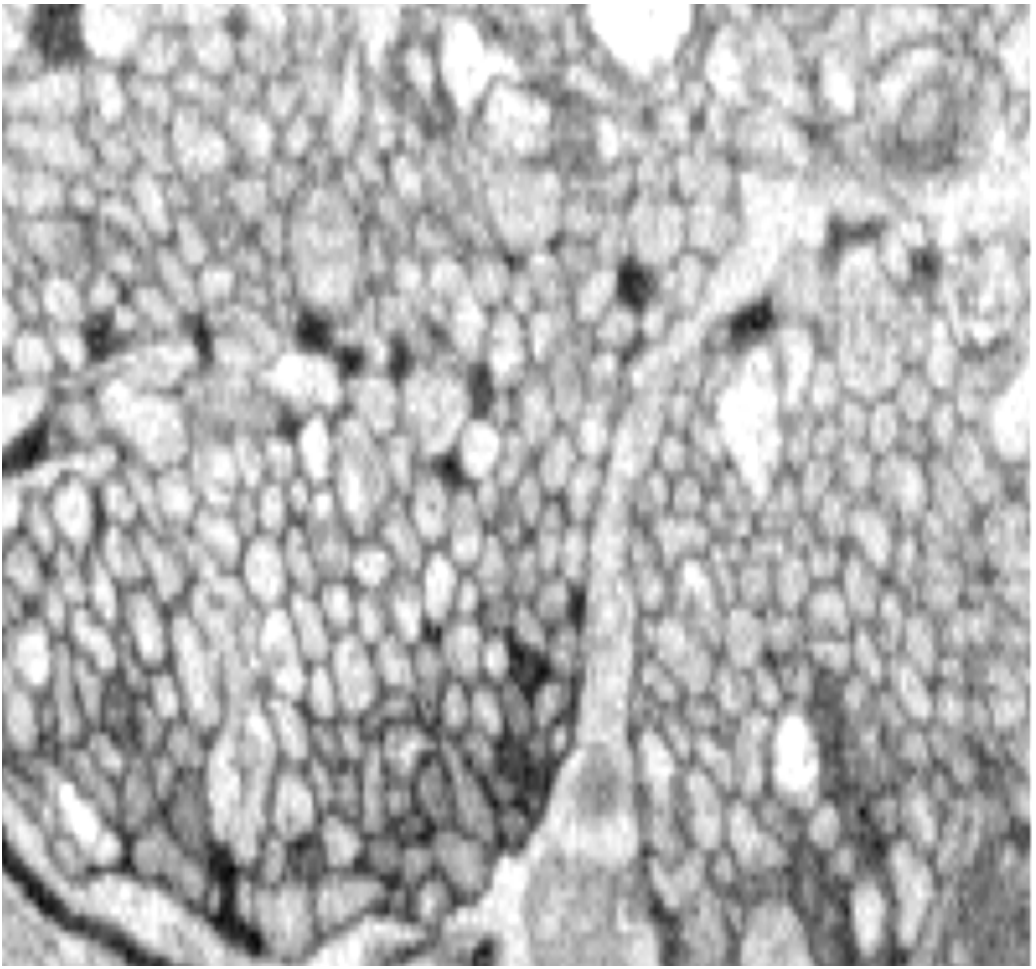
(c2)

