

Computer Vision

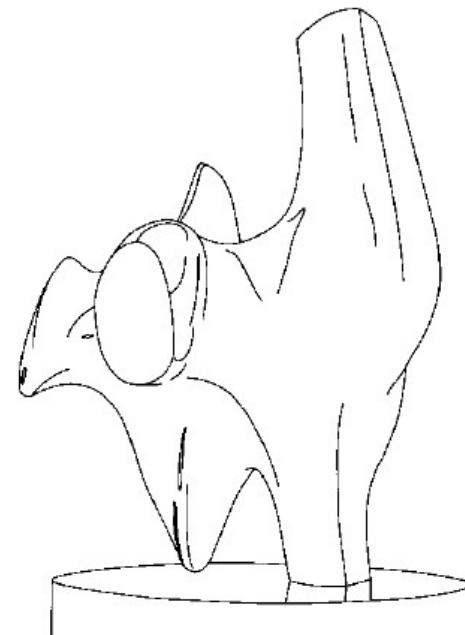
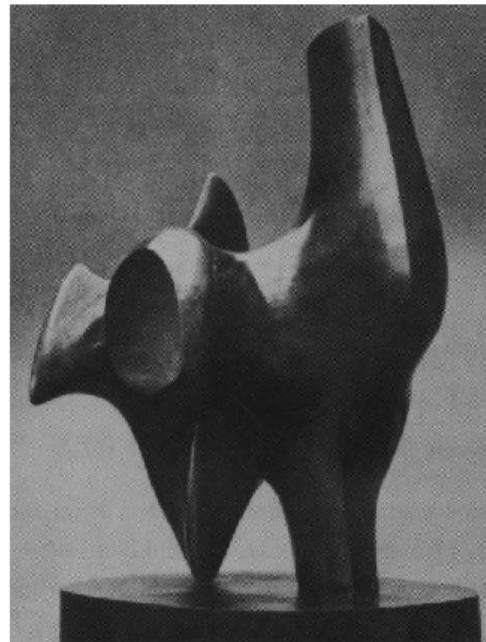
Lecture 2: Edge detection

SHADOW

From [Sandlot Science](#)

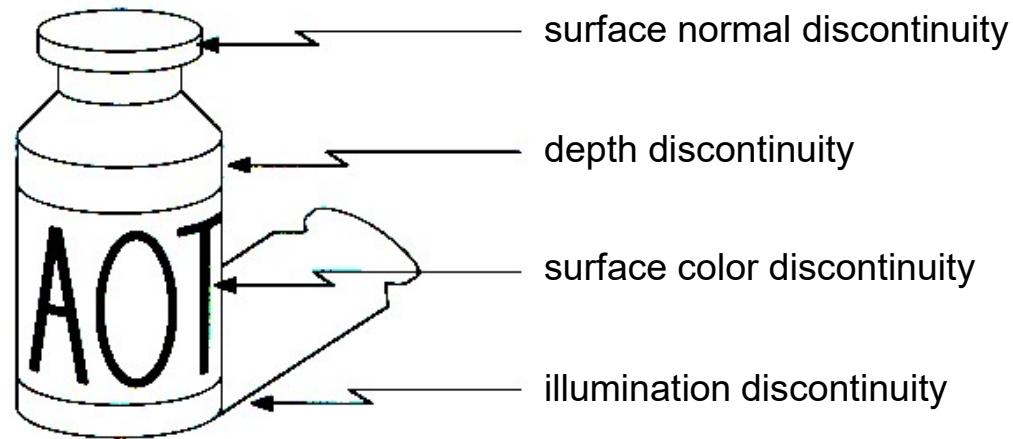


Edge detection



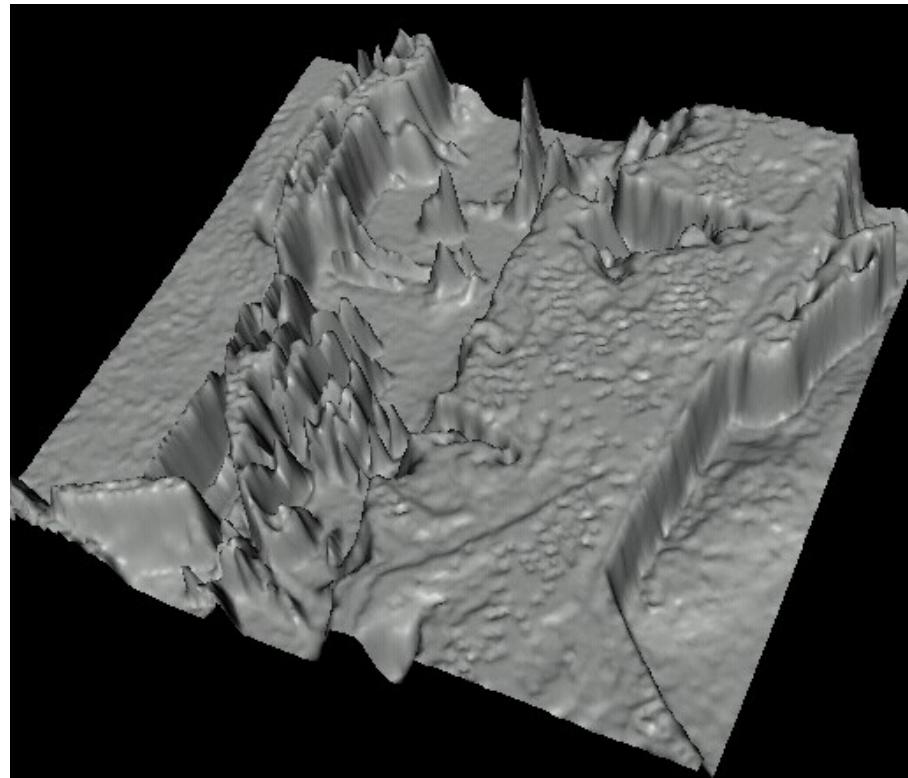
- Convert a 2D image into a set of curves
 - Extracts salient features of the scene
 - More compact than pixels

Origin of edges



- Edges are caused by a variety of factors

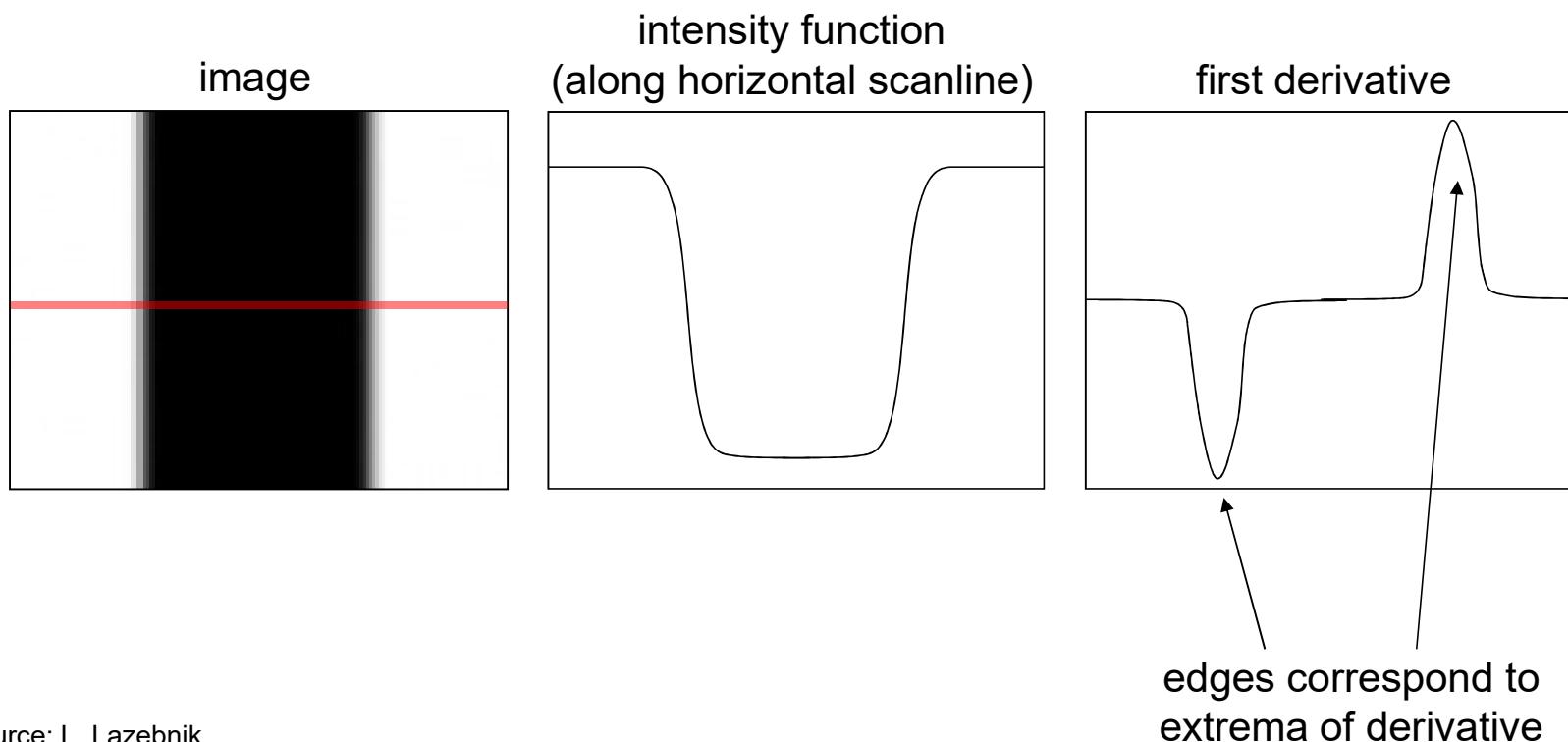
Images as functions...



