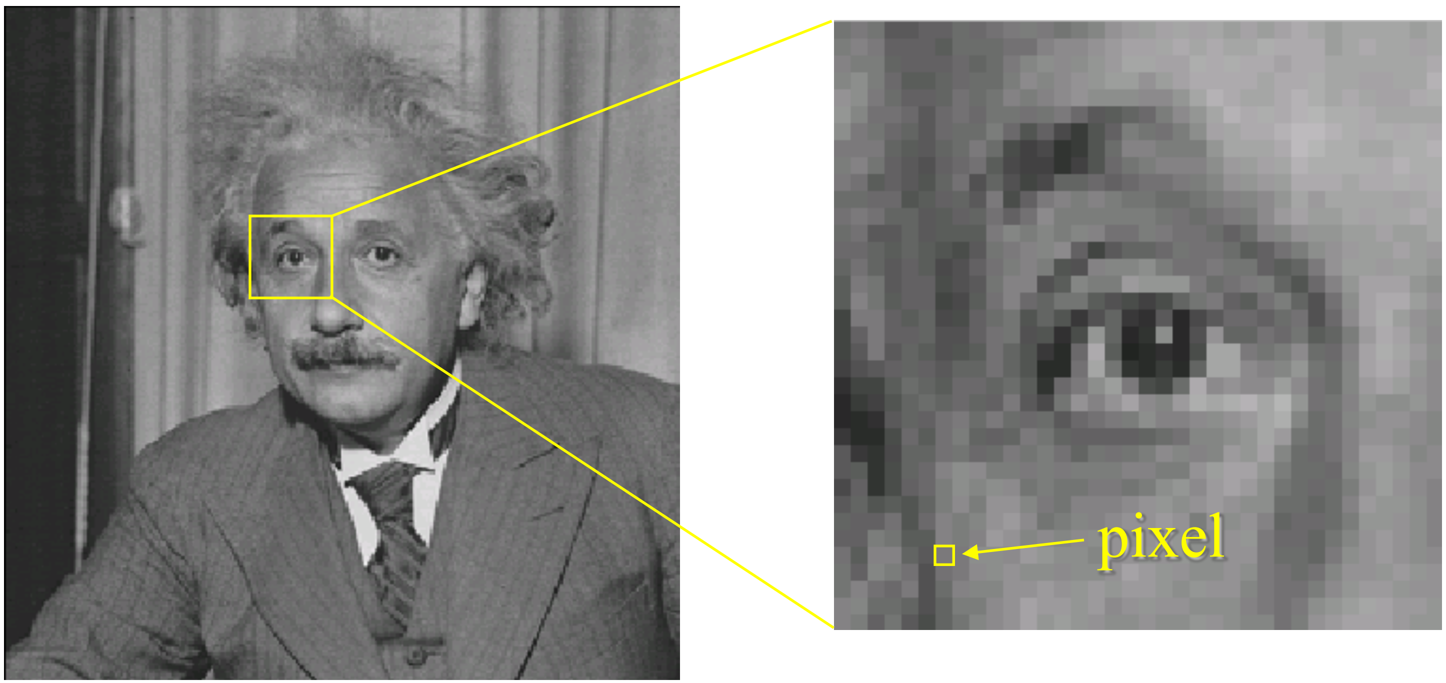
Topic 4:

Local analysis of image patches

- What do we mean by an image “patch”?
- Applications of local image analysis
- Visualizing 1D and 2D intensity functions

Local Image Patches

So far, we have considered pixels completely independently of each other (as a 2D array of numbers or RGB values)



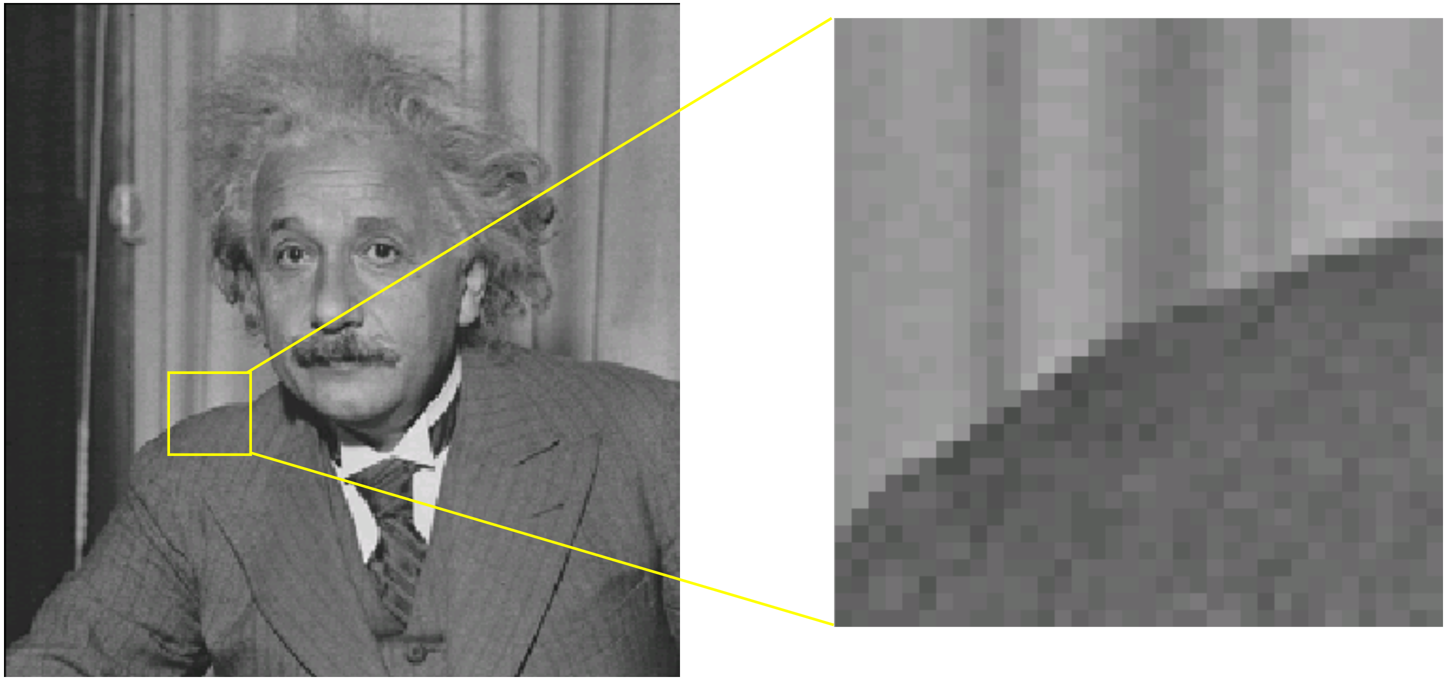
In reality, photos have a great deal of structure

This structure can be analyzed at a local level (eg., small groups of nearby pixels) or a global one (eg. entire image)

Local Image Patches

There are many different types of patches in an image

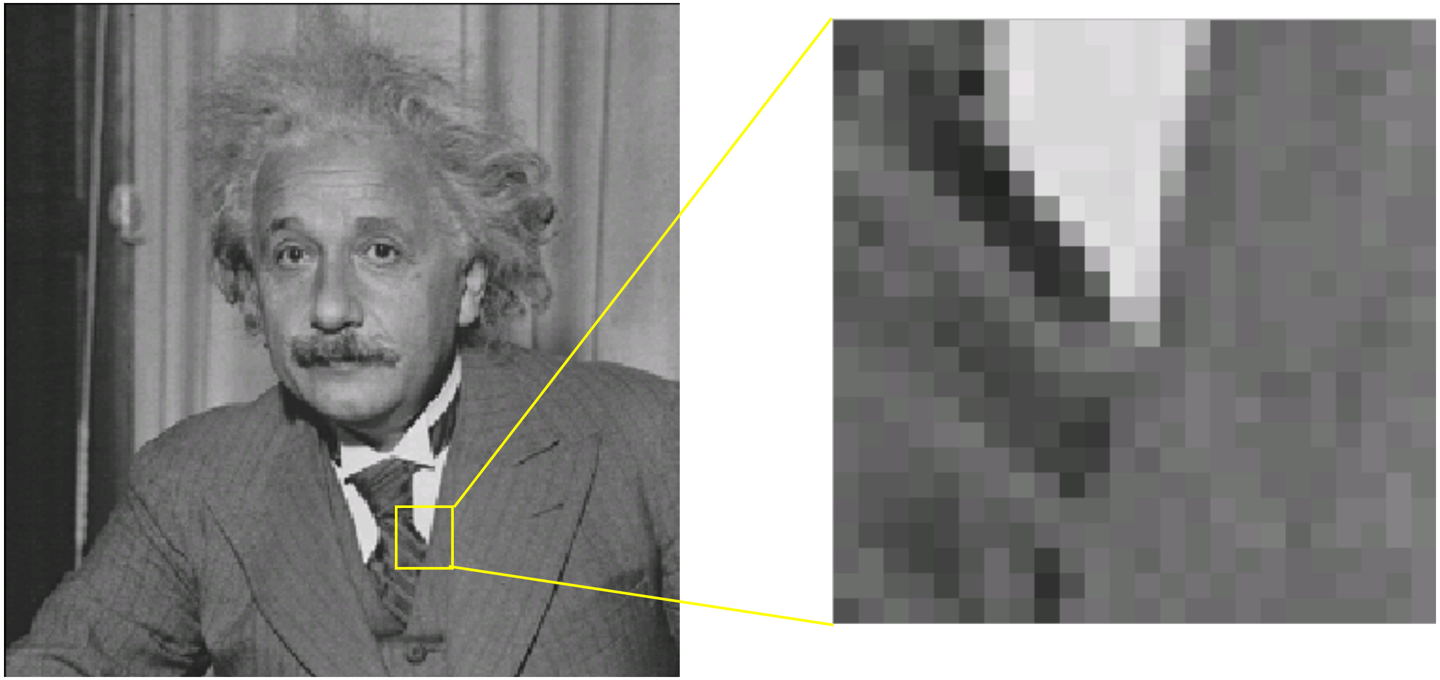
Patches corresponding to an “edge” in the image



Local Image Patches

There are many different types of patches in an image

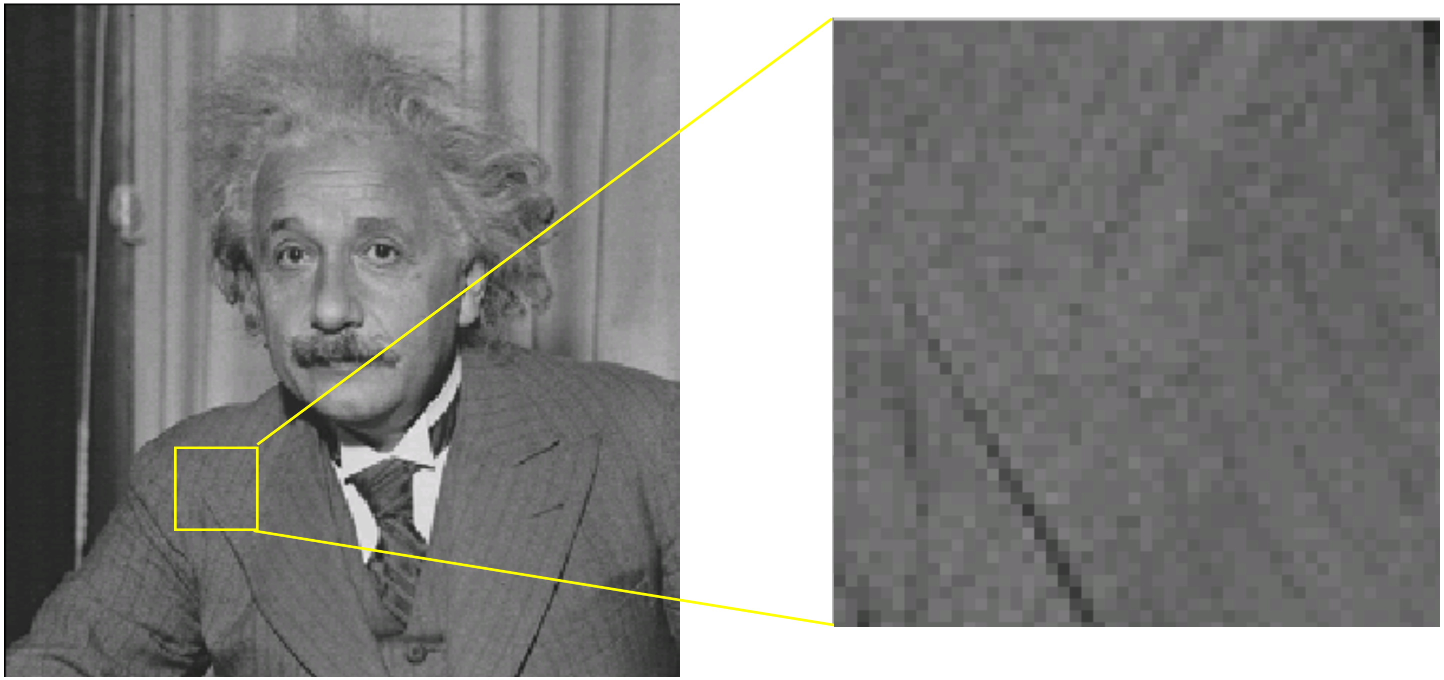
Patches corresponding to a “corner” in the image



Local Image Patches

There are many different types of patches in an image

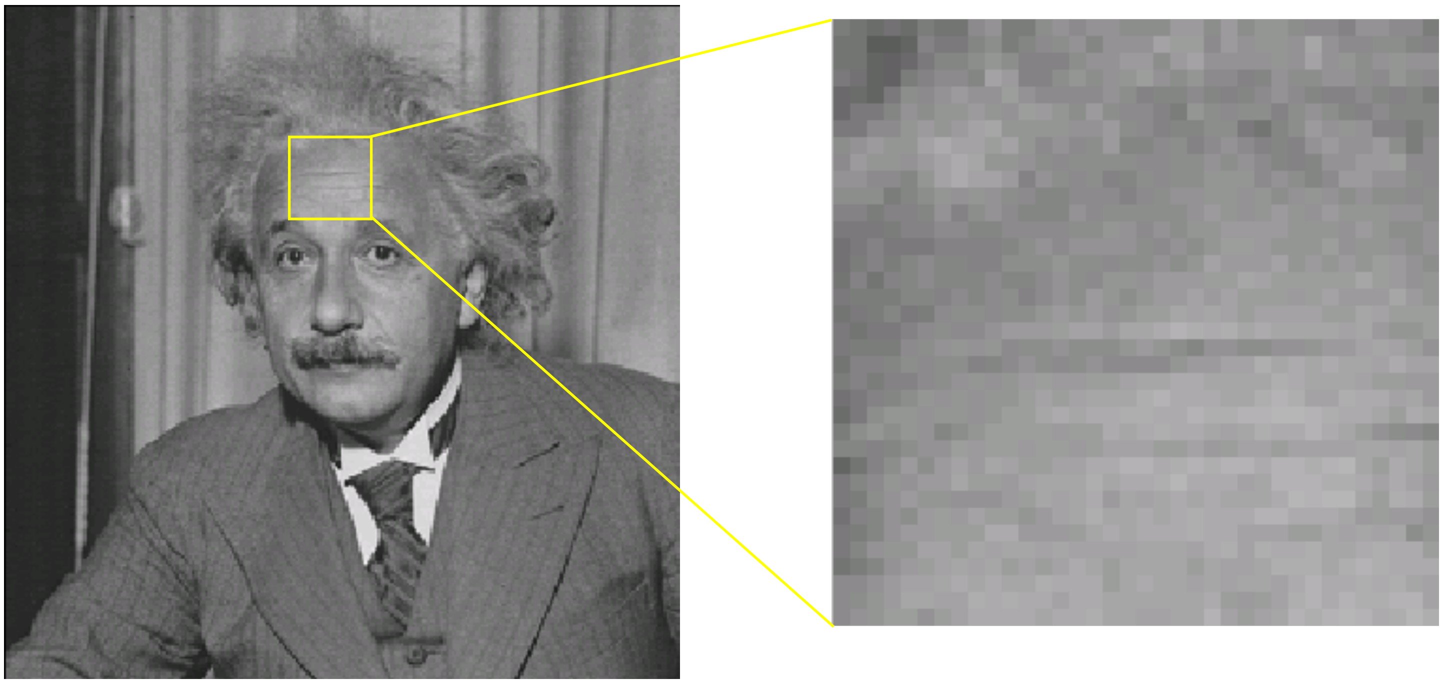
Patches of uniform texture



Local Image Patches

There are many different types of patches in an image

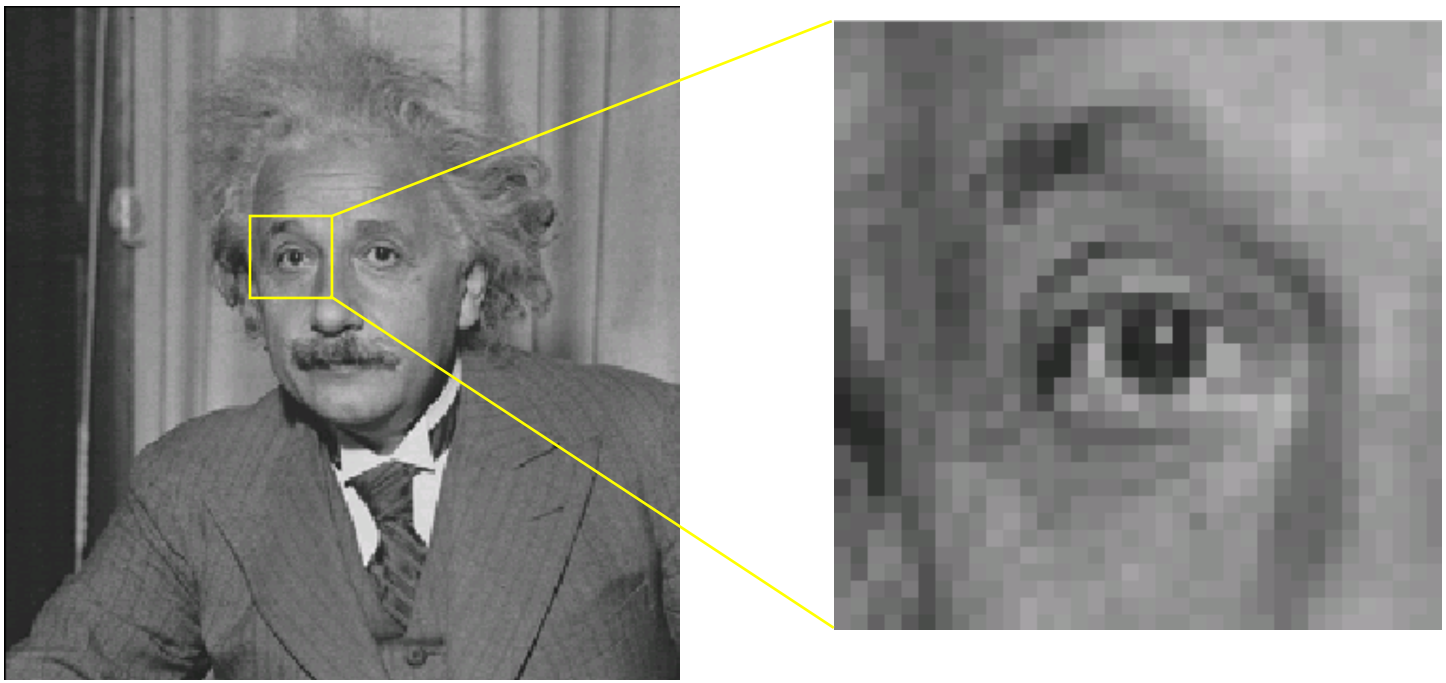
Patches associated with a single surface



Local Image Patches

There are many different types of patches in an image

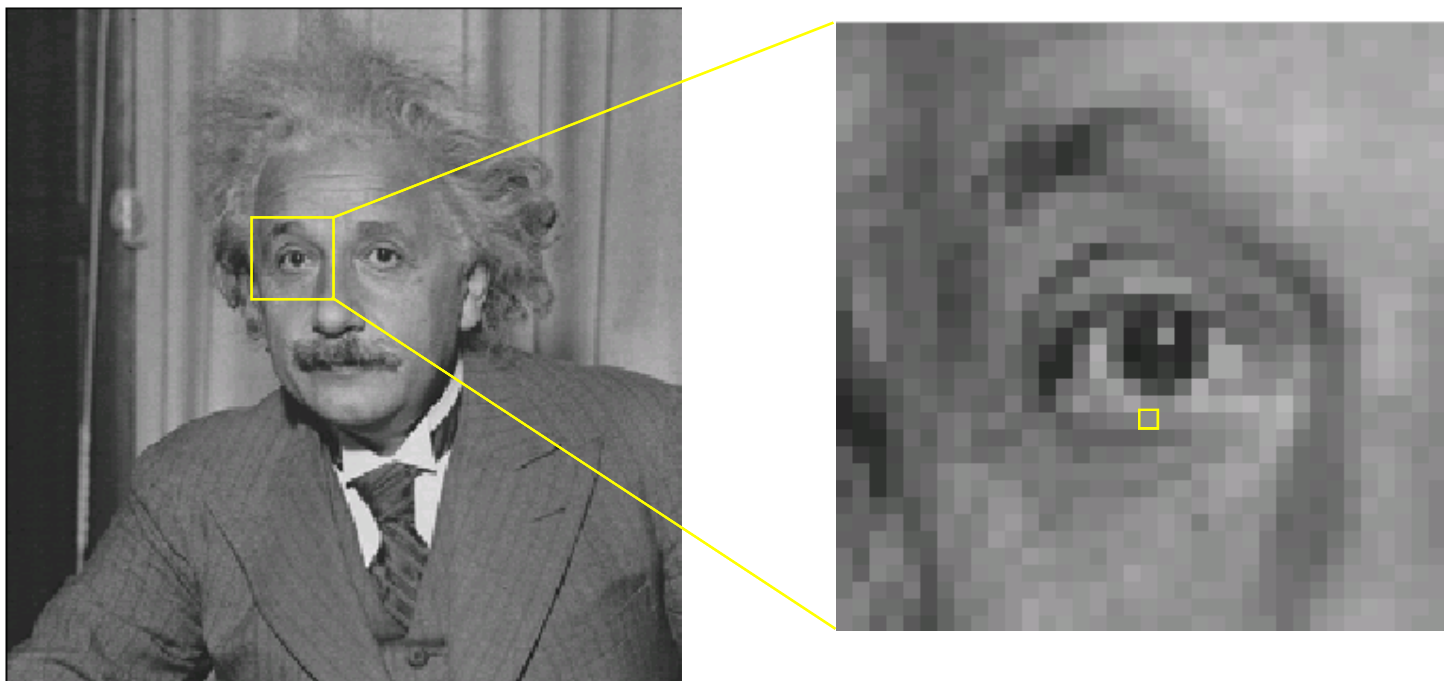
Perceptually-significant “features”



Local Image Patches

When is a group of pixels considered a local patch?