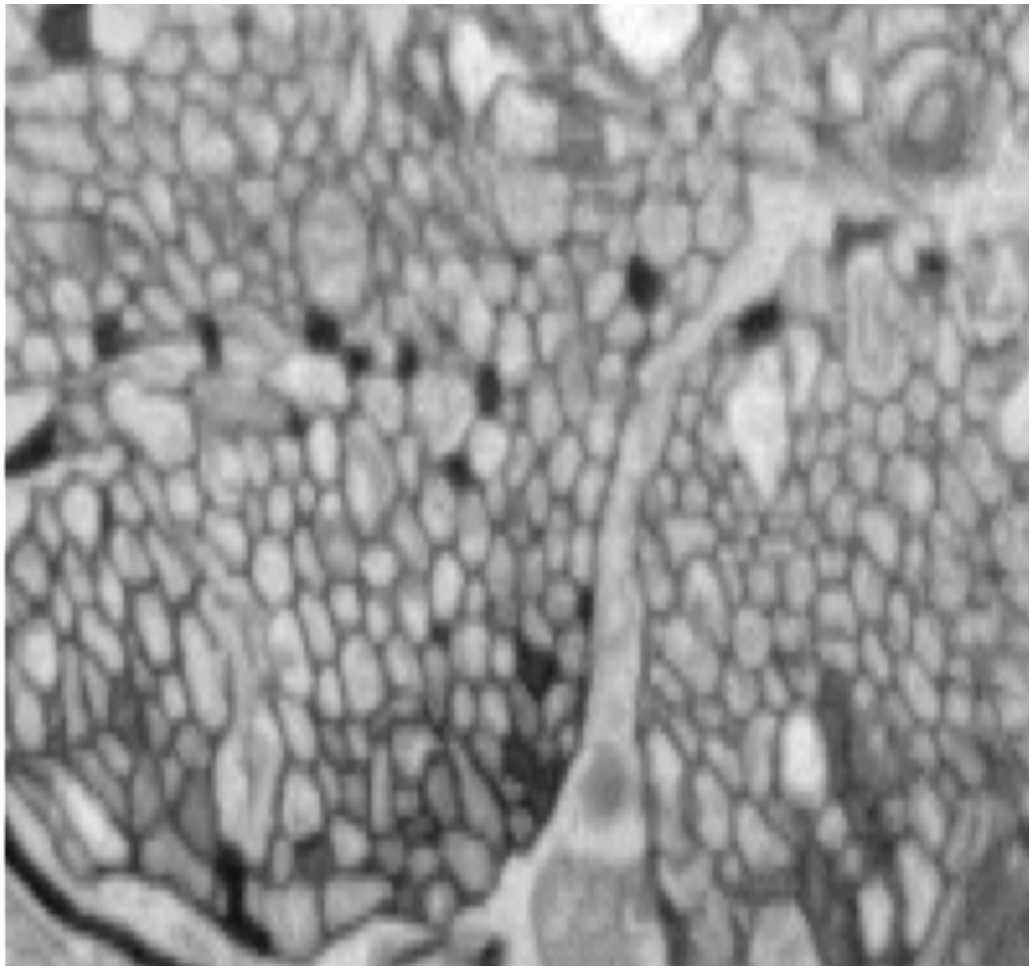
(e2)