- Edges look like steep cliffs

Characterizing edges

- An edge is a place of *rapid change* in the image intensity function



Source: L. Lazebnik

Image derivatives

- How can we differentiate a *digital* image $F[x,y]$?
 - Option 1: reconstruct a continuous image, f , then compute the derivative
 - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x, y] \approx F[x + 1, y] - F[x, y]$$

How would you implement this as a linear filter?

$$\frac{\partial f}{\partial x} : \begin{array}{|c|c|c|}\hline & & \\ \hline \end{array}$$

H_x

$$\frac{\partial f}{\partial y} : \begin{array}{|c|c|c|}\hline & & \\ \hline \end{array}$$

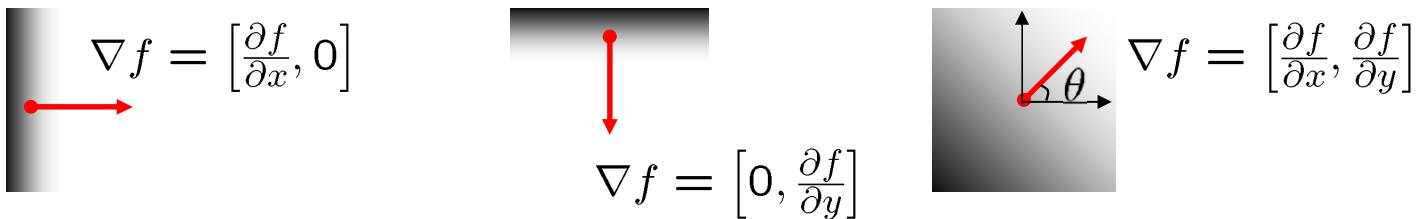
H_y

Source: S. Seitz

Image gradient

- The *gradient* of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$

The gradient points in the direction of most rapid increase in intensity



The *edge strength* is given by the gradient magnitude:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

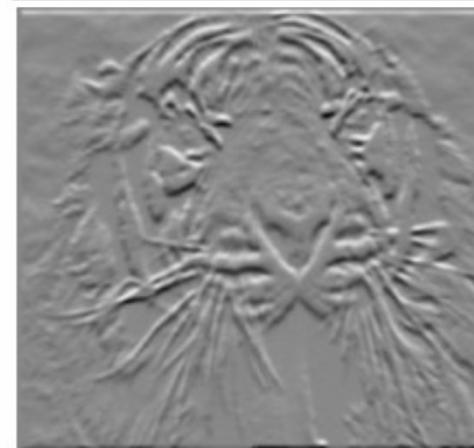
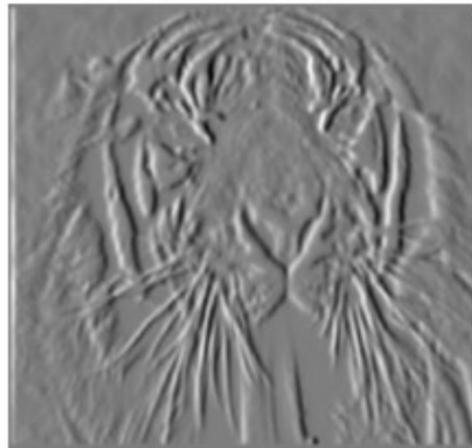
The gradient direction is given by:

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

- how does this relate to the direction of the edge?

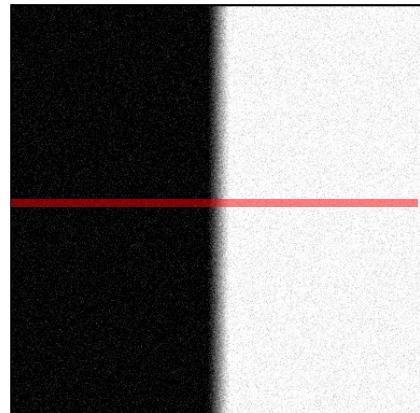
Source: Steve Seitz

Image gradient



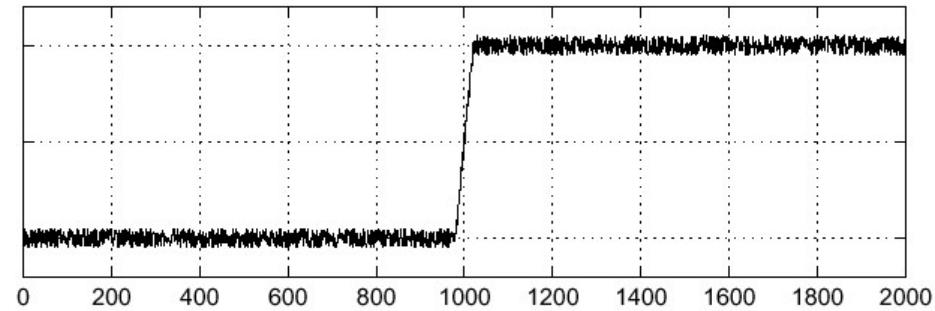
Source: L. Lazebnik

Effects of noise

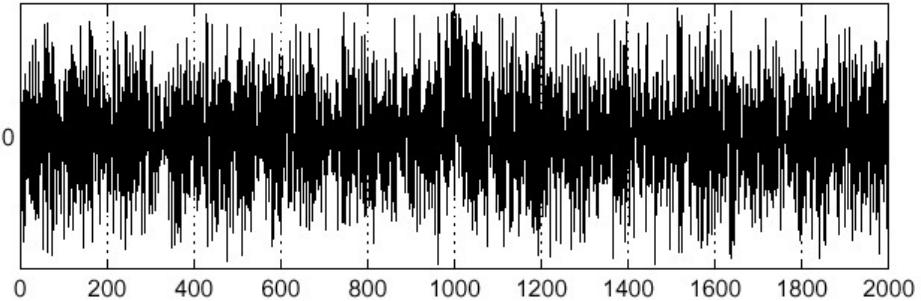


Noisy input image

$$f(x)$$



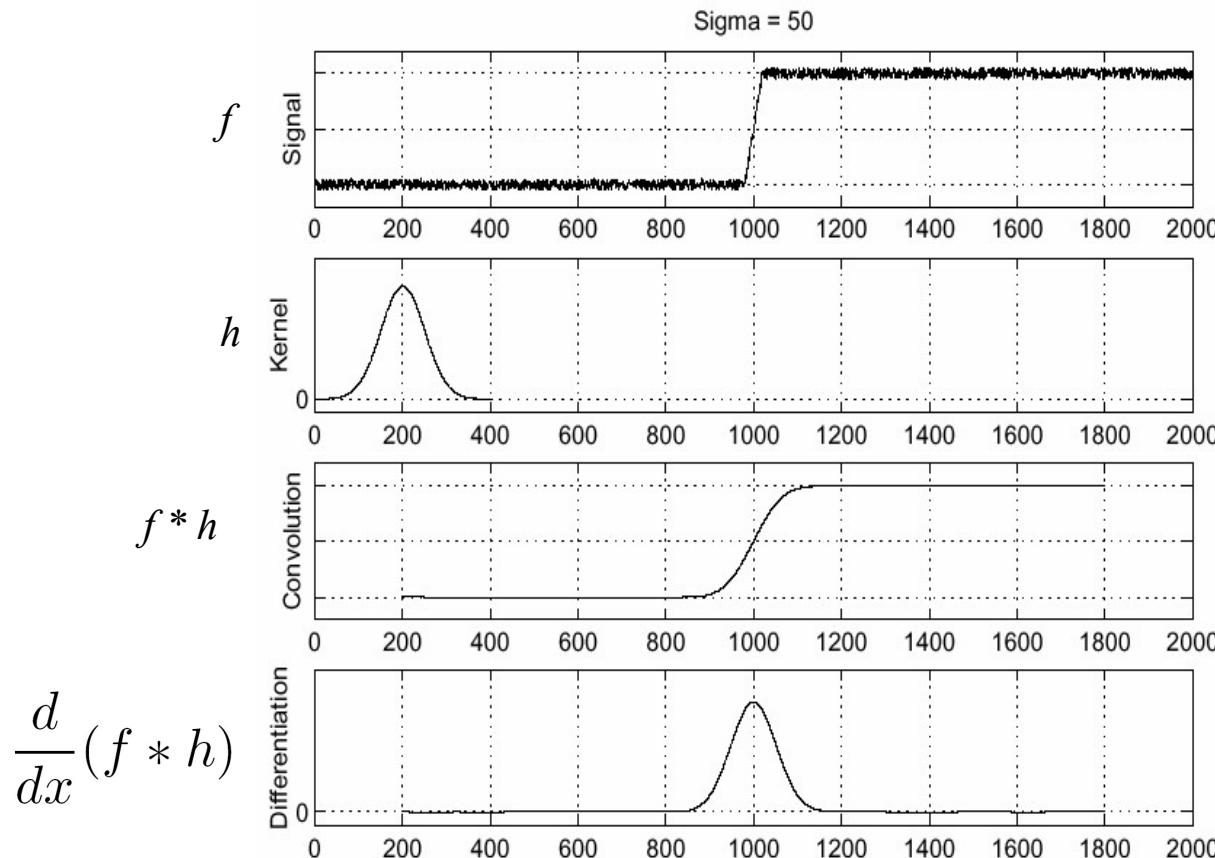
$$\frac{d}{dx}f(x)$$



Where is the edge?