There is no answer to this question!

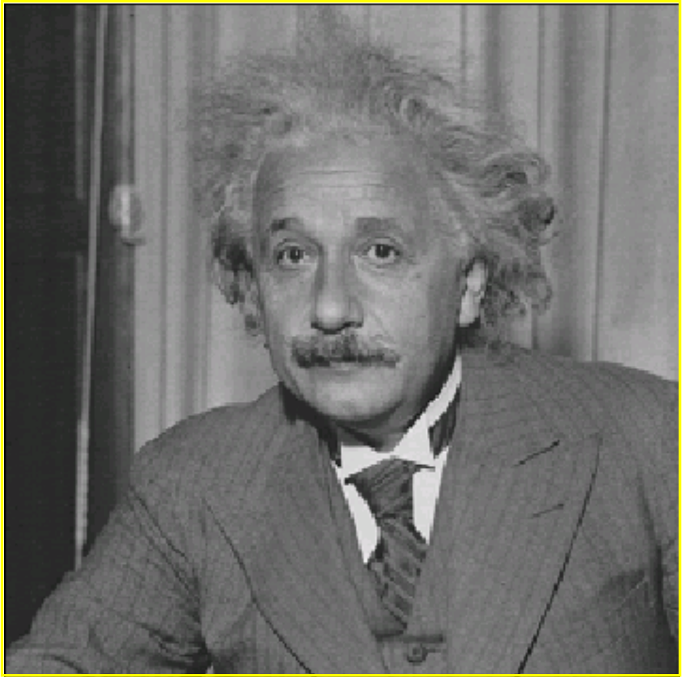


The notion of a patch is relative---it can be a single pixel

Local Image Patches

When is a group of pixels considered a local patch?

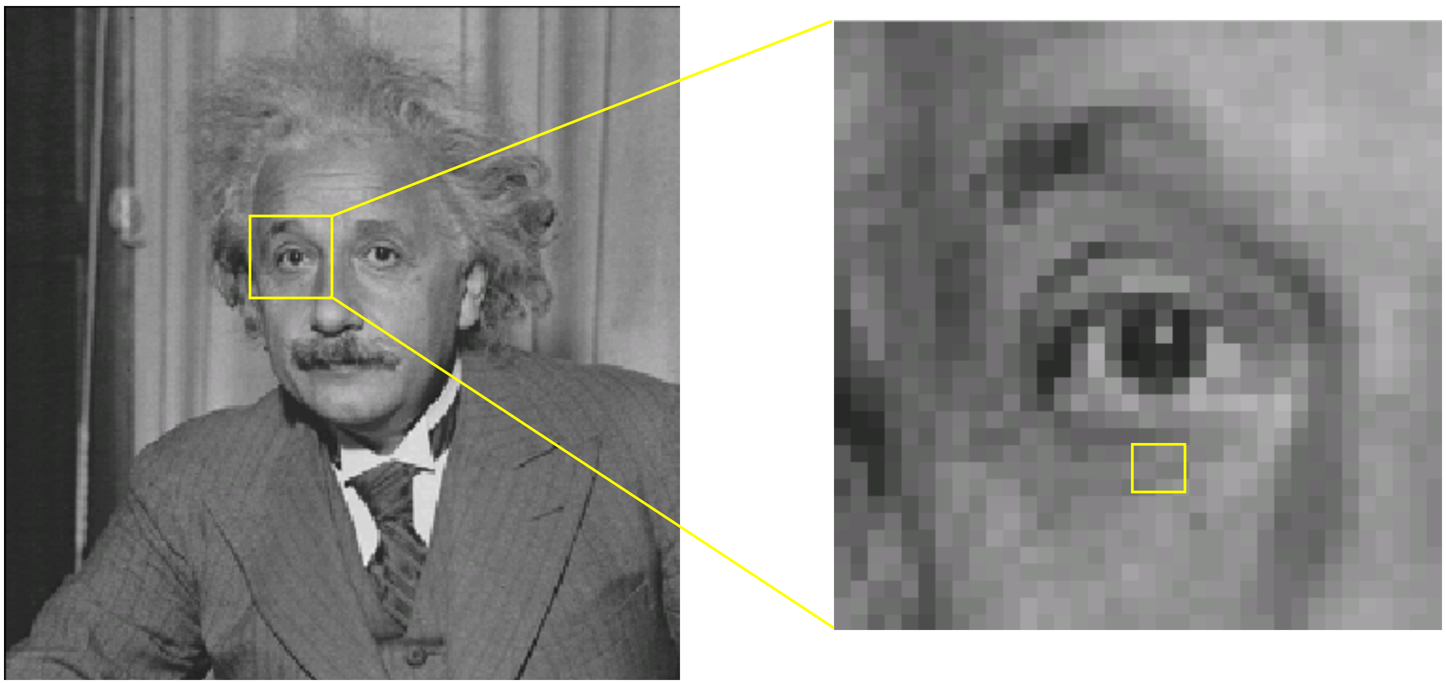
There is no answer to this question!



The notion of a patch is relative---it can be the entire image

Local Image Patches

We will begin with mathematical descriptions that apply mostly to very small patches (e.g., 3x3)



… and eventually consider descriptions that apply to entire images

Topic 4:

Local analysis of image patches

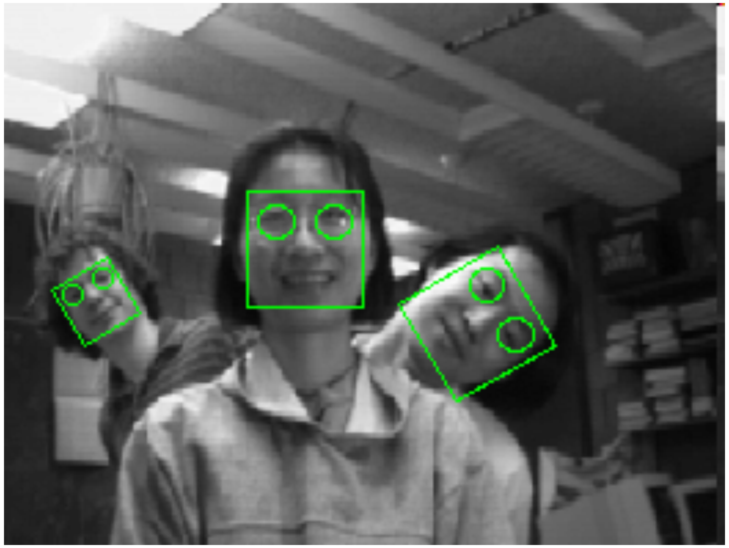
- What do we mean by an image "patch"?
- Applications of local image analysis
- Visualizing 1D and 2D intensity functions

Why Do We Care?

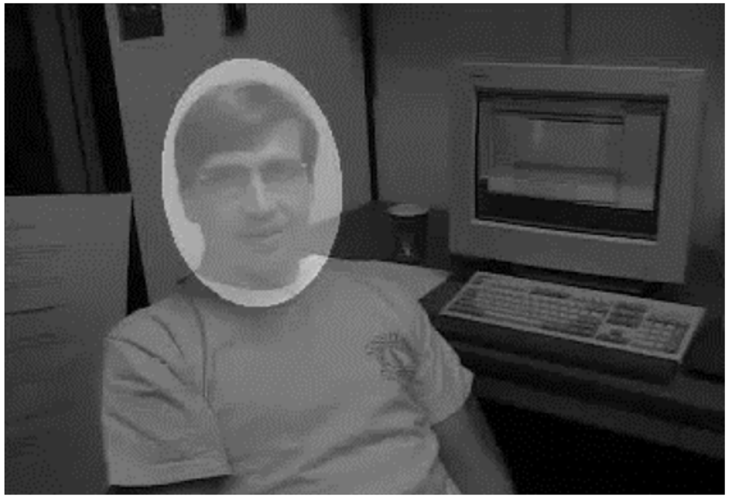
Many applications…

- Recognition
- Inspection
- Video-based tracking
- Special effects

Recognition & Tracking



(Rowley et al, PAMI'98)



(El-Maraghi et al, CVPR'01)

Editing & Manipulating Photos

Object removal from a photo

Original

New

(Criminisi et al, CVPR 2003)

Editing & Manipulating Photos

Colorization of black and white photos

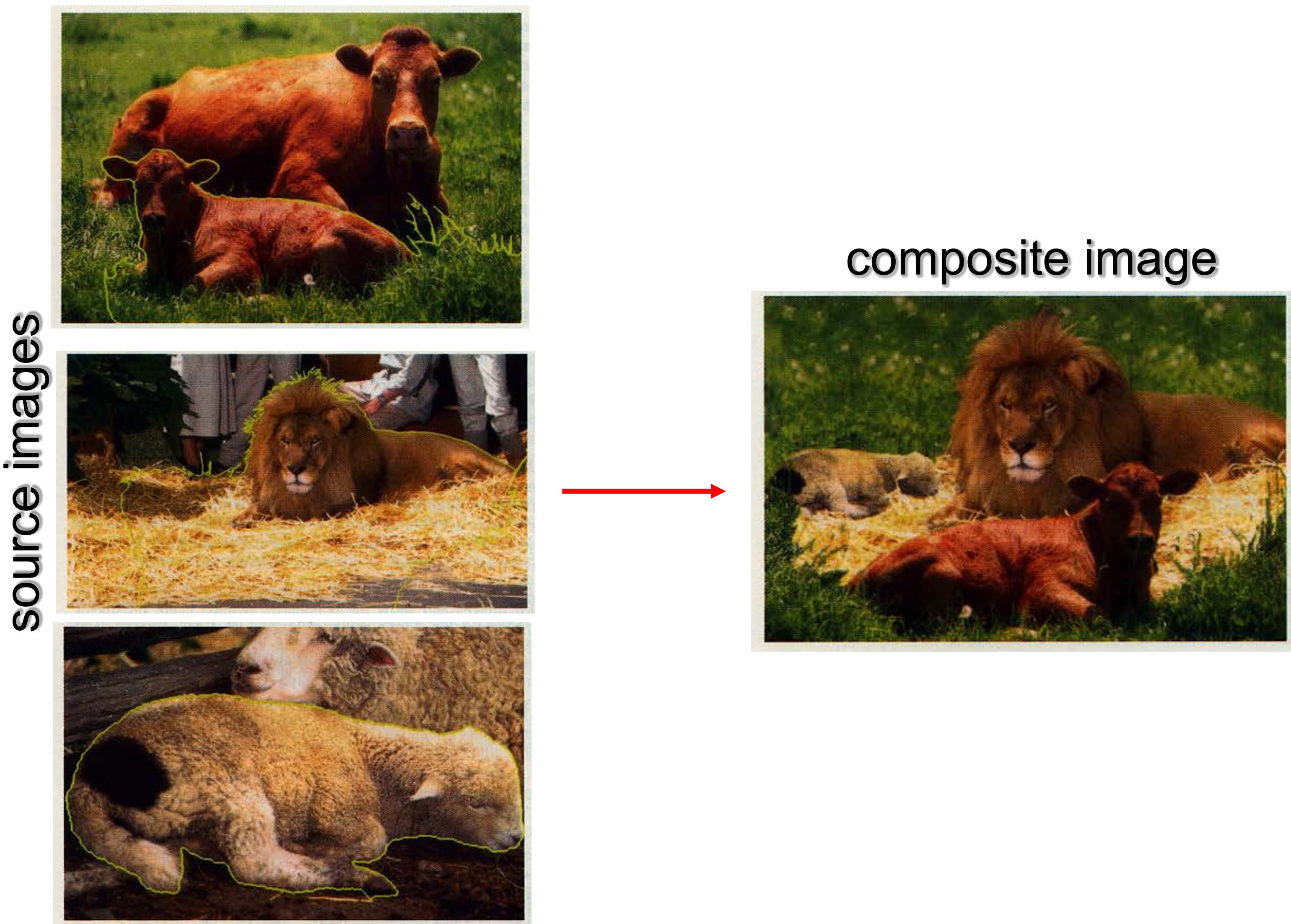
Original (B&W)

New (Color)

(Levin & Weiss, SIGGRAPH 2004)

Editing & Manipulating Photos

Scissoring objects from a photo



Giving Photos a “Painted” Look

Case study: From P. Litwinowicz’s SIGGRAPH’97 paper
“Processing Images and Videos for an Impressionist Effect”



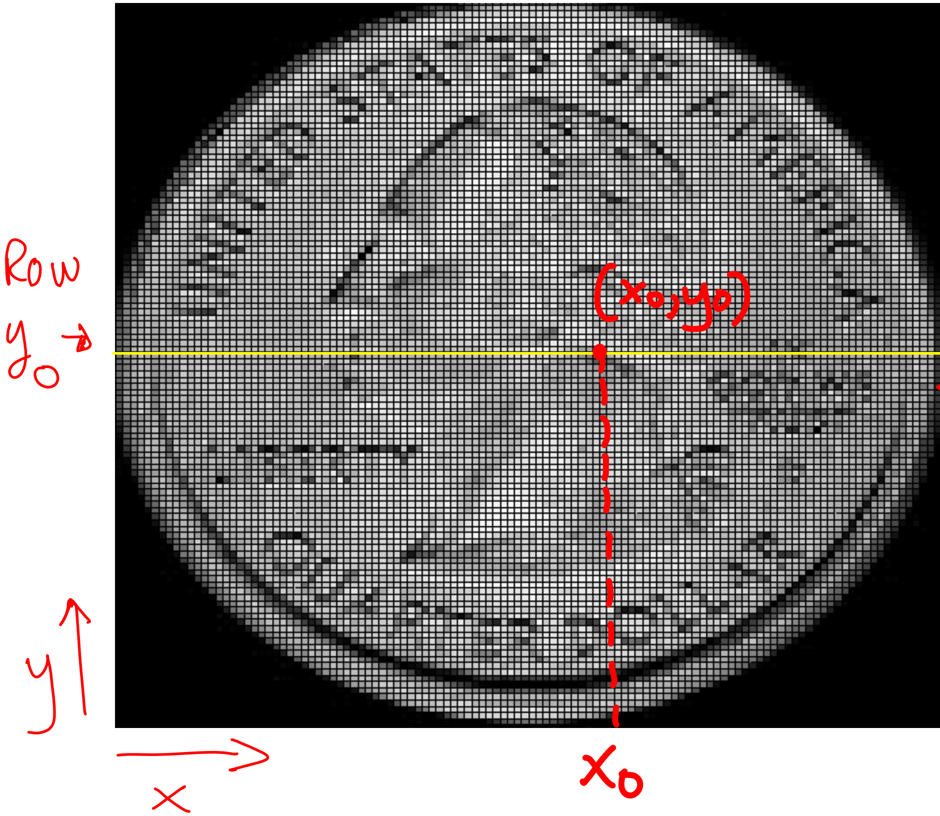
Topic 4:

Local analysis of image patches

- What do we mean by an image “patch”?
- Applications of local image analysis
- Visualizing 1D and 2D intensity functions

Image row or column ⇔ Graph in 2D

Gray-scale image



Graph in 2D

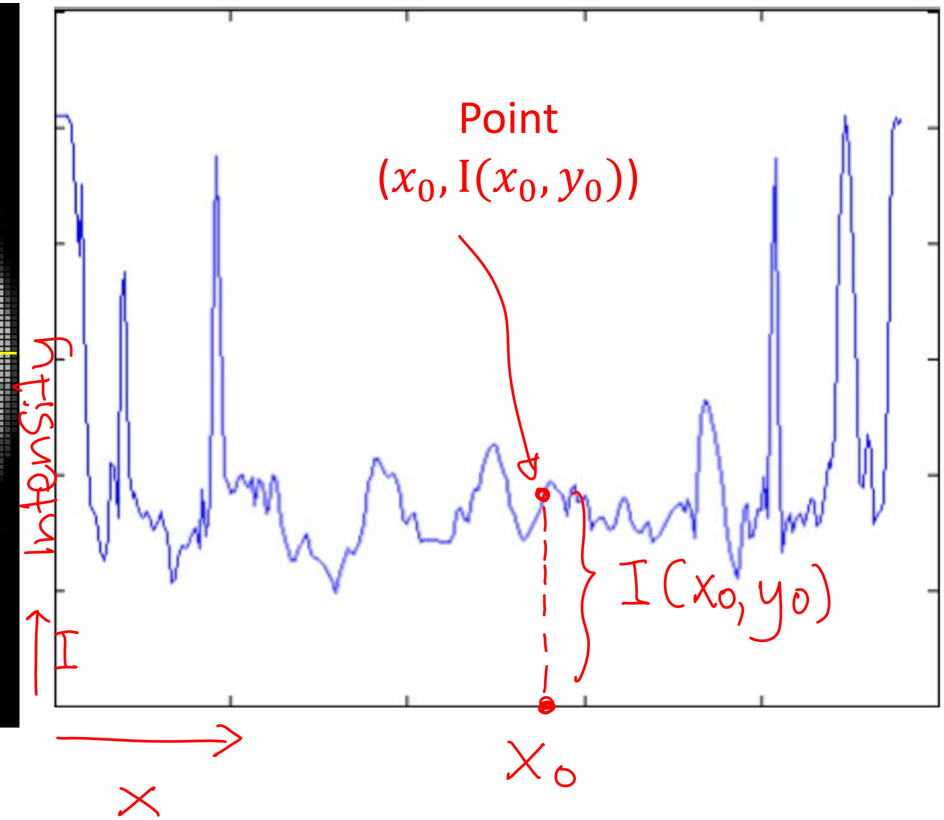
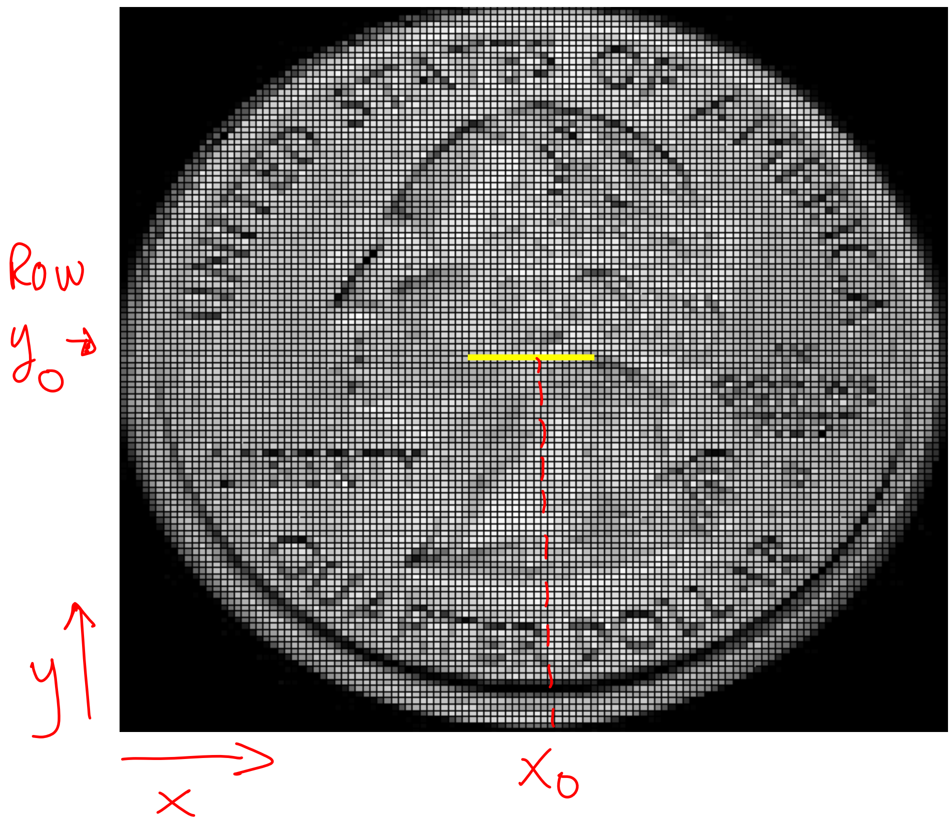