Patch-Based Filtering

- Results on textured images

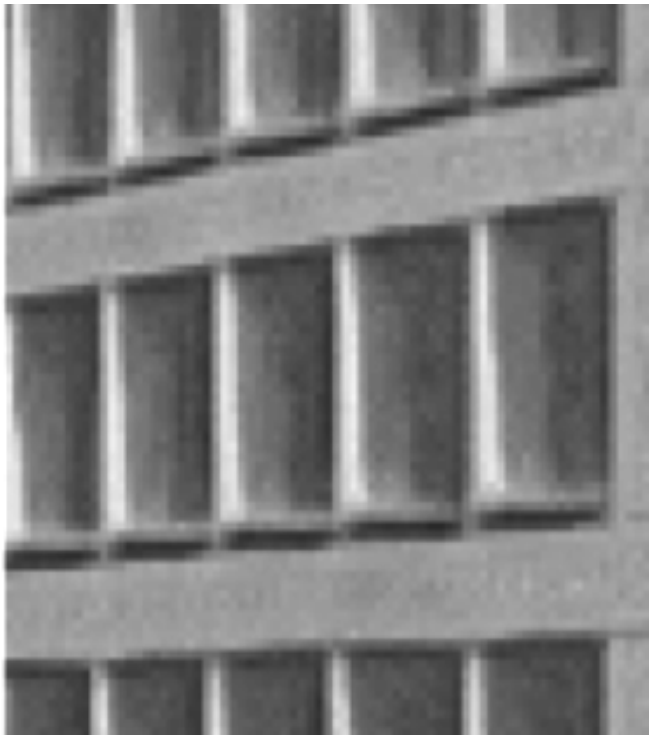




Patch-Based Filtering

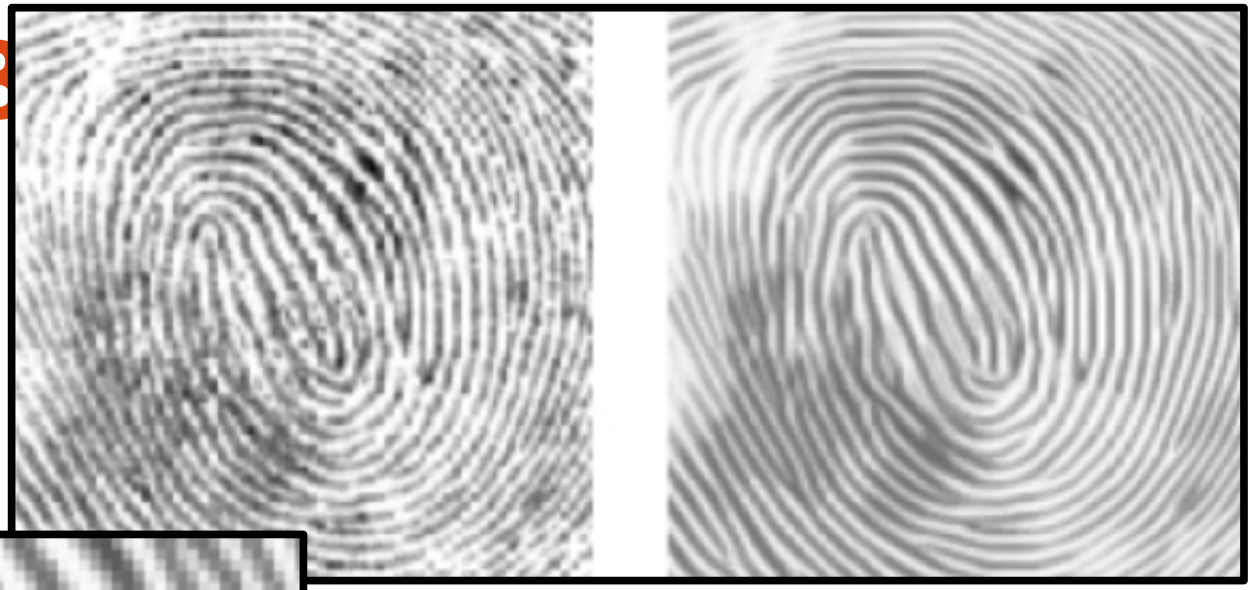
- Results on textured images

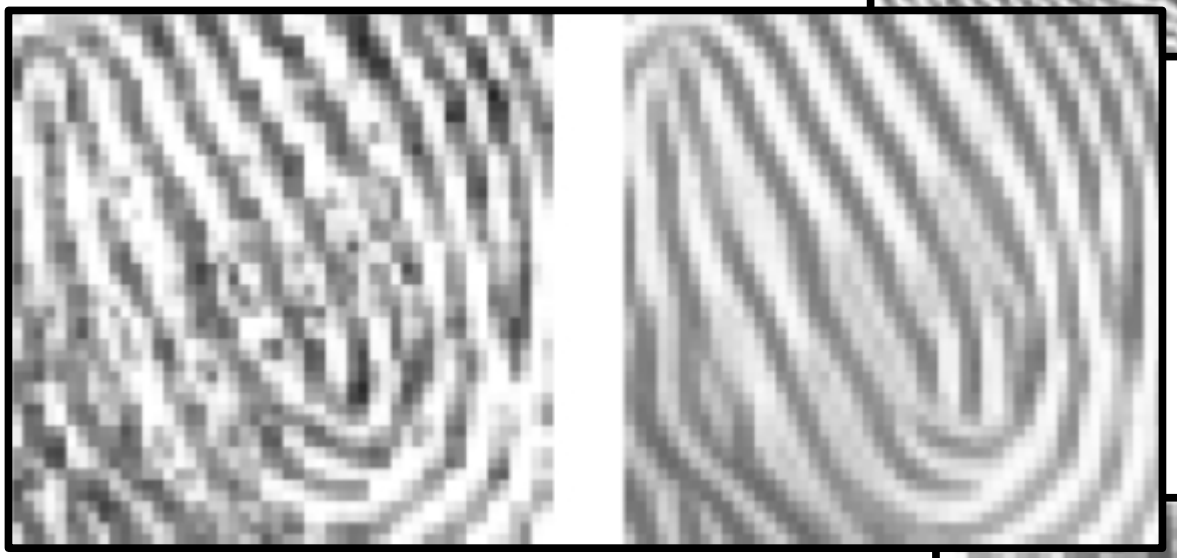


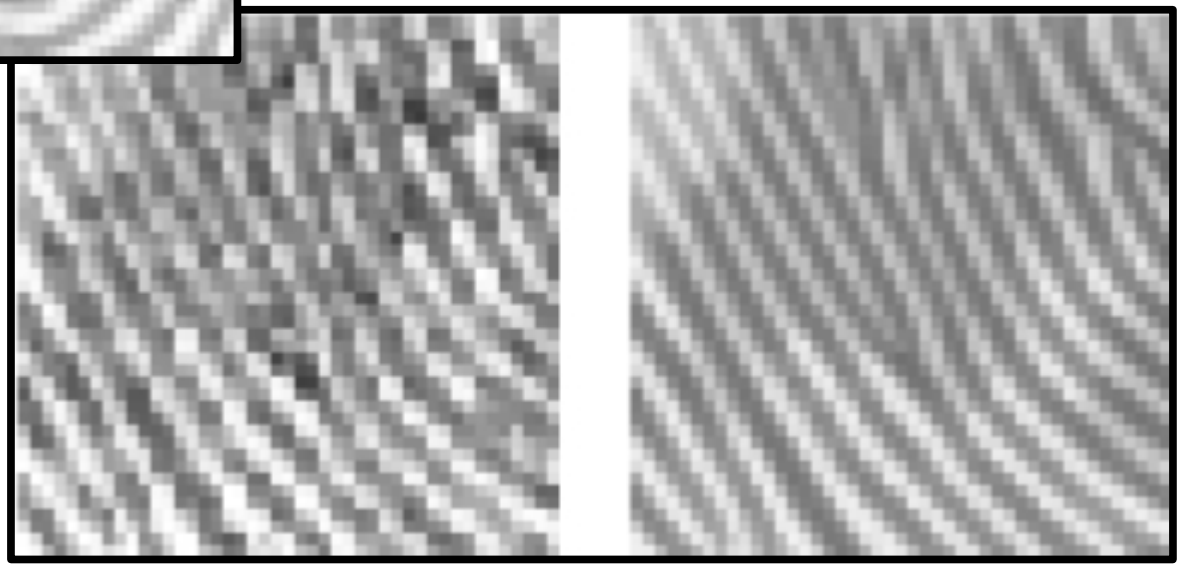


Patch-B

- Results on textured images