Source: S. Seitz

Solution: smooth first

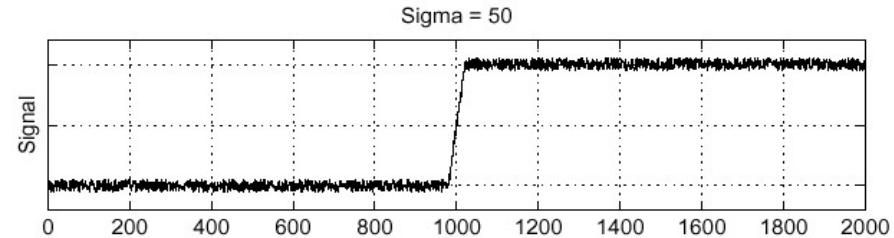


To find edges, look for peaks in $\frac{d}{dx}(f * h)$

Source: S. Seitz

Associative property of convolution

- Differentiation is convolution, and convolution is associative: $\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$
- This saves us one operation: f

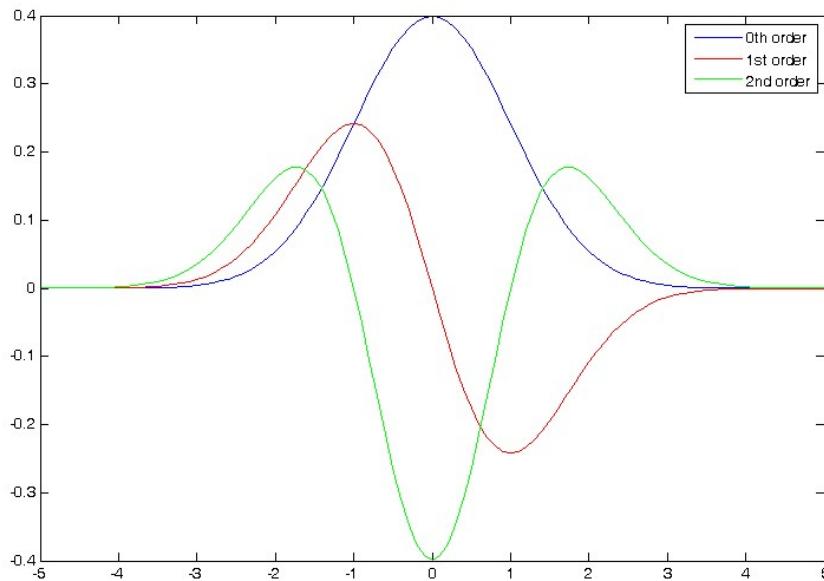


Source: S. Seitz

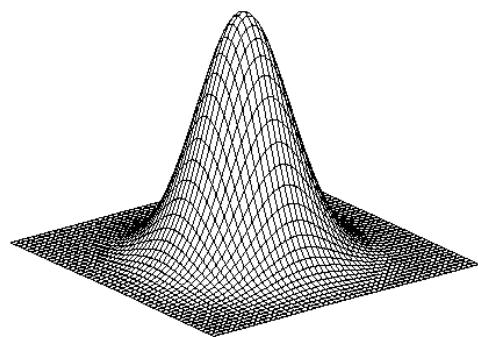
The 1D Gaussian and its derivatives

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma} \left(\frac{x}{\sigma}\right) G_\sigma(x)$$

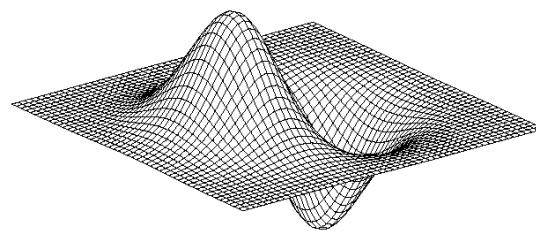


2D edge detection filters



Gaussian

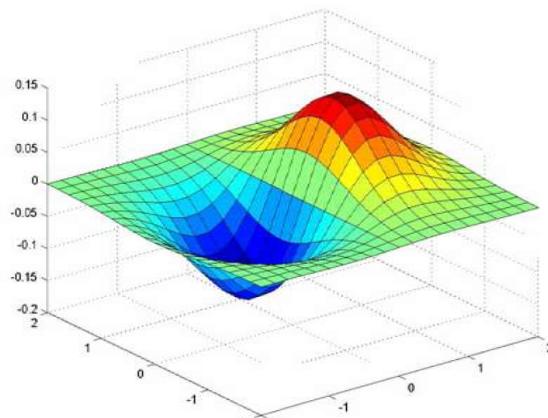
$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



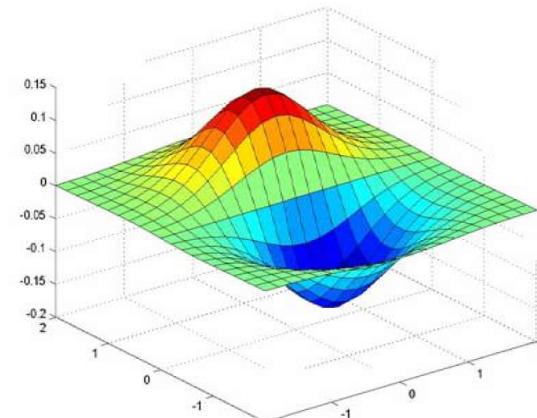
derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u, v)$$

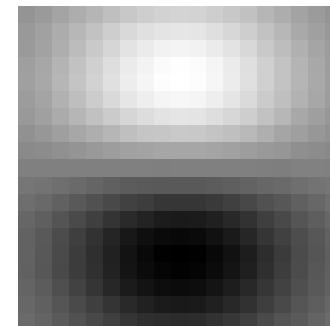
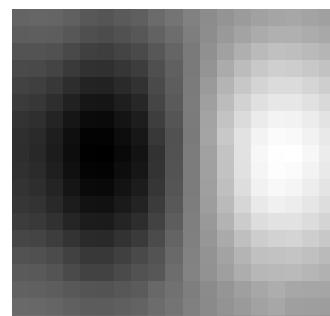
Derivative of Gaussian filter



x-direction



y-direction



The Sobel operator

- Common approximation of derivative of Gaussian

$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

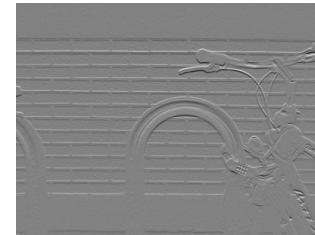
s_x

$$\frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

s_y

- The standard definition of the Sobel operator omits the $1/8$ term
 - doesn't make a difference for edge detection
 - the $1/8$ term **is** needed to get the right gradient magnitude

Sobel operator: example



Source: Wikipedia

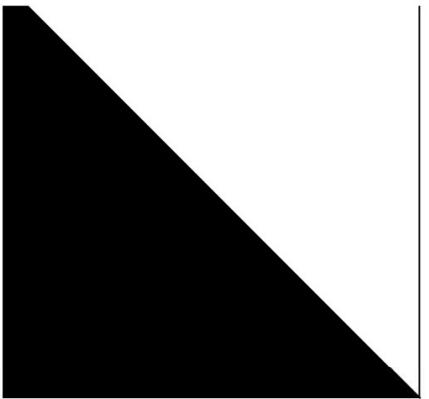
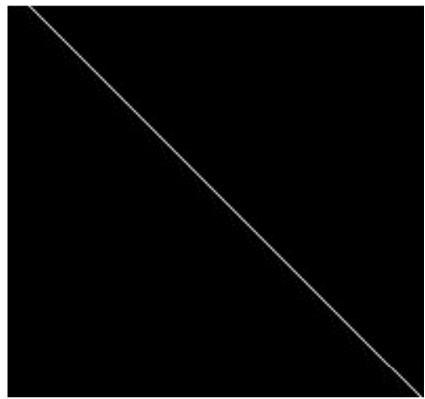


Image with Edge



Edge Location

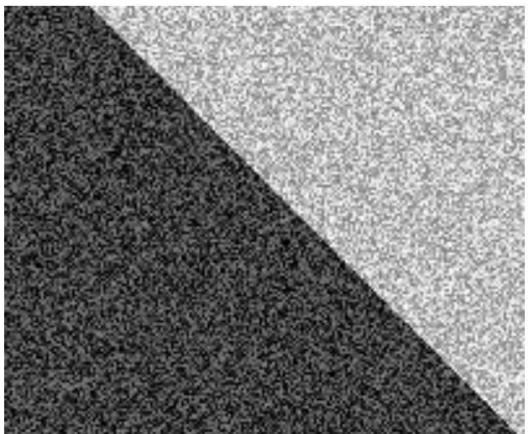
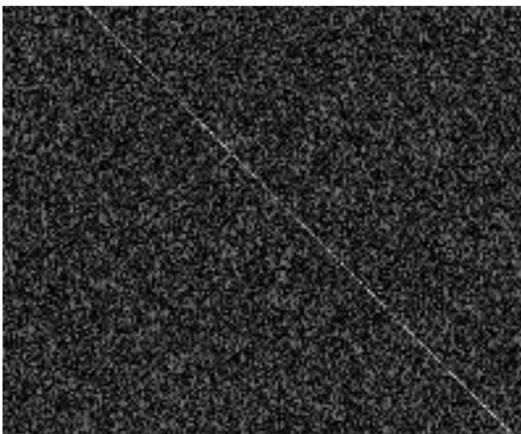
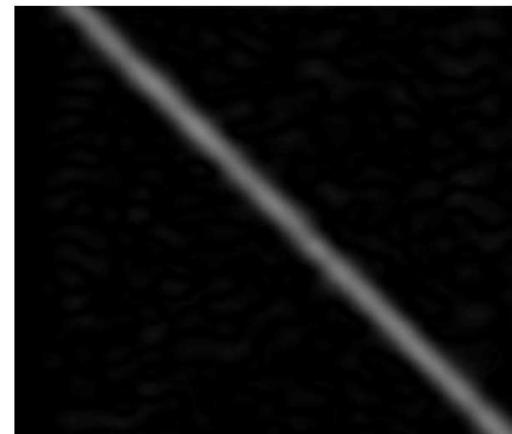


