

# Digital Image Processing

## Image Smoothing for Noise Reduction: Patch-Based Filtering

Suyash P. Awate

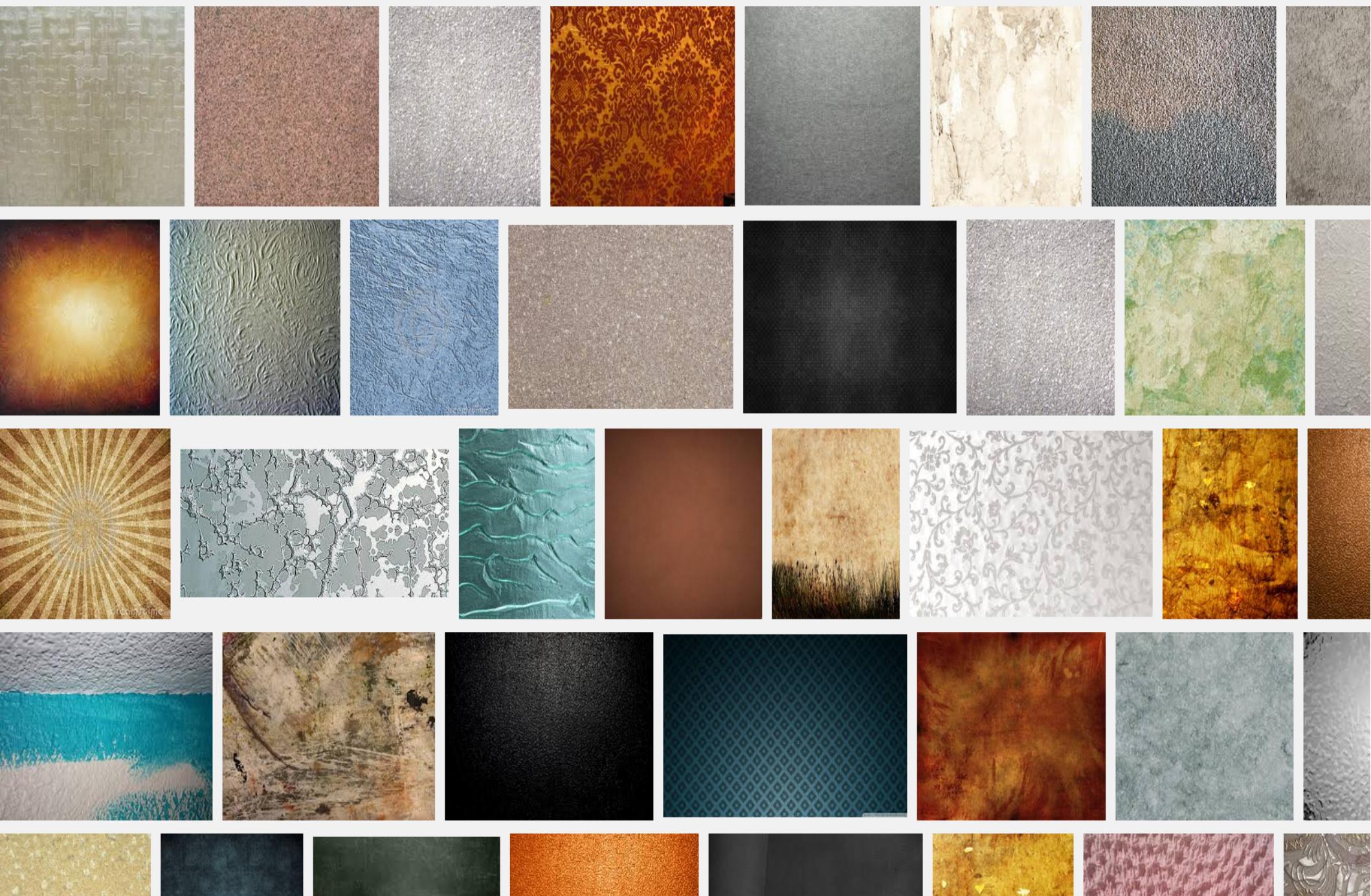
# Image Denoising

- Bilateral filtering for gray and color images
  - IEEE Int. Conf. Computer Vision. 1998.
  - [https://en.wikipedia.org/wiki/Bilateral\\_filter](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](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







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

Bruno Espinosa<sup>1</sup>, Víctor Igual<sup>2</sup>, Fabián R.<sup>3</sup>, Pedro Pérez<sup>4</sup>,  
Eduardo González<sup>5</sup>, Gorka Martínez<sup>6</sup>, Raúl Rojas<sup>7</sup>,  
Jesús González<sup>8</sup>, Juan Gómez<sup>9</sup>, and Raúl Urtasun<sup>10</sup>

<sup>1</sup> Department of Computer Science, UPM,  
<sup>2</sup> Technical University of Valencia, Spain,  
<sup>3</sup> University of Zaragoza, Zaragoza, Spain,  
<sup>4</sup> University of Zaragoza, Zaragoza, Spain,  
<sup>5</sup> Computer Vision Laboratory, Universidad  
Politècnica de Madrid, Madrid, Spain,  
<sup>6</sup> Autodesk Perception Inc., USA  
<sup>7</sup> University of California, Berkeley, USA

<sup>8</sup> Microsoft Research, Cambridge, MA, USA  
<sup>9</sup> Microsoft Research, Cambridge, MA, USA  
<sup>10</sup> Microsoft Research, Redmond, WA, USA

**Abstract.** In this technical demonstration, we will show our framework for learning and detecting 3D objects in cluttered scenes, even when the objects don't have a texture. The detection is mainly based on the novel training strategy of learning a 3D model of the object and then detecting it using a camera-based 3D sensor. We have performed extensive experiments on several datasets.

In this demonstration, we will show each step of our pipeline, from the acquisition of the 3D model to the learning and detection of the object. We will also show how our framework can be used for different applications, such as autonomous navigation (e.g. in robotic applications).

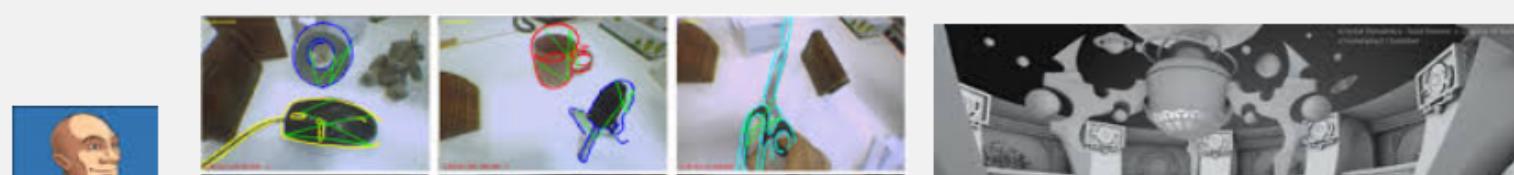
**1. Introduction**

Many current vision applications, such as autonomous driving, driver HMI [1], or airport security [2], can be made much easier through the analysis of depth, which is usually obtained by cameras. However, cameras are not the only sensor. Vision, which is important to efficiently and accurately detect objects and estimate their pose, can be done using other sensors, such as LiDAR [3] or stereo cameras [4]. LiDAR is an efficient sensor that explores both depth and geometry, but it is expensive and has a limited range. Stereo cameras are cheap and can measure depth and geometry, but they are less accurate than LiDAR and have a limited range. In this demonstration, we will show how our framework can be used for different types of objects, such as vehicles, people, and animals, without the need of a camera or a LiDAR.

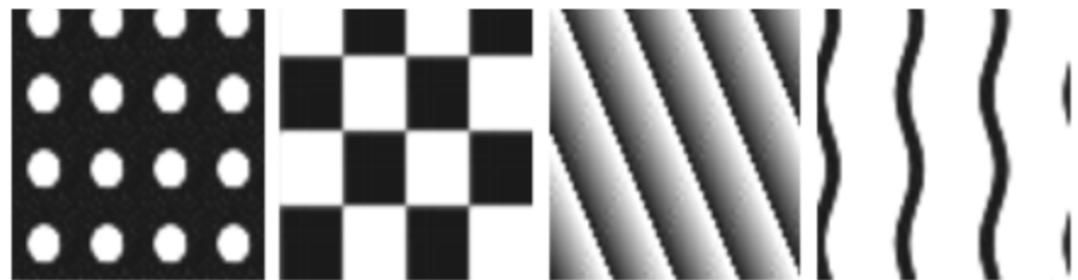
**2. Related Work**

The field of LiDAR-based vision is very large. First, Google's car-based solution, which is difficult to control and should be legal [5]

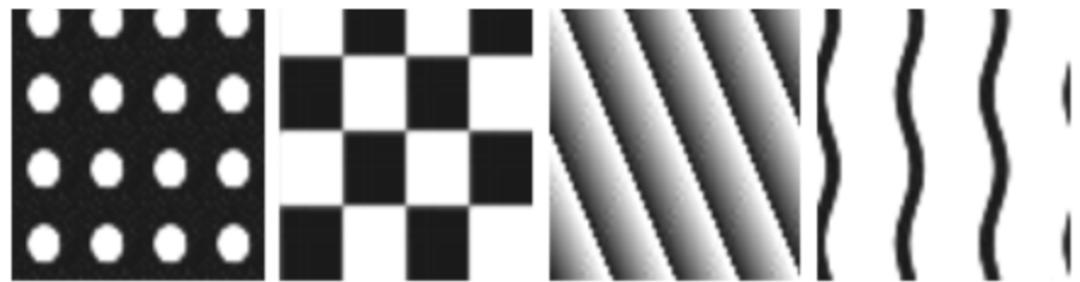
1. [Funding](#) This work was partially funded by the Spanish Ministry of Science and Innovation under grants ECO2009-07200 and ECO2012-34340.