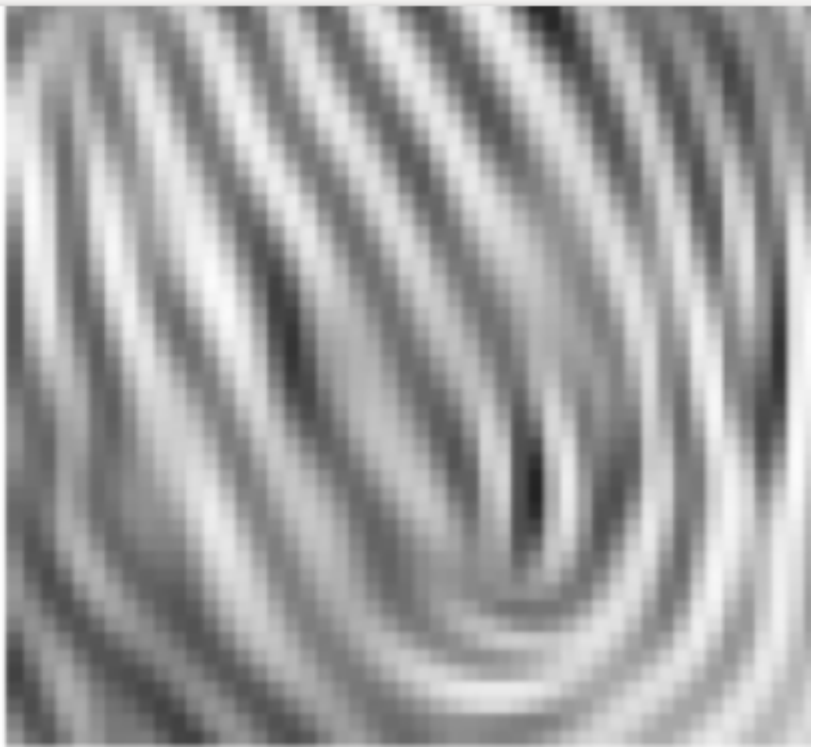






Patch-Based Filtering

- Results on textured images
  - Non patch-based filtering
  - **Bottom right = bilateral filter**
  - Bottom left = anisotropic diffusion
    - Perona-Malik



Patch-Based Filtering

- Results on piecewise smooth images

(a1)

(c1)

(e1)