Image + Noise



Derivatives detect
edge *and* noise



Smoothed derivative removes
noise, but blurs edge

Example



original image

Demo: <http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/>

Image credit: Joseph Redmon

Finding edges



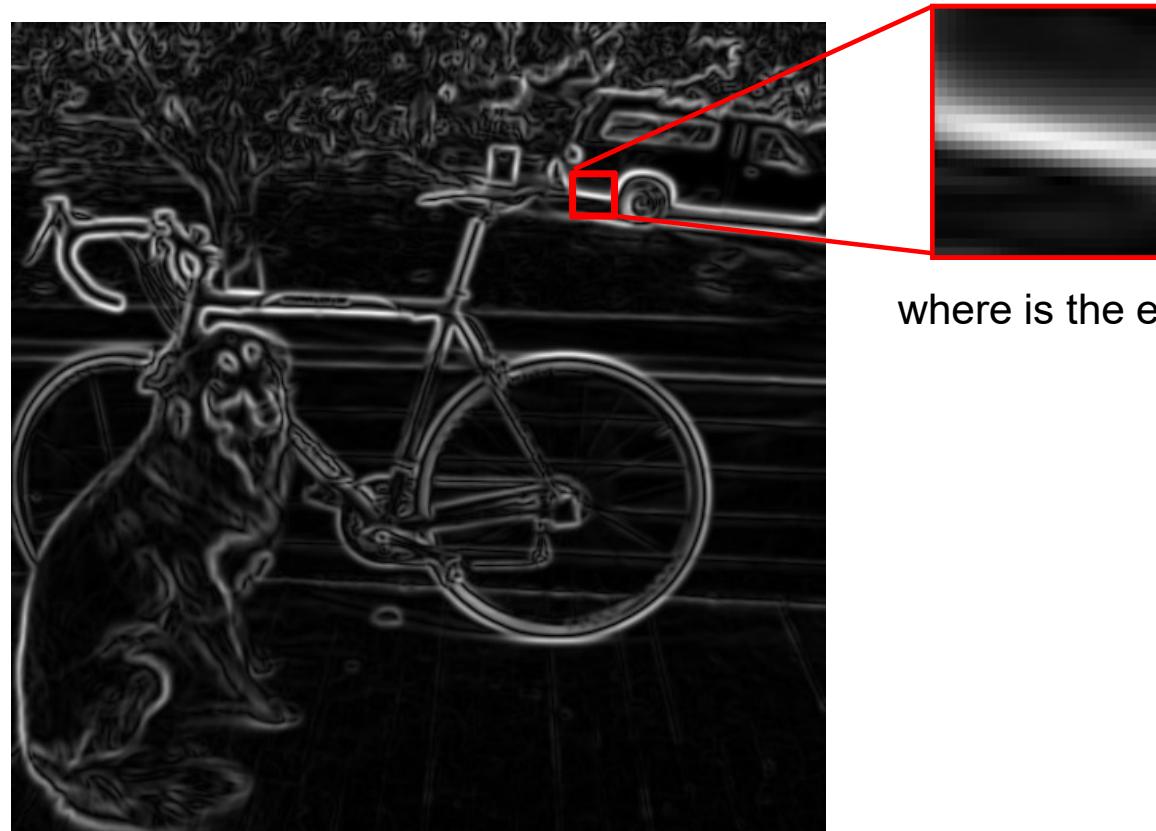
smoothed gradient magnitude

Finding edges



smoothed gradient magnitude

Finding edges

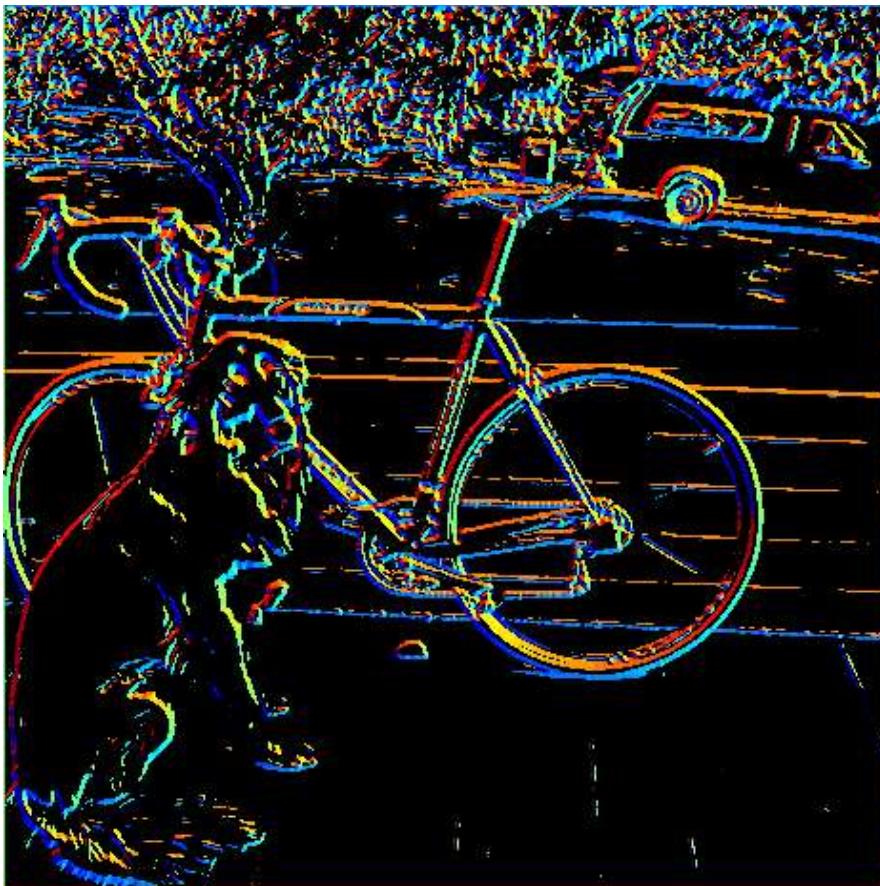


where is the edge?

thresholding

Get Orientation at Each Pixel

- Get orientation (below, threshold at minimum gradient magnitude)