Image row or column ⇔ Graph in 2D

Gray-scale image



Graph in 2D

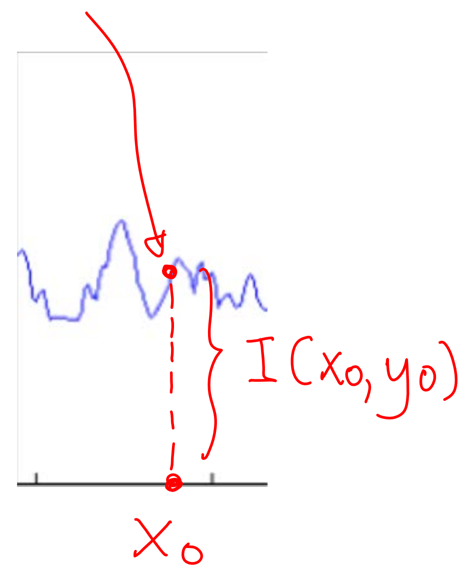


Image ⇔ Surface in 3D

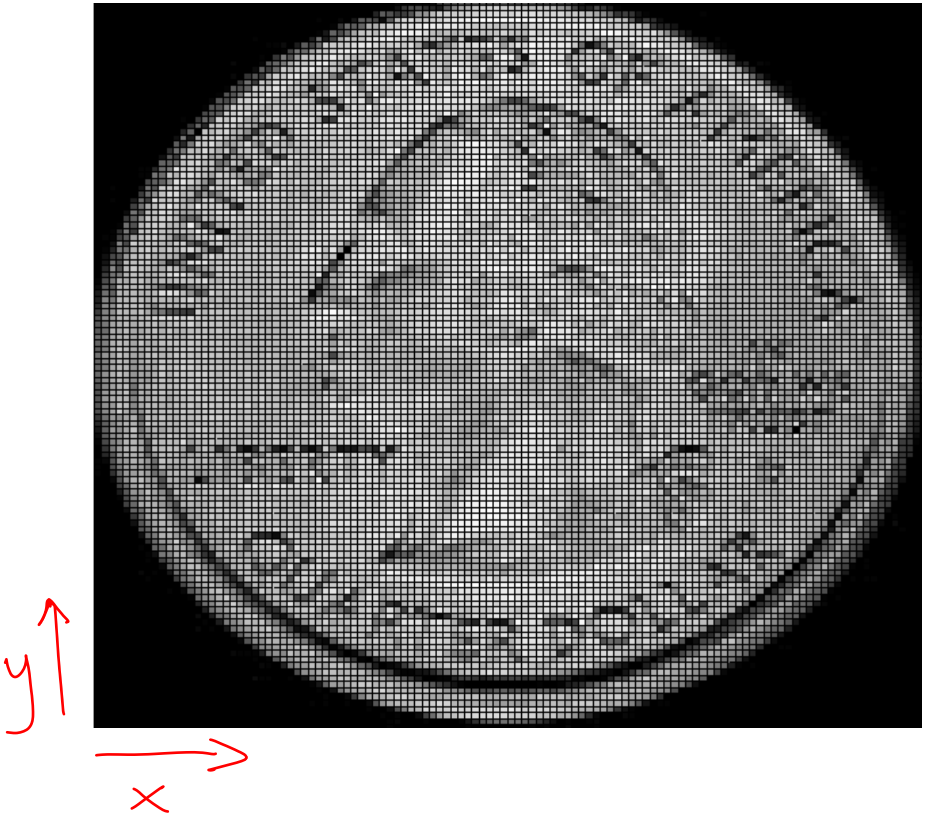
Gray-scale image $\mathrm{I}(x, y)$

y

x

Image ⇔ Surface in 3D

Gray-scale image $\mathrm{I}(x,y)$



Surface $\mathrm{z} = \mathrm{I}(x,y)$

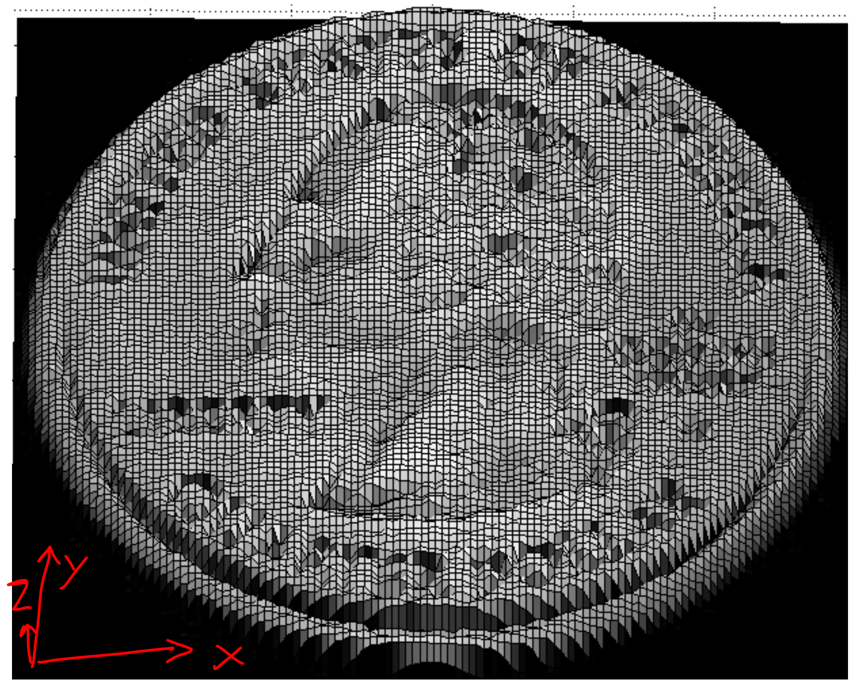
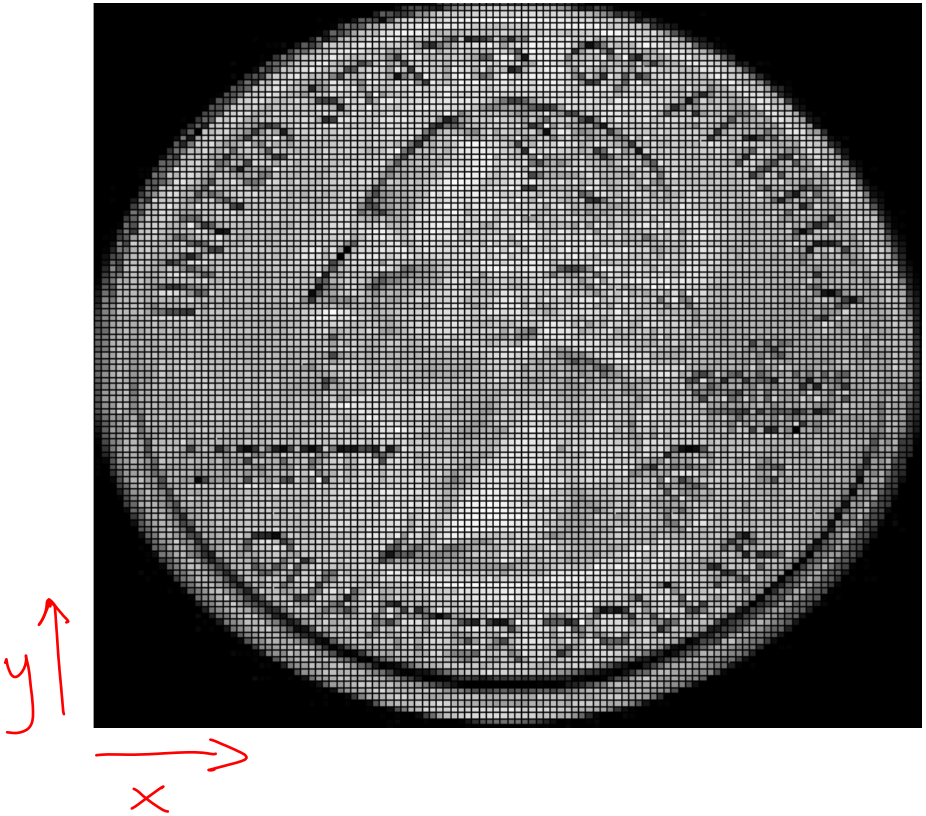
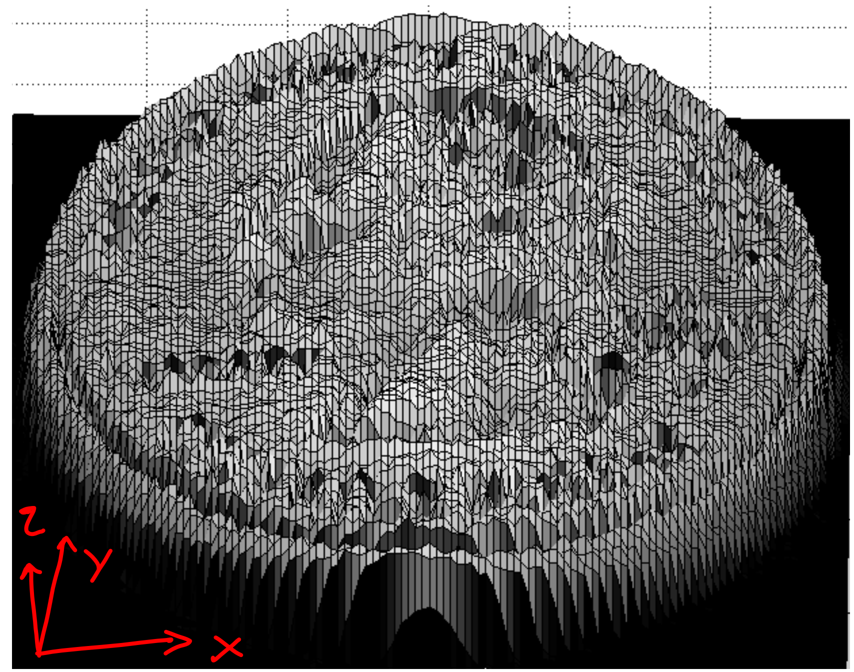


Image ⇔ Surface in 3D

Gray-scale image $\mathrm{I}(x, y)$

Surface $z = \mathrm{I}(x, y)$





- The height of the surface at (x, y) is $\mathrm{I}(x, y)$
- The surface contains point $(x, y, \mathrm{I}(x, y))$

Image ⇔ Surface in 3D

Gray-scale image

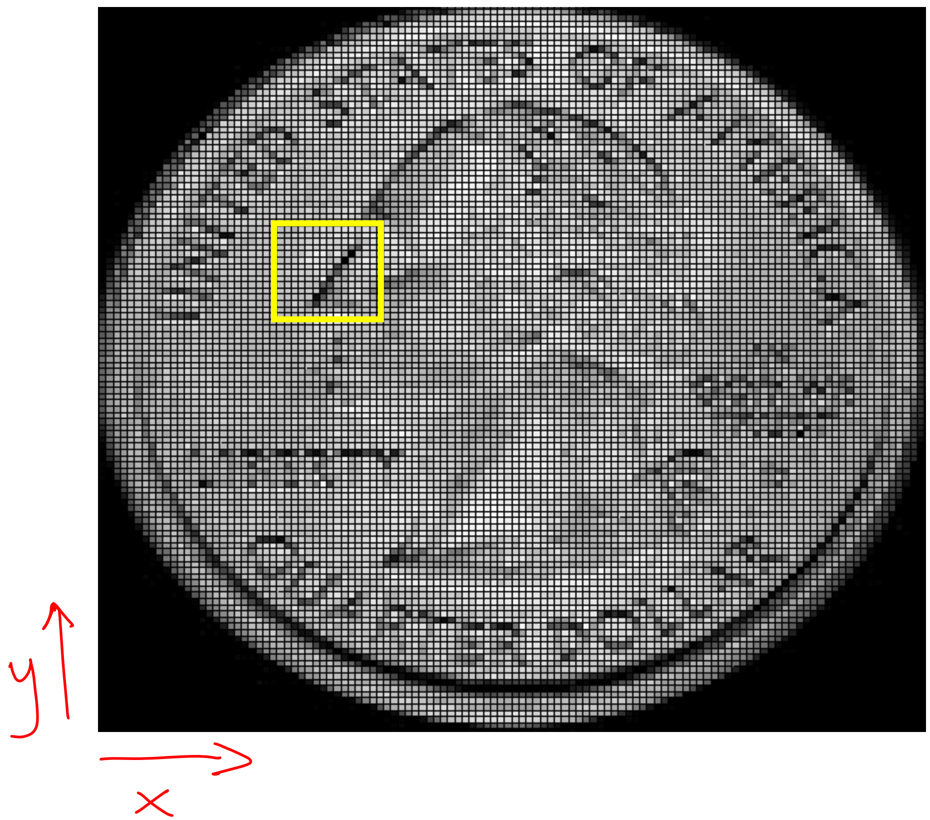
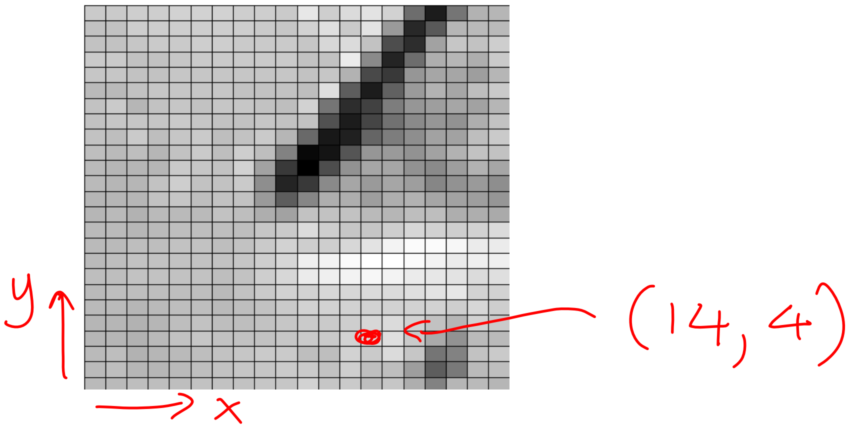
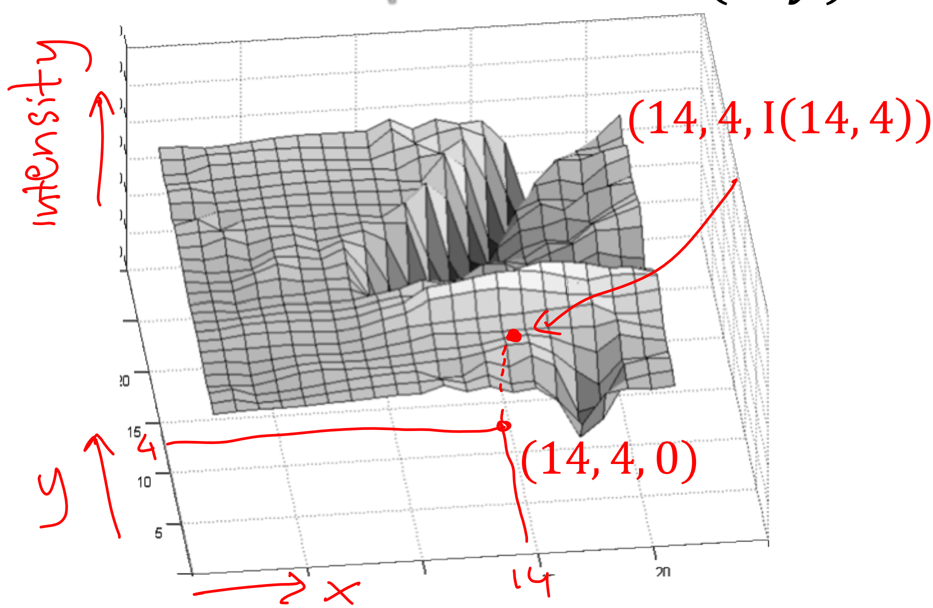


Image patch



Surface patch $z = \mathrm{I}(x, y)$



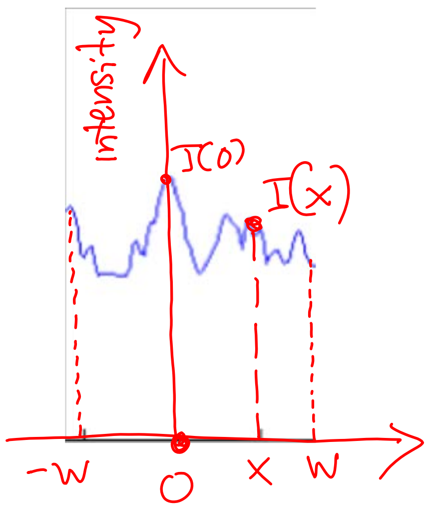
Topic 4.1: Polynomial fitting (Local analysis of 1D image patches as an example)

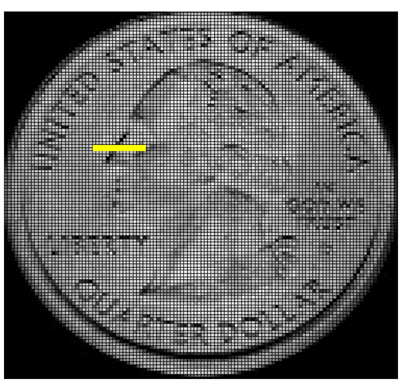
- Taylor series approximation of 1D intensity patches
- Estimating derivatives of 1D intensity patches:
  - Least-squares fitting
  - Weighted least-squares fitting
  - Robust polynomial fitting: RANSAC

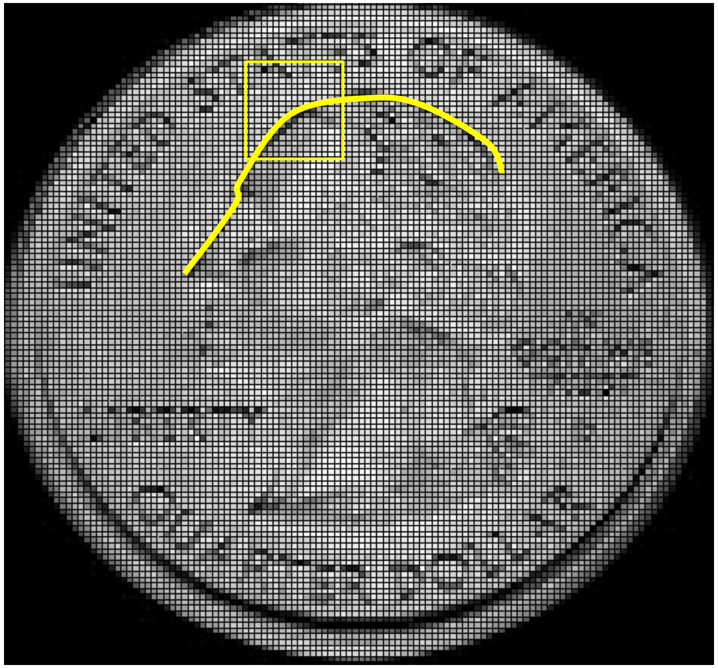
Polynomial curve fitting/regression

Pixel intensities as graph in 2D

(our example)







Also useful for 2D curves in image, etc..