- Results

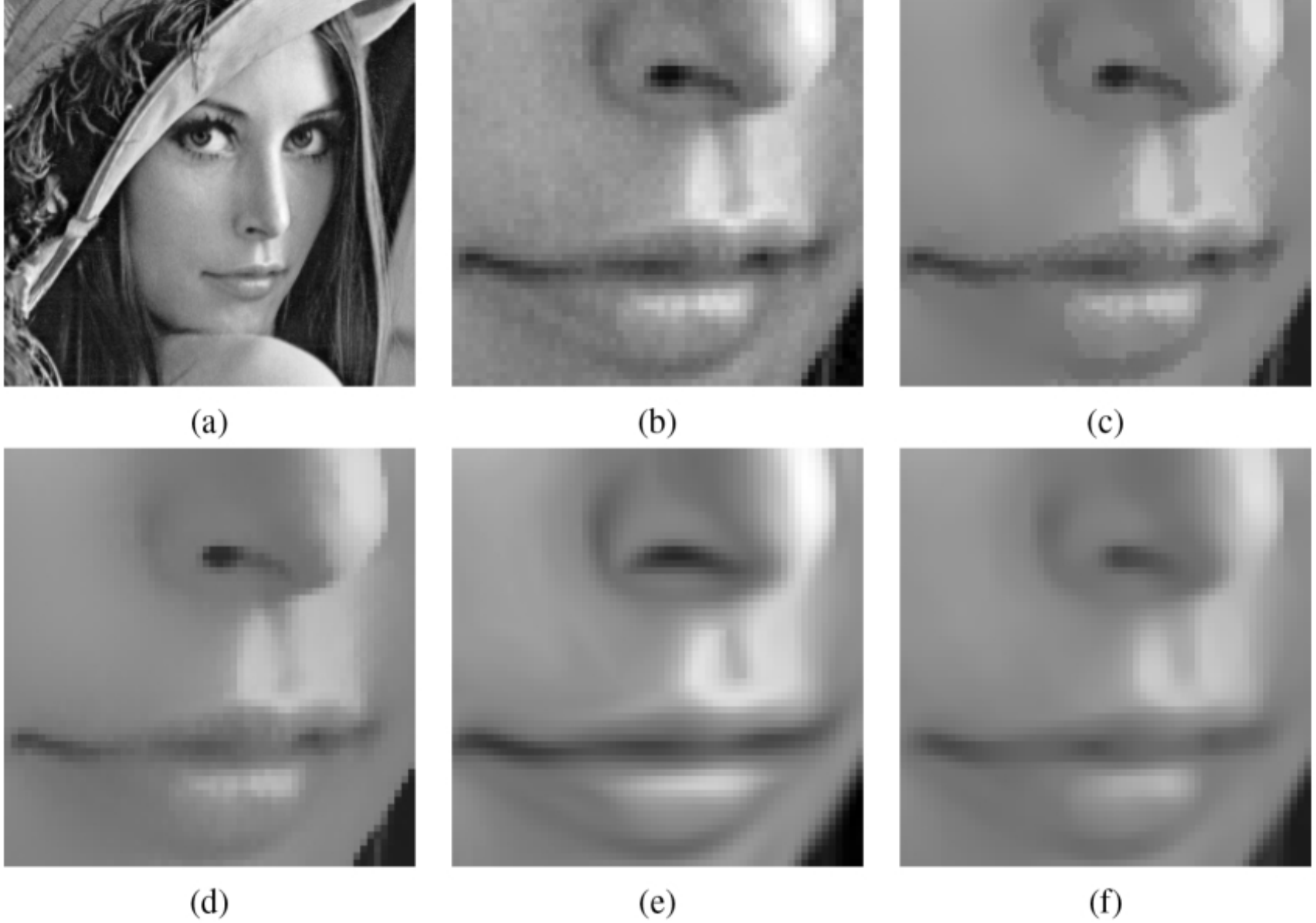


Figure 4.1. Comparison of with prevalent strategies. (a) Degraded *Lena* image: grayscale values range:0–100 grayscale unit (G.U.). Zoomed insets of: (b) the degraded image; (c) anisotropic diffusion: K=0.5 G.U.s, 20 iterations, (d) bilateral filtering: σ_{domain}=3 pixels, σ_{range}=12 G.U., (e) coherence-enhancing diffusion: σ=0.1 pixels, ρ=2 pixels, α=0.0001, C=0.0001, 15 iterations, and (f) curvature flow: time step=0.2, 8 iterations.