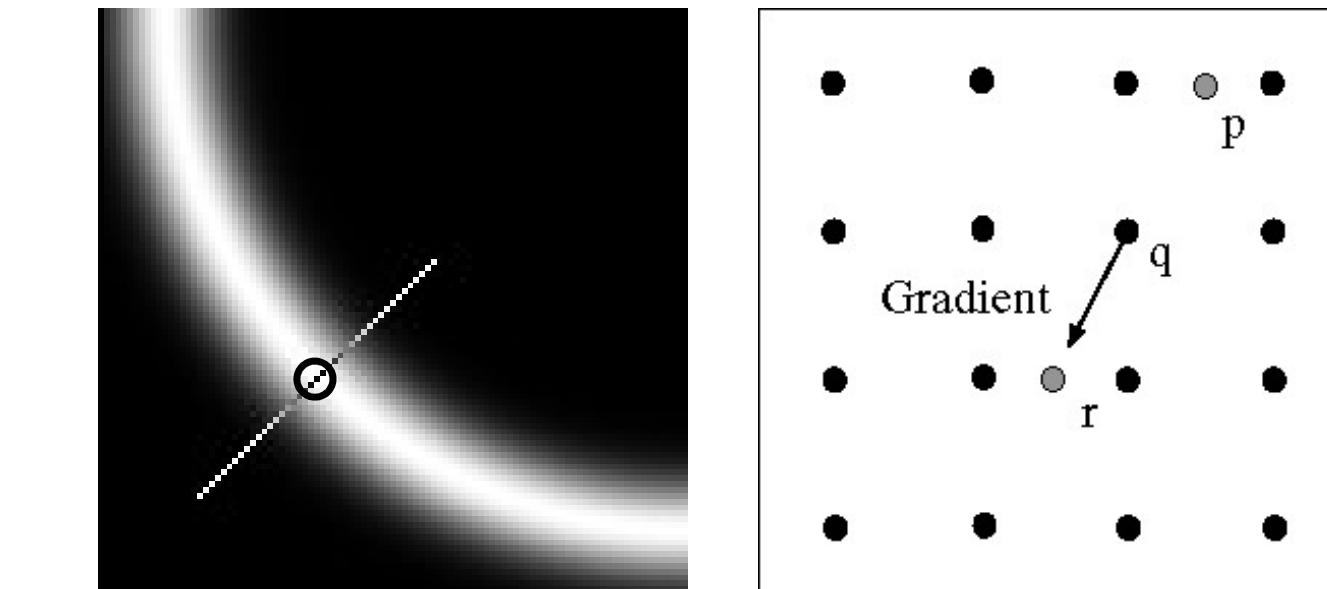
$\theta = \text{atan2}(g_y, g_x)$

360

0

Gradient orientation angle

Non-maximum suppression

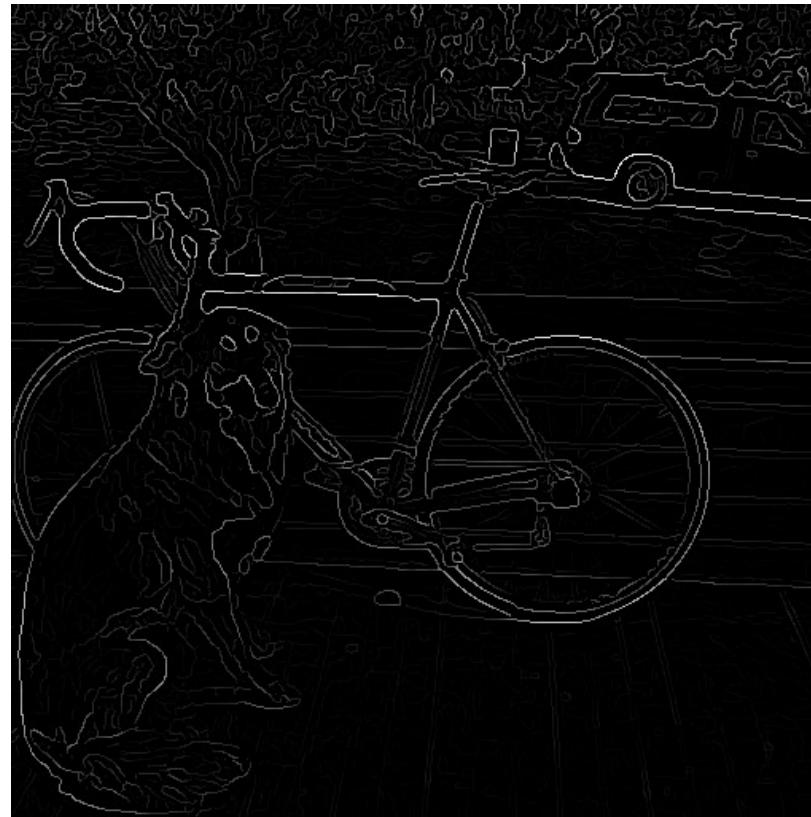


- Check if pixel is local maximum along gradient direction
 - requires *interpolating* pixels p and r

Before Non-max Suppression



After Non-max Suppression



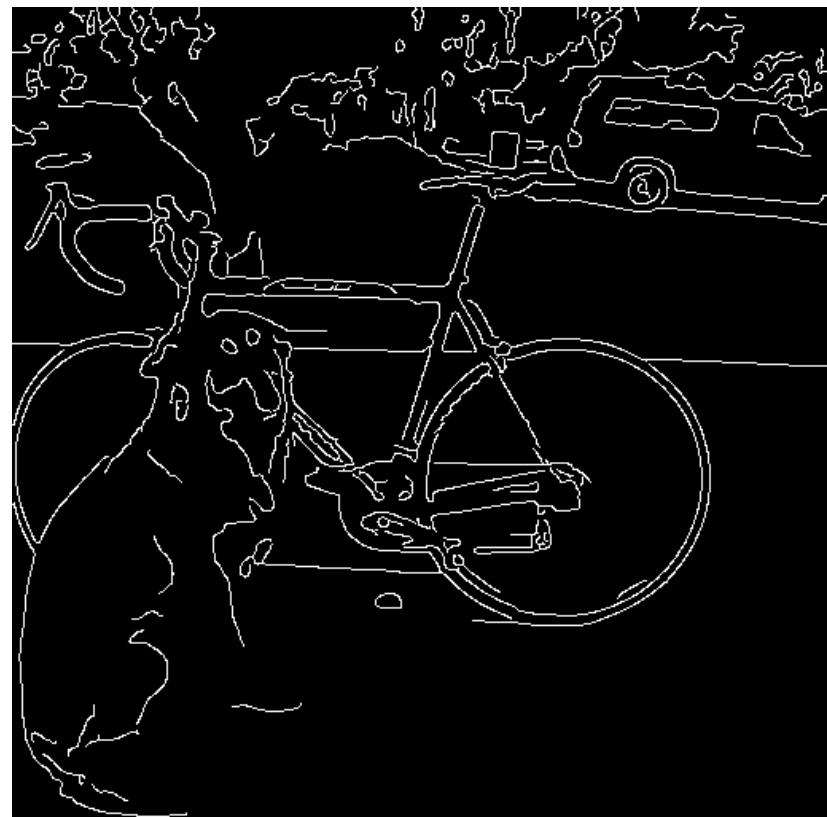
Thresholding edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
 - $R > T$: strong edge
 - $R < T$ but $R > t$: weak edge
 - $R < t$: no edge
- Why two thresholds?



Connecting edges

- Strong edges are edges!
- Weak edges are edges iff they connect to strong
- Look in some neighborhood (usually 8 closest)



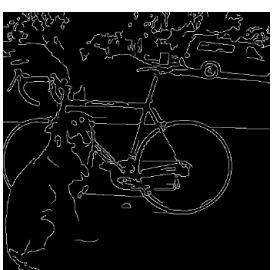
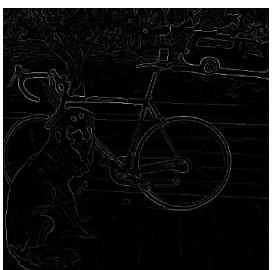
Canny edge detector



MATLAB: `edge(image, 'canny')`



1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them



Source: D. Lowe, L. Fei-Fei, J. Redmon

Canny edge detector

- Our first computer vision pipeline!
- Still a widely used edge detector in computer vision

J. Canny, [A Computational Approach To Edge Detection](#), IEEE
Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

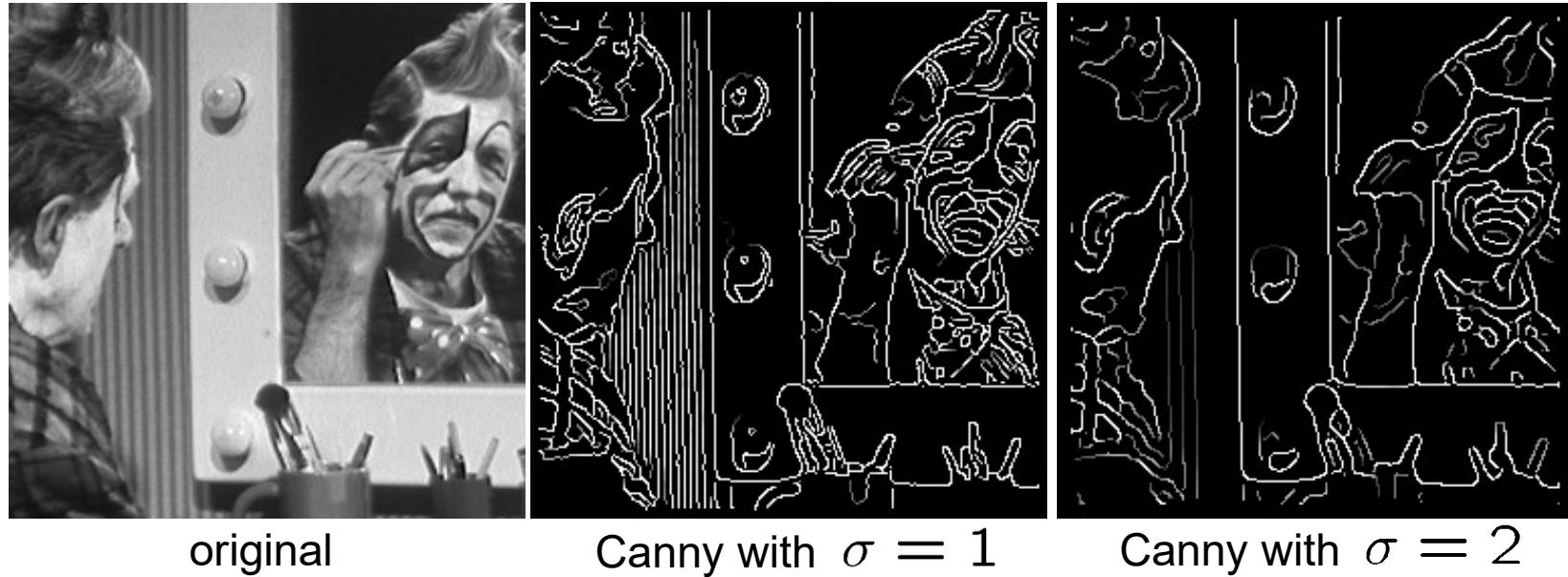
- Depends on several parameters:

high threshold

low threshold

σ : width of the Gaussian blur

Canny edge detector



- The choice of σ depends on desired behavior
 - large σ detects “large-scale” edges
 - small σ detects fine edges

Source: S. Seitz

Questions?