Estimating Derivatives For Image Row r

"Sliding window" algorithm:

- Define a "pixel window" centered at pixel (w,r)

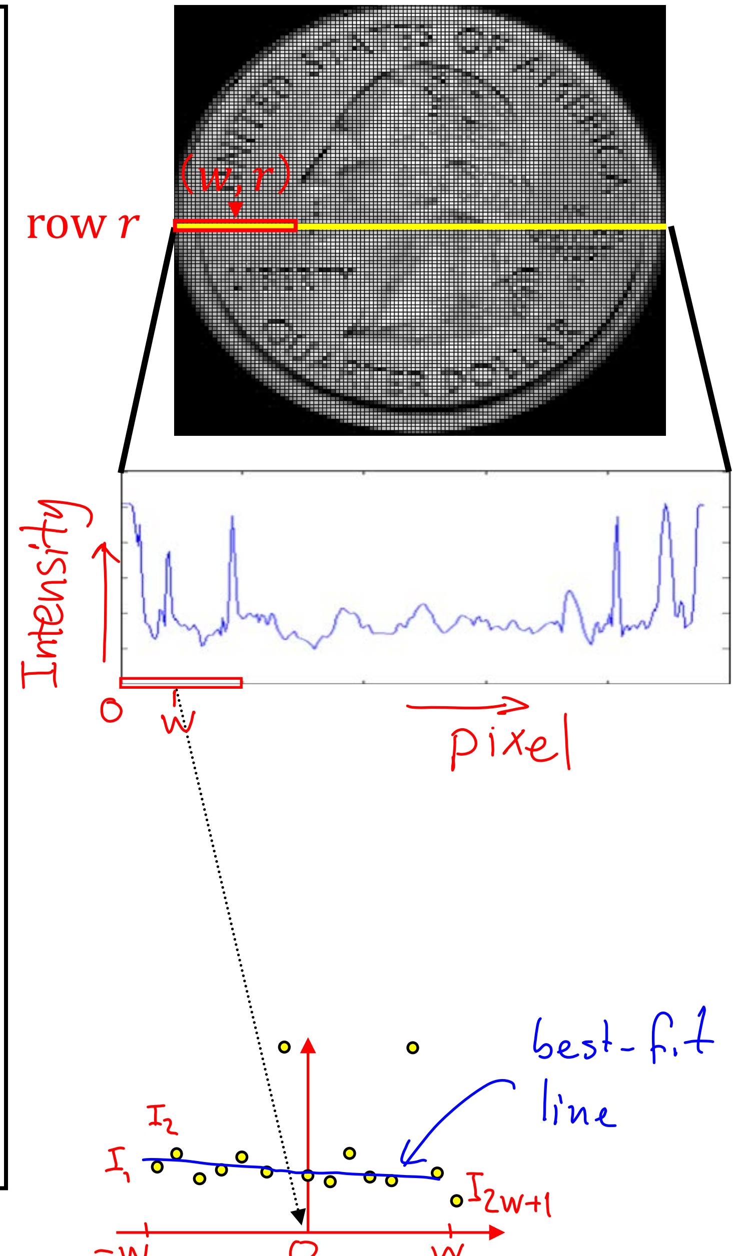
In this demonstration, the pixel index starts from 0, so the patch is (0,r) to (2w,r), centering at w.

- Fit n-degree poly to window's intensities (usually n=1 or 2)
- Assign the poly's derivatives at x=0 to pixel at window's center

$$\frac{dI}{dx}(\mathrm{w}) \longleftrightarrow \frac{dI}{dx}(0)$$

image coordinate patch coordinate

- "Slide" window one pixel over, so that it is centered at pixel (w+1,r)
- Repeat 1-4 until window reaches right image border



Estimating Derivatives For Image Row r

"Sliding window" algorithm:

- Define a "pixel window" centered at pixel (w,r)

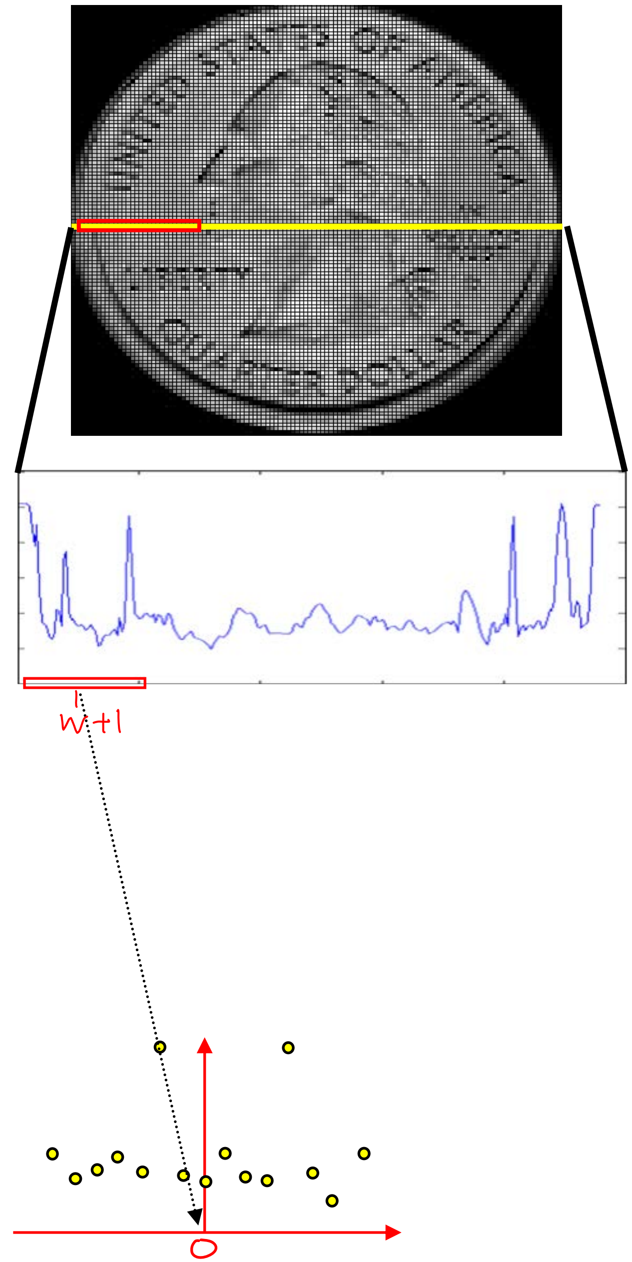
In this demonstration, the pixel index starts from 0, so the patch is (0,r) to (2w,r), centering at w.

- Fit n-degree poly to window's intensities (usually n=1 or 2)
- Assign the poly's derivatives at x=0 to pixel at window's center

$$\frac{dI}{dx}(\mathrm{w}) \longleftrightarrow \frac{dI}{dx}(0)$$

image coordinate   patch coordinate

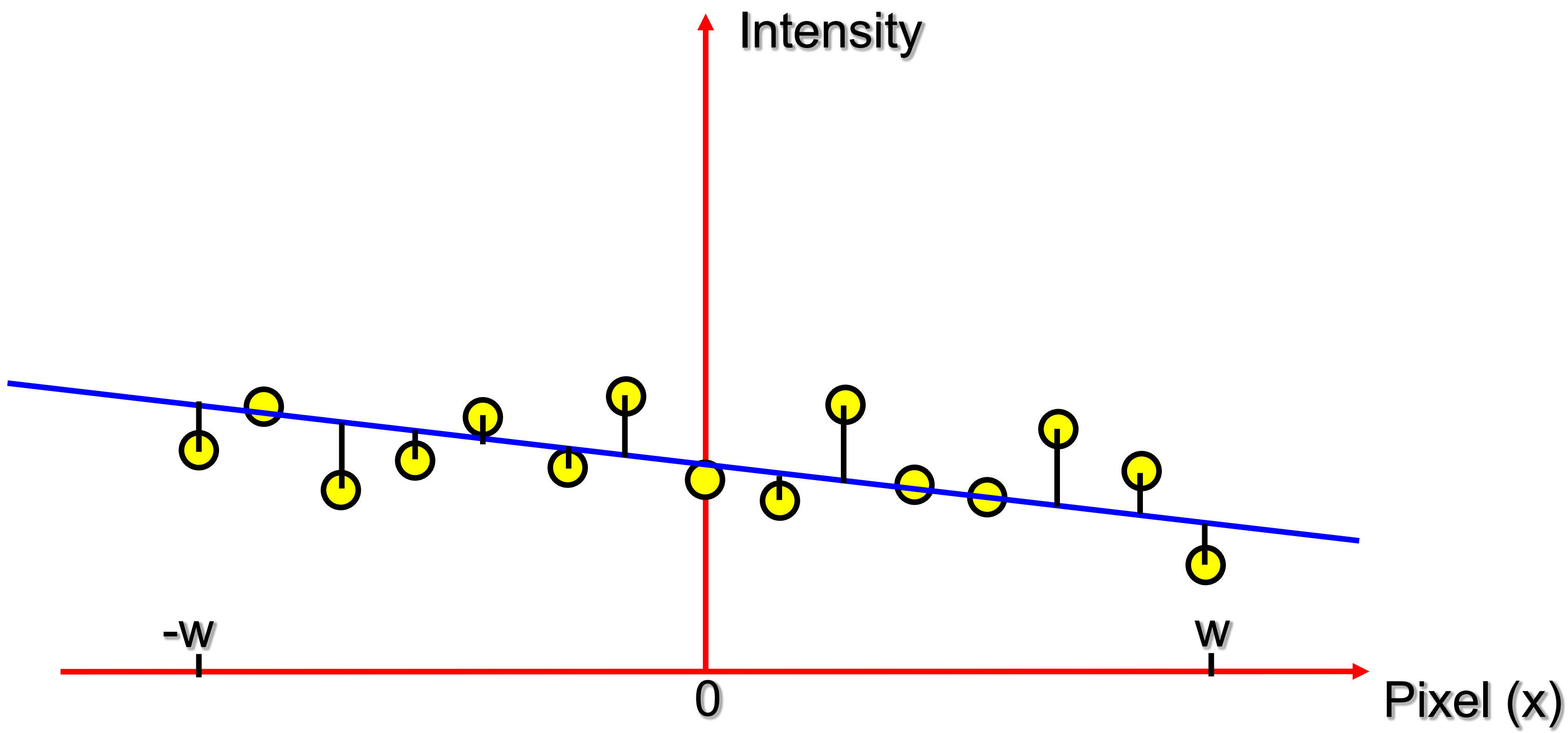
- "Slide" window one pixel over, so that it is centered at pixel (w+1,r)
- Repeat 1-4 until window reaches right image border



Least-Squares Polynomial Fitting

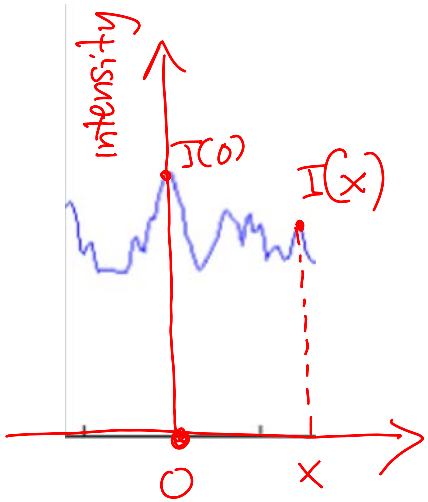
Scenario #1:

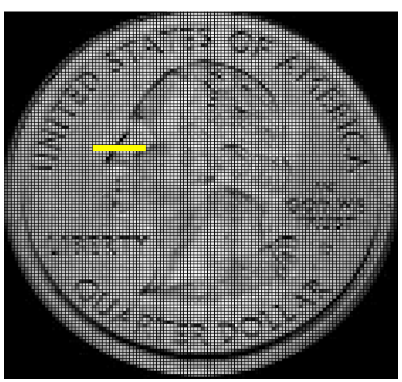
- Fit polynomial to ALL pixel intensities in a patch
- All pixels contribute equally to estimate of derivative(s) at patch center (i.e., at x=0)



Taylor-Series Approximation of I(x)

As graph in 2D





Taylor series expansion of $I(x)$ near the "patch" center 0

$$I(x) = \underbrace{\underbrace{\underbrace{I(0)}_{\text{0-th order approximation}} + x\cdot\frac{dI}{dx}(0)}_{\text{1st-order approx. of } I} + \frac{1}{2}x^2\frac{d^2I}{dx^2}(0)}_{\text{2nd-order approx of } I}$$

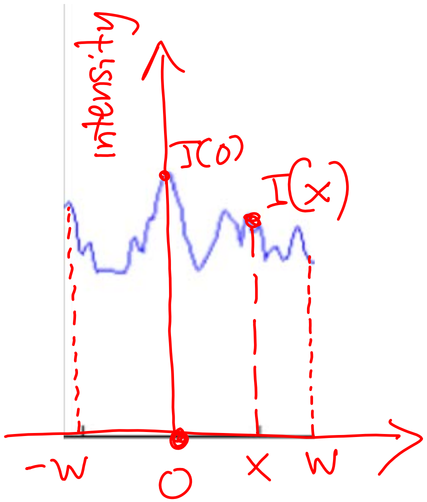
$$\underbrace{+ \cdots + \frac{1}{n!}x^n\frac{d^nI}{dx^n}(0)}_{\text{n-th order approx}} + R_{n+1}(x)$$

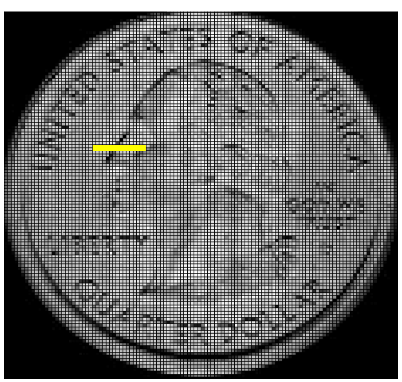
The residual $R_{n+1}(x)$ satisfies

$$\lim_{x\to 0} R_{n+1}(x) = 0$$

Taylor-Series Approximation of I(x)

As graph in 2D





Taylor series expansion of $I(x)$ near the "patch" center 0

n^{th}-order approximation in matrix notation:

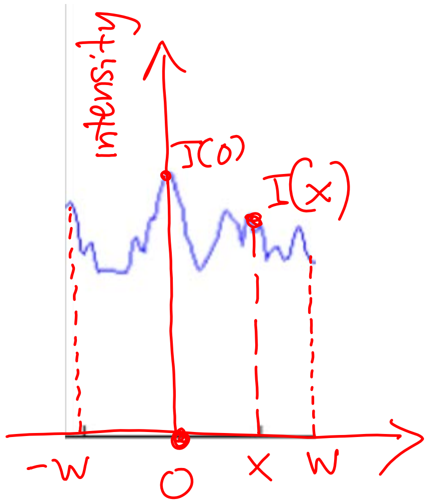
for $x \in (-w, w)$

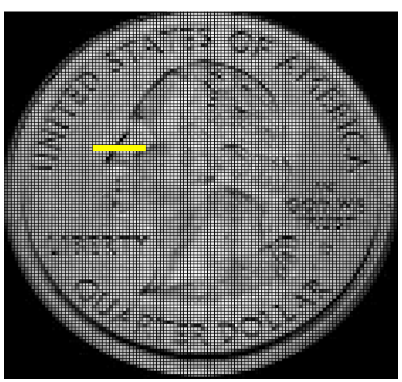
$$I(x) \simeq \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 & \cdots & \frac{1}{n!}x^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

for a given x, approximation depends on (n+1) constants corresponding to the intensity derivatives at the patch origin

Taylor-Series Approximation of I(x)

As graph in 2D

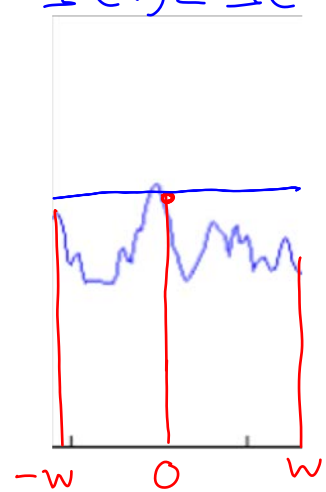




Taylor series expansion of I(x) near the "patch" center 0

Example: 0th-order approx

$I(x) = I(0)$

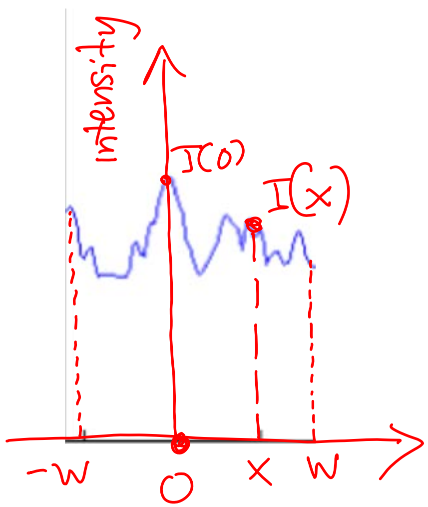


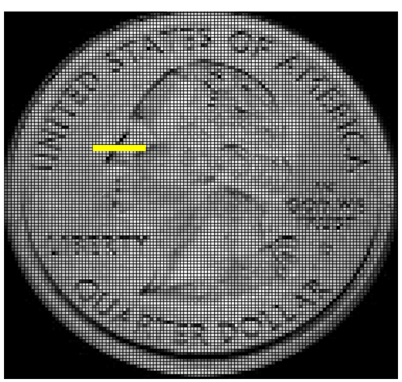
0th order approximation of I

$$I(x) \approx \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 & \cdots & \frac{1}{n!}x^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

Taylor-Series Approximation of I(x)

As graph in 2D

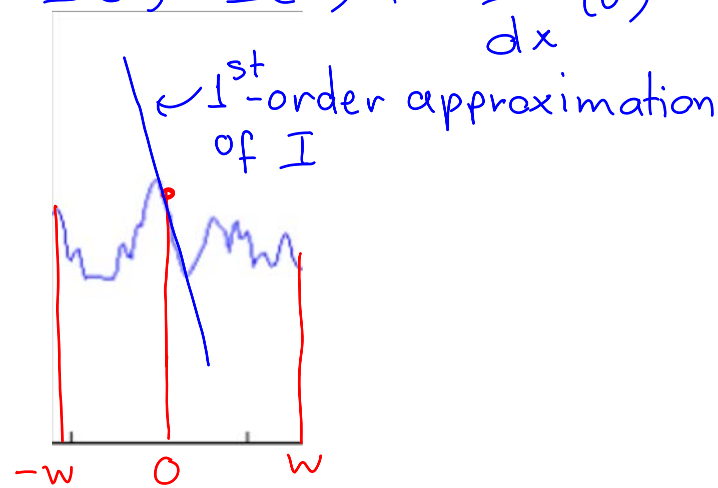




Taylor series expansion of I(x) near the "patch" center 0

Example: 1st-order approx

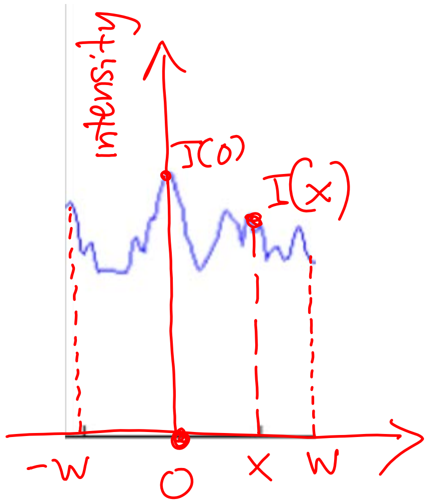
$$I(x) = I(0) + x \cdot \frac{dI}{dx}(0)$$

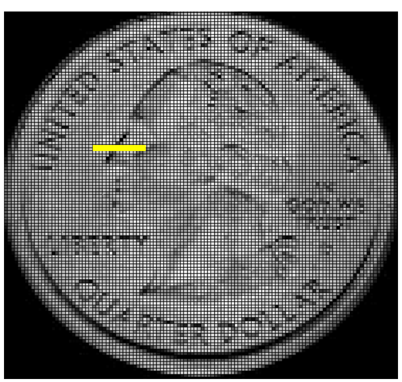


$$I(x) \simeq \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 & \cdots & \frac{1}{n!}x^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

Taylor-Series Approximation of I(x)

As graph in 2D



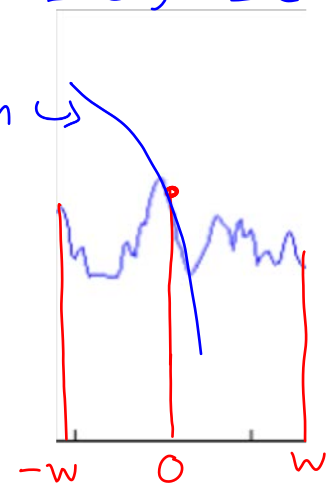


Taylor series expansion of I(x) near the "patch" center 0

Example: 2nd-order approx

$$I(x) = I(0) + x \cdot \frac{dI}{dx}(0) + \frac{x^2}{2}\frac{d^2I}{dx^2}(0)$$

2nd-order approximation of I



$$I(x) \approx \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 & \cdots & \frac{1}{n!}x^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