**Artificial/periodic:**



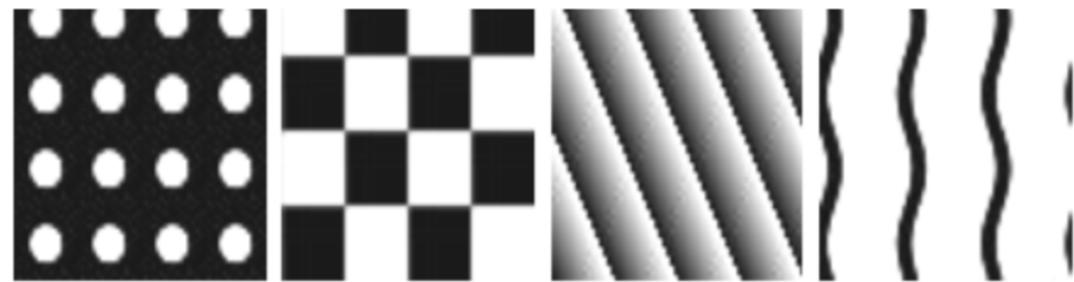
**Artificial/periodic:**



**Artificial/non-periodic:**



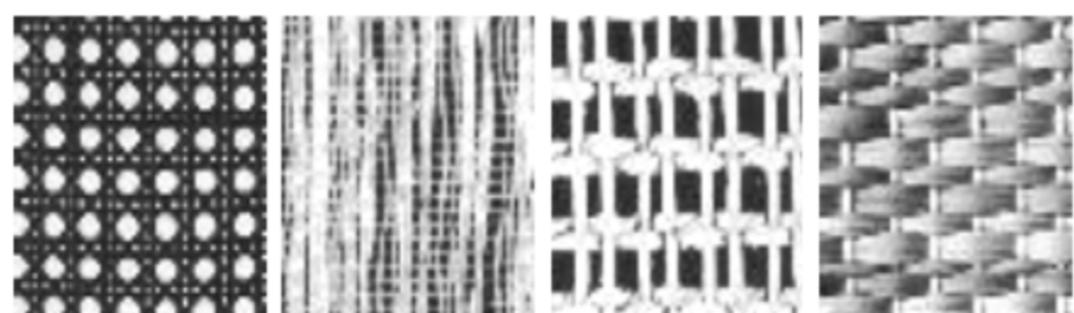
**Artificial/periodic:**



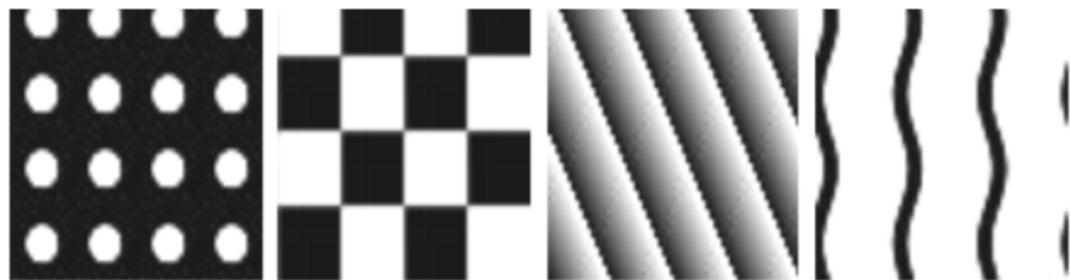
**Artificial/non-periodic:**



**Photographic/quasi-periodic:**



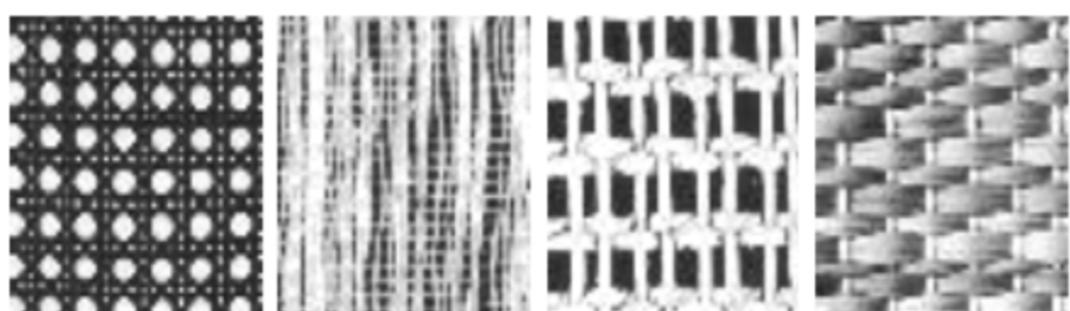
**Artificial/periodic:**



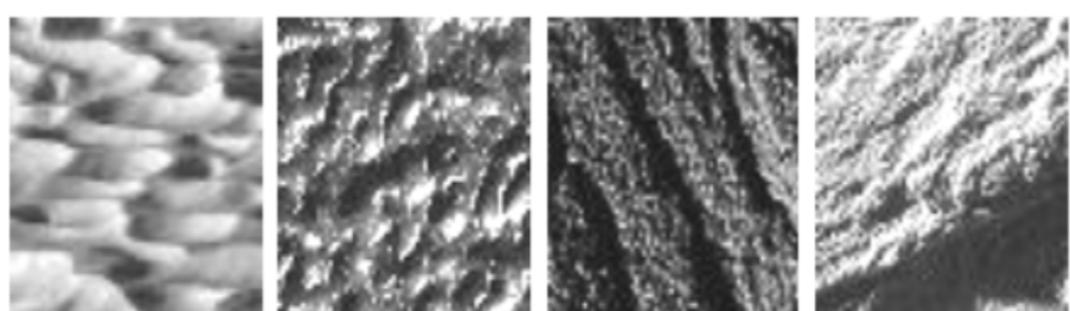
**Artificial/non-periodic:**



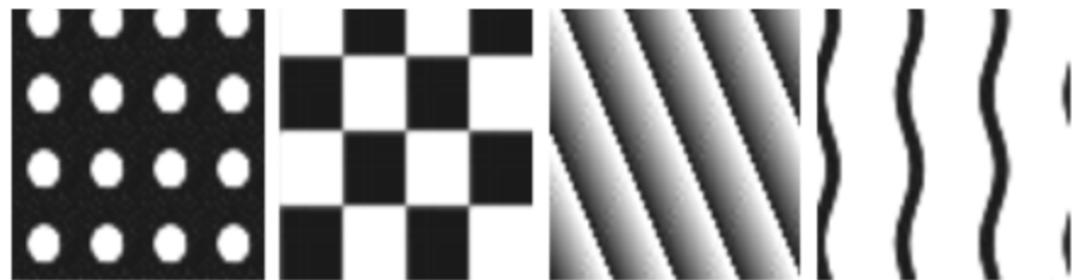
**Photographic/quasi-periodic:**



**Photographic/random:**



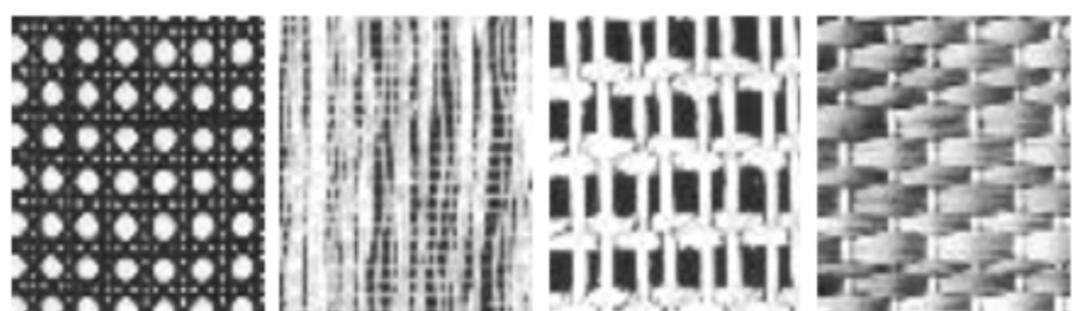
**Artificial/periodic:**



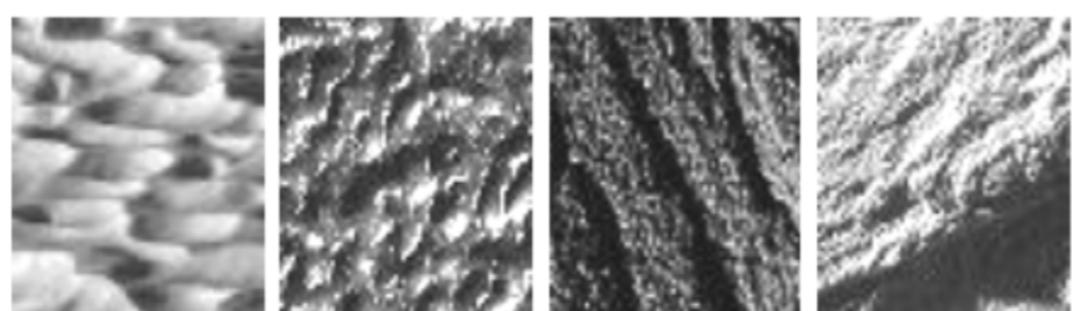
**Artificial/non-periodic:**



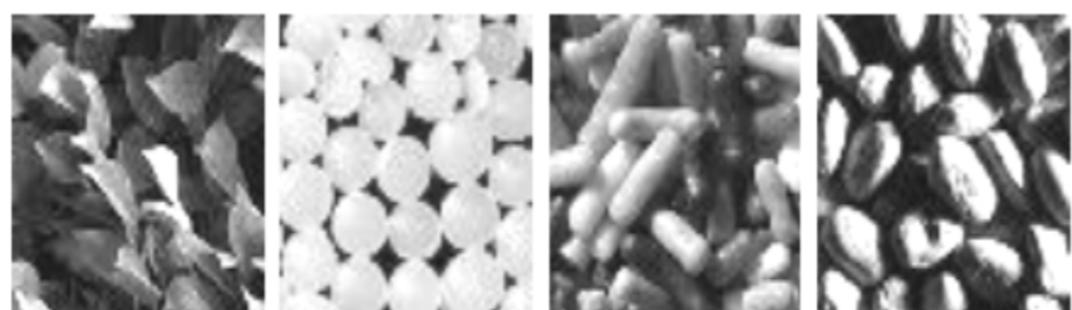
**Photographic/quasi-periodic:**



**Photographic/random:**



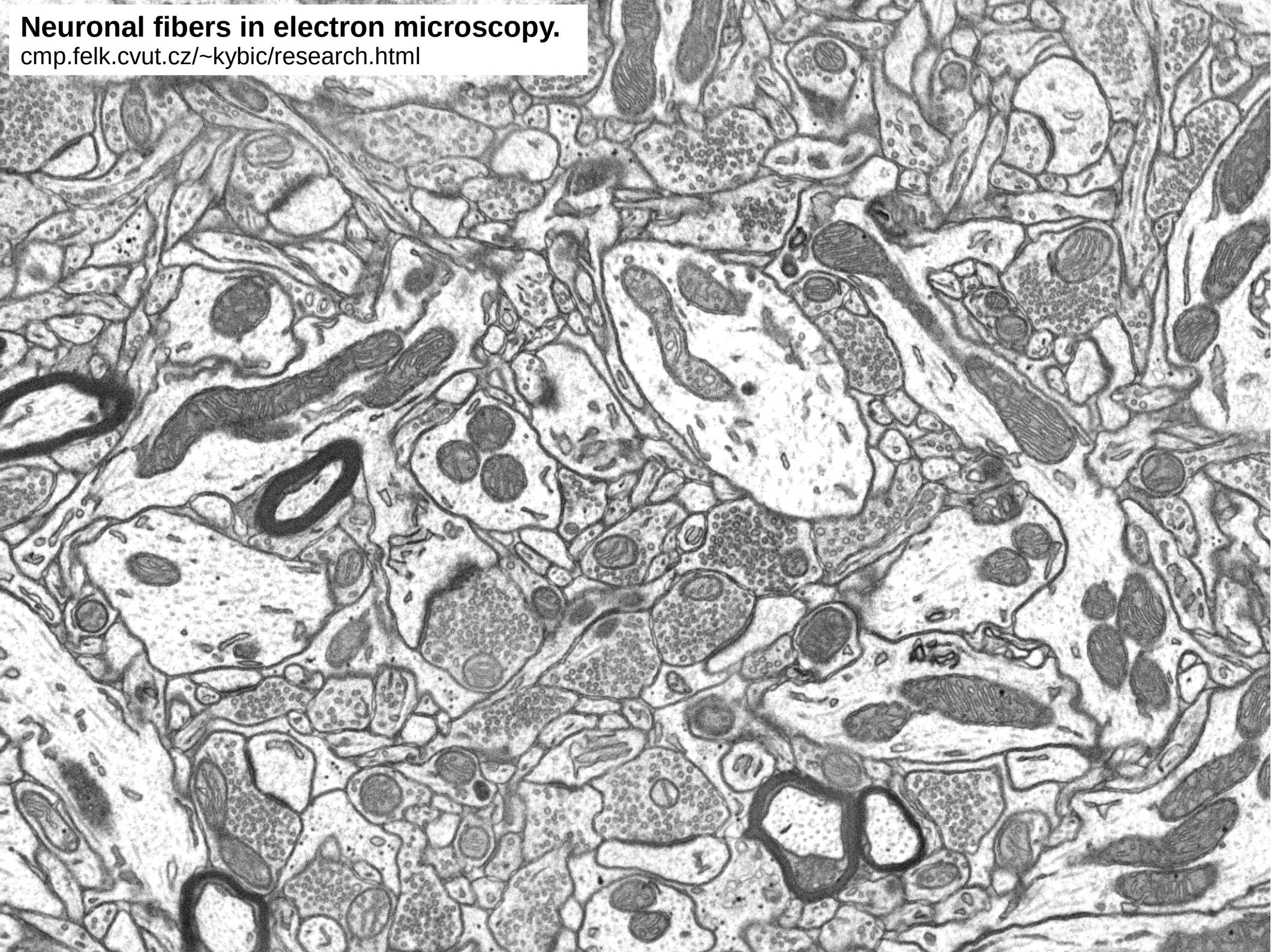
**Photographic/structured:**





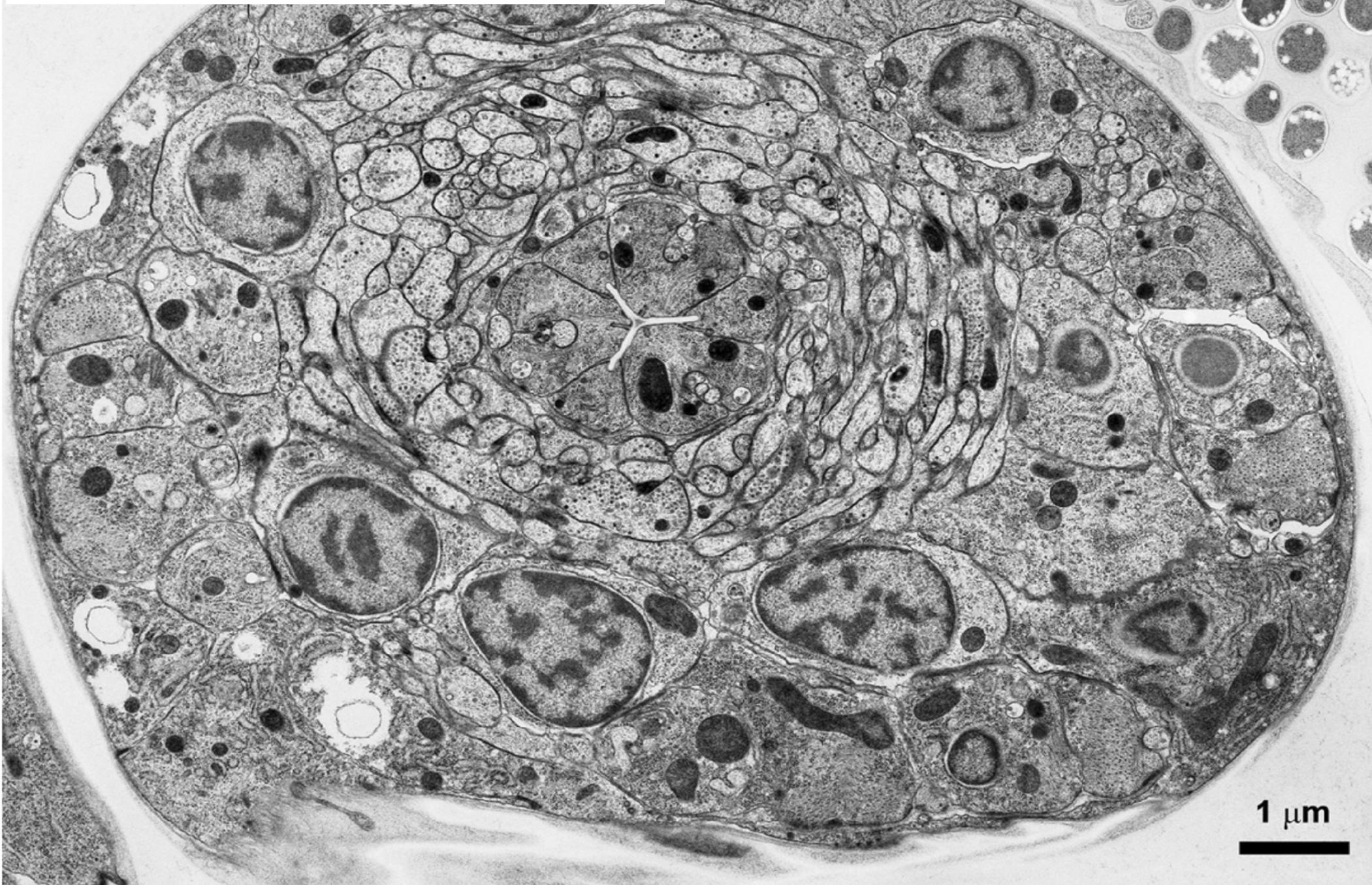
# Neuronal fibers in electron microscopy.

[cmp.felk.cvut.cz/~kybic/research.html](http://cmp.felk.cvut.cz/~kybic/research.html)



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

R. FETTER (2/27/04)



## Retinal Veins

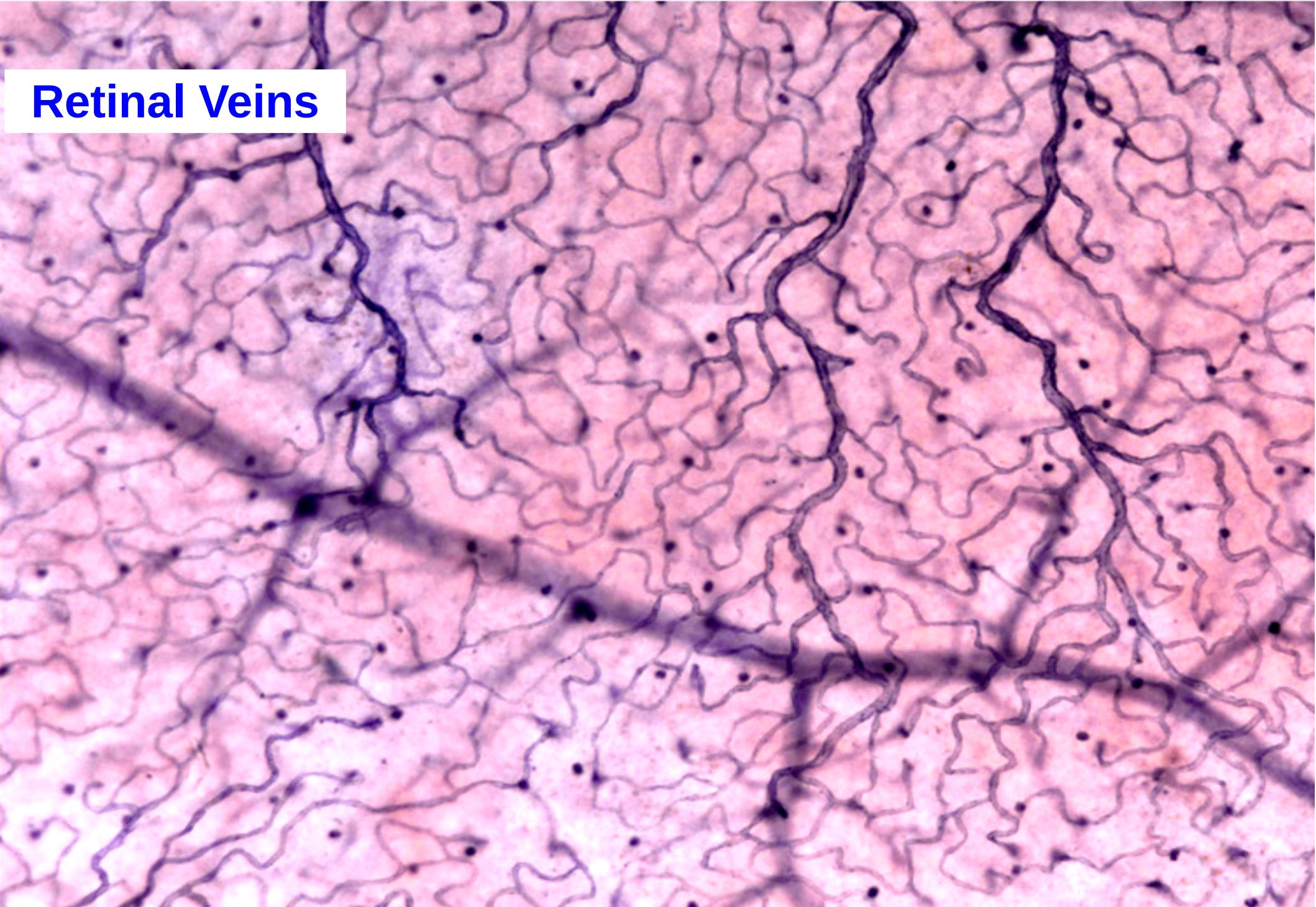
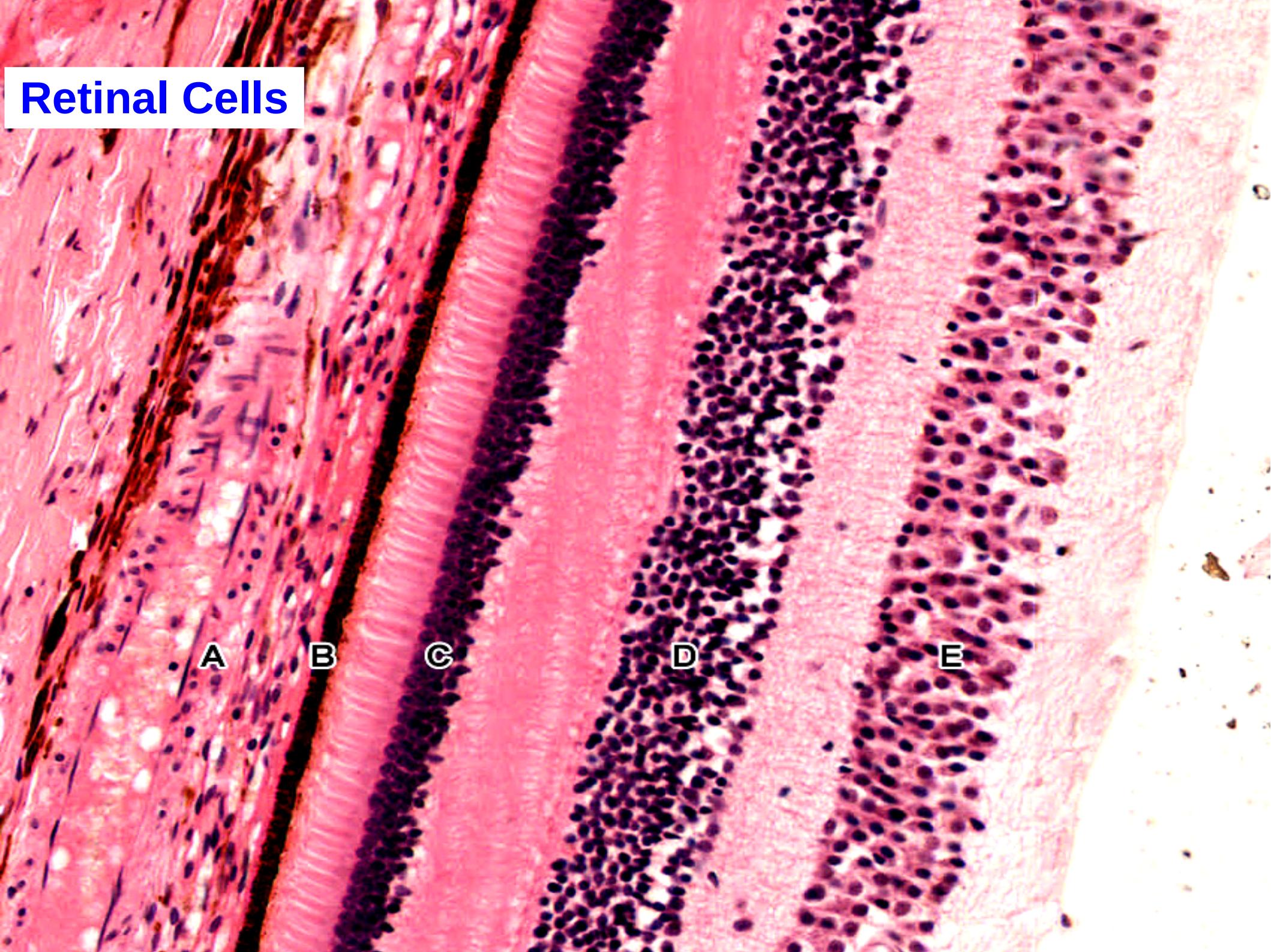
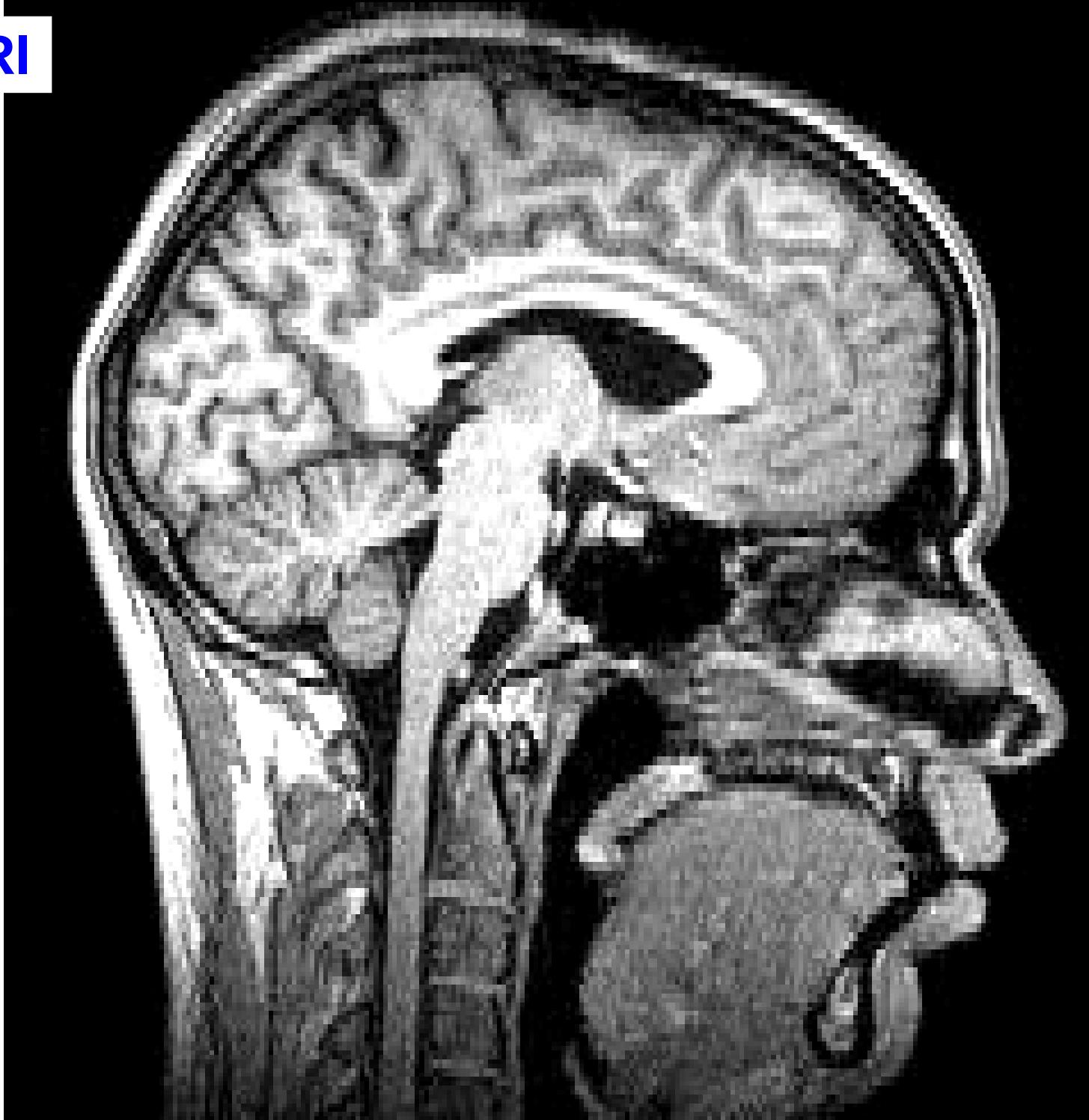


Fig. 18b. Flatmount view of a rat retina stained with NADPH-diaphorase at the level of focus of a major vein and venules. (Courtesy of Toby Holmes, Moran Eye Center).