Computer Vision

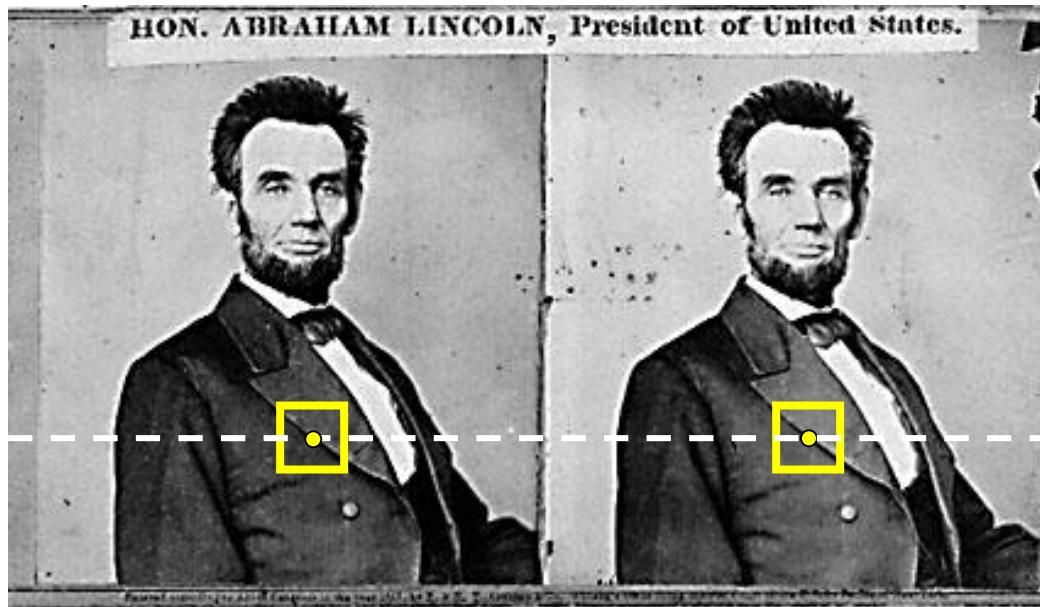
Two-view geometry



Reading

- Reading: Szeliski (2nd Edition), 12.1 12.6

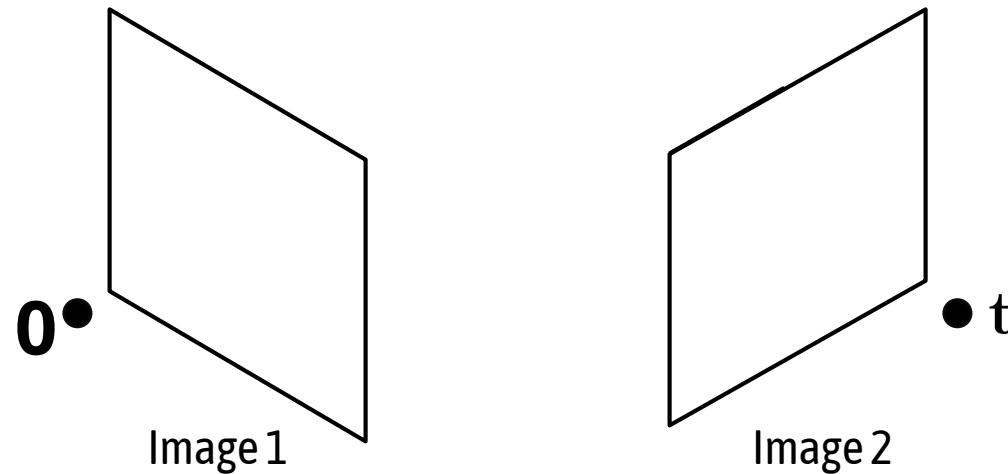
Back to stereo



- Where do epipolar lines come from?

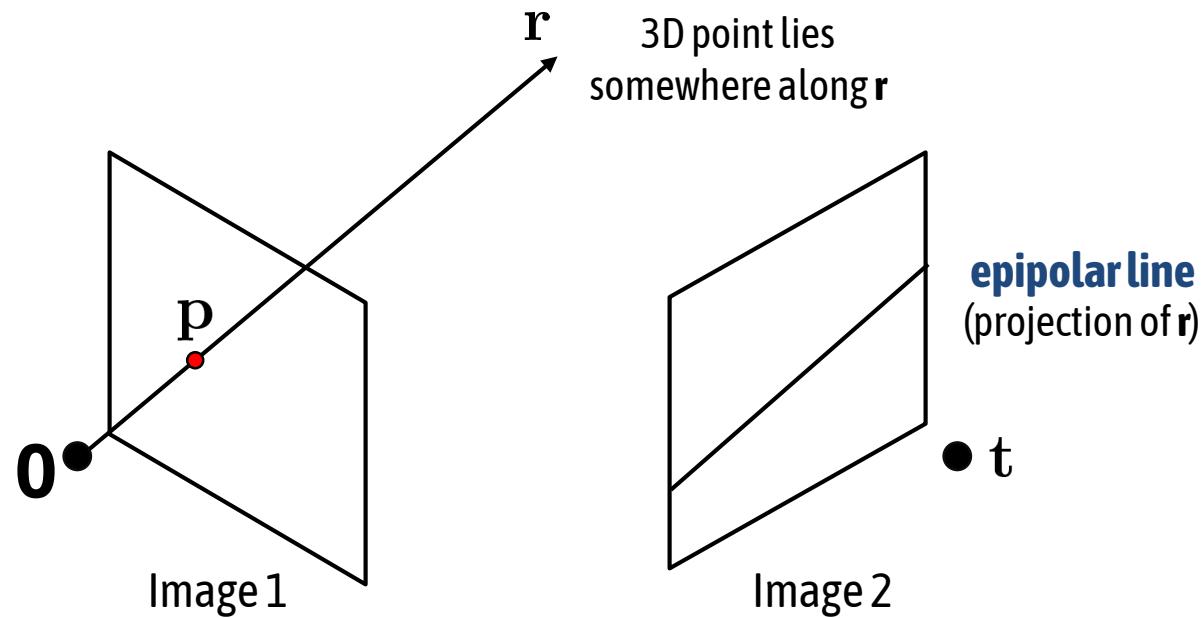
Two-view geometry

- Where do epipolar lines come from?



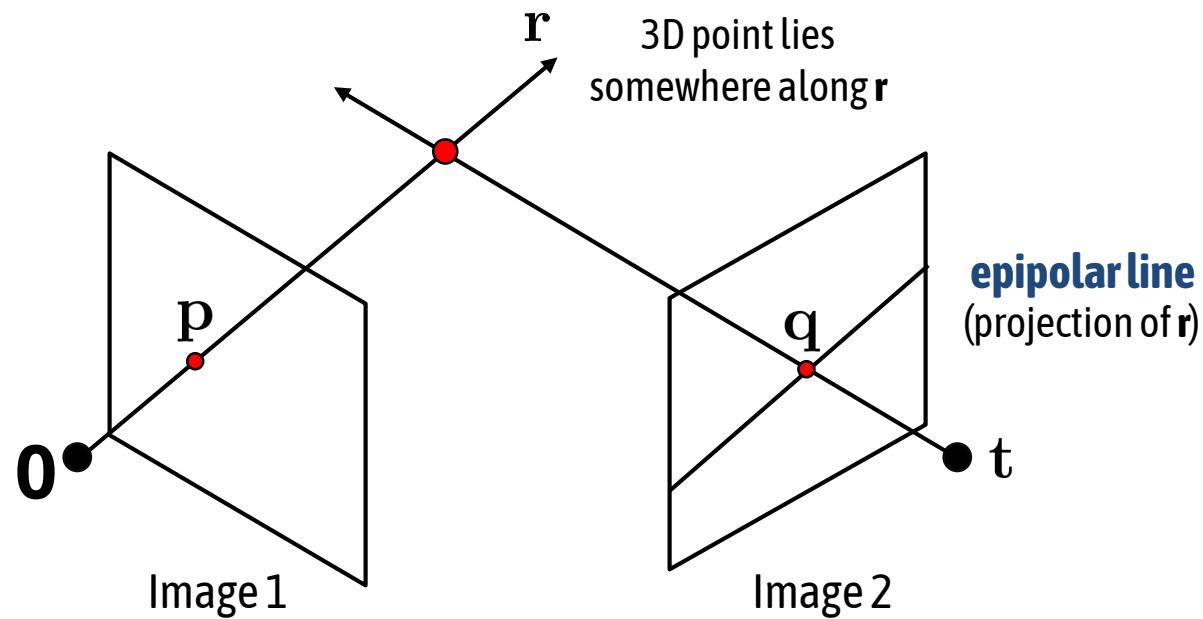
Two-view geometry

- Where do epipolar lines come from?