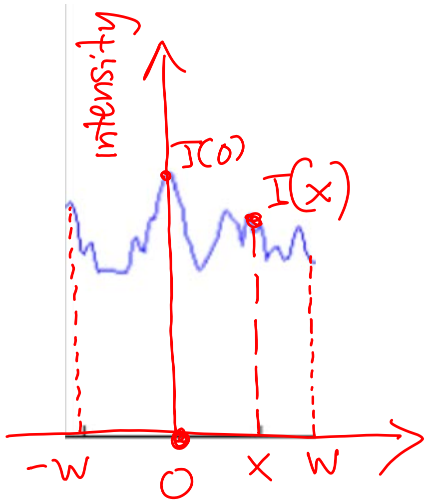
Topic 4.1:

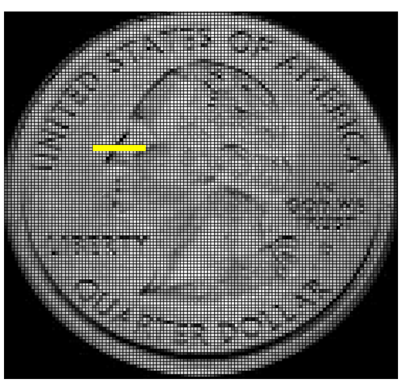
Local analysis of 1D image patches

- Taylor series approximation of 1D intensity patches
- Estimating derivatives of 1D intensity patches:
  - Least-squares fitting
  - Weighted least-squares fitting
  - Robust polynomial fitting: RANSAC

Least-Squares Polynomial Fitting of I(x)

As graph in 2D





Our first "patch descriptor": intensity derivatives

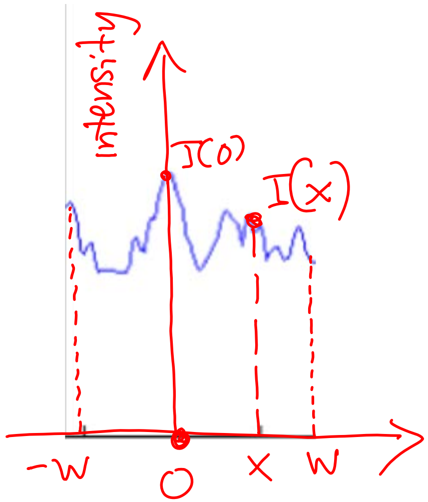
To compute the n derivatives at pixel 0:

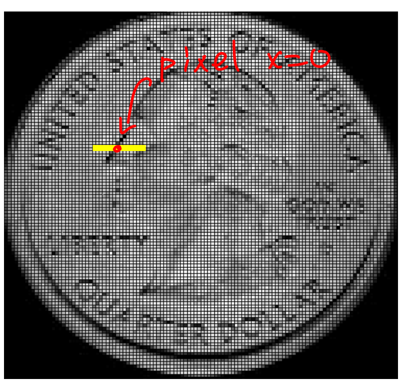
$$\boxed{\text{fit a polynomial of degree } n \text{ to the patch intensities}}$$

$$I(x) \simeq \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 & \cdots & \frac{1}{n!}x^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

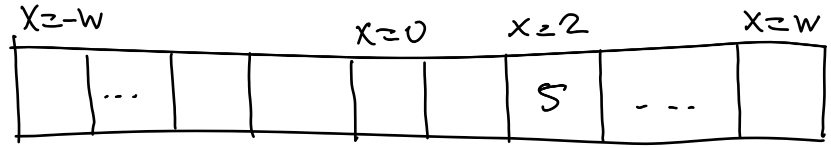
Least-Squares Polynomial Fitting of I(x)

As graph in 2D





Patch (2w+1 pixels)



1 linear eq. for every pixel

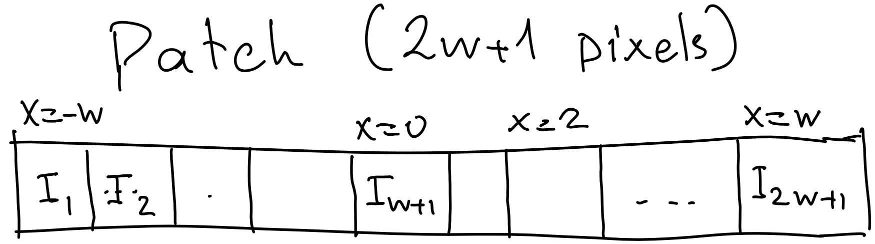
$$5 = \begin{bmatrix} 1 & 2 & \frac{1}{2}2^2 & \frac{1}{6}2^3 & \cdots & \frac{1}{n!}2^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

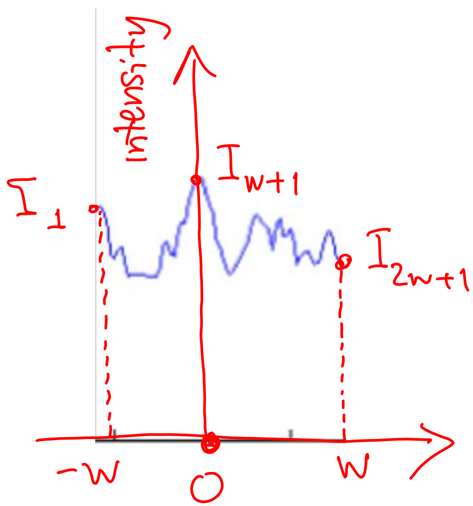
$\Longleftrightarrow$ have 2w+1 eqs for the 2w+1 pixels

$$I(x) \simeq \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 & \cdots & \frac{1}{n!}x^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

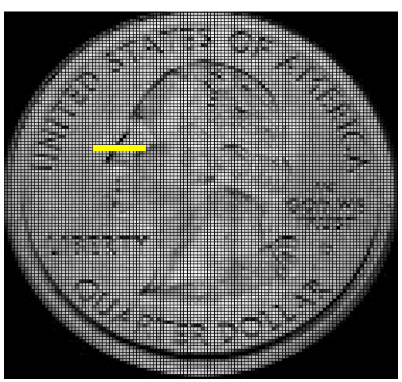
Least-Squares Polynomial Fitting of I(x)

As graph in 2D





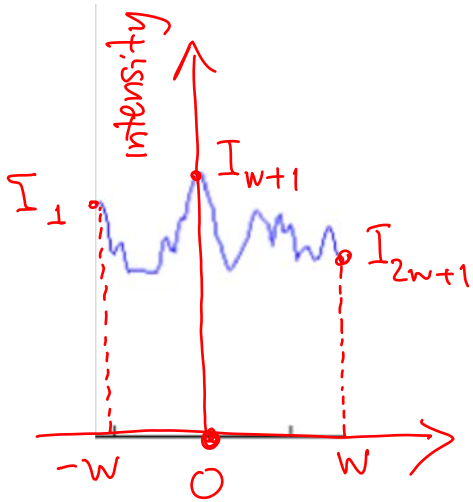
$$I_{(2w+1)\times 1} = X_{(2w+1)\times(n+1)}\, d_{(n+1)\times 1}$$

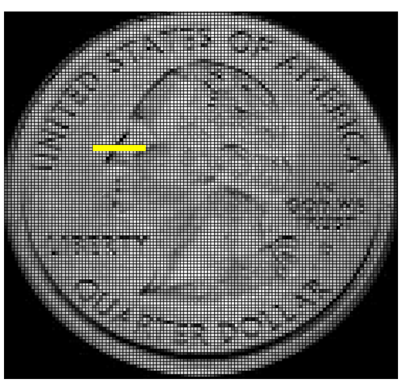


$$I(x) \simeq \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 & \cdots & \frac{1}{n!}x^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

Least-Squares Polynomial Fitting of I(x)

As graph in 2D





Patch (2w+1 pixels)

x=-w				x=0	x=2			x=w
I_1	I_2	.		I_{w+1}			...	I_{2w+1}

$$I_{(2w+1)\times 1} = X_{(2w+1)\times(n+1)}\, d_{(n+1)\times 1}$$

intensities (known) — positions (known) — derivatives (unknown)

Solving linear system in terms of d minimizes the "fit error"

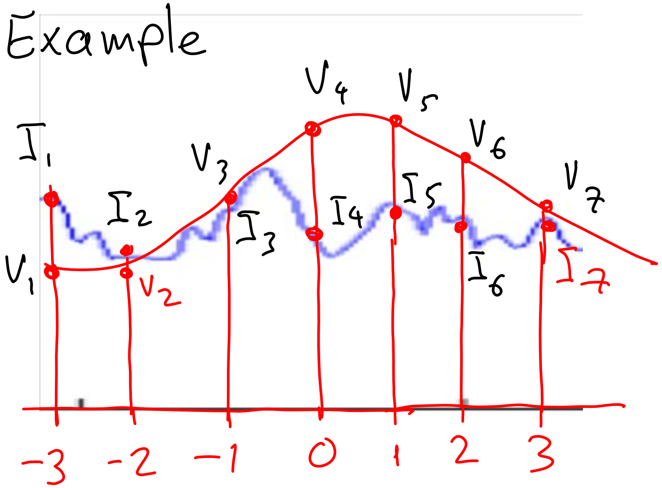
$$\| I - Xd \|^2$$

Definition (α-norm $\|v\|^\alpha$ of vector v)

for $v = [v_1\ v_2 \dots v_m]$, $\|v\|^\alpha = \left(\sum_{i=1}^{m} |v_i|^\alpha \right)^{1/\alpha}$

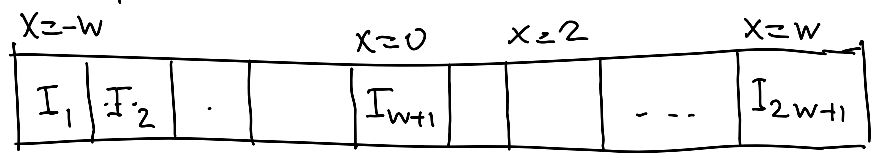
Least-Squares Polynomial Fitting of I(x)

Example



- For the solution d, the vector $v = Xd$ gives us the values of the polynomial at $(-w, \dots, 0, \dots, w)$
- This solution minimizes the 2-norm (i.e. the length) of the error vector $(I-v)$:
$$\left(\sum_{i=1}^{2w+1} (I_i - v_i)^2 \right)^{1/2}$$

Patch (2w+1 pixels)



$$I_{(2w+1)\times 1} = X_{(2w+1)\times(n+1)}\, d_{(n+1)\times 1}$$

intensities (known) — positions (known) — derivatives (unknown)

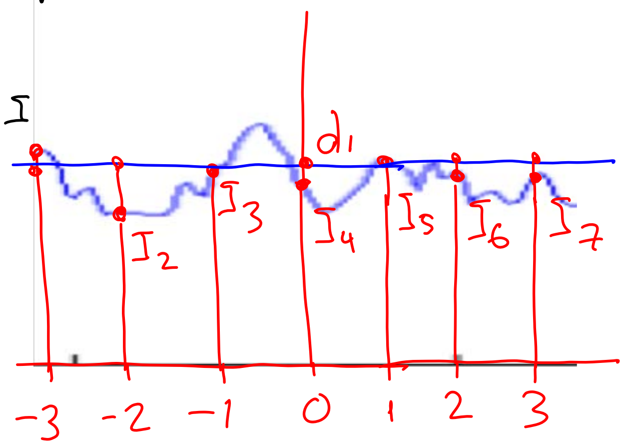
Solving linear system in terms of d minimizes the "fit error"

$$\| I - Xd \|^2$$

Solution d is called a <u>least-squares fit</u>

0th-Order (Constant) Estimation of I(x)

Special case:



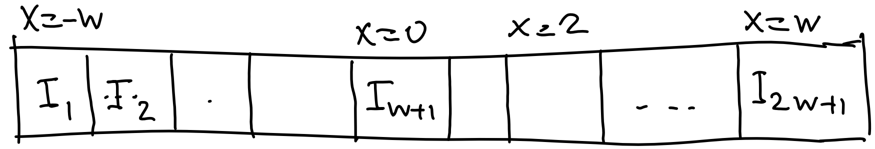
- Solution minimizes

$$\sum_{i=1}^{2w+1} (I_i - d_1)^2$$

- Solution is the mean intensity of the patch:

$$d_1 = \frac{1}{2w+1} \sum_{i=1}^{2w+1} I_i$$

Patch (2w+1 pixels)



$$I_{(2w+1)\times 1} = X_{(2w+1)\times 1} \; d_{1\times 1}$$

intensities (known) — positions (known) — one unknown (equal to I(0))

$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} d_1 \end{bmatrix}$$

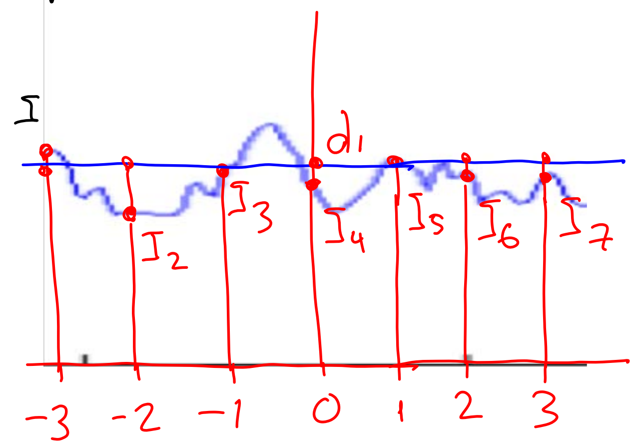
$d_1 \leftarrow I(0)$

Solving linear system in terms of d minimizes the "fit error"

$$\| I - Xd \|^2$$

0^{th}-Order (Constant) Estimation of I(x)

Special case:



- Solution minimizes
$$\sum_{i=1}^{2w+1} (I_i - d_1)^2$$

- Solution is the mean intensity of the patch:
$$d_1 = \frac{1}{2w+1} \sum_{i=1}^{2w+1} I_i$$

Proof

- Let $E(x) = \sum_{i=1}^{2w+1} (I_i - x)^2$

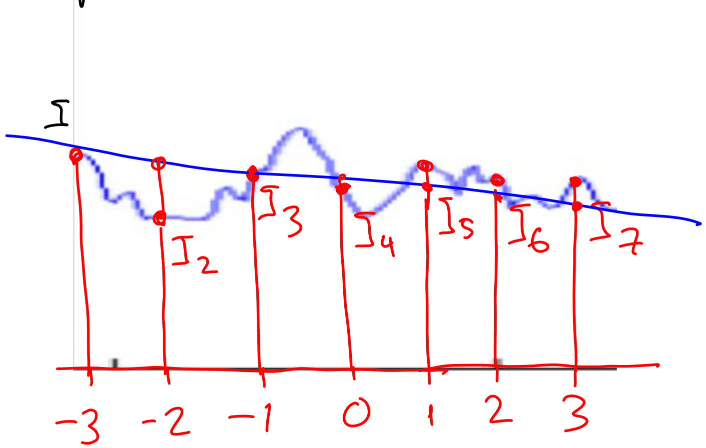
- At the minimum of $E(x)$, the derivative $\frac{d}{dx}E(x)$ must be zero

- $$\frac{d}{dx}E(x) = \sum_{i=1}^{2w+1} \frac{d}{dx}\left[(I_i - x)^2\right]$$
$$= \sum_{i=1}^{2w+1} 2(I_i - x)\cdot(-1)$$
$$= -2\left[\sum_{i=1}^{2w+1} (I_i - x)\right]$$
$$= -2\left(\sum_{i=1}^{2w+1} I_i\right) + 2(2w+1)x$$

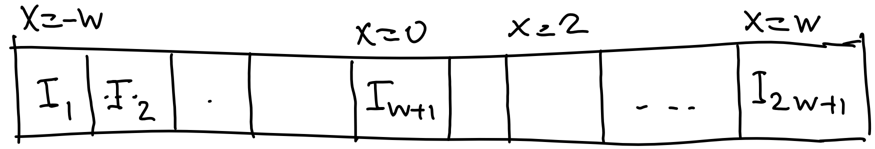
- $$\frac{d}{dx}E(x) = 0 \iff x = \frac{1}{2w+1}\left(\sum_{i=1}^{2w+1} I_i\right).$$

1st-Order (Linear) Estimation of I(x)

Special case:



Patch (2w+1 pixels)



$$I_{(2w+1)\times 1} = X_{(2w+1)\times 2}\, d_{2\times 1}$$

intensities (known) — positions (known) — two unknowns

$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} 1 & -3 \\ 1 & -2 \\ \vdots & \vdots \\ 1 & 3 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

d_1 ← $I(0)$

d_2 ← $\frac{dI}{dx}(0)$

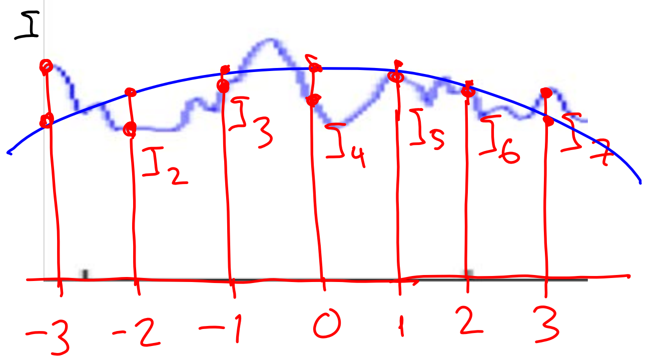
- Solution minimizes sum of "vetrical" squared distances between line and image intensities
- Gives us an estimate of $I(0)$ and $\frac{dI}{dx}(0)$ (i.e. value & derivative at 0)

Solving linear system in terms of d minimizes the "fit error"

$$\| I - Xd \|^2$$

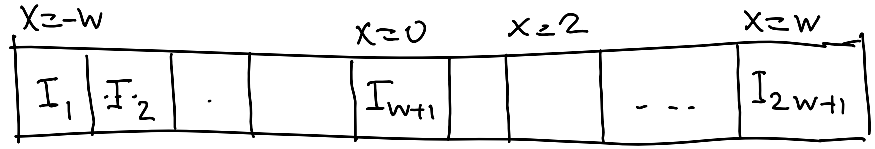
2^{nd}-Order (Quadratic) Estimation of I(x)

Special case:



- Fits a parabola/ hyperbola/ ellipse
- Gives us an estimate of 1st & 2nd image derivative at patch center

Patch (2w+1 pixels)



$$I_{(2w+1)\times 1} = X_{(2w+1)\times 3} \; d_{3\times 1}$$

intensities (known) | positions (known) | 3 unknowns

$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} 1 & -3 & 9/2 \\ 1 & -2 & 2 \\ \vdots & \vdots & \\ 1 & 3 & 9/2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

$d_3 \leftarrow \frac{d^2 I}{dx^2}(0)$

Topic 4.1:

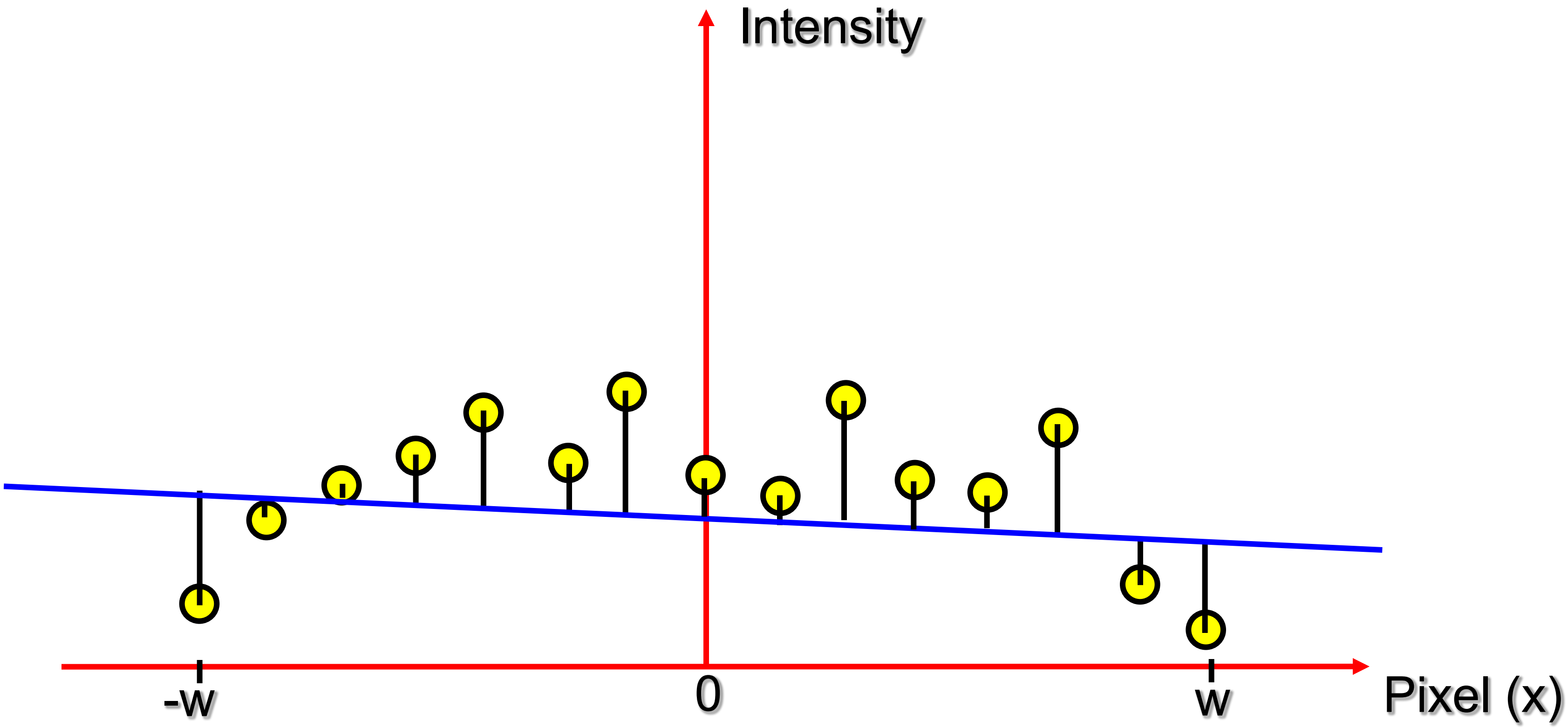
Local analysis of 1D image patches

- **Taylor series approximation of 1D intensity patches**
- Estimating derivatives of 1D intensity patches:
  - Least-squares fitting
  - Weighted least-squares fitting
  - Robust polynomial fitting: RANSAC

Weighted Least Squares Polynomial Fitting

Scenario #1:

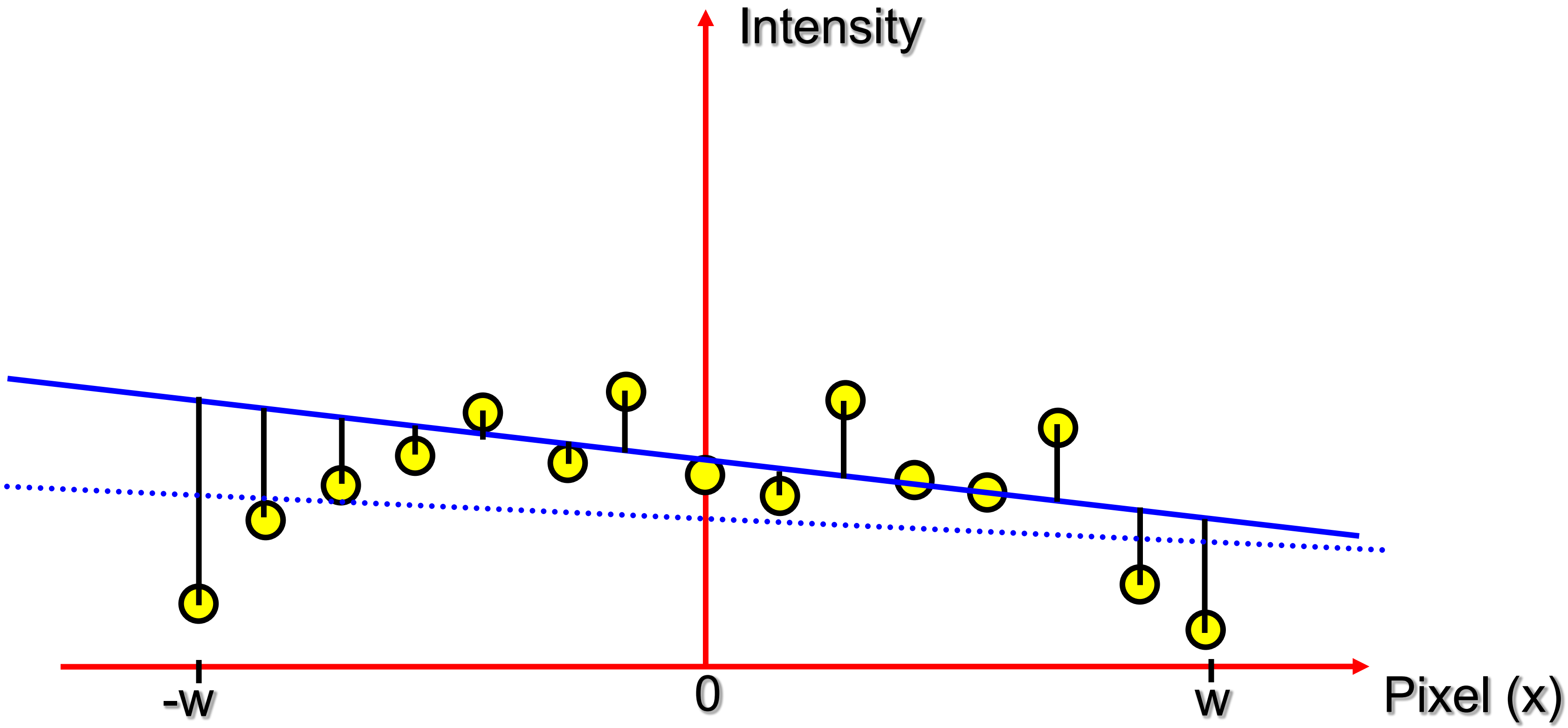
- Fit polynomial to ALL pixel intensities in a patch



Weighted Least Squares Polynomial Fitting

Scenario #2:

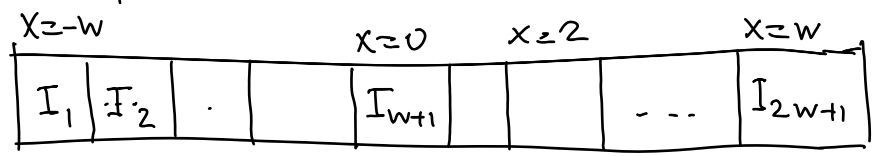
- Fit polynomial to ALL pixel intensities in a patch
- Pixels contribute to estimate of derivative(s) at center according to a weight function $\Omega(x)$



Polynomial Fitting: A Linear Formulation

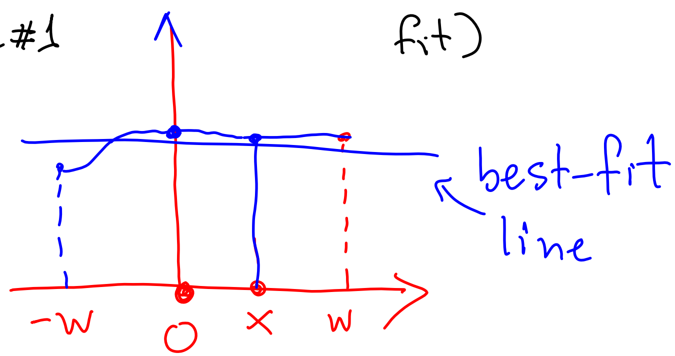
Q: Will the estimate of $\frac{dI}{dx}(0)$ be the same or different in the two cases below? (assume a 1^{st} order fit)

Patch (2w+1 pixels)

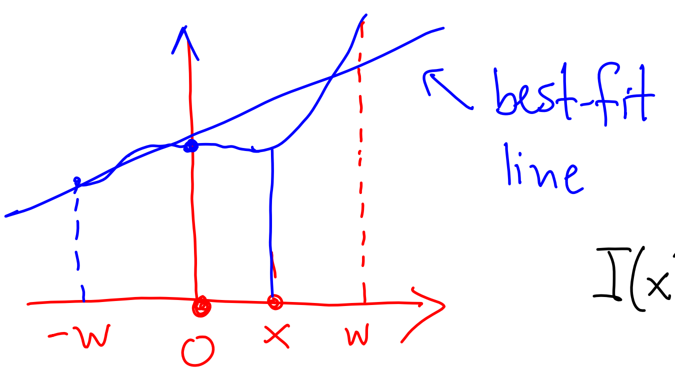


$$I_{(2w+1)\times 1} = X_{(2w+1)\times(n+1)}\, d_{(n+1)\times 1}$$

case #1



case #2



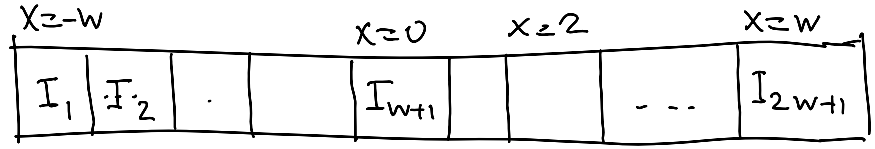
Ans: the values will differ because all patch pixels contribute equally to the linear system!

$$I(x) \simeq \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 & \cdots & \frac{1}{n!}x^n \end{bmatrix} \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

Weighted Least-Squares Estimation of I(x)

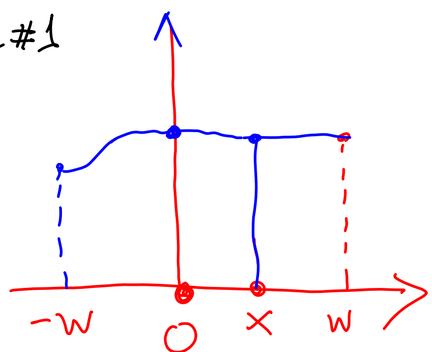
Q: How can we bias our estimate of $\frac{d}{dx}I(0)$ toward the patch center?

Patch (2w+1 pixels)



$$I_{(2w+1)\times 1} = X_{(2w+1)\times(n+1)}\, d_{(n+1)\times 1}$$

case #1

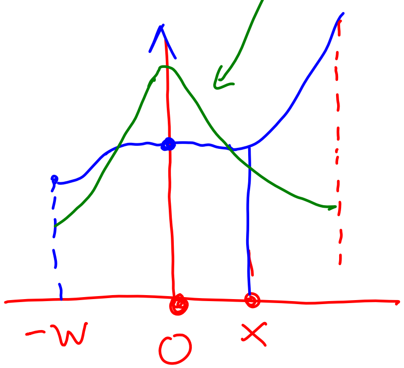


Idea: Weigh pixels near center more than pixels away from it

$$\begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

weight function $\Omega(x)$
(e.g. $\Omega(x) = e^{-x^2}$)

case #2



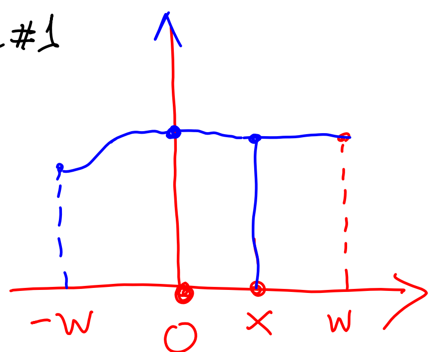
New equation for pixel x:

$$\Omega(x)\, I(x) = \Omega(x) \cdot \left[1 \quad x \quad \frac{1}{2}x^2 \quad \frac{1}{6}x^3 \quad \cdots \quad \frac{1}{n!}x^n \right]$$

Weighted Least-Squares Estimation of I(x)

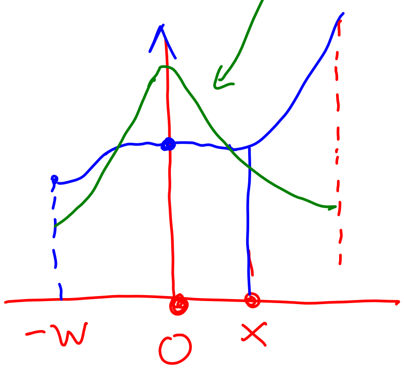
Q: How can we bias our estimate of $\frac{d}{dx}I(0)$ toward the patch center?

case #1

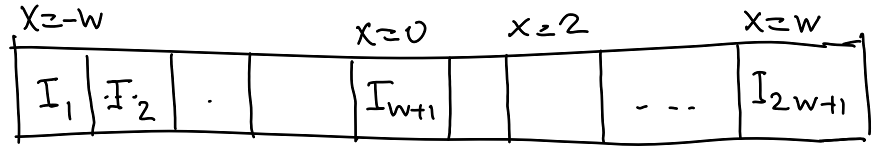


weight function $\Omega(x)$ (e.g. $\Omega(x) = e^{-x^2}$)

case #2



Patch (2w+1 pixels)



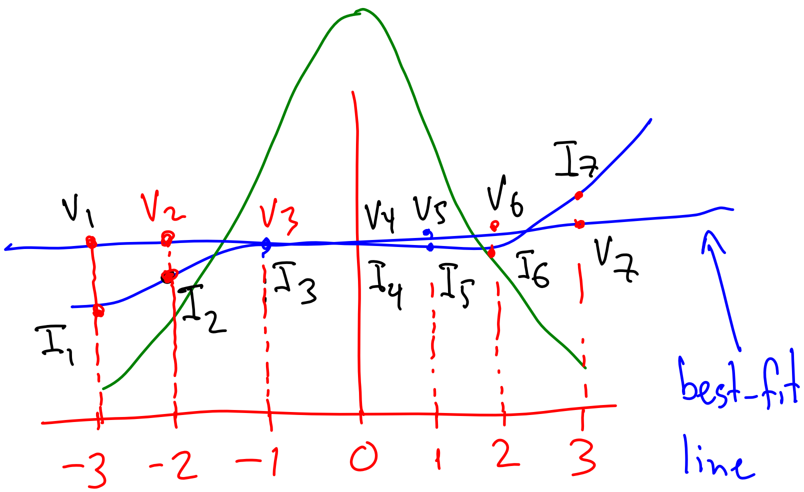
$$\begin{bmatrix} \Omega_1 & & 0 \\ & \ddots & \\ 0 & & \Omega_{2w+1} \end{bmatrix} I = \begin{bmatrix} \Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1} \end{bmatrix} X d$$

Idea: Weigh pixels near center more than pixels away from it

Solution d minimizes the norm

$$\left\| \begin{bmatrix} \Omega_1 & & 0 \\ & \ddots & \\ 0 & & \Omega_{2w+1} \end{bmatrix} (I - Xd) \right\|^2$$

Weighted Least-Squares Estimation of I(x)



Patch (2w+1 pixels)

x=-w				x=0		x=2		x=w
I_1	I_2	.		I_{w+1}			...	I_{2w+1}

$$\begin{bmatrix}\Omega_1 & & 0\\ & \ddots & \\ 0 & & \Omega_{2w+1}\end{bmatrix} I = \begin{bmatrix}\Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1}\end{bmatrix} X d$$

Idea: Weigh pixels near center more than pixels away from it

- For the solution d, the vector $v = Xd$ gives us the values of the polynomial at $(-w, \ldots, 0, \ldots, w)$
- This solution minimizes the 2-norm (i.e. the length) of the weighted error vector:

$$\left(\sum_{i=1}^{2w+1} \left[\Omega_i (I_i - v_i) \right]^2 \right)^{1/2}$$

Solution d minimizes the norm

$$\left\| \begin{bmatrix}\Omega_1 & & 0\\ & \ddots & \\ 0 & & \Omega_{2w+1}\end{bmatrix} (I - Xd) \right\|^2$$

Topic 4.1:

Local analysis of 1D image patches

- **Taylor series approximation of 1D intensity patches**
- Estimating derivatives of 1D intensity patches:
  - Least-squares fitting
  - Weighted least-squares fitting
  - Robust polynomial fitting: RANSAC

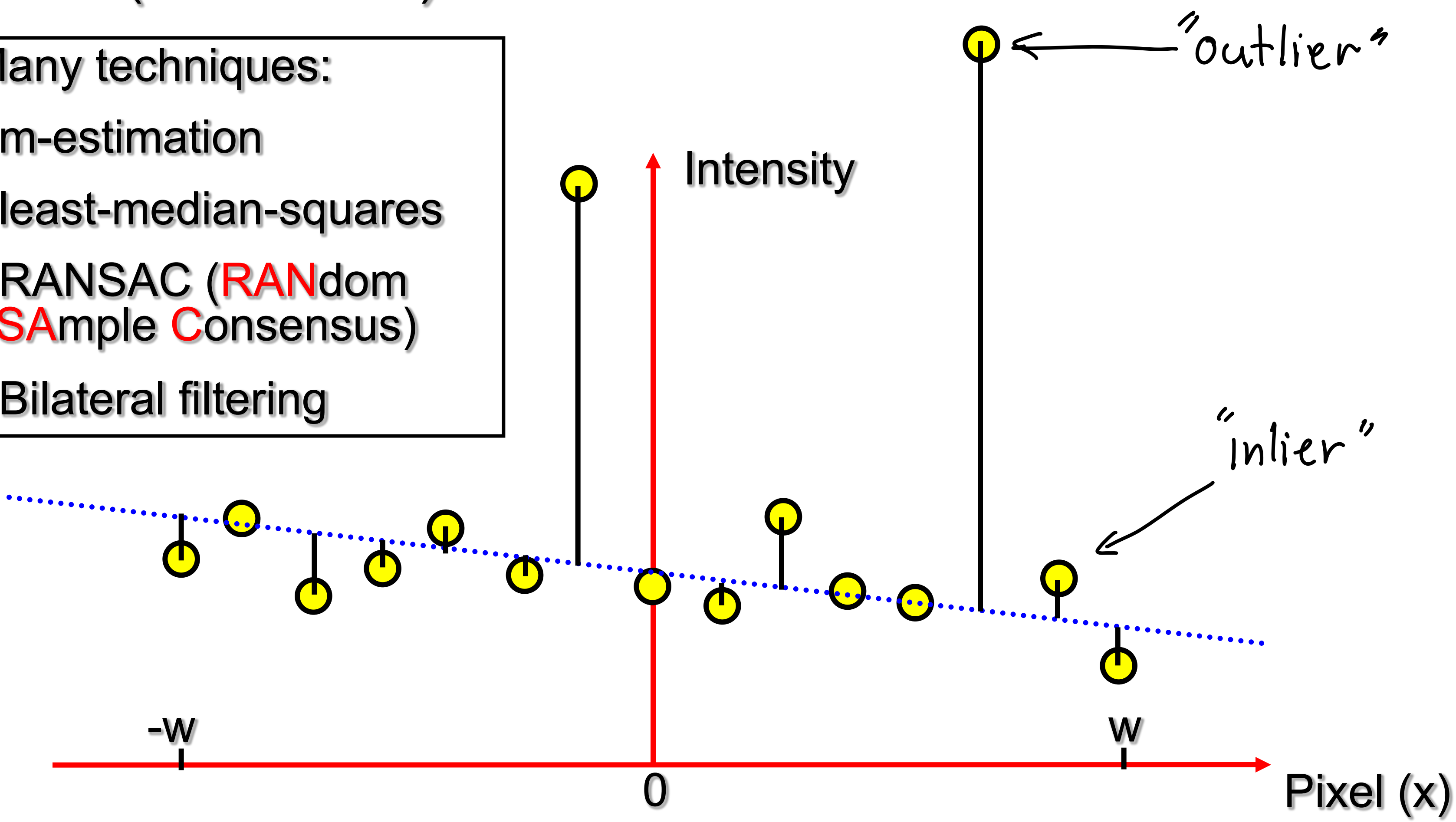
Robust Polynomial Fitting

Scenario #3:

- Fit polynomial only to SOME pixel intensities in a patch (the "inliers")

Many techniques:

- m-estimation
- least-median-squares
- RANSAC (RANdom SAmple Consensus)
- Bilateral filtering



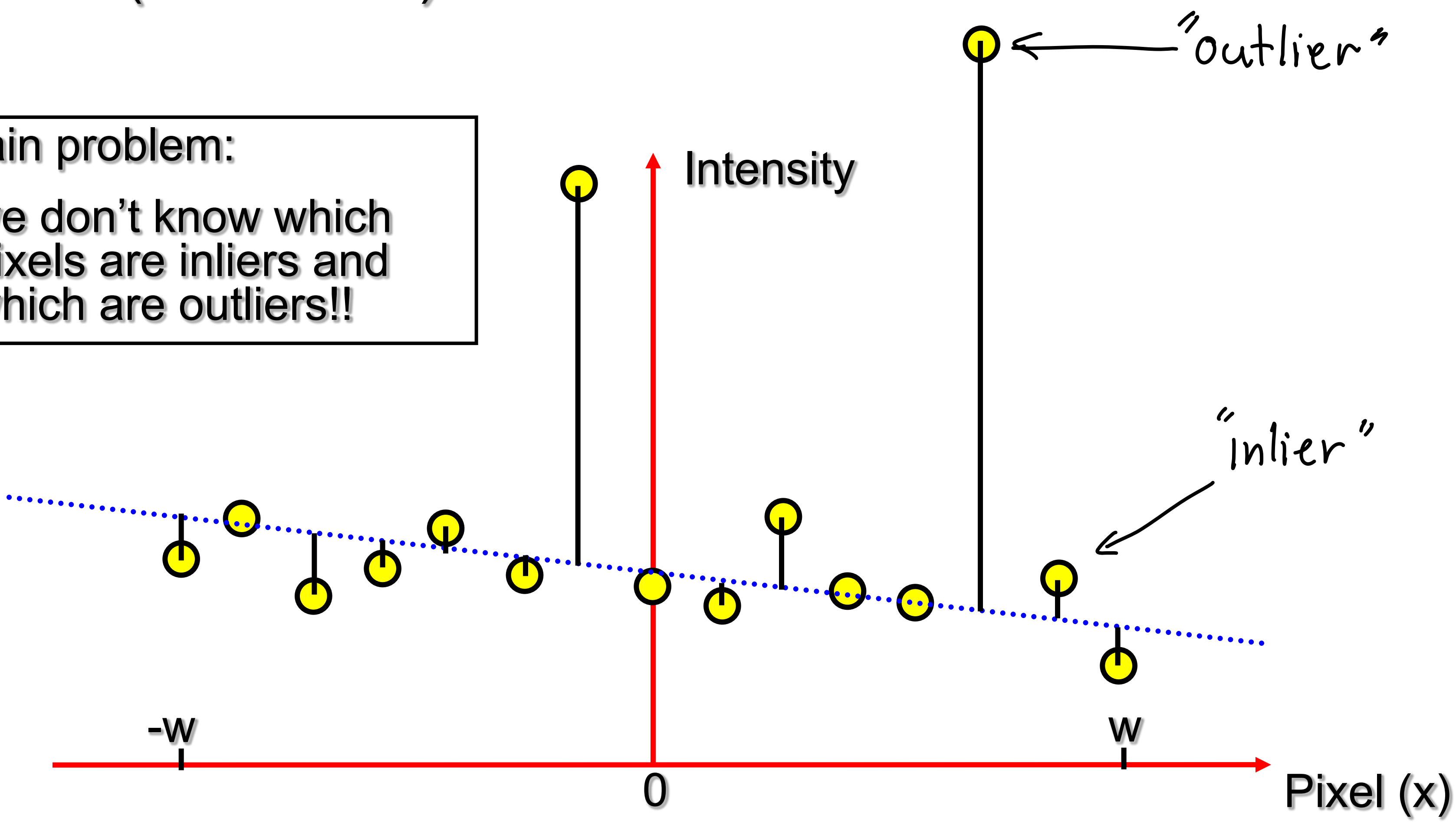
Robust Polynomial Fitting

Scenario #3:

- Fit polynomial only to SOME pixel intensities in a patch (the “inliers”)

Main problem:

we don’t know which pixels are inliers and which are outliers!!