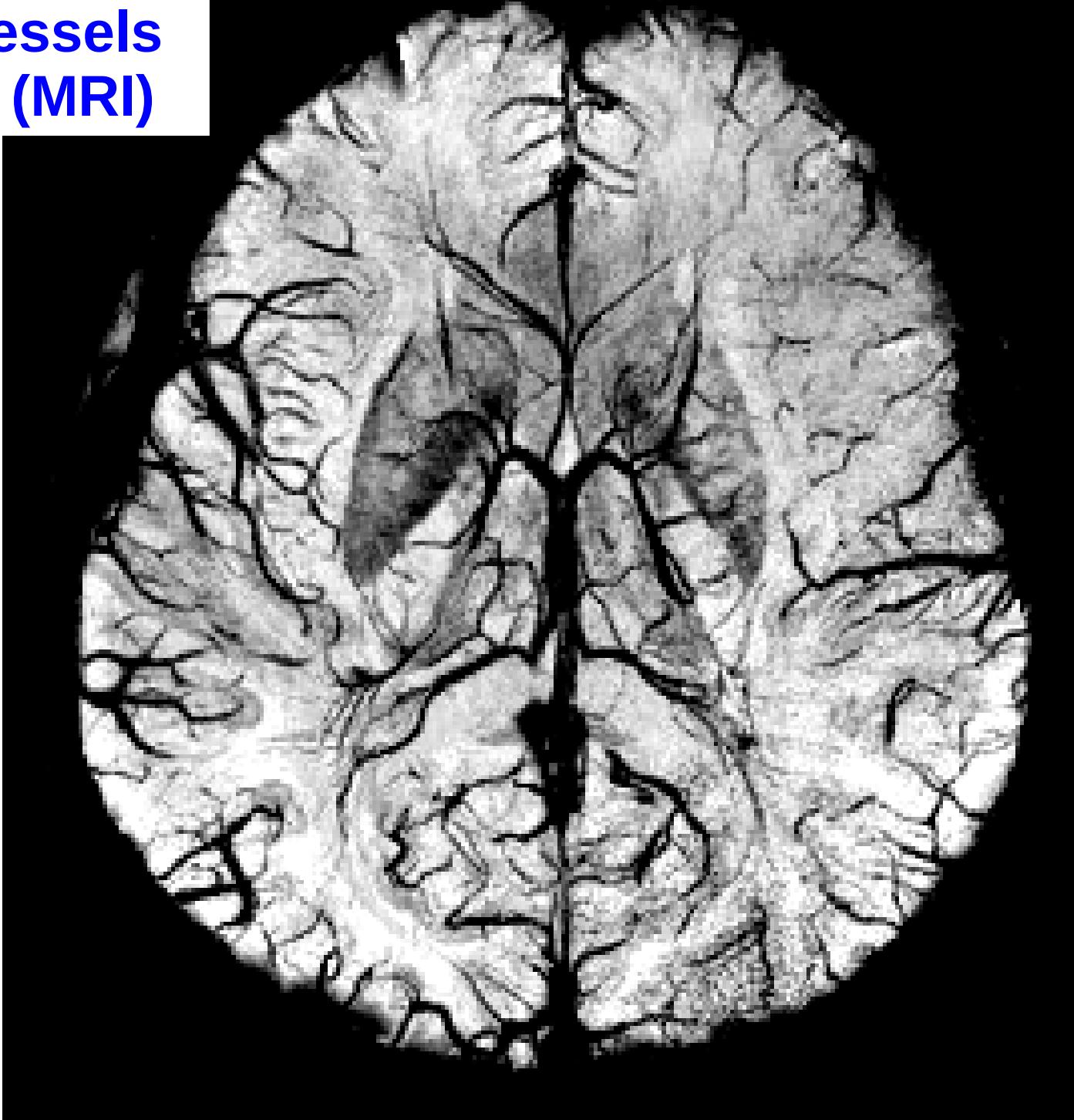
## Retinal Cells

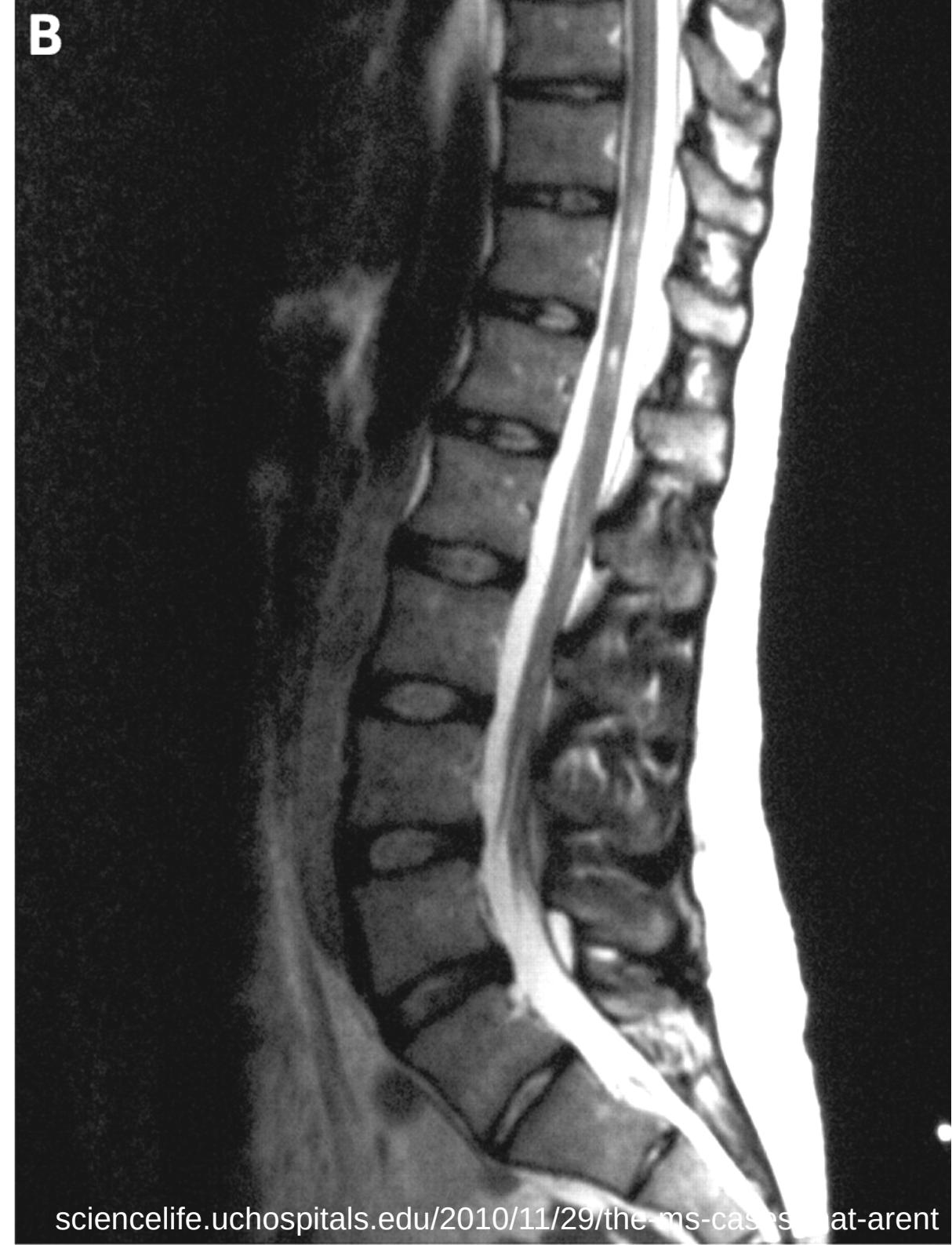


# Head MRI



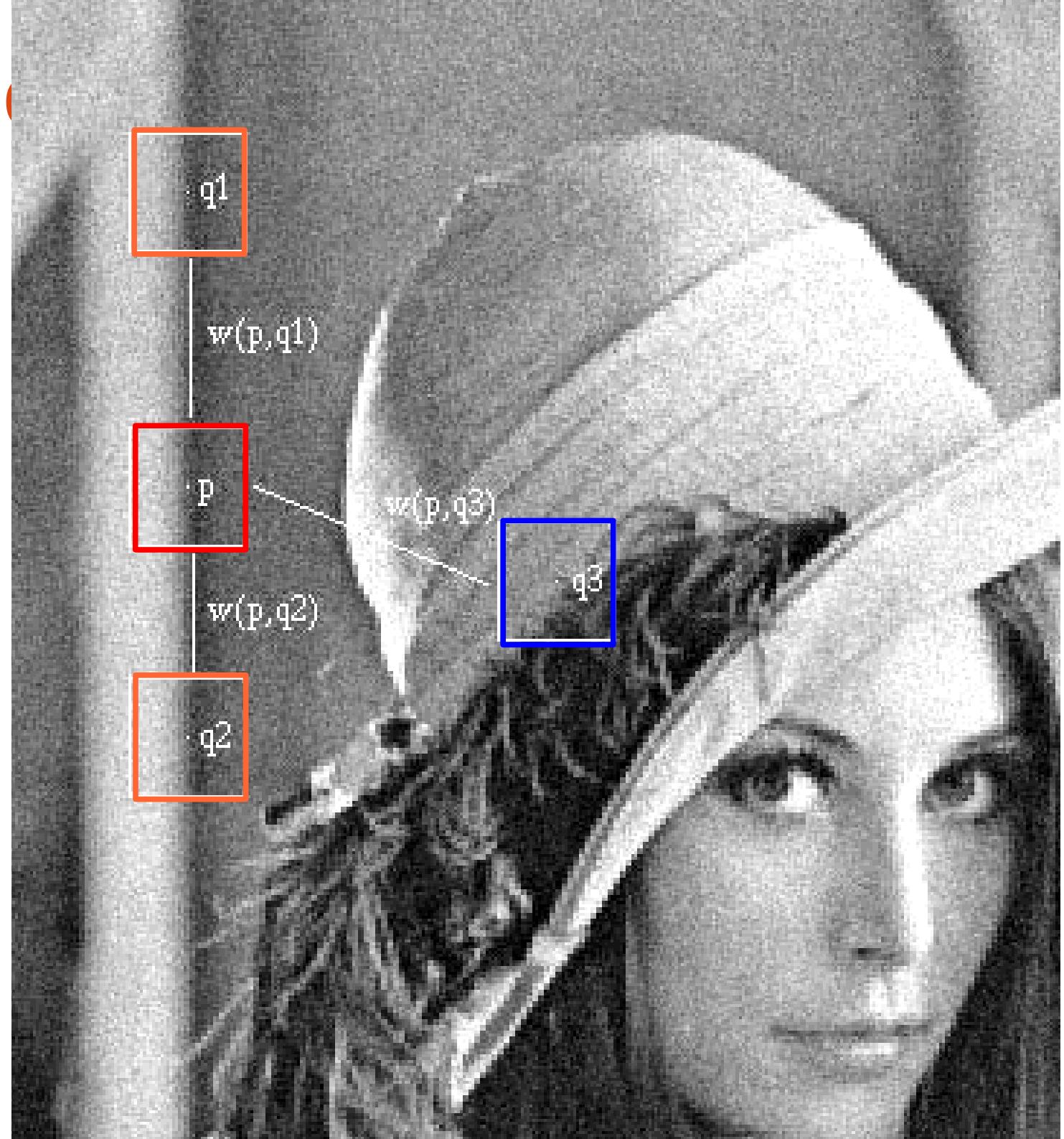
## Blood Vessels in Brain (MRI)





# Patch-Based

- 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 ?
  - 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

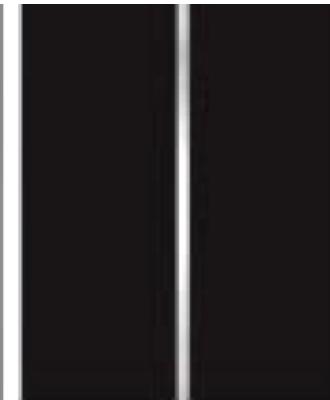
$$\sum_{j \in \Lambda} \omega_h(j, k) f(j)$$

- 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

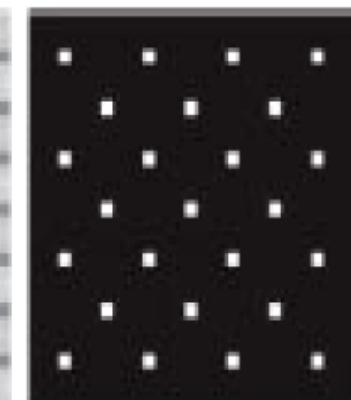
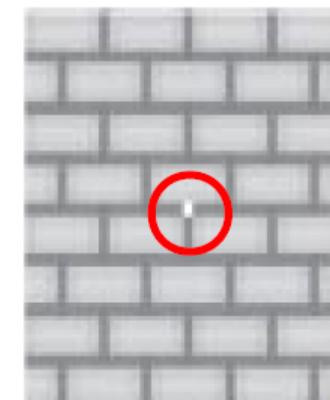
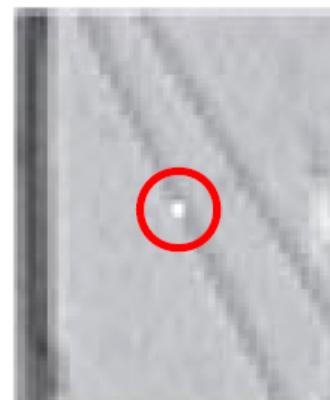
- Pixel weights (right)  
for a chosen pixel in an image (left)



(a)

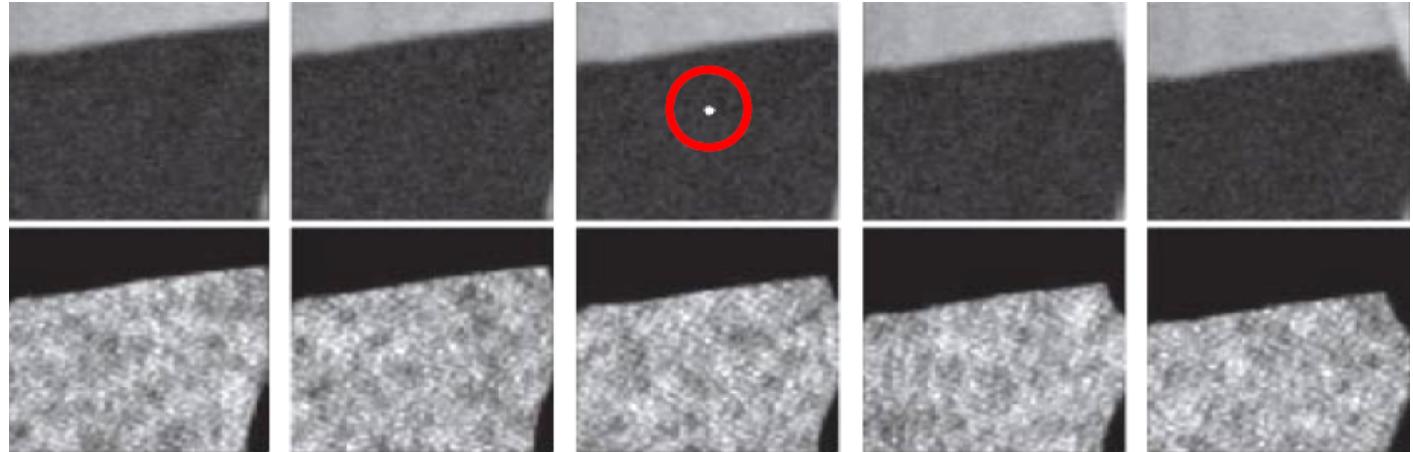
(b)

(c)

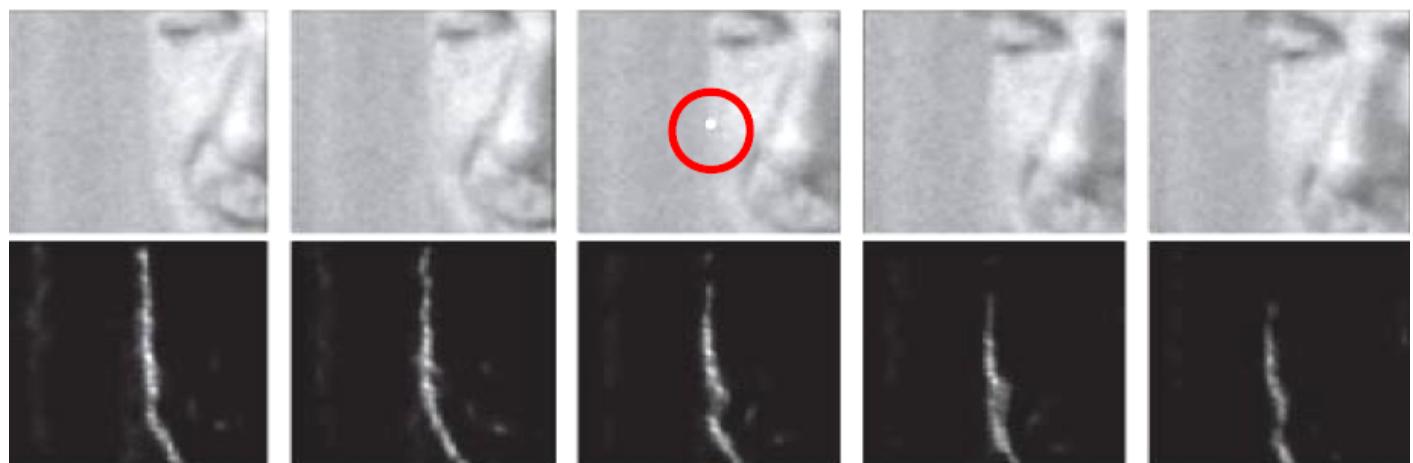


# Patch-Based

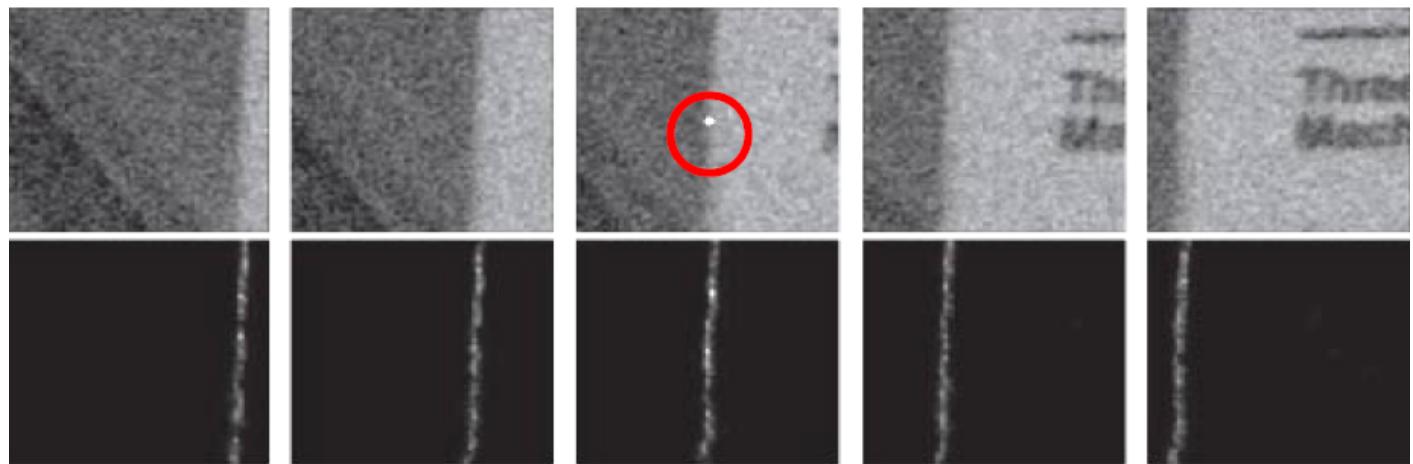
- Pixel weights  
(bottom)  
for a chosen  
pixel in a  
video (top)



(a)



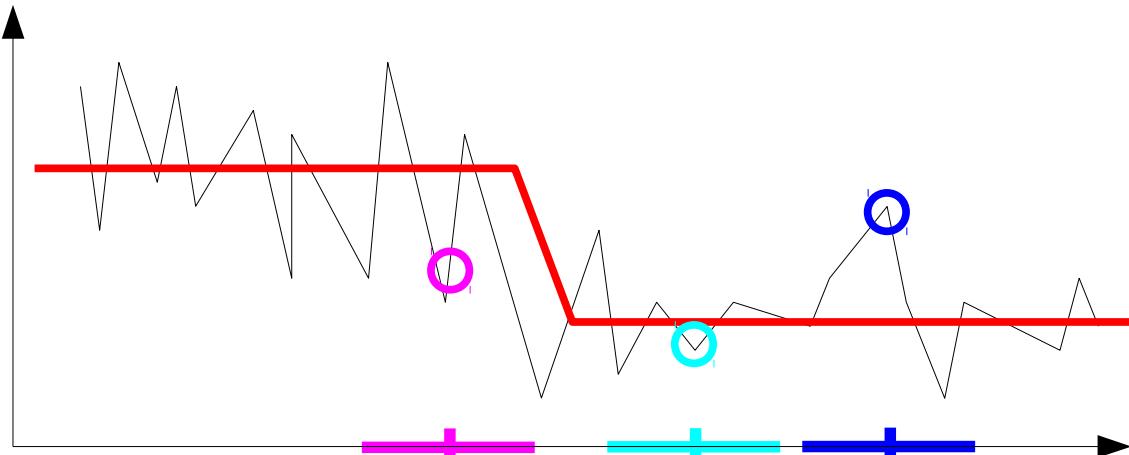
(b)



# 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



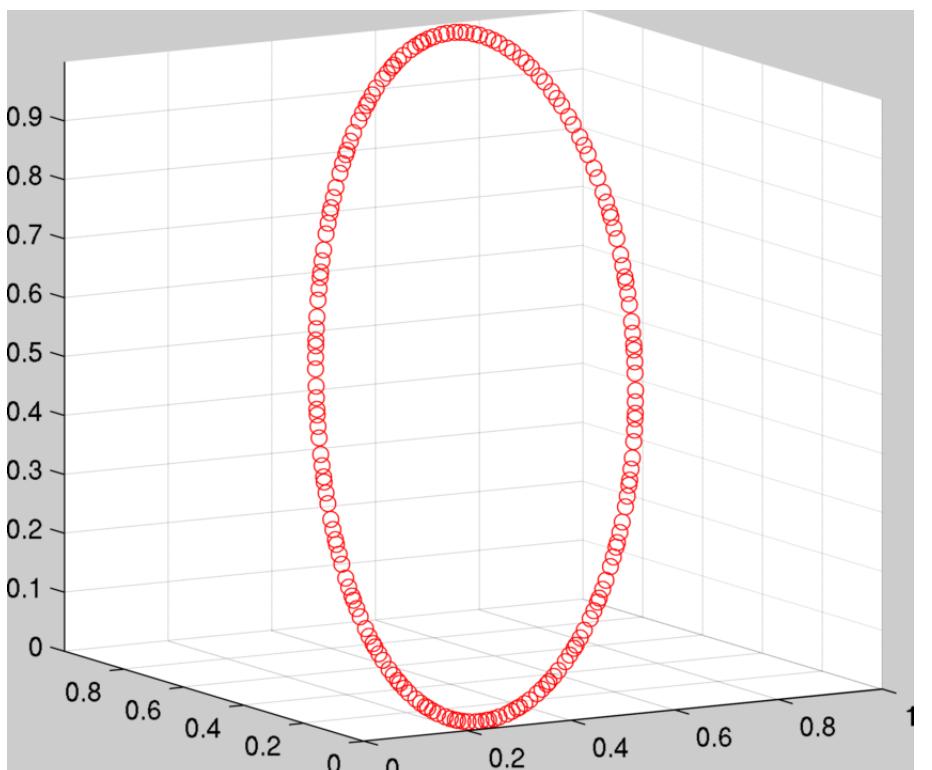
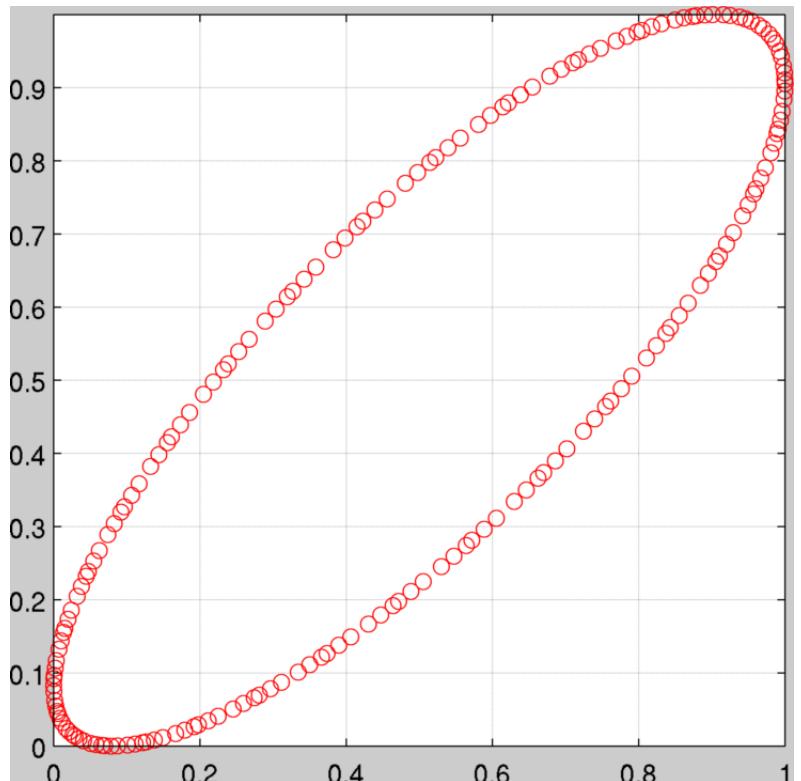
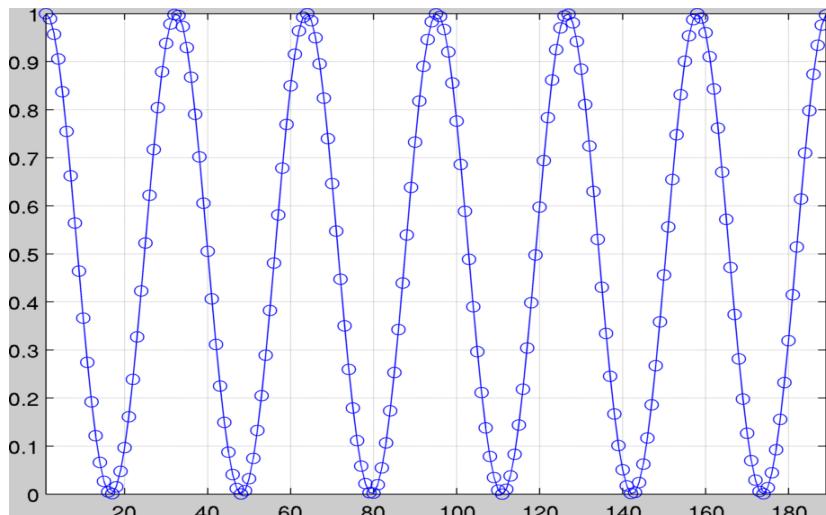
- **Blue** intensity patch much more similar to **cyan** intensity patch than **magenta** intensity patch
  - Desirable

# Patch-Based Filtering

- “Typical” patch sizes =  $5 \times 5 \leftrightarrow 9 \times 9 \leftrightarrow 15 \times 15$
- 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 =  $21 \times 21 \leftrightarrow 101 \times 101$
- How to choose Gaussian standard deviation “h” ?
  - This is the important parameter
- Insights into the algorithm → see next ...

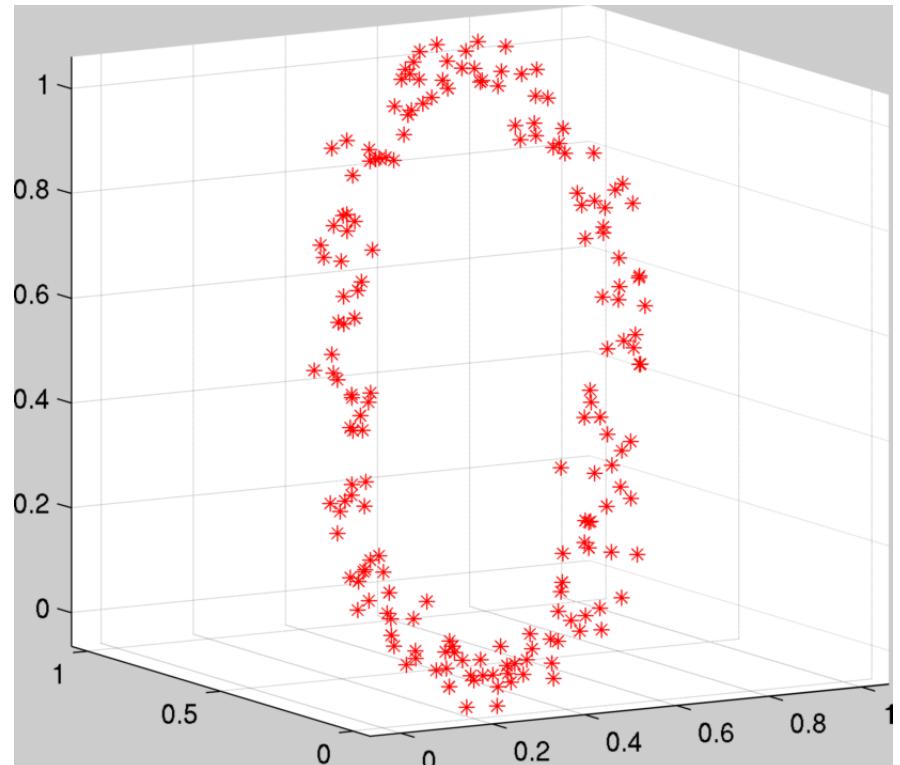
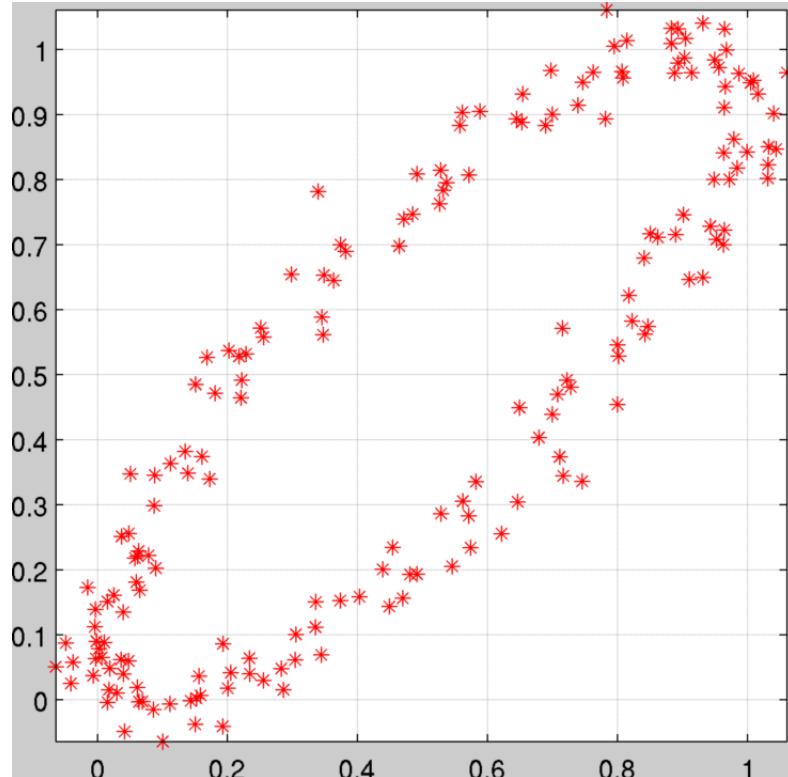
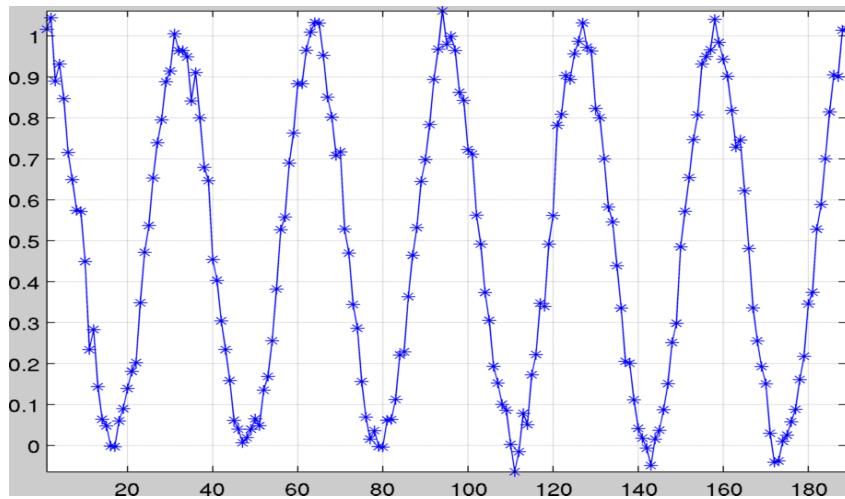
# Patch-Based Filtering

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



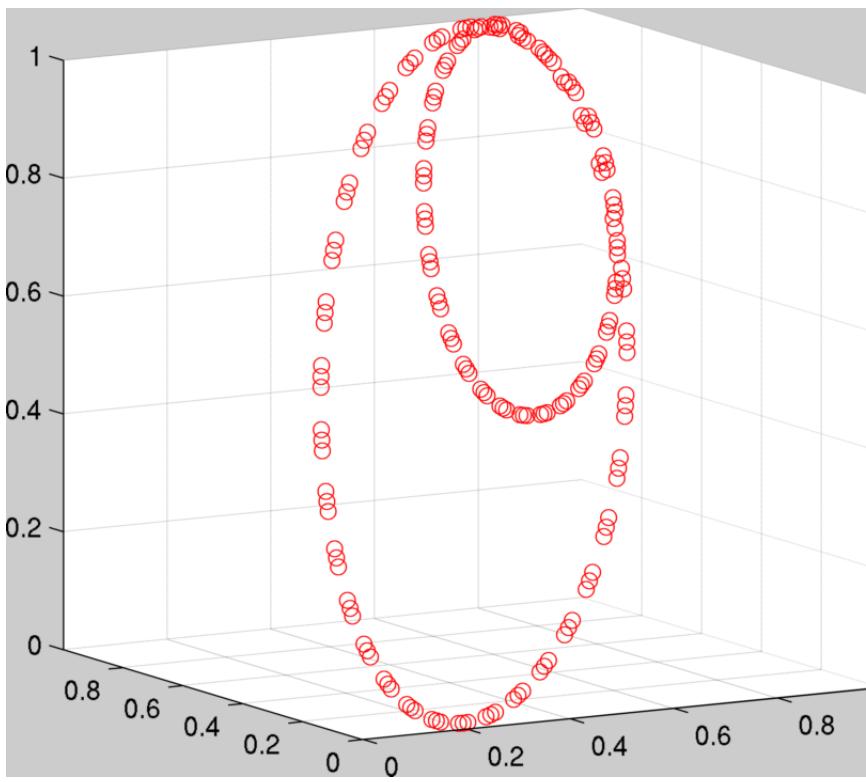
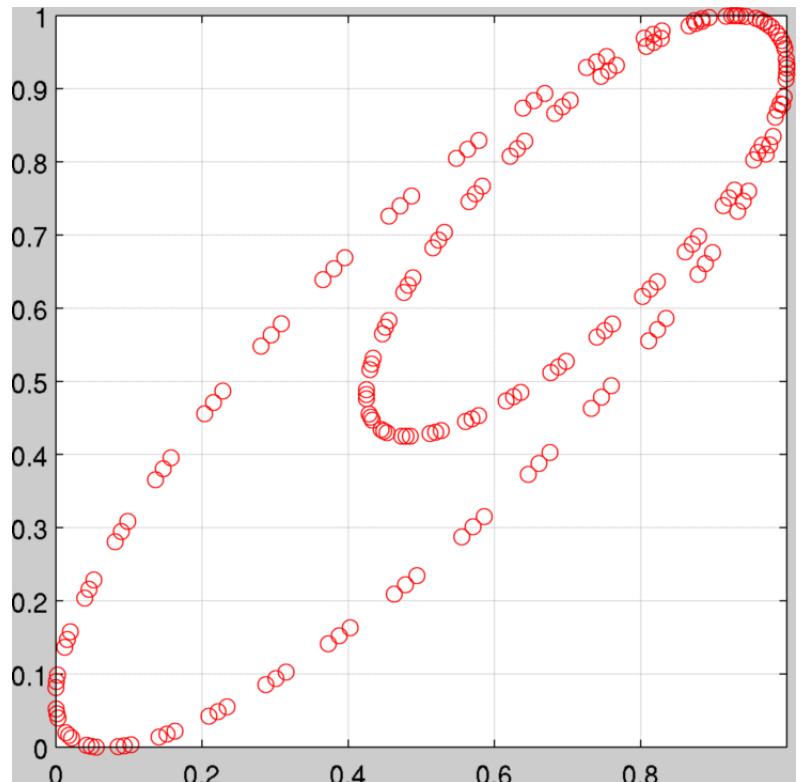
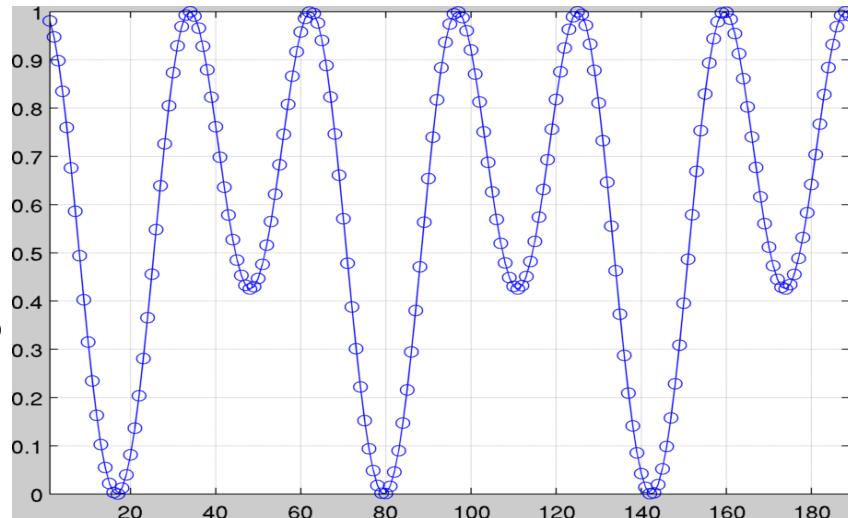
# Patch-Based Filtering

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



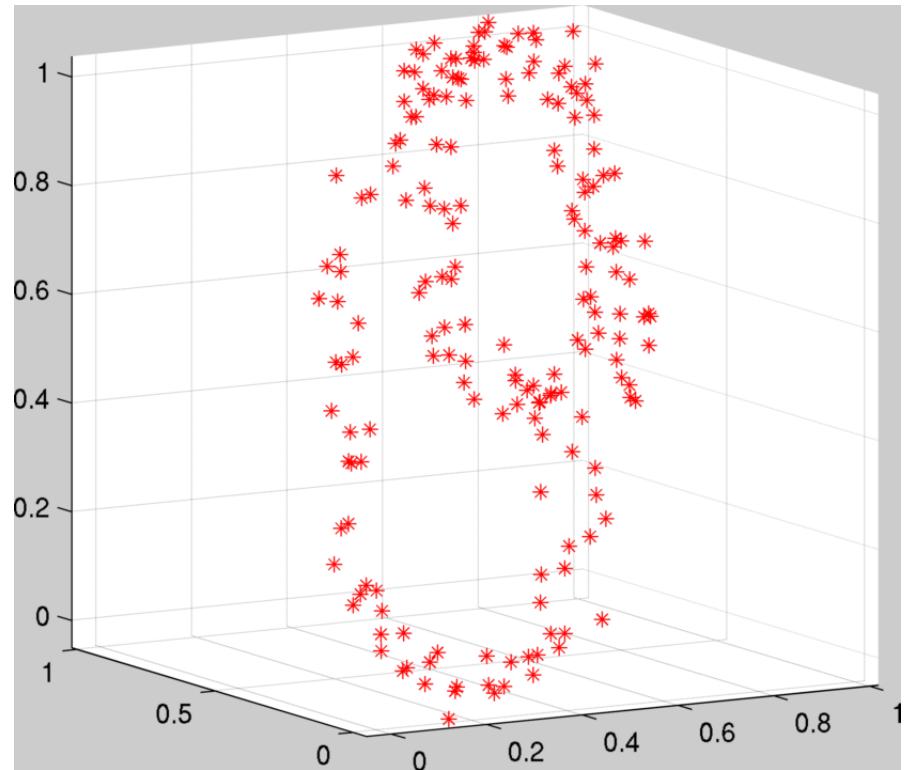
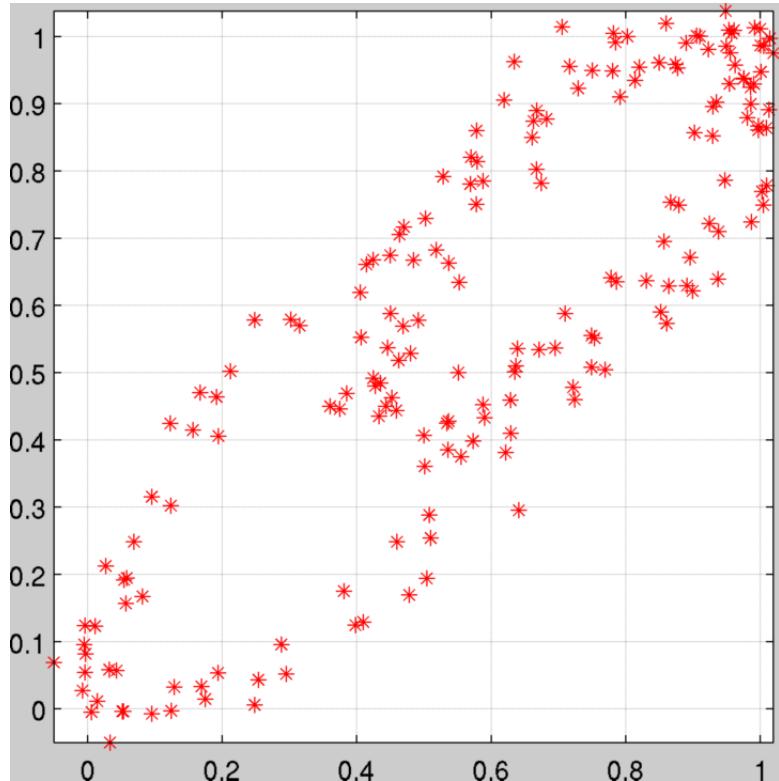
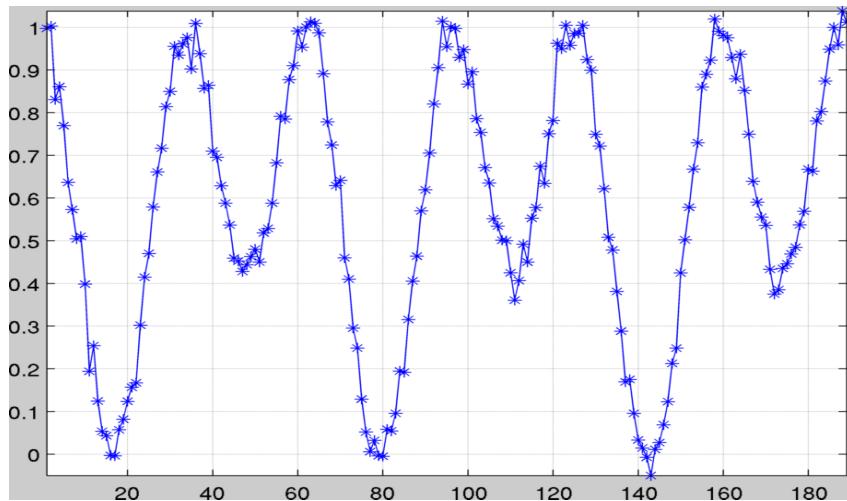
# Patch-Based Filtering

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



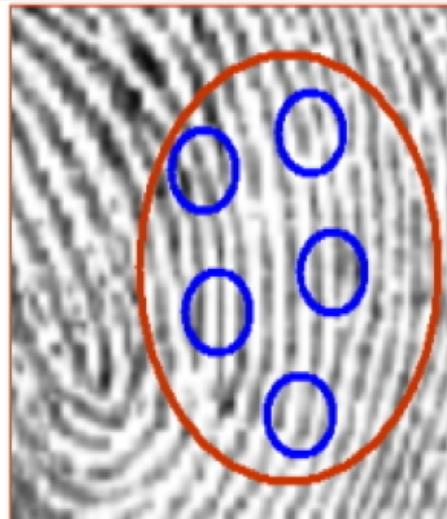
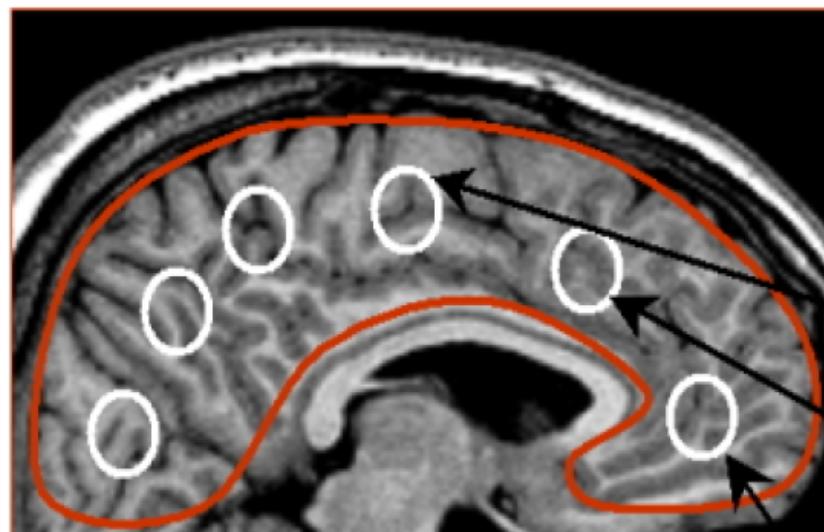
# Patch-Based Filtering

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

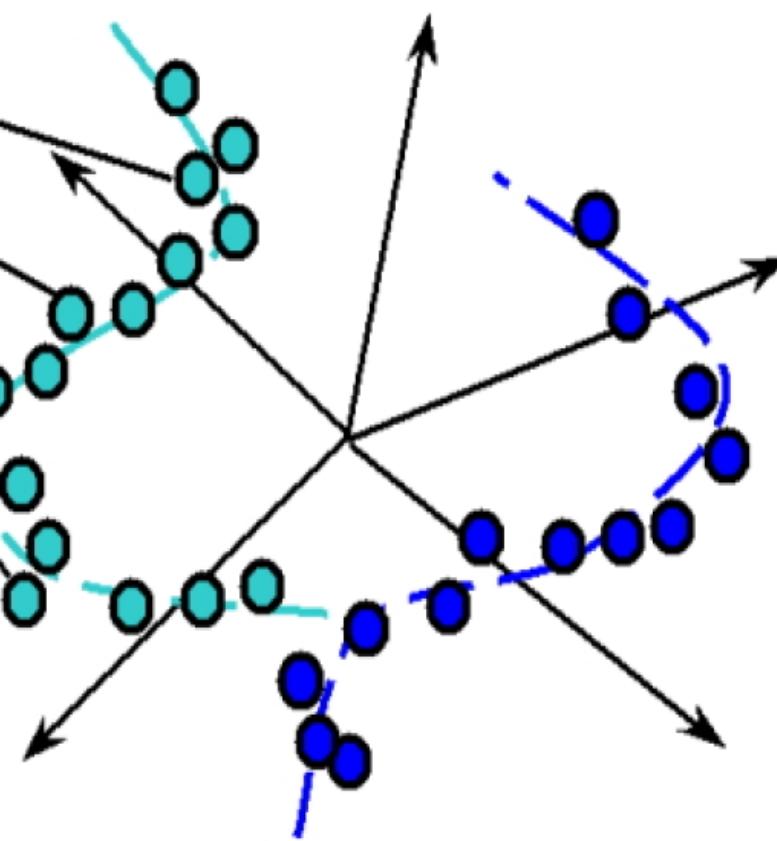


# Patch-Based Filtering

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



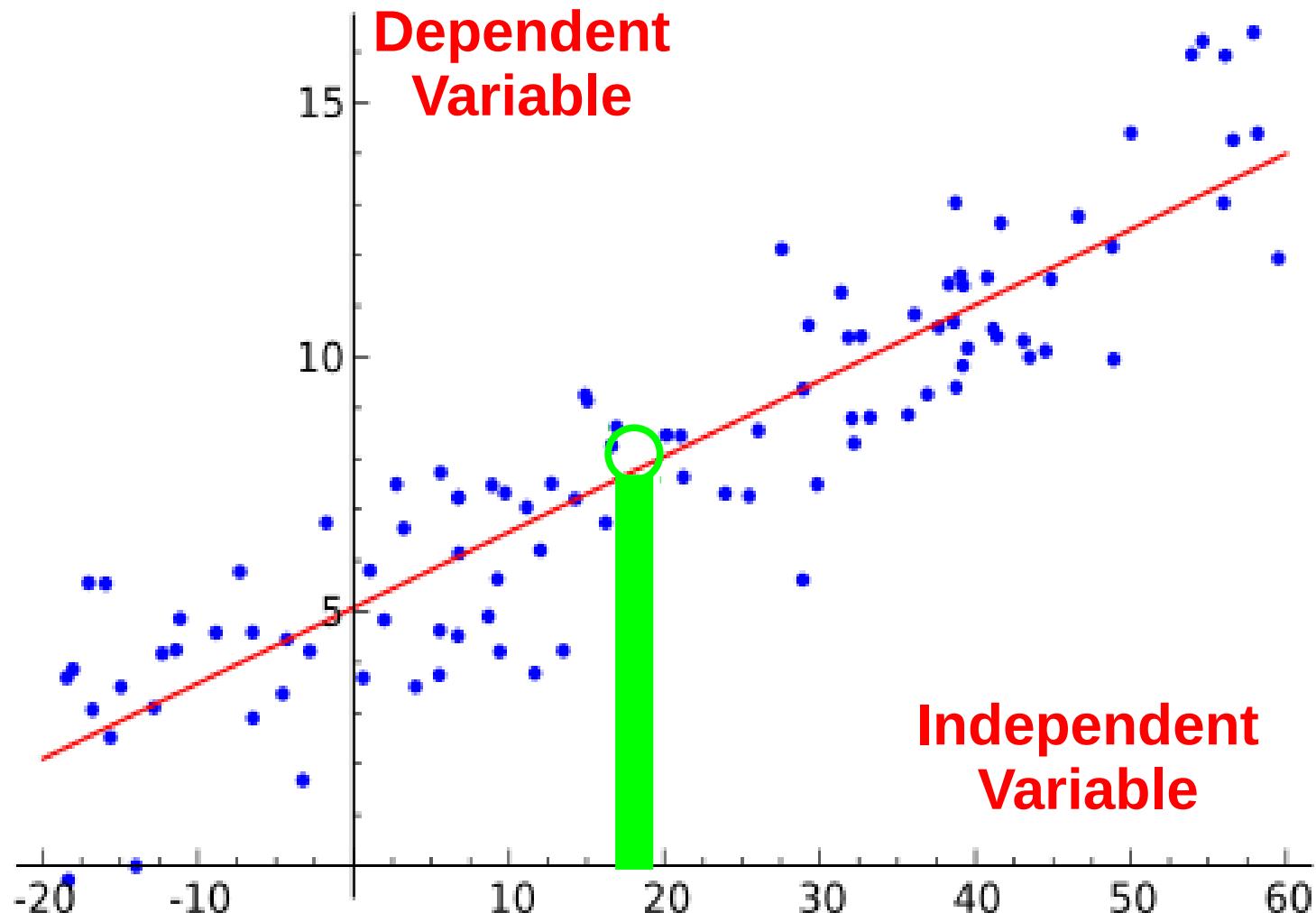
High-dimensional *feature space* of image neighborhoods



# Patch-Based Filtering

- **Regression**

- Model / estimate relationship between variables
- Linear
- **Given  $(x_i, y_i)$** 
  - $i = 1, \dots, n$
- **Given  $x_0$**
- **Find  $y_0$**

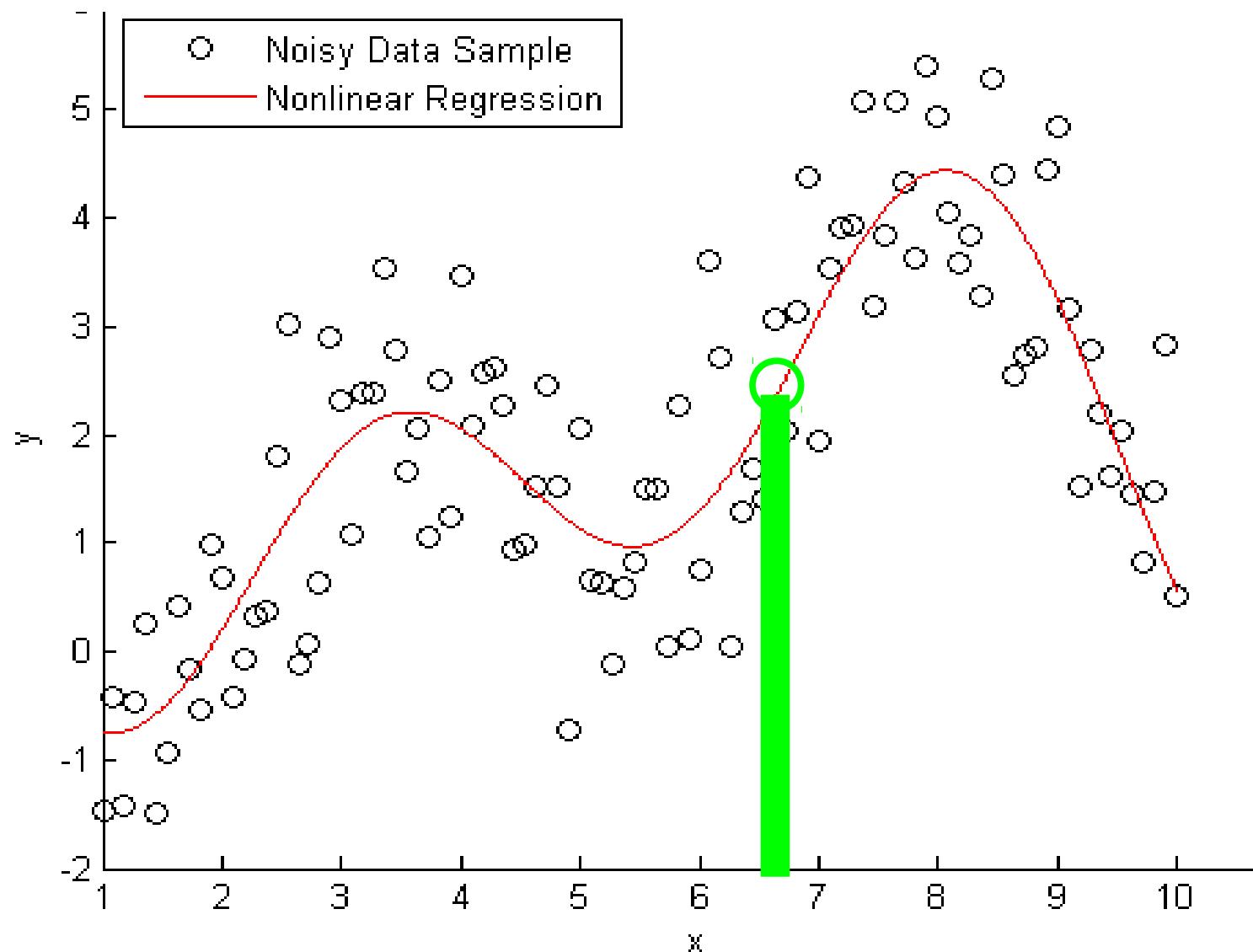


# Patch-Based Filtering

- 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  $(x_i, y_i)$ 
    - $i = 1, \dots, n$
  - Given  $x_0$
  - Find  $y_0$



# Patch-Based Filtering

- Nonlinear regression
  - Given  $(x_i, y_i)$ ,  $i=1, \dots, n$ .    Given  $x_0$ .    Find  $y_0$ .
  - Principle 1
    - If we have a very large number of pairs  $(x_j, y_j)$ , where  $x_j = x_0$  (i.e., X coordinates match)
    - Then, we'd like  $y_0 \rightarrow$  average of  $y_j$

# Patch-Based Filtering

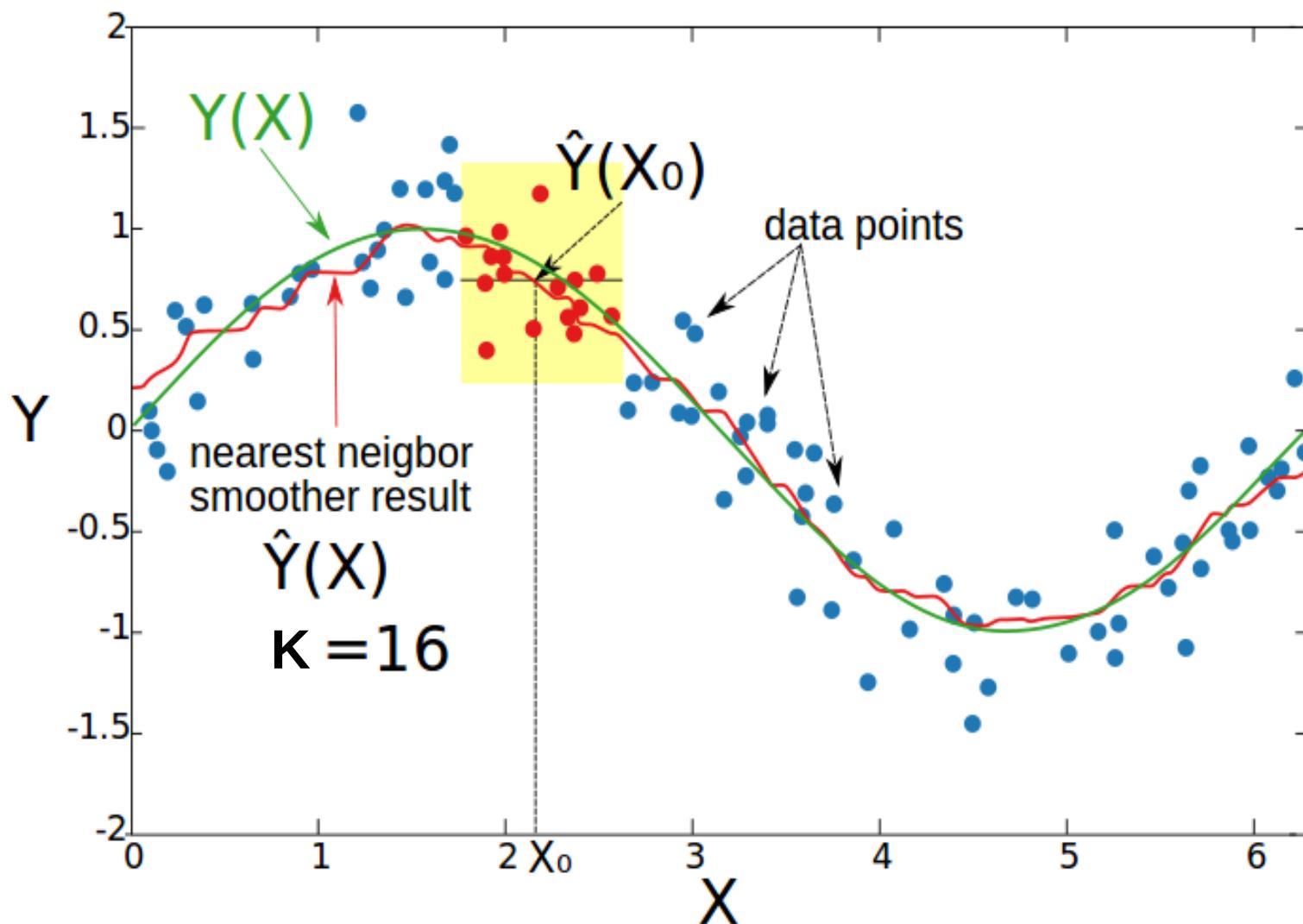
- Nonlinear regression
  - Given  $(x_i, y_i)$ ,  $i=1, \dots, n$ .    Given  $x_0$ .    Find  $y_0$ .
  - Principle 1
    - If we have a very large number of pairs  $(x_j, y_j)$ , where  $x_j = x_0$  (i.e., X coordinates match)
    - Then, we'd like  $y_0 \rightarrow$  average of  $y_j$
  - Principle 2
    - In practice, we may have NO pair  $(x_j, y_j)$ , where  $x_j = x_0$
    - But, we may have pairs  $(x_k, y_k)$ , where  $x_k$  is close to  $x_0$
    - Then, we want  $y_0 \rightarrow$  weighted average of  $y_k$ , where weights are decreasing (non-increasing) with increasing distance  $\|x_0 - x_k\|$

# Patch-Based Filtering

- Nonlinear regression

- K Nearest Neighbor

- Given  $x_0$
    - Find K nearest neighbors  $\{x_j\}$
    - Average associated values  $\{y_j\}$
    - Weights = 0 or  $1/K$  (binary)



# Patch-Based Filtering

- Nonlinear regression
  - Given  $(x_i, y_i)$ ,  $i=1, \dots, n$ .    Given  $x_0$ .    Find  $y_0$ .
  - Kernel regression

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

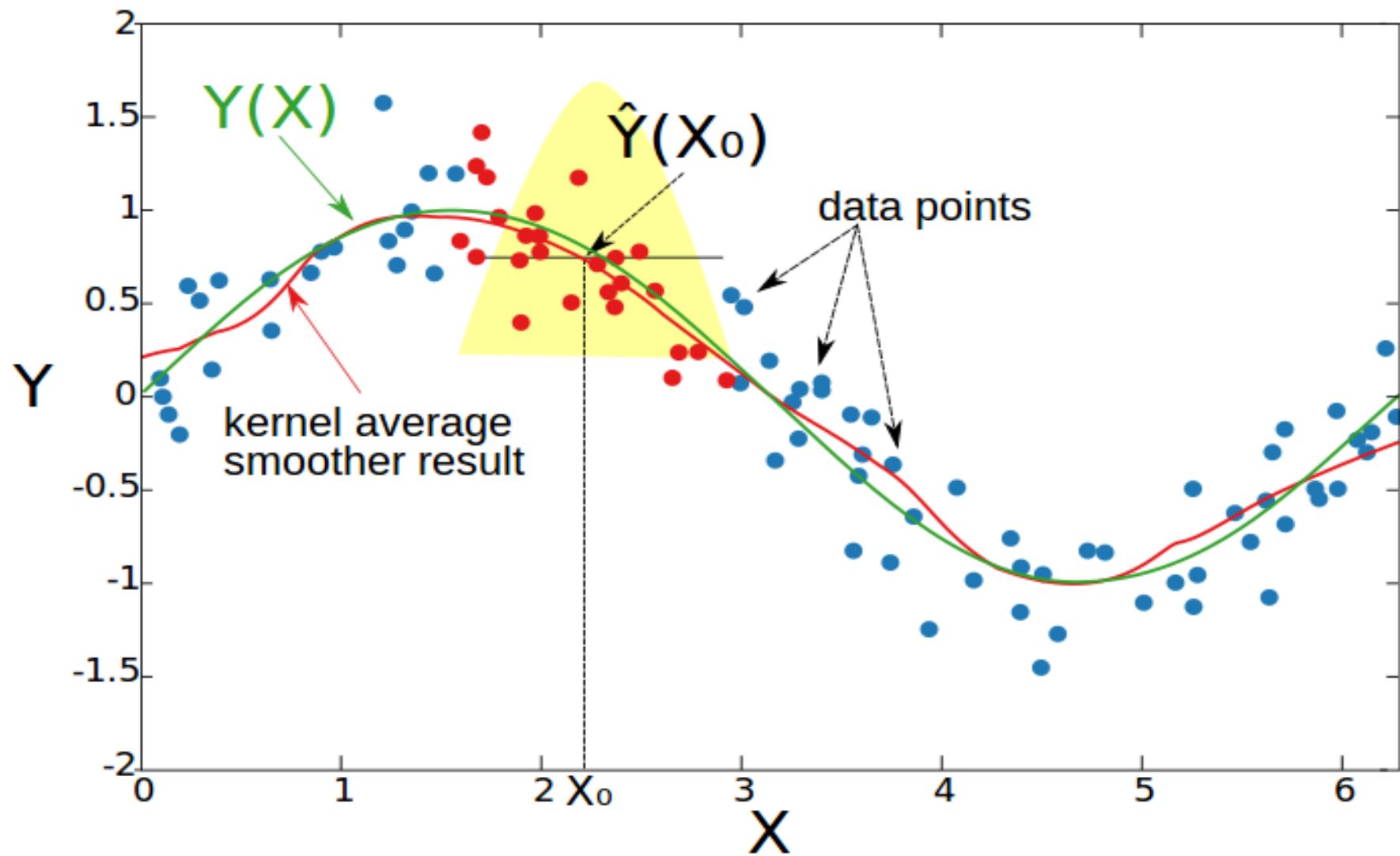
where  $K(\cdot)$  is a kernel with bandwidth  $h$

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

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

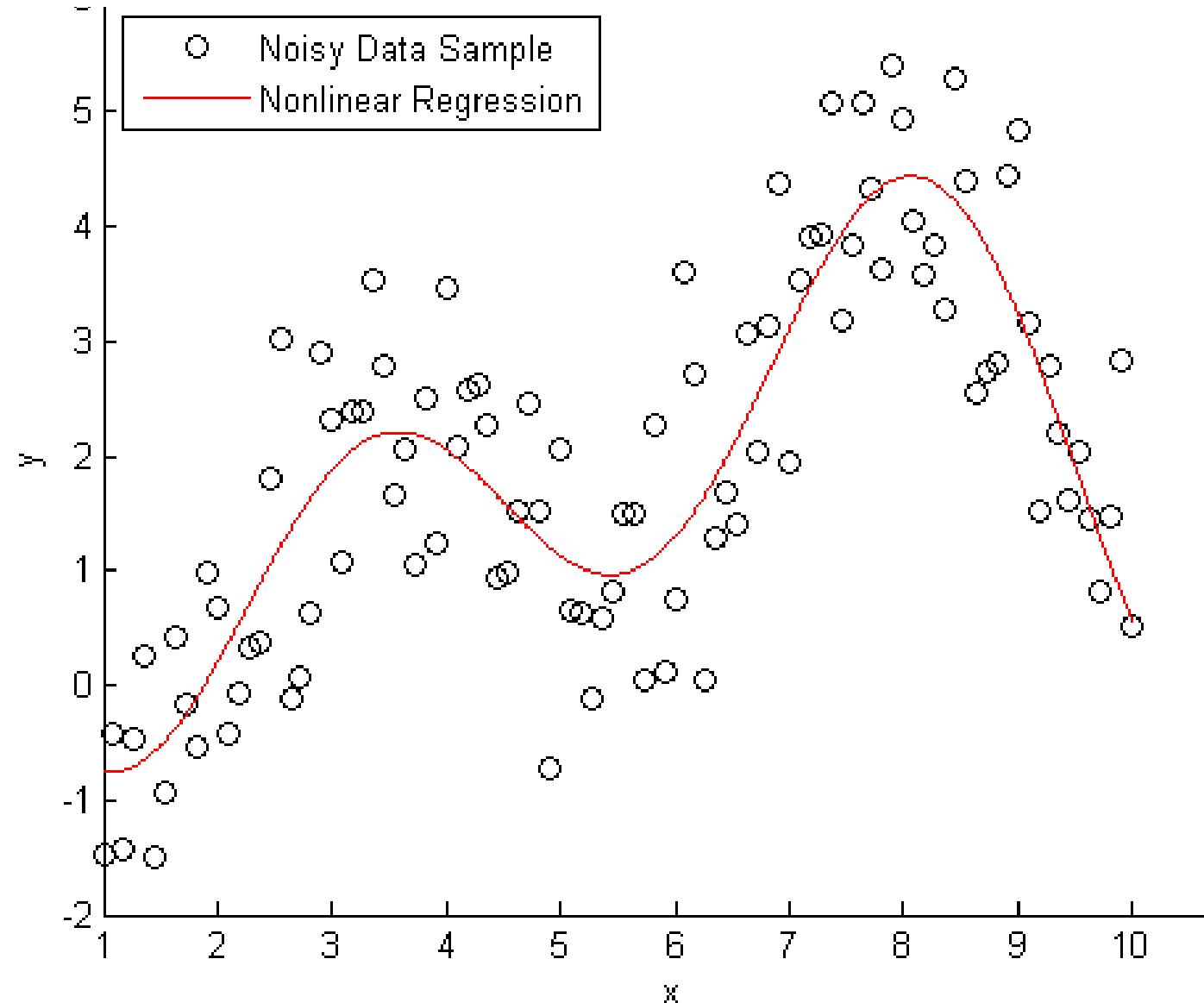
# Patch-Based Filtering

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



# Patch-Based Filtering

- Nonlinear regression
  - How to select standard deviation  $h$  ?
  - What if  $h \rightarrow \infty$  ?
  - What if  $h \rightarrow 0$  ?



# Patch-Based Filter

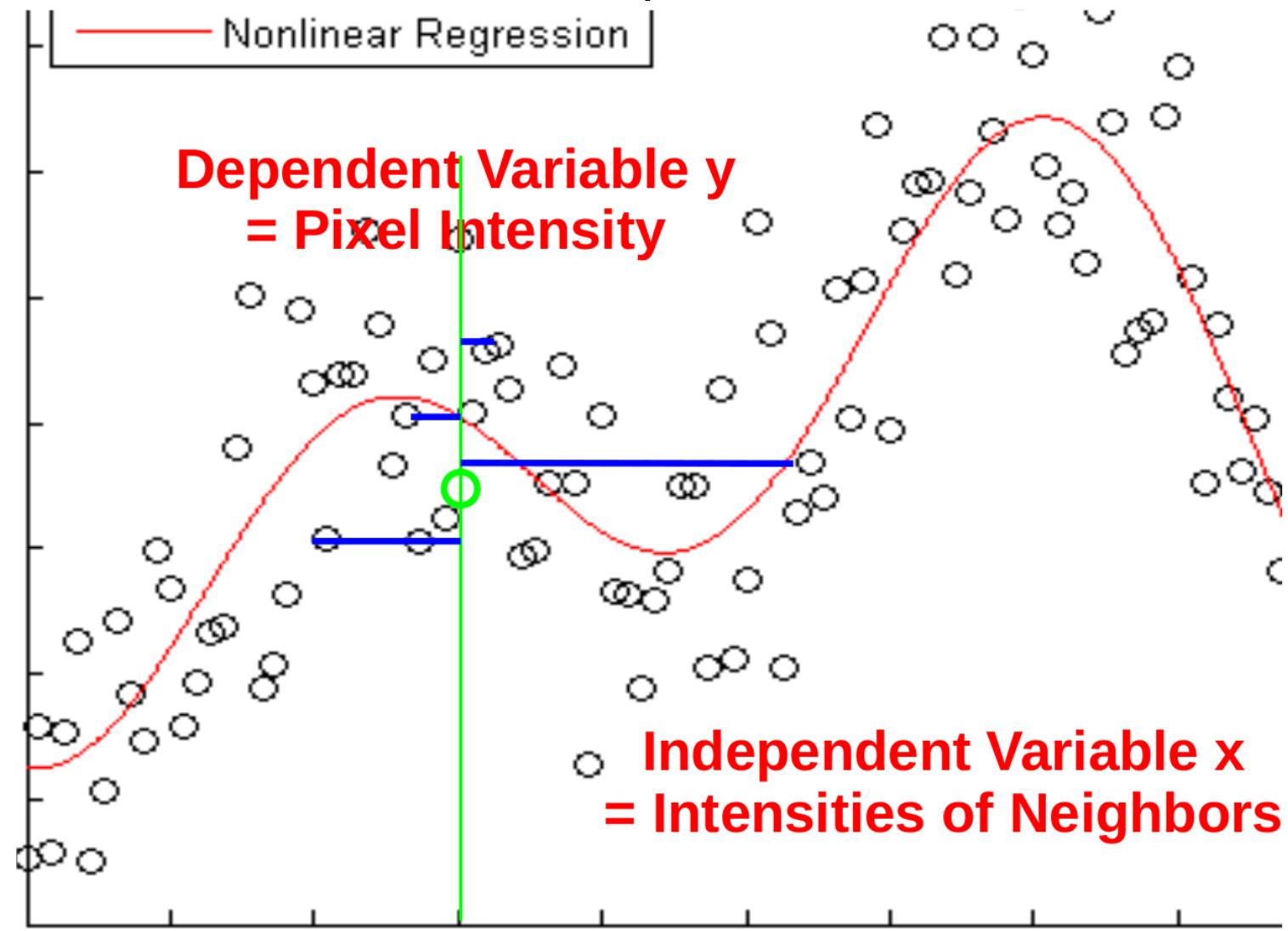
- Image Filtering + Nonlinear:

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

- Weights = **distances between neighborhoods**

- Neighborhood **doesn't** include center pixel !

- How to update **green** patch to reduce scatter ?



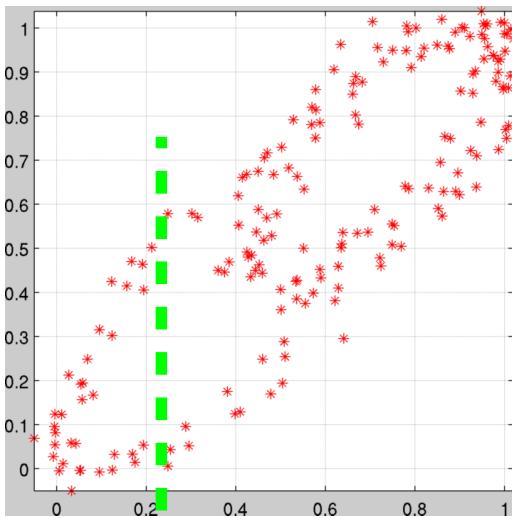
# Patch-Based Filter

- Image Filtering + Nonlinear

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

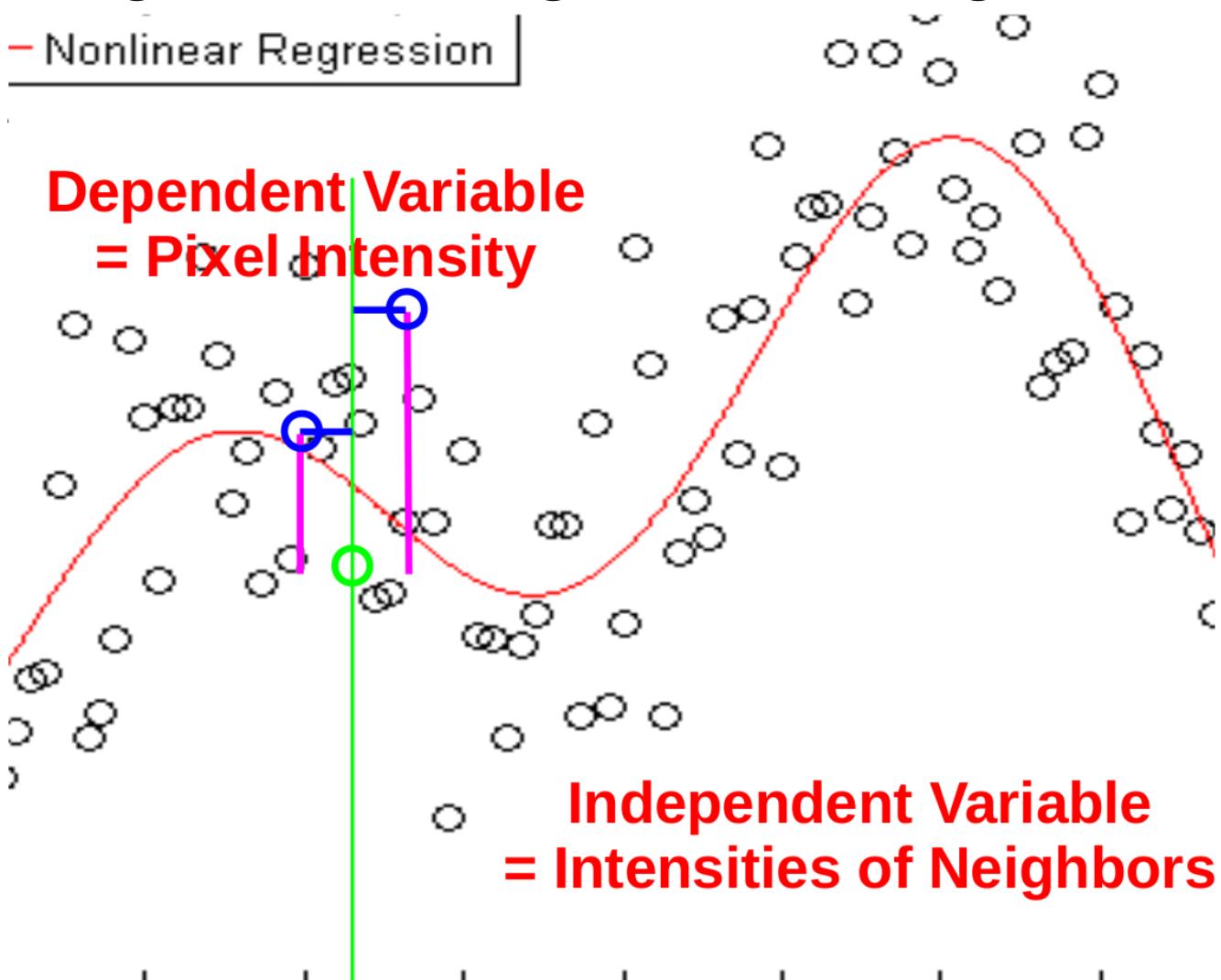
- How to update **green** patch to reduce scatter ?
- **Why should blue neighborhoods get same weight ?**

- One **blue** patch has **center pixel** intensity that is **closer** to center pixel intensity in green patch



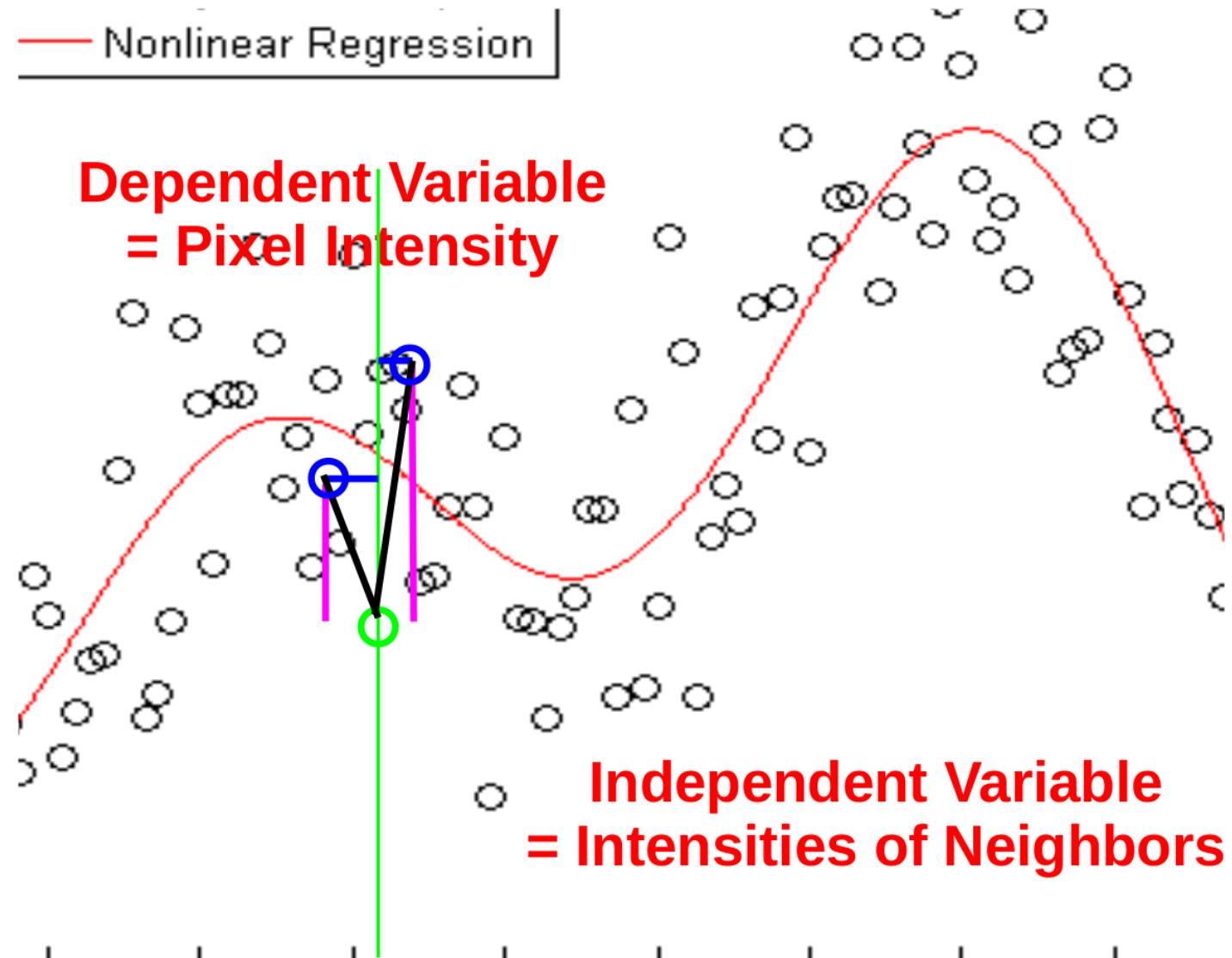
— Nonlinear Regression

**Dependent Variable  
= Pixel Intensity**



# 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 ?
    - 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)

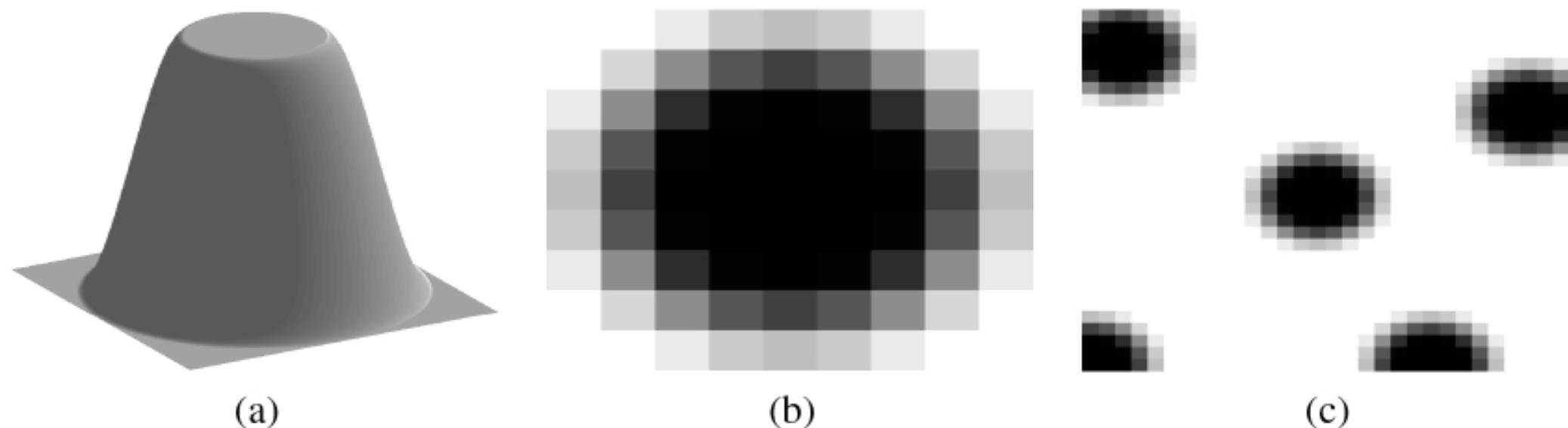
$$\sum_{j \in \Lambda} \omega_h(j, k) f(j)$$

# 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”



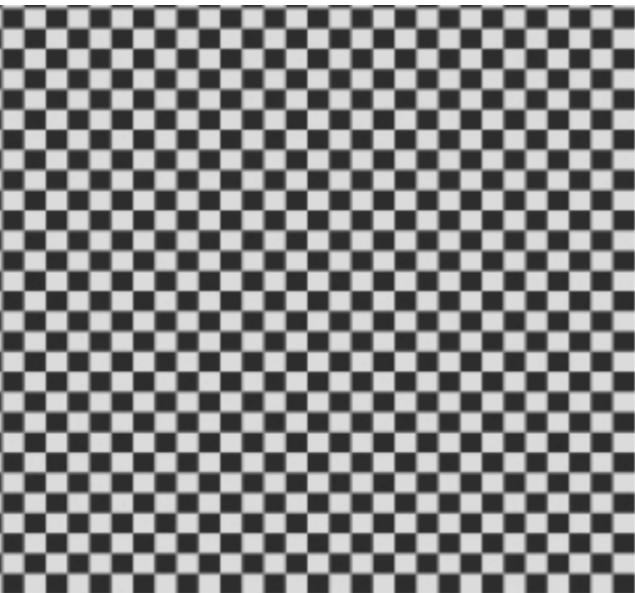
**Figure 3.3.** Neighborhood shapes. (a) Preserving rotational invariance via a neighborhood mask consisting of a flat central circular plateau with cubic splines on the sides. (b) The discrete sampling of the mask ( $\text{black} \equiv 1$ ,  $\text{white} \equiv 0$ ) for a  $9 \times 9$  pixels neighborhood.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

# Patch-Based Filtering

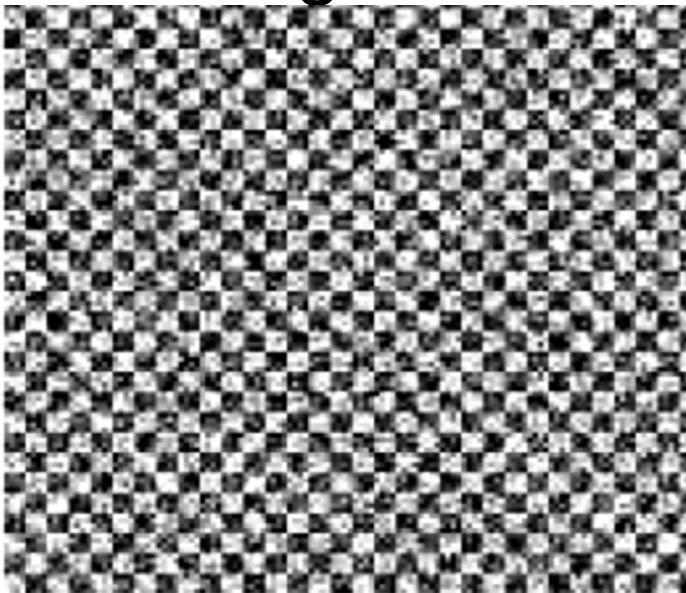
- Computational cost
  - Image  $M \times N$
  - Patch  $P \times Q$
  - Window for searching patches  $R \times S$
  - Number of operations =  $MN PQ RS$
- There are ways to make this fast,  
while preserving accuracy
  - Non-trivial

# Patch-Based Filtering

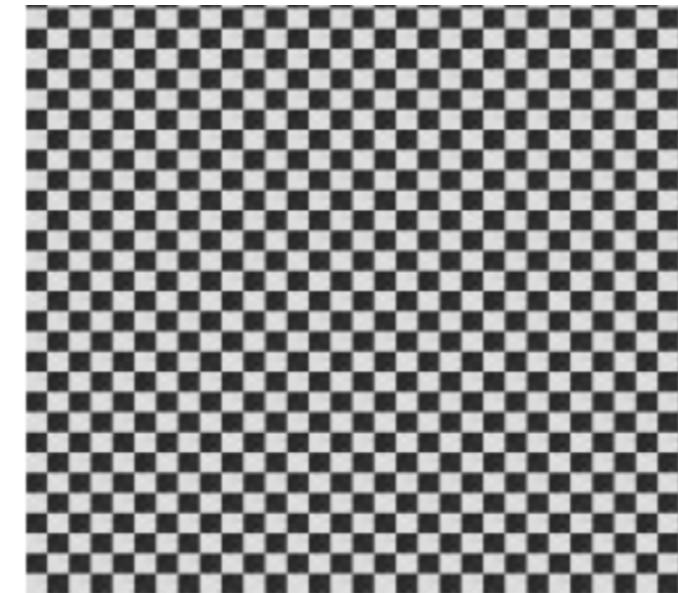
- Results on textured images



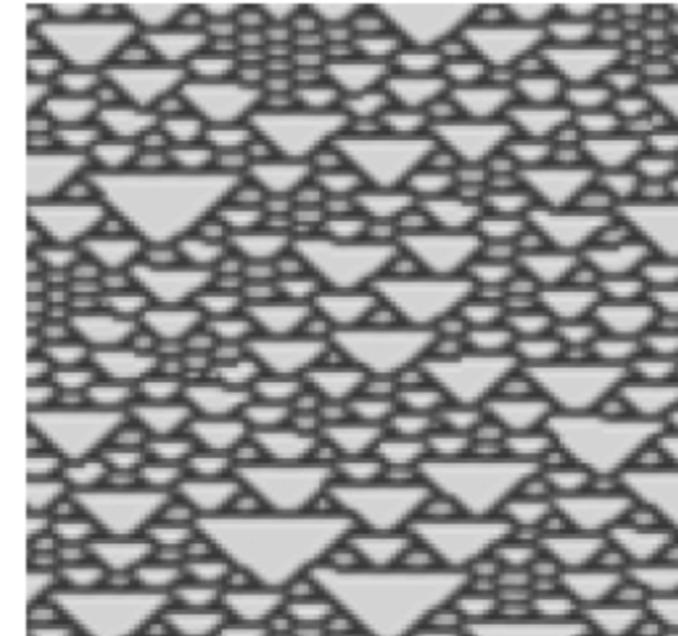
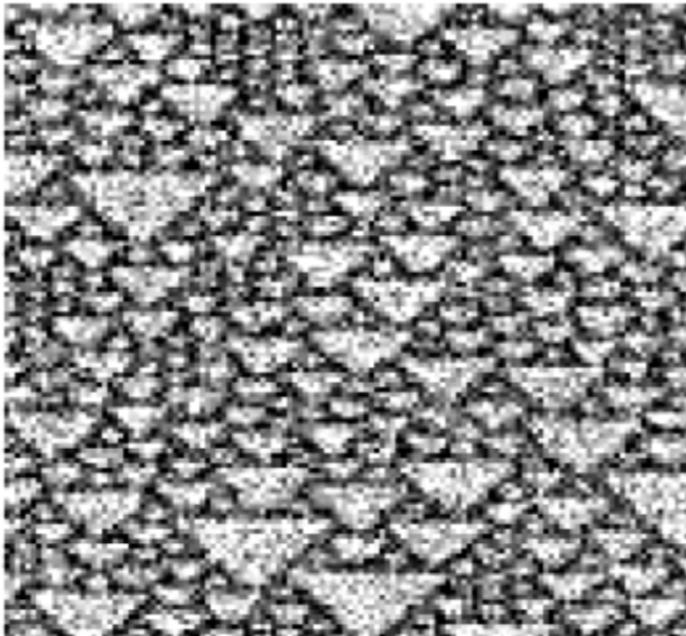
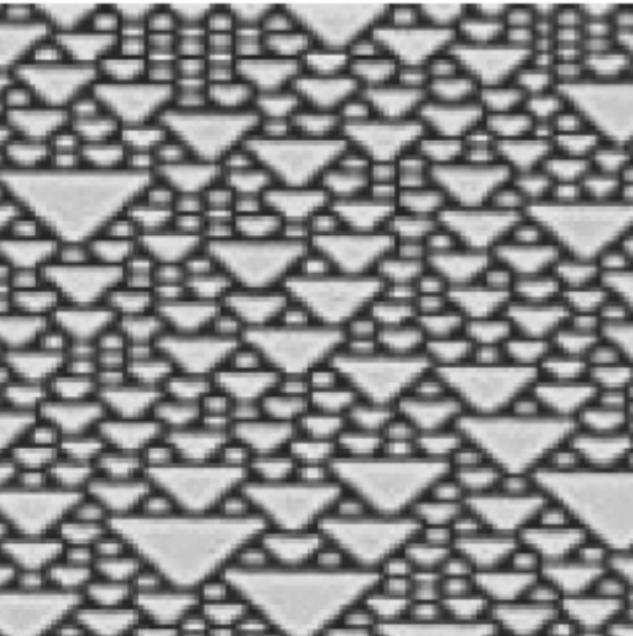
(a)



(b)

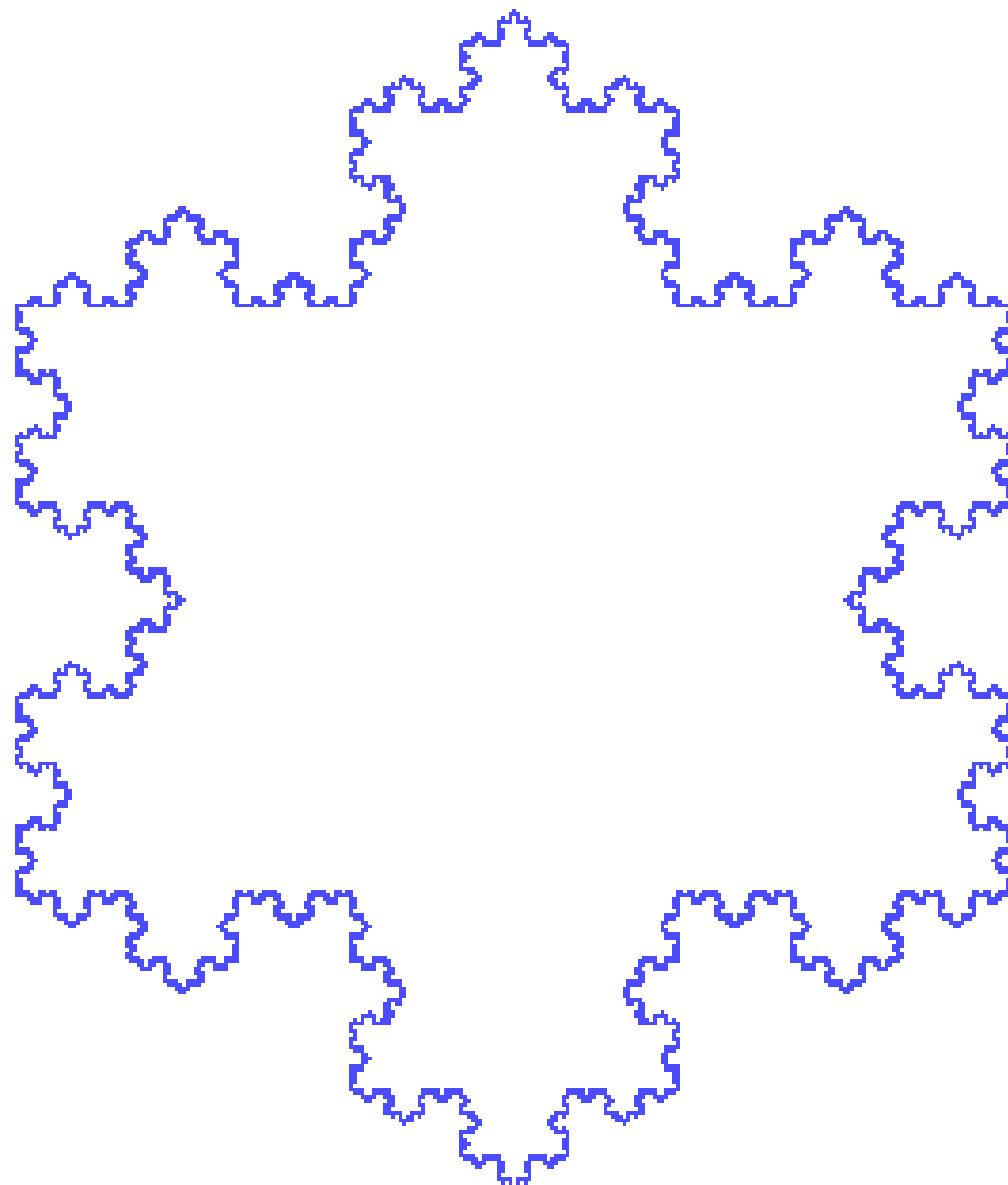
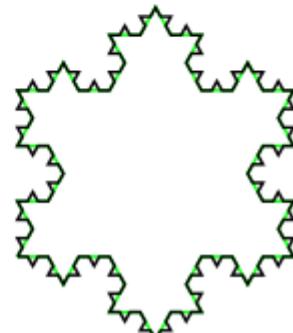
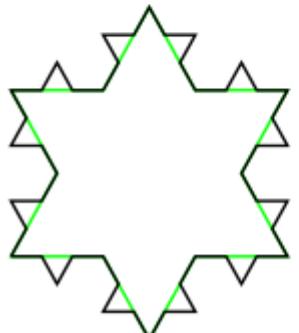
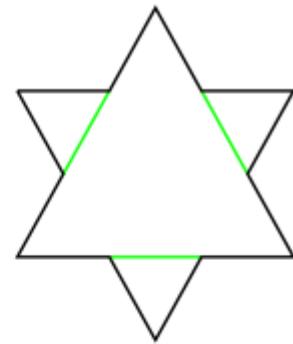
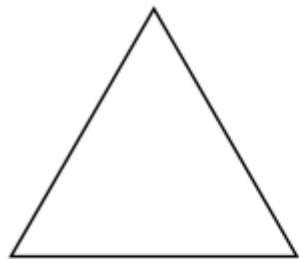


(c)



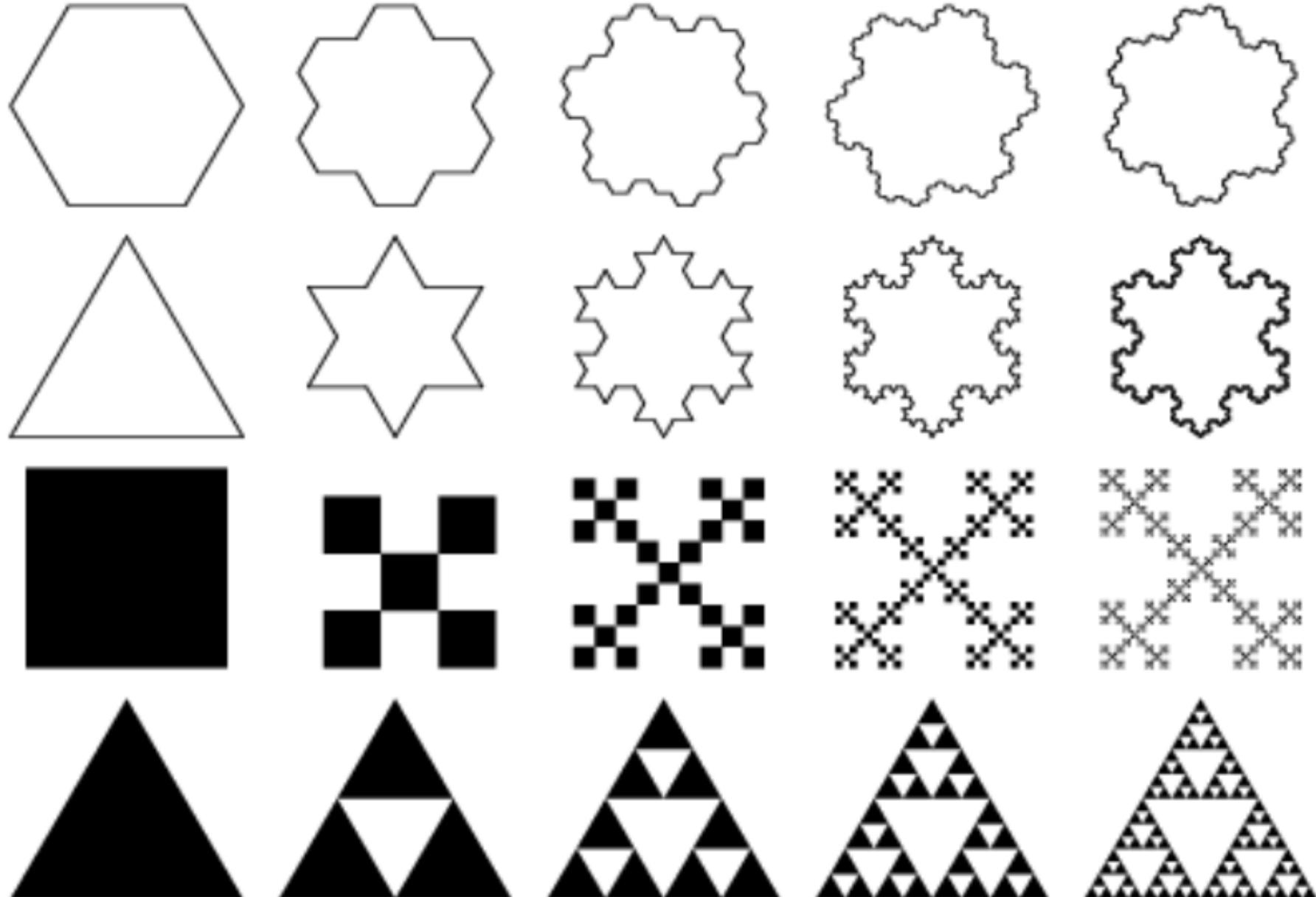
# Patch-Based Filtering

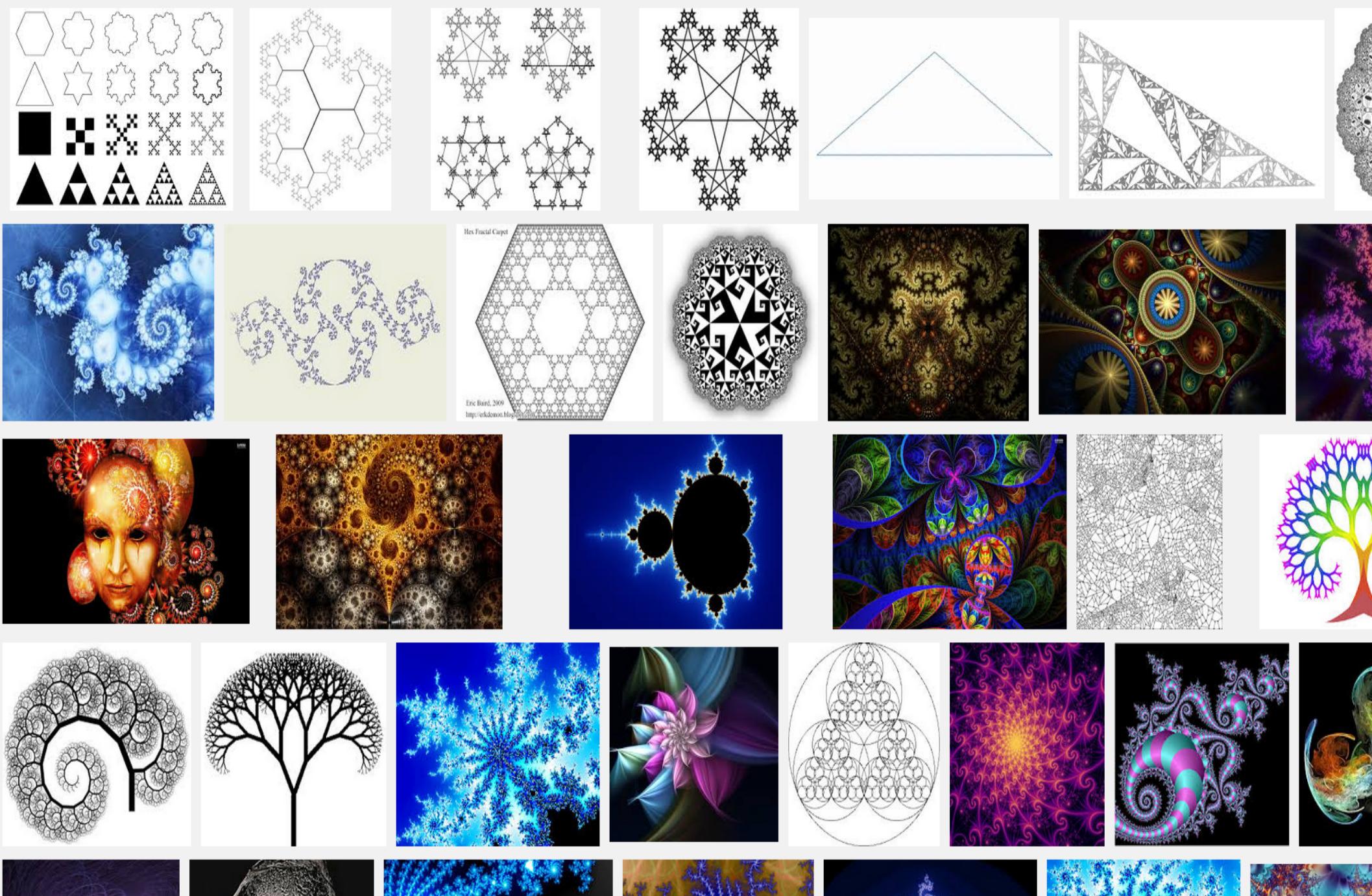
- Fractal = repeating pattern (texture) at all scales
  - Koch snowflake



# Patch-Based Filtering

- Fractal = repeating pattern (texture) at all scales





# Patch-Based Filtering

- Fractals in nature (snowflakes)



# Patch-Based Filtering

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



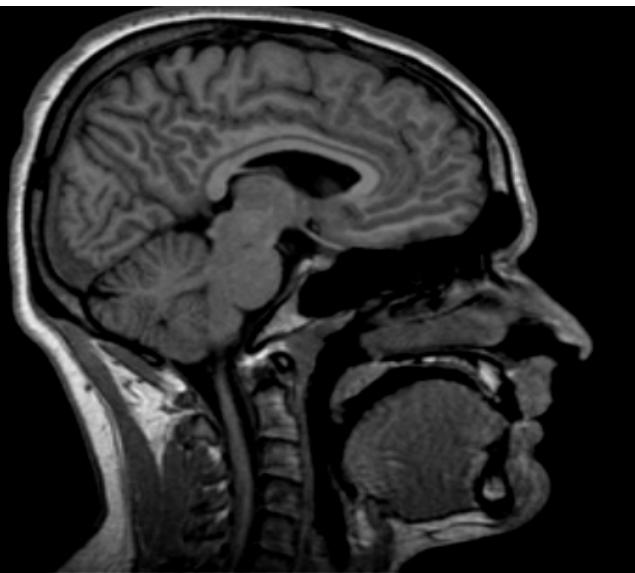
# Patch-Based Filtering

- Fractals in nature (Romanesque cauliflower)



# Patch-Based Filtering

- Results on textured images



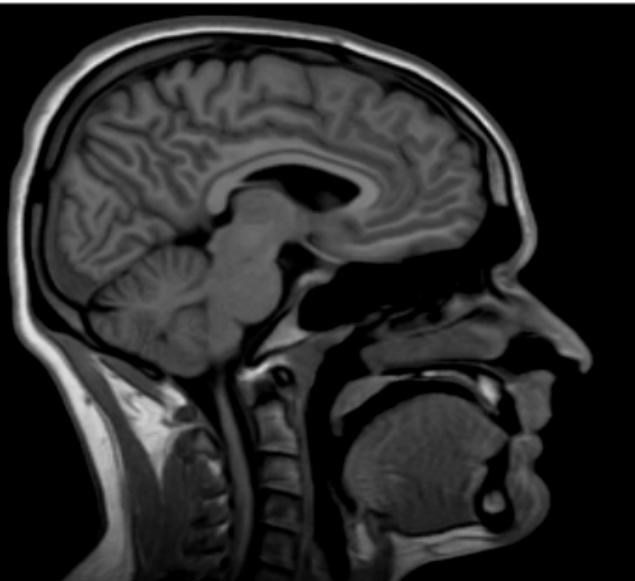
(a2)



(c2)

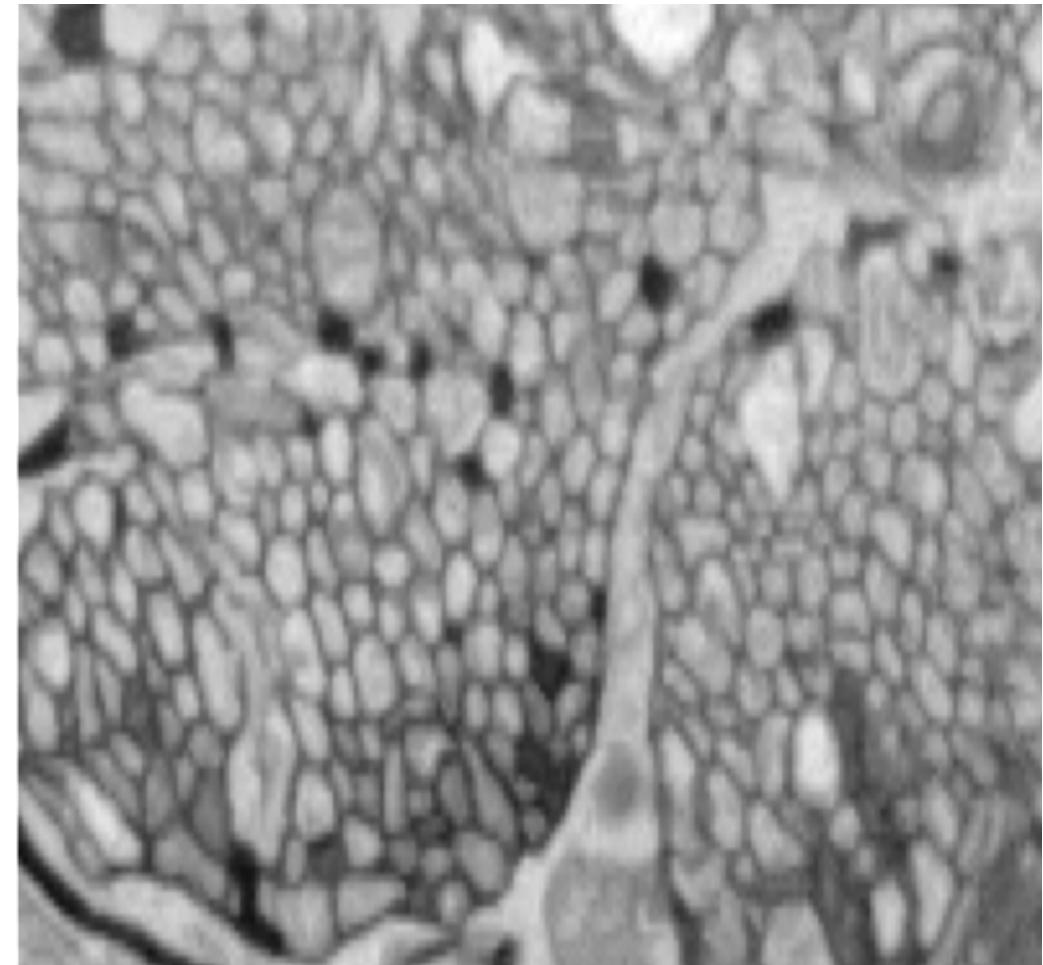
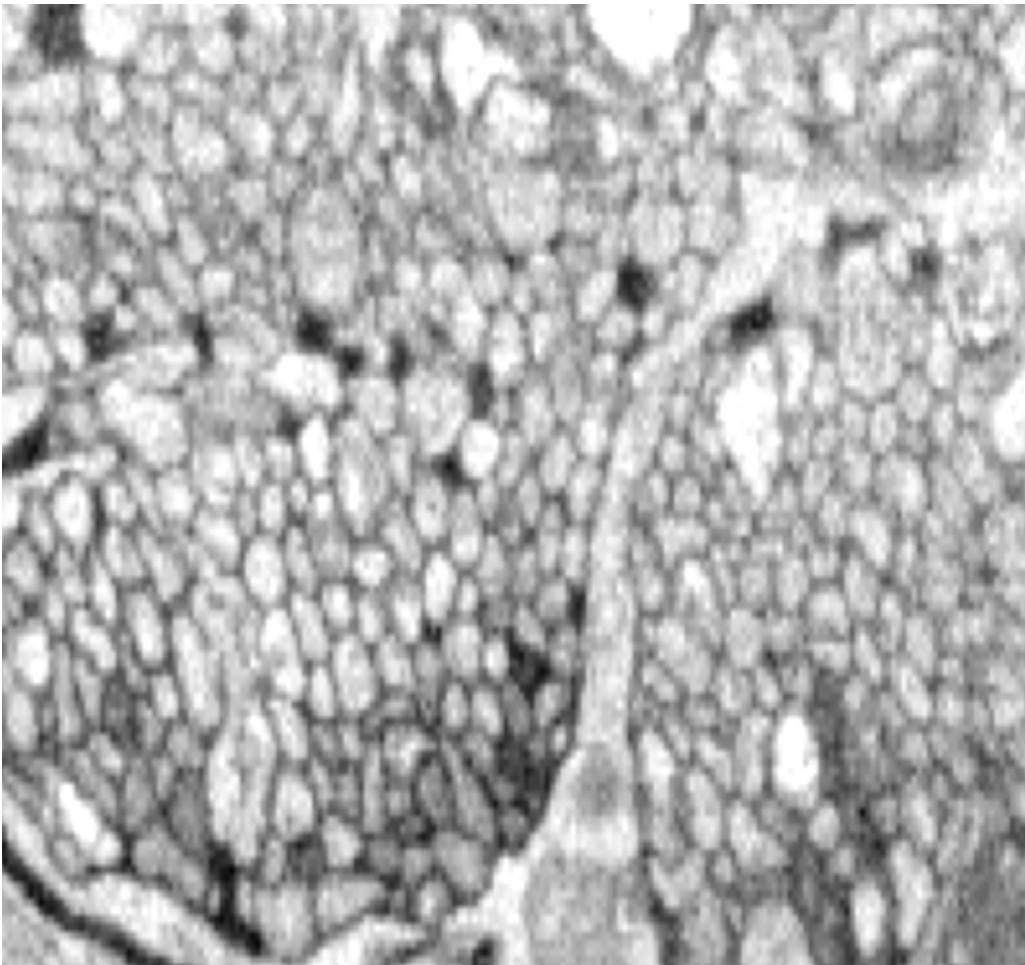


(e2)



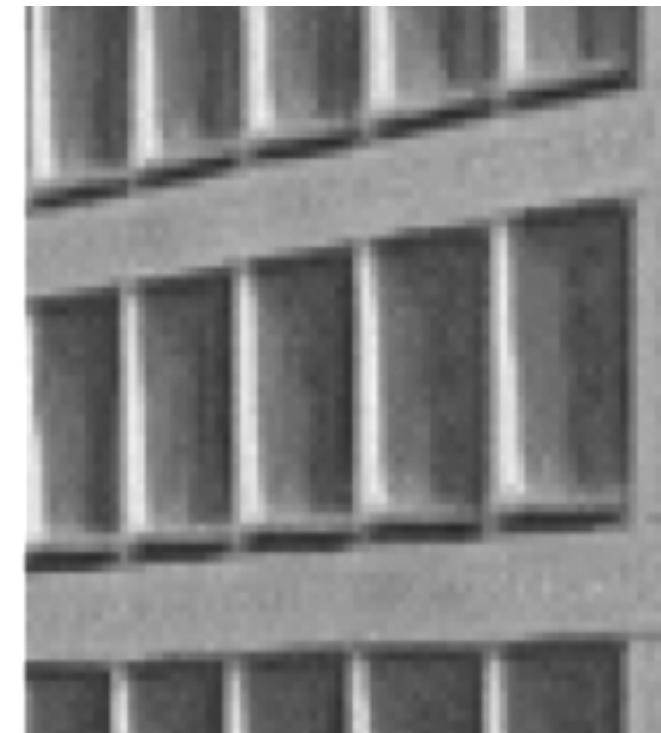
# Patch-Based Filtering

- Results on textured images



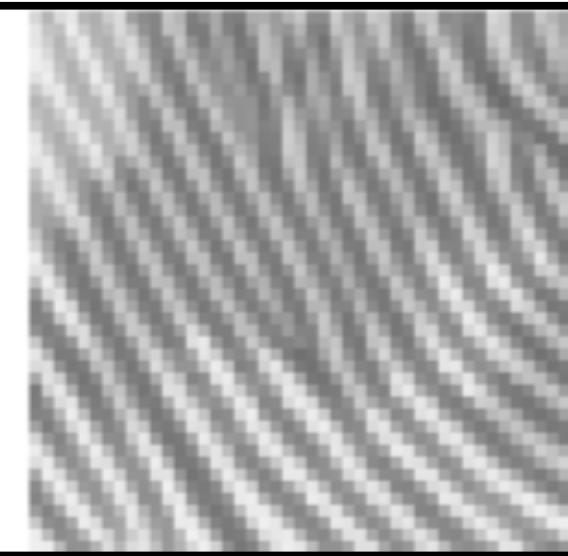
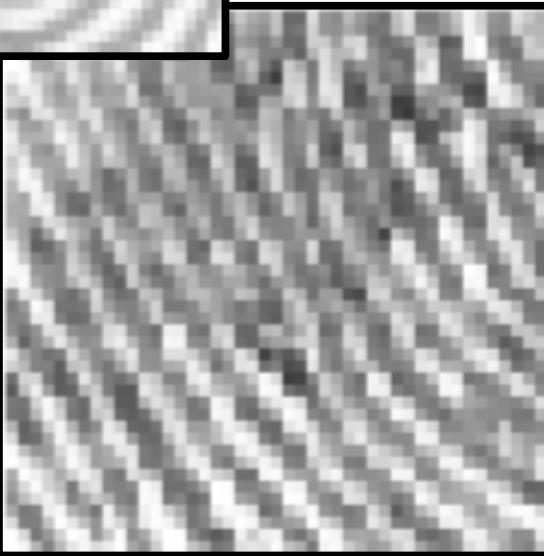
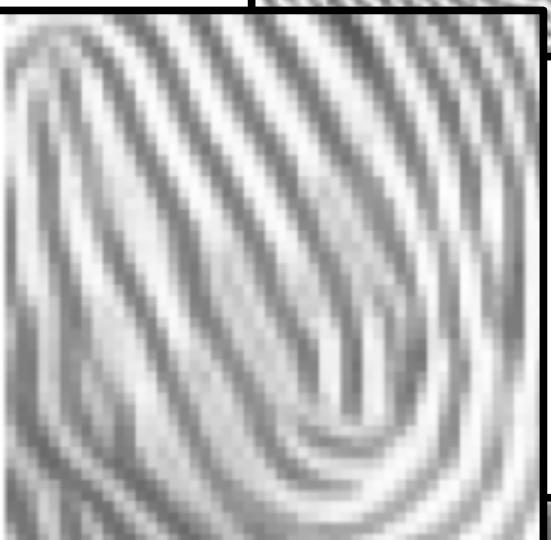
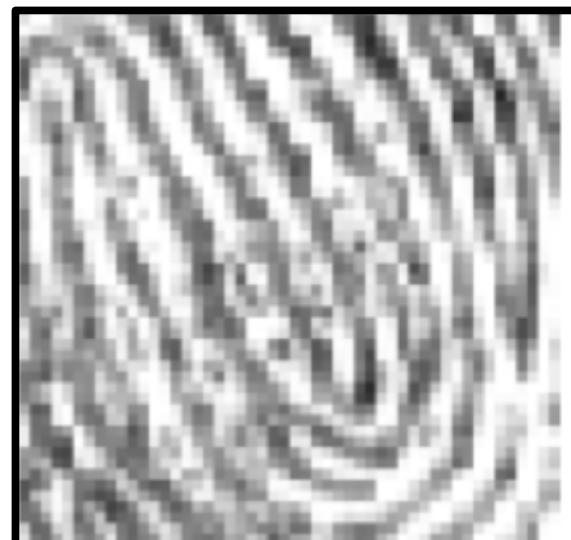
# Patch-Based Filtering

- 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)



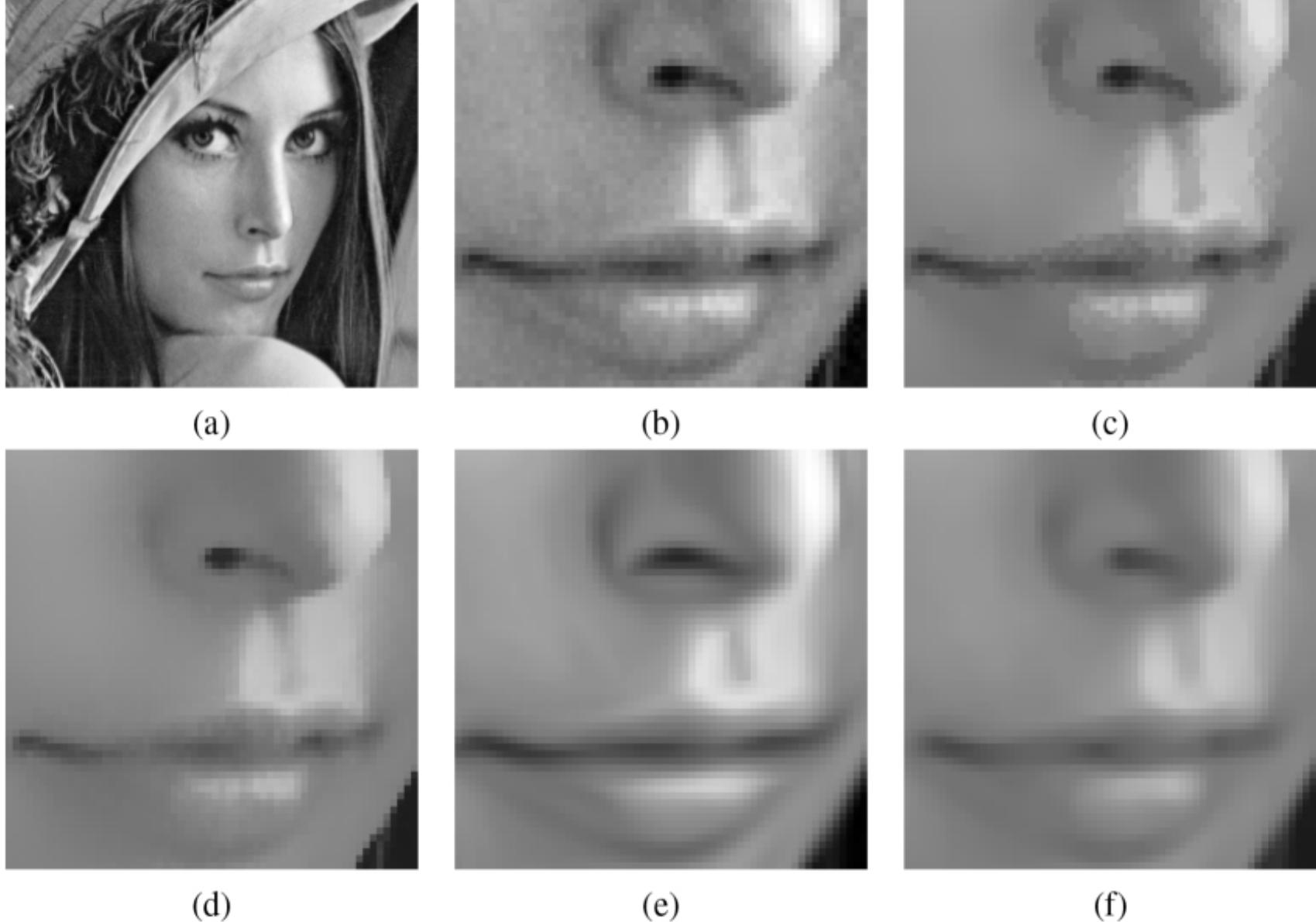
(c1)



(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:  $\sigma_{\text{domain}}=3$  pixels,  $\sigma_{\text{range}}=12$  G.U., (e) coherence-enhancing diffusion:  $\sigma=0.1$  pixels,  $\rho=2$  pixels,  $\alpha=0.0001$ ,  $C=0.0001$ , 15 iterations, and (f) curvature flow: time step=0.2, 8 iterations.