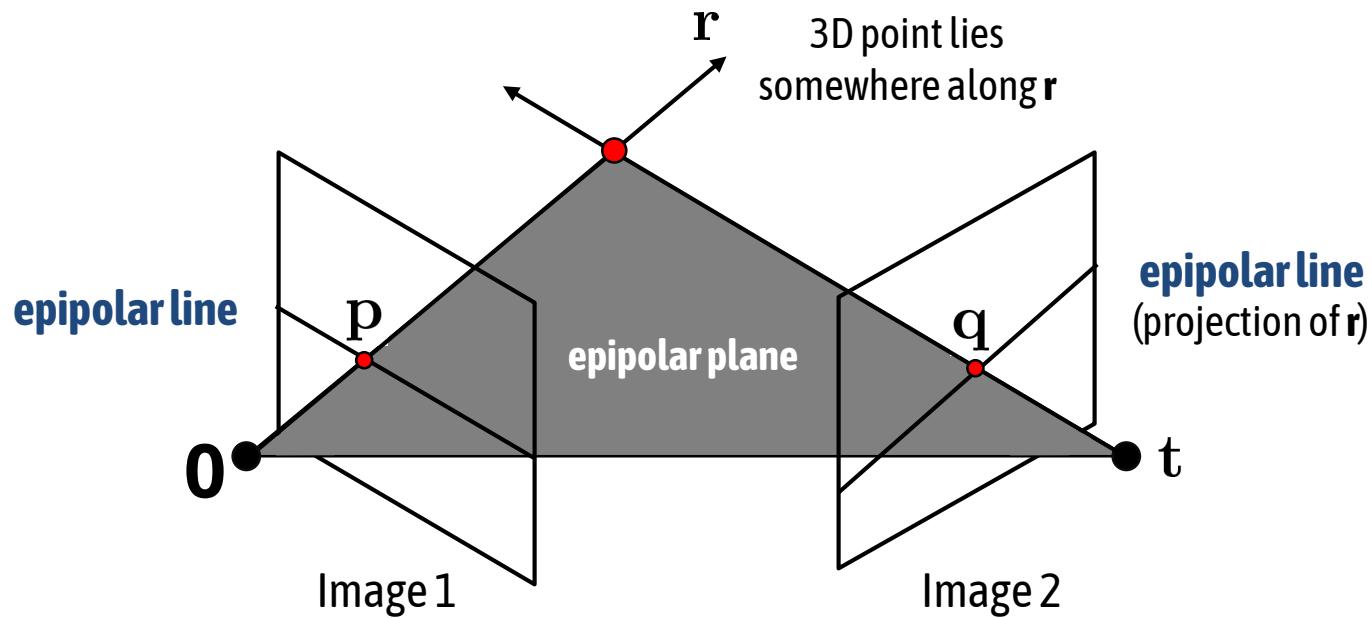
Two-view geometry

- Where do epipolar lines come from?

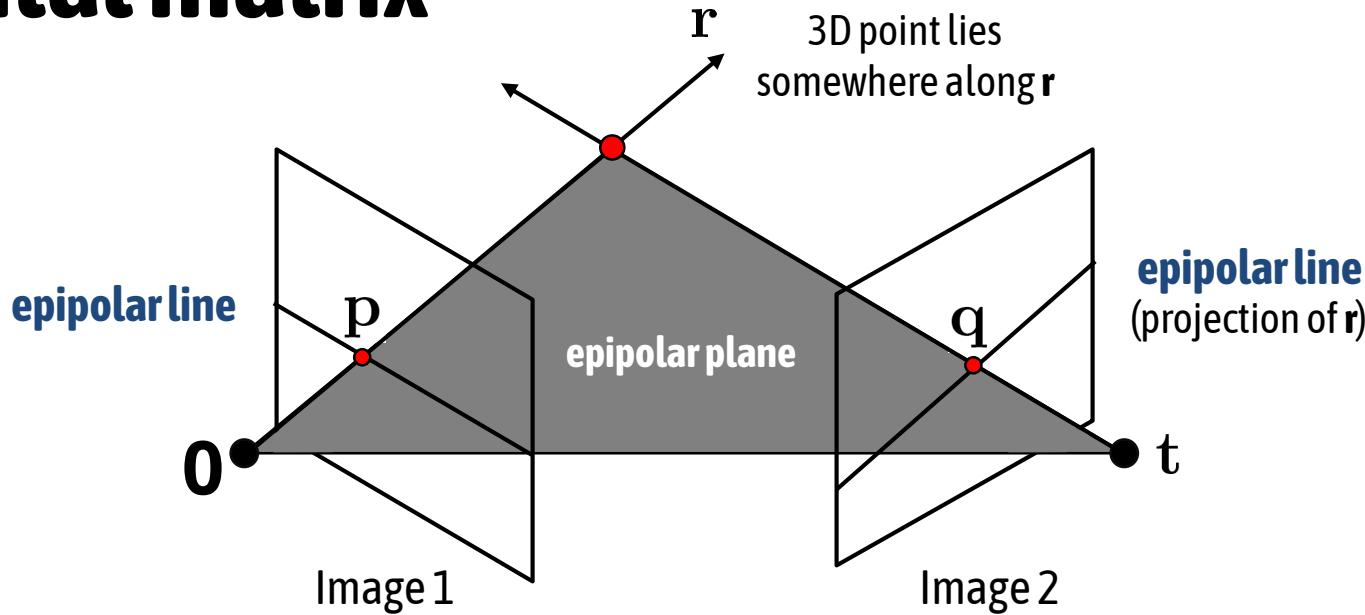


Two-view geometry

- Where do epipolar lines come from?

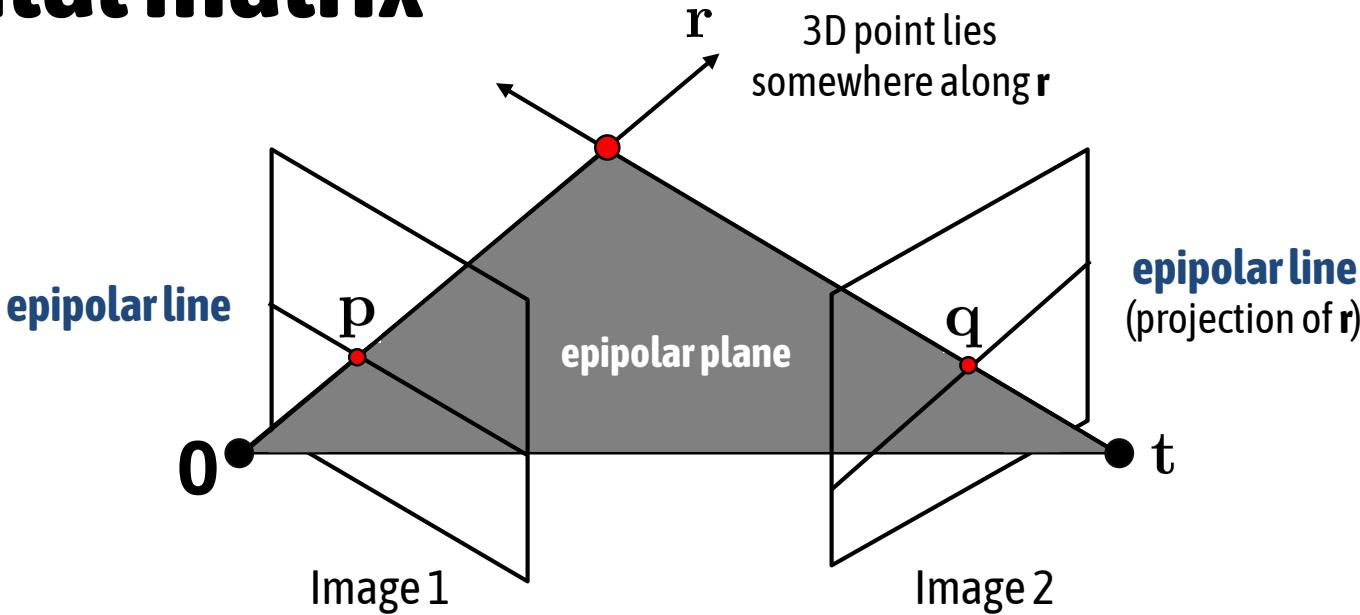


Fundamental matrix



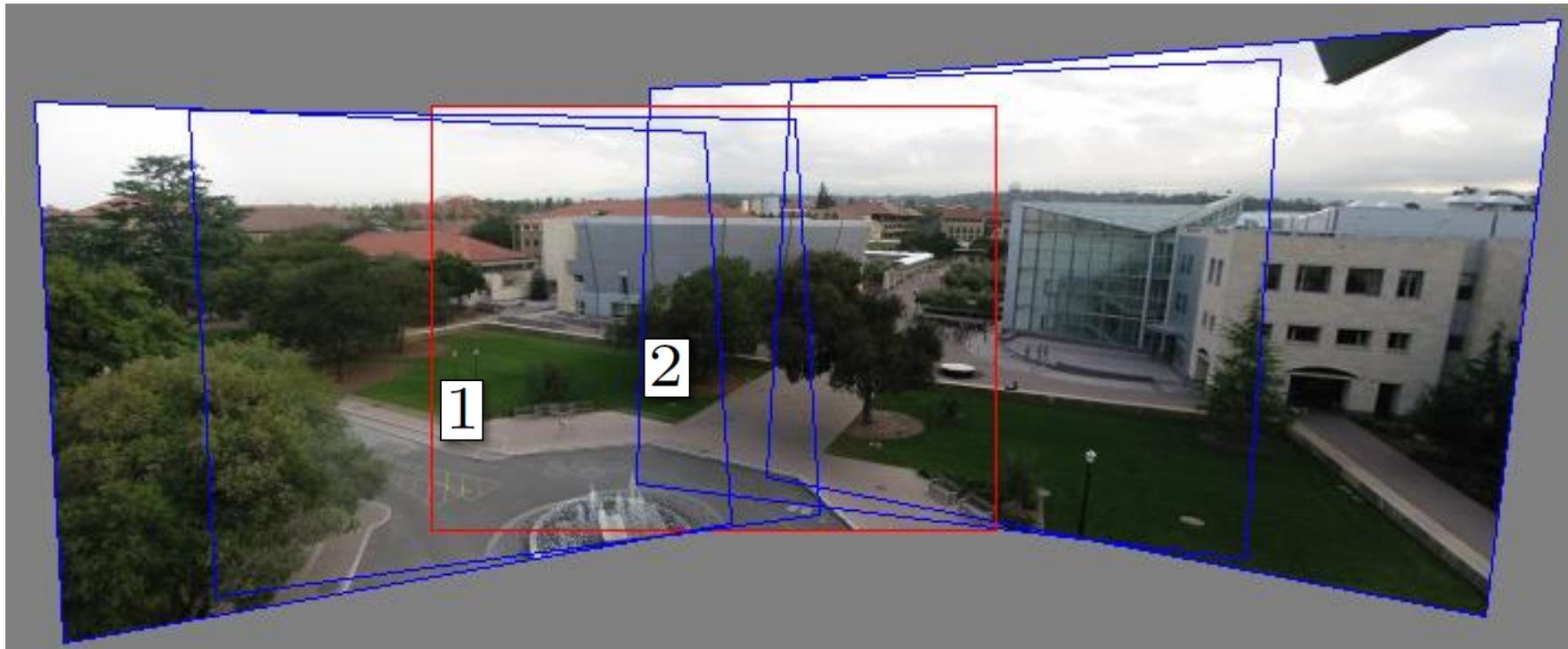
- This *epipolar geometry* of two views is described by a very special 3×3 matrix, called the *fundamental matrix* F

Fundamental matrix



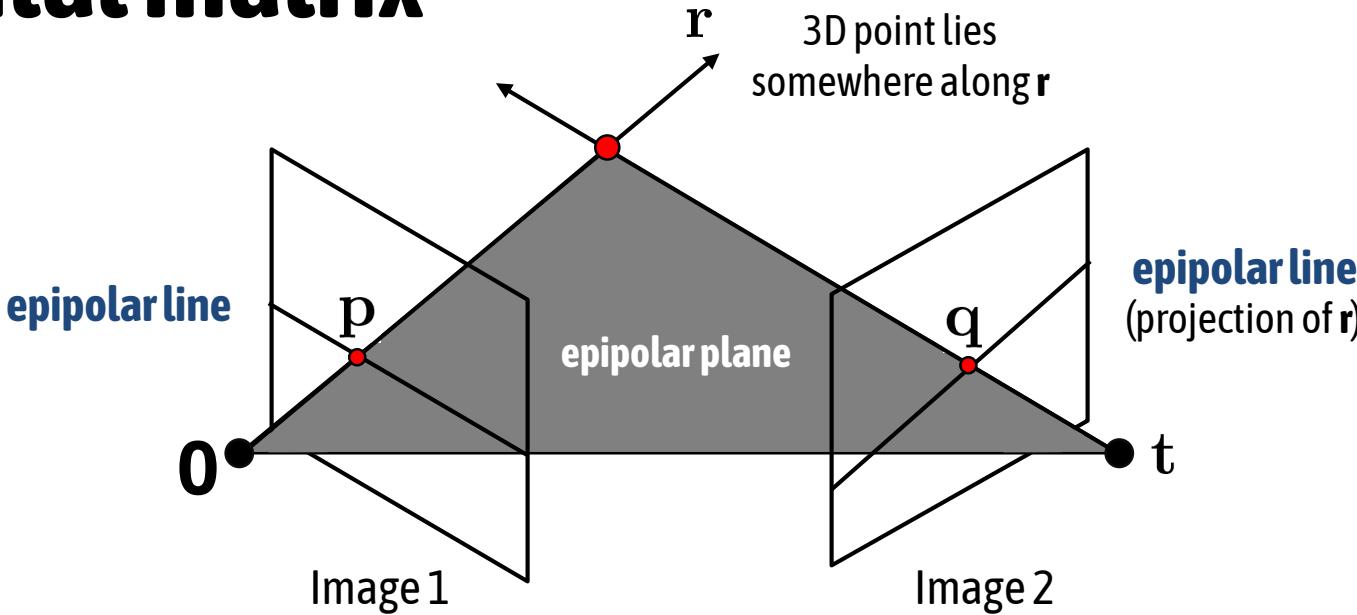
- Epipolar geometry, very special 3×3 fundamental matrix \mathbf{F}
- \mathbf{F} maps (homogeneous) *points* in image 1 to *lines* in image 2!

Relationship between F matrix and homography?



Images taken from the same center of projection? Use a homography!

Fundamental matrix



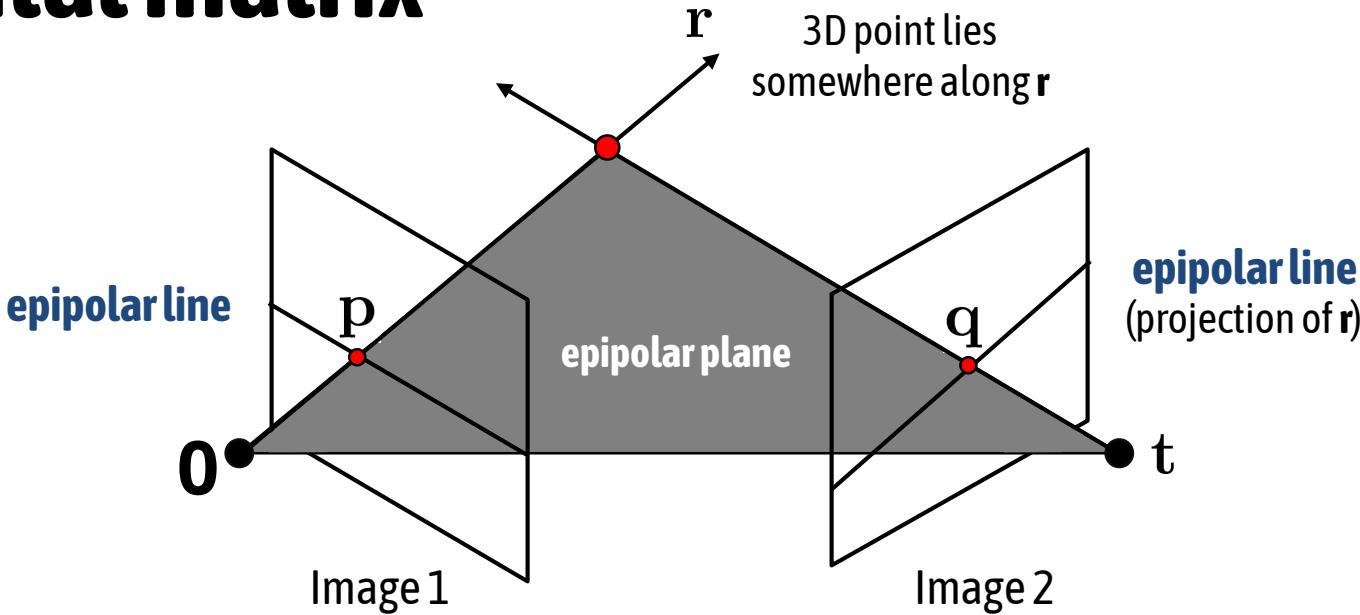
- Epipolar geometry, very special 3×3 fundamental matrix \mathbf{F}
- \mathbf{F} maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point p is: $\mathbf{F}^T \mathbf{p}$

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} q_x \\ q_y \\ 1 \end{bmatrix}$$

$$\mathbf{l}' = \mathbf{F}\mathbf{p} = \begin{bmatrix} l'_a \\ l'_b \\ l'_c \end{bmatrix}$$

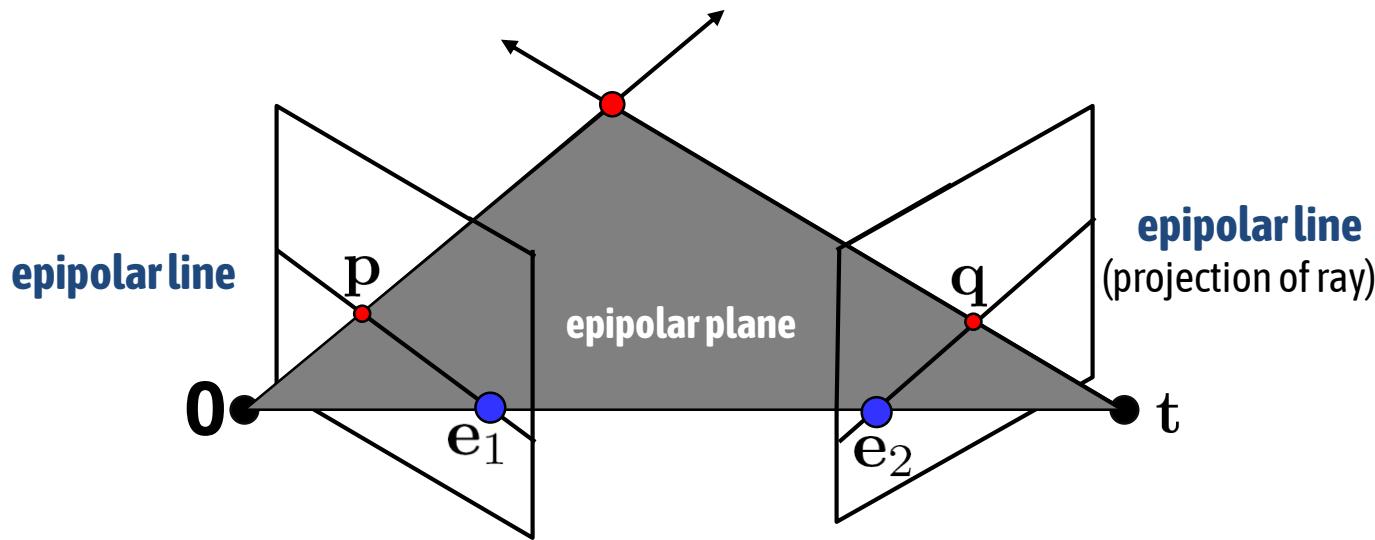
$$\mathbf{q}^T \mathbf{l}' = [q_x \quad q_y \quad 1] \begin{bmatrix