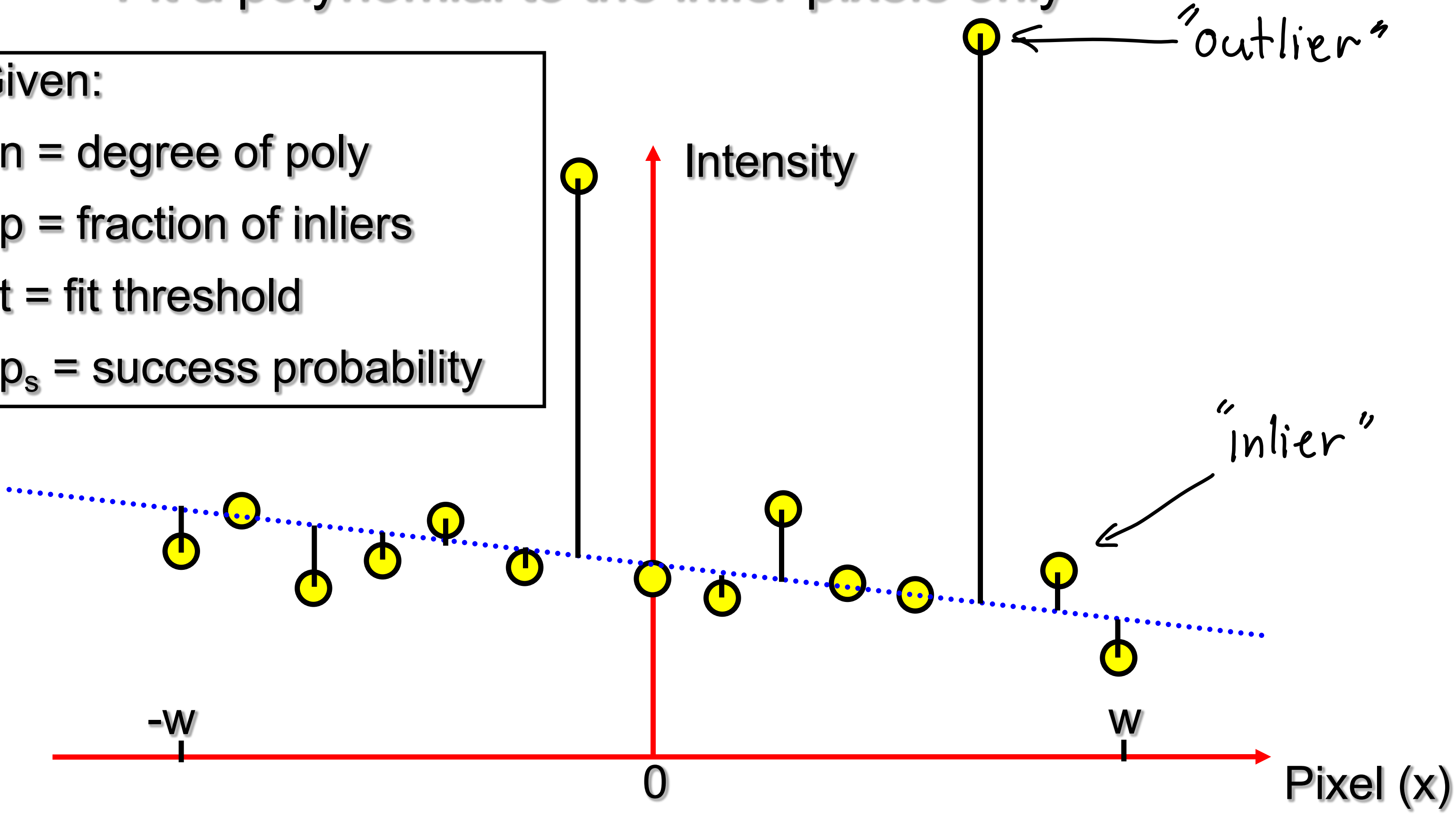
Polynomial Fitting Using RANSAC

Scenario #3:

- Find the “inlier” pixels in a patch of radius w
- Fit a polynomial to the inlier pixels only

Given:

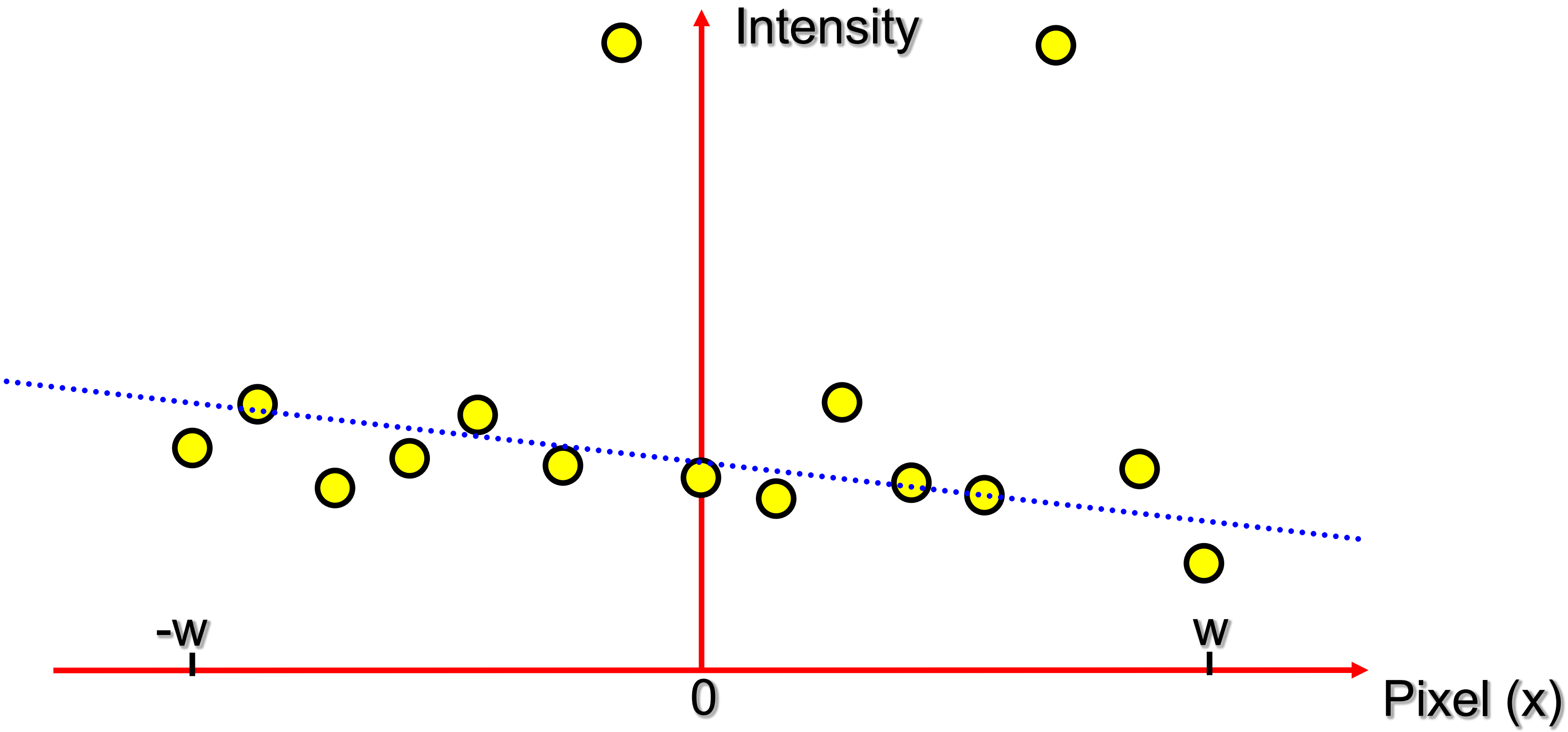
- n = degree of poly
- p = fraction of inliers
- t = fit threshold
- p_s = success probability



RANSAC Algorithm

Example: Line fitting using RANSAC (i.e., n=2 unknown polynomial coefficients)

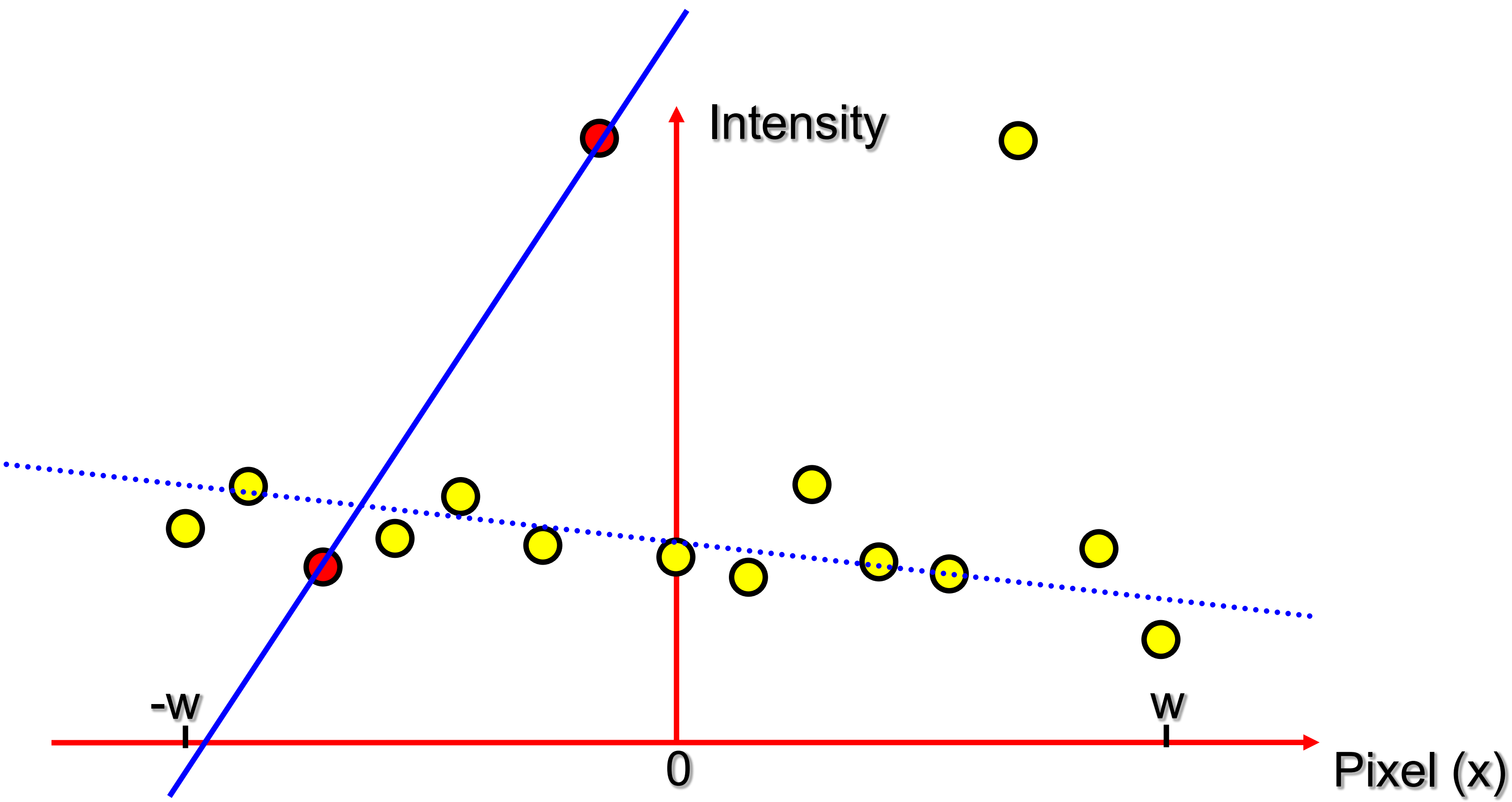
- Step 1: Randomly choose n pixels from the patch



RANSAC Algorithm

Example: Line fitting using RANSAC (i.e., n=2 unknown polynomial coefficients)

- Step 2: Fit the poly using the chosen pixels/intensities

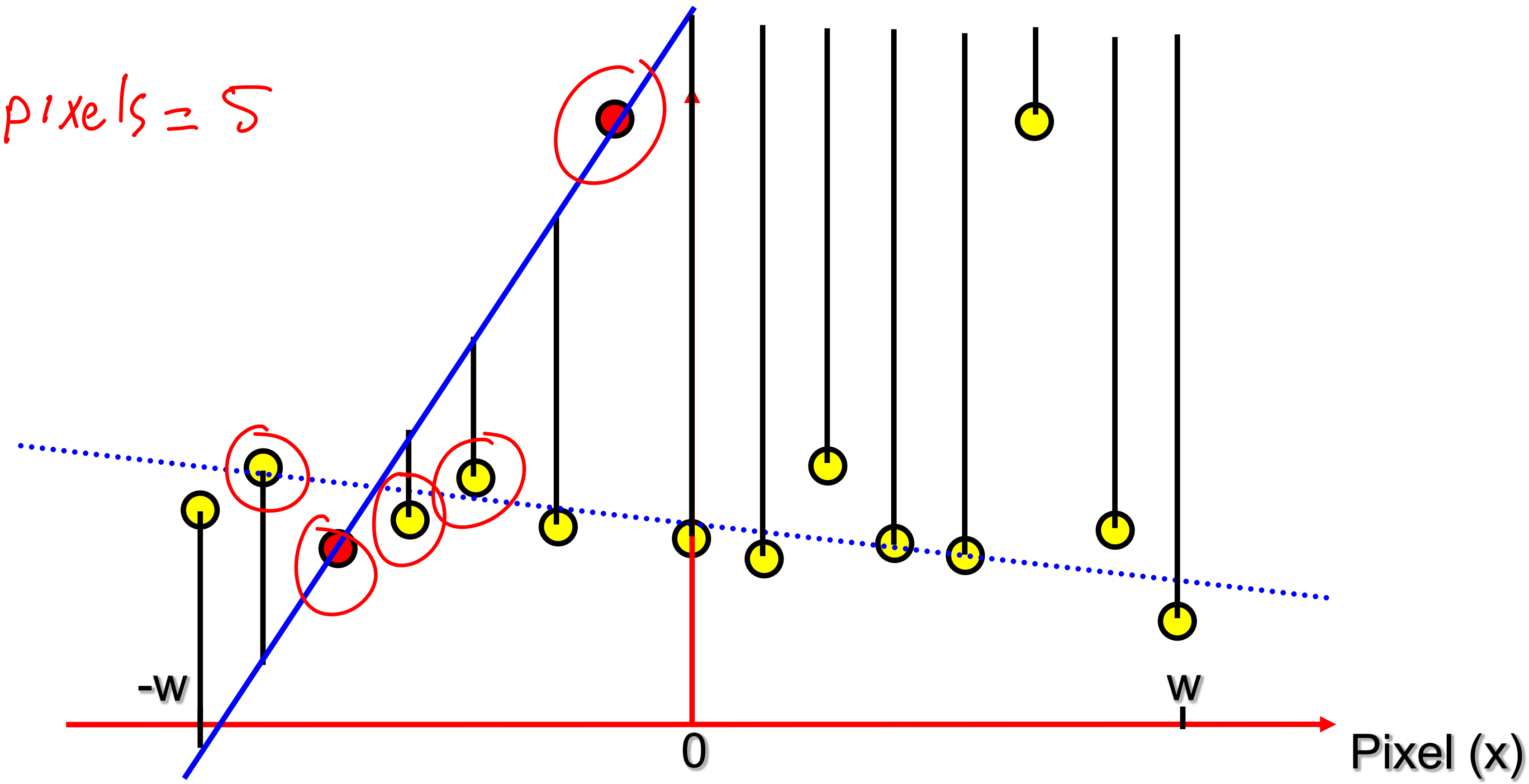


RANSAC Algorithm

Example: Line fitting using RANSAC (i.e., n=2 unknown polynomial coefficients)

- Step 3: Count pixels with vertical distance < threshold t

#pixels = 5



RANSAC Algorithm

Example: Line fitting using RANSAC (i.e., n=2 unknown polynomial coefficients)

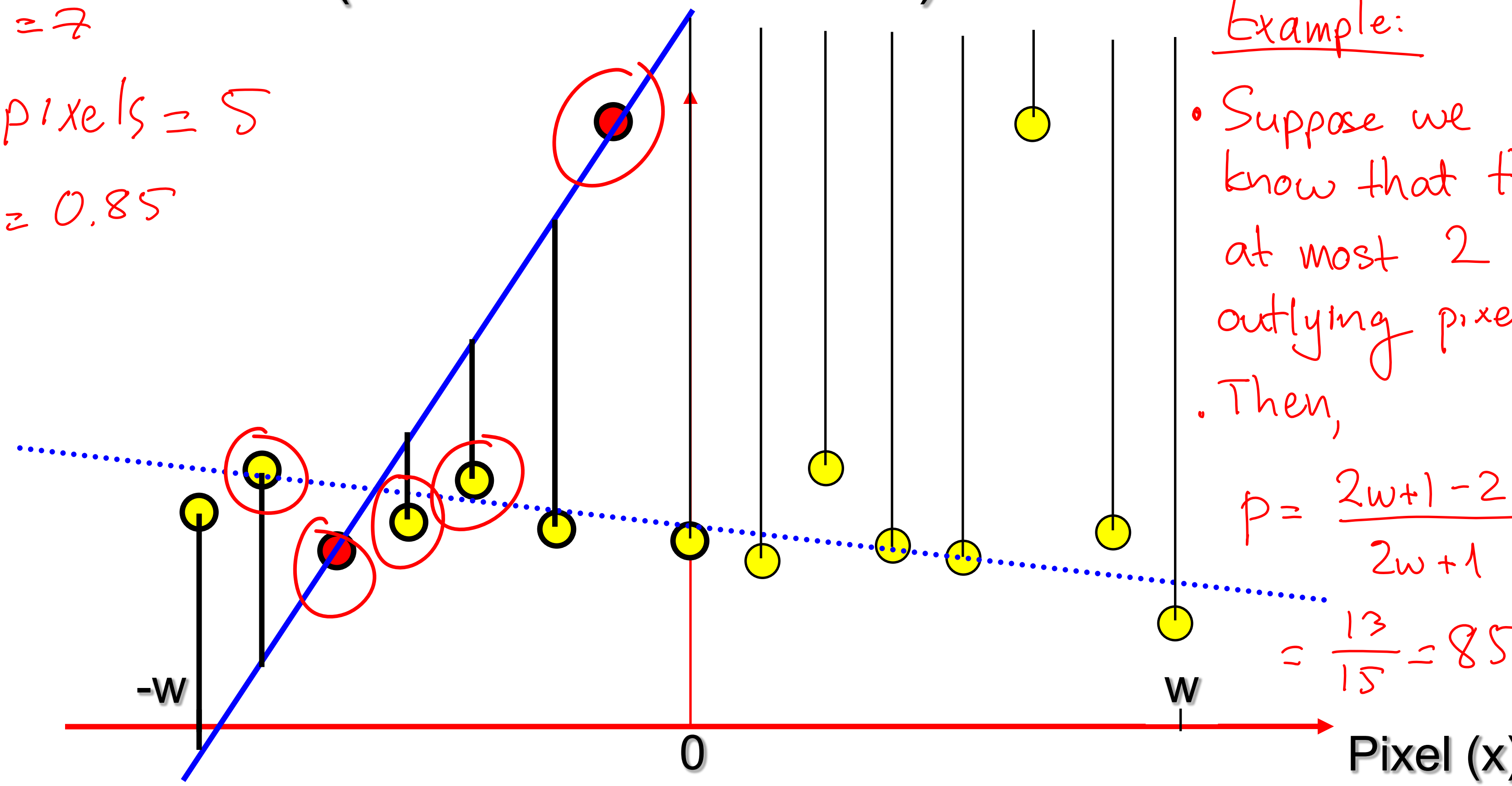
$= (2w+1) \cdot p$ pixels

- Step 4: If there aren't "enough" such pixels, REPEAT (not more than K times)

$w = 7$

$\#\text{pixels} = 5$

$p = 0.85$



Example:

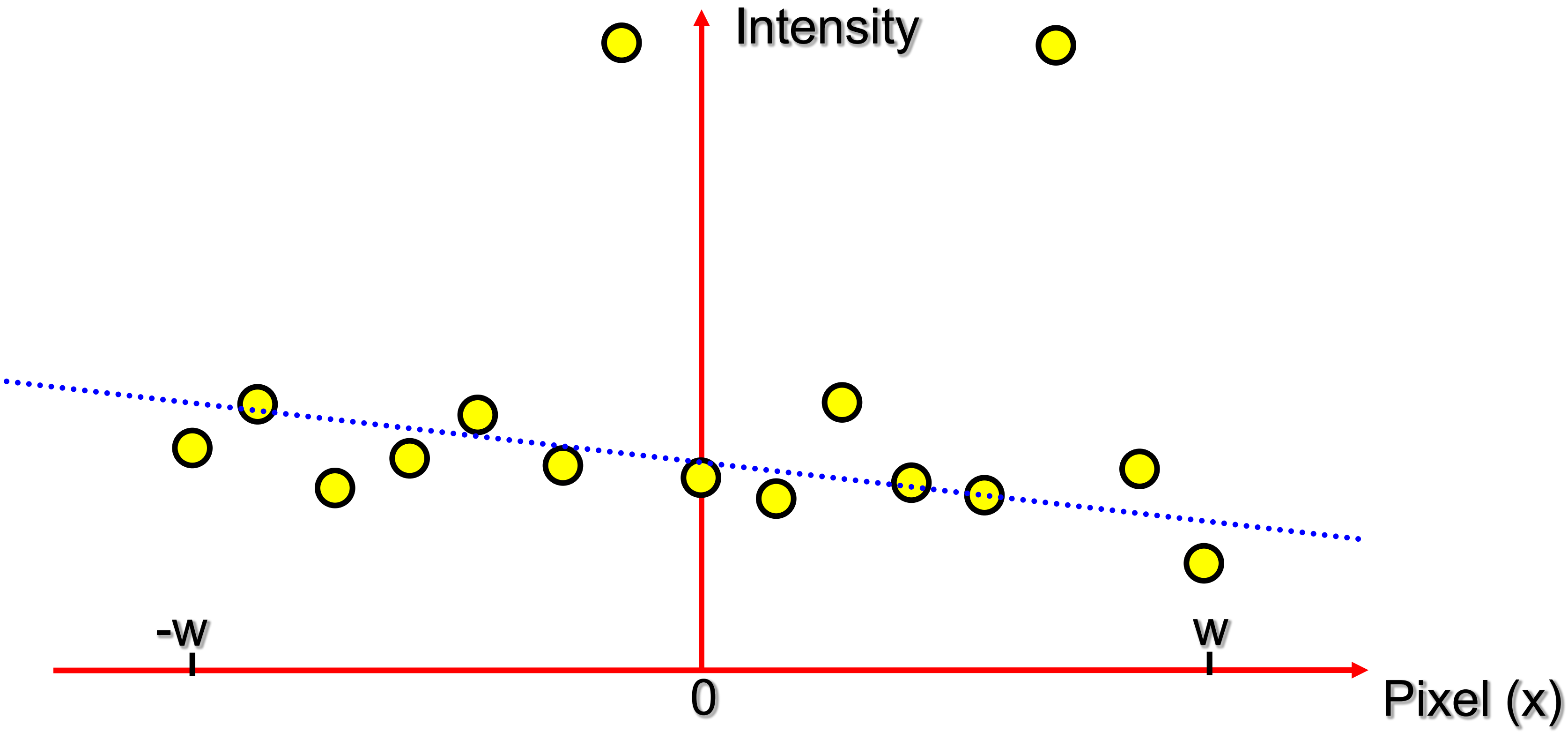
- Suppose we know that there at most 2 outlying pixels.
- Then,

$$p = \frac{2w+1-2}{2w+1} = \frac{13}{15} = 85\%$$

RANSAC Algorithm

Example: Line fitting using RANSAC (i.e., n=2 unknown polynomial coefficients)

- Step 1: Randomly choose n pixels from the patch



RANSAC Algorithm

Example: Line fitting using RANSAC (i.e., n=2 unknown polynomial coefficients)

- Step 4: If there are "enough" such pixels, STOP
  Label them as "inliers" & do a least-squares fit to the INLIER pixels only

$w = 7$

$\#$ pixels $= 13$

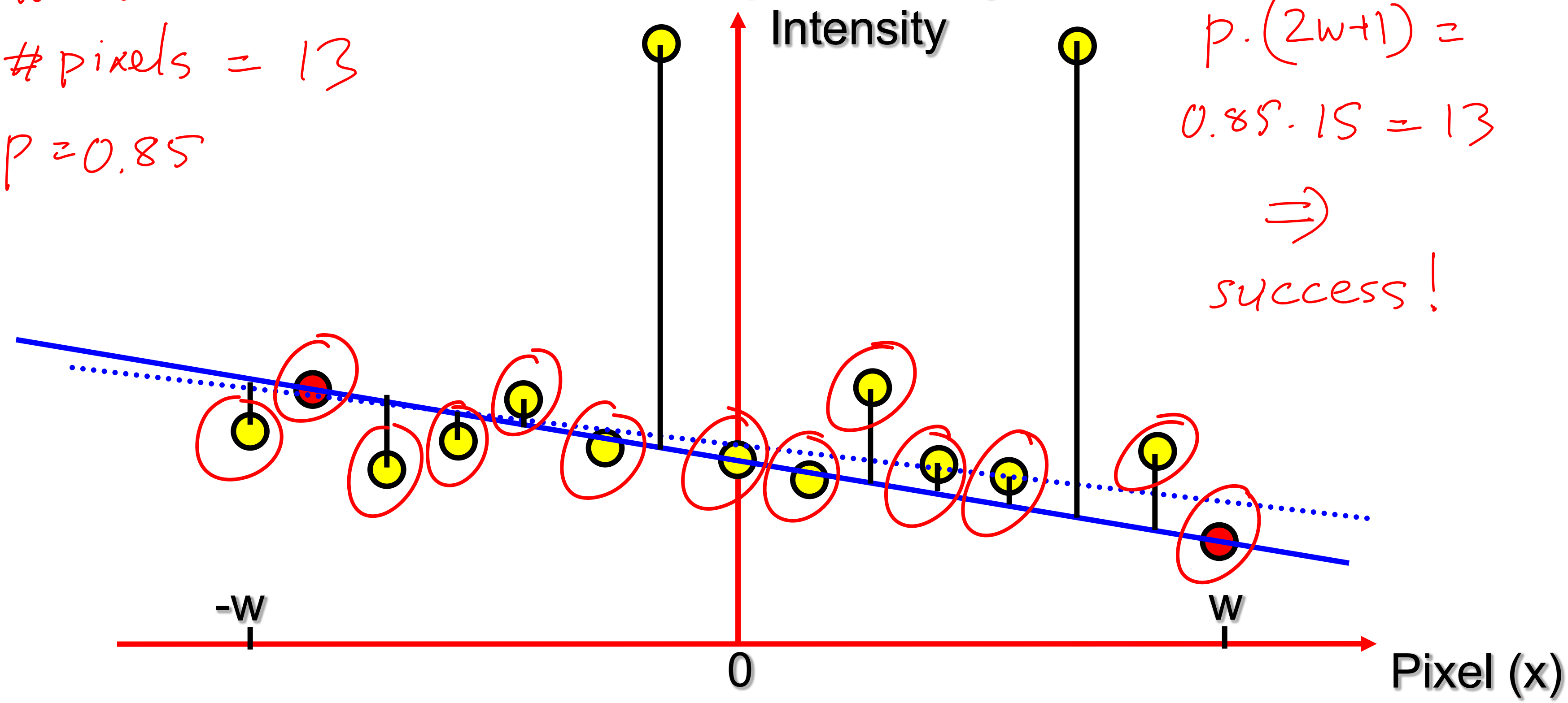
$p = 0.85$

$p \cdot (2w+1) =$

$0.85 \cdot 15 = 13$

$\Rightarrow$

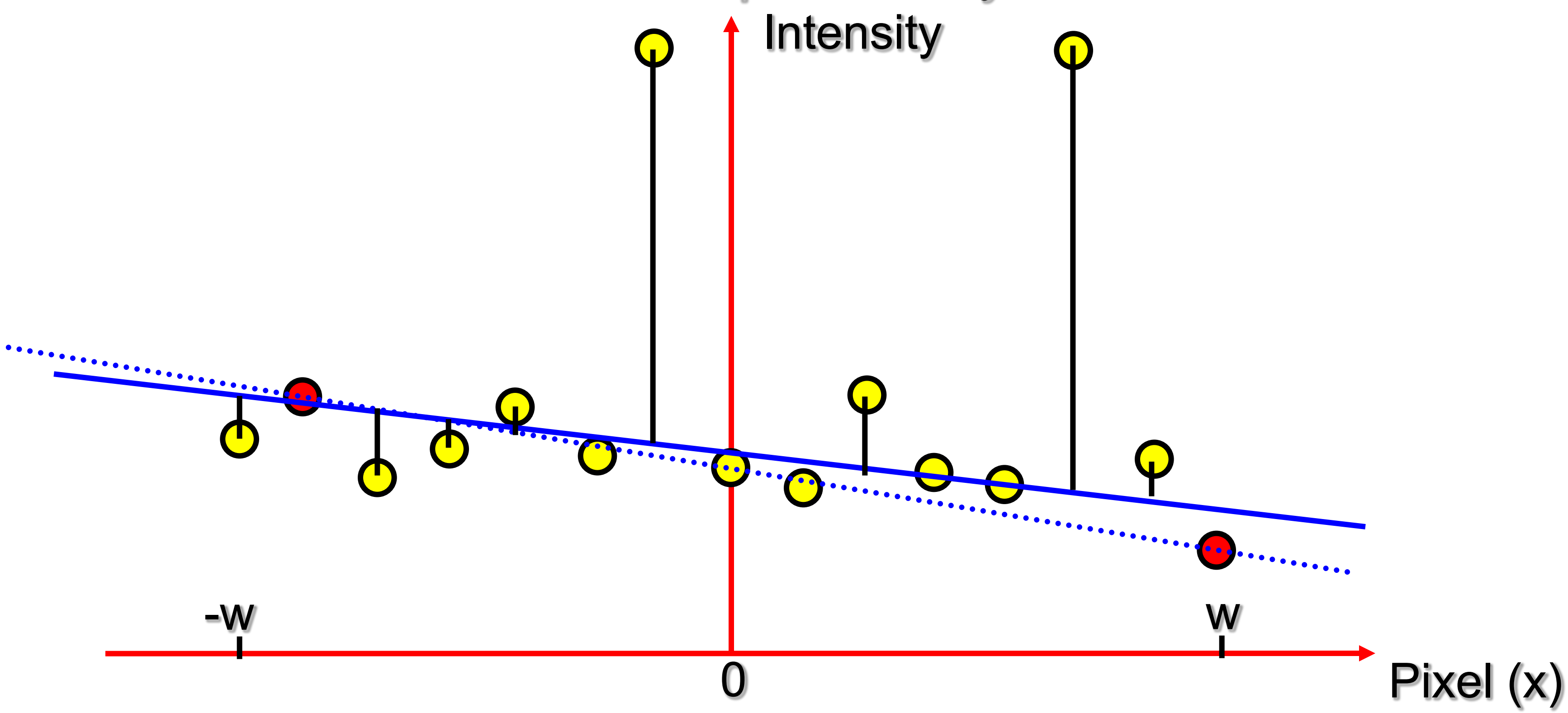
success!



RANSAC Algorithm

Example: Line fitting using RANSAC (i.e., n=2 unknown polynomial coefficients)

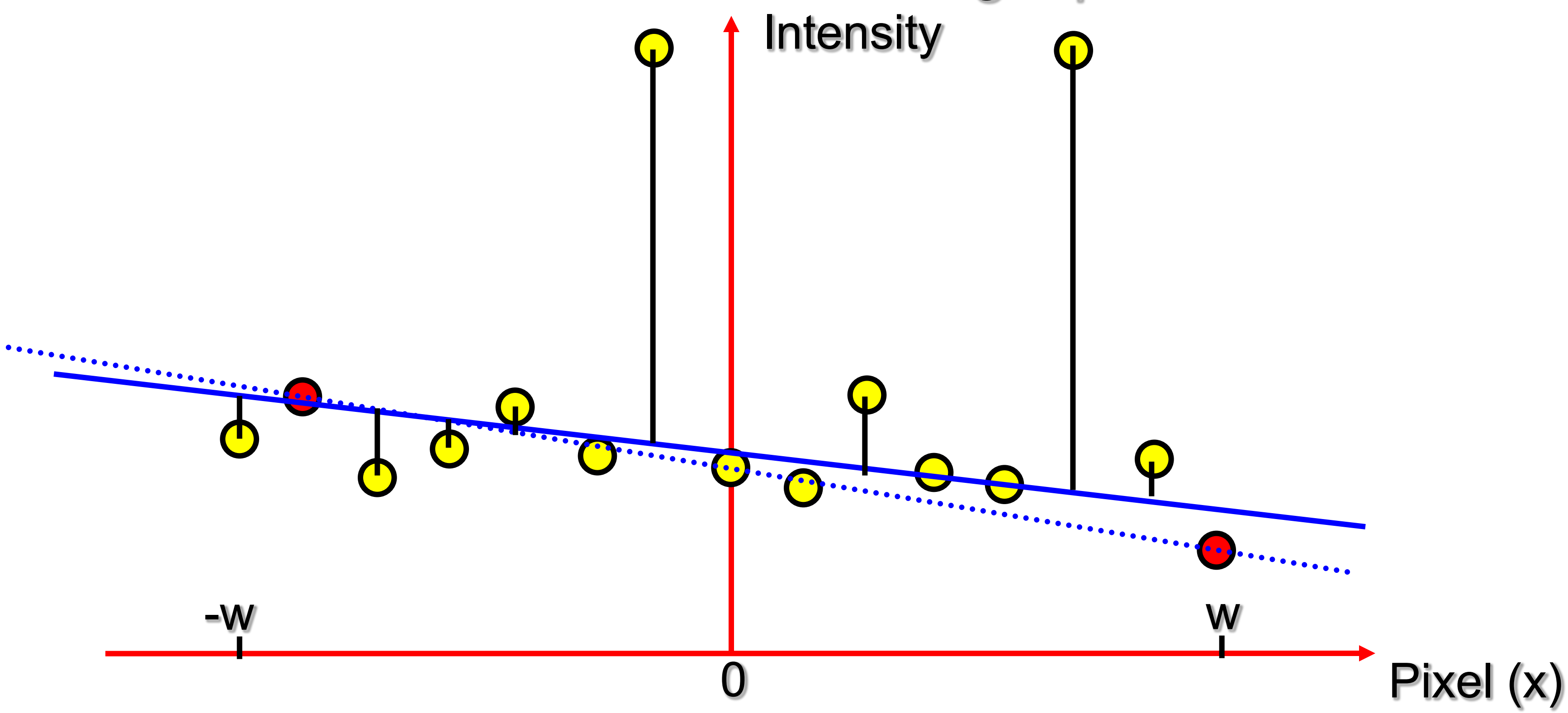
- Step 4: If there are “enough” such pixels, STOP
  Label them as “inliers” & do a least-squares fit to the INLIER pixels only



RANSAC Algorithm

Example: Line fitting using RANSAC (i.e., n=2 unknown polynomial coefficients)

- Idea: Eventually, after "enough" trials, all of the chosen pixels will be inliers ⇒ poly will have vertical distance below t for "enough" pixels



RANSAC Algorithm

Given:

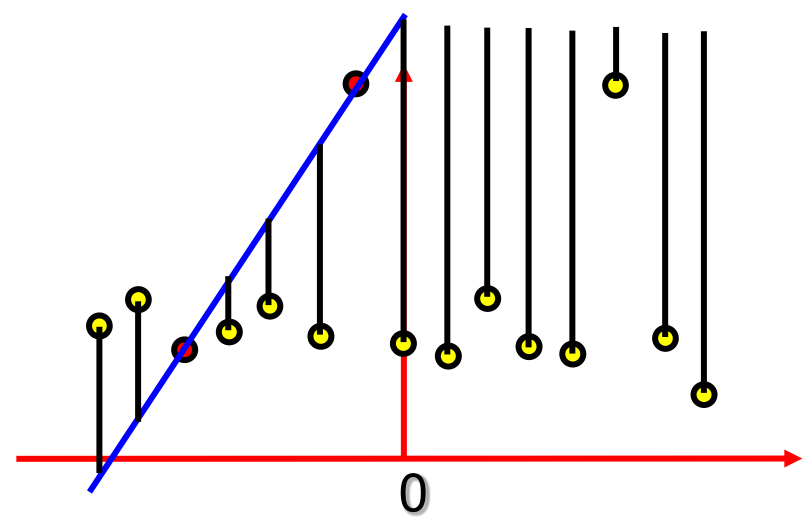
- n = degree of poly
- p = fraction of inliers
- t = fit threshold
- p_s = success probability

Repeat at most K times:

1. Randomly choose n+1 pixels
2. Fit n-degree poly
3. Count pixels whose vertical distance from poly is < t
4. If there are at least (2w+1)p pixels, EXIT LOOP
   a. Label them as inliers
   b. Fit n-degree poly to all inlier pixels

Q: What should K be?

- Probability we chose an inlier pixel: p
- Probability we chose (n+1) inlier pixels: p^{n+1}
- Prob at least 1 outlier chosen: $1-p^{n+1}$
- Prob at least 1 outlier chosen in all K trials: $(1-p^{n+1})^K$



RANSAC Algorithm

Given:

- n = degree of poly
- p = fraction of inliers
- t = fit threshold
- p_s = success probability

Repeat at most K times:

1. Randomly choose n+1 pixels
2. Fit n-degree poly
3. Count pixels whose vertical distance from poly is < t
4. If there are at least (2w+1)p pixels, EXIT LOOP
   a. Label them as inliers
   b. Fit n-degree poly to all inlier pixels

Q: What should K be?

- Probability we chose an inlier pixel: p
- Probability we chose (n+1) inlier pixels: p^{n+1}
- Prob at least 1 outlier chosen: $1-p^{n+1}$
- Prob at least 1 outlier chosen in all K trials: $(1-p^{n+1})^K$
- Failure probability: $(1-p^{n+1})^K$
- Success probability $p_s = 1-(1-p^{n+1})^K$
- By taking logs on both sides

$$\boxed{K = \frac{\log(1-p_s)}{\log(1-p^{n+1})}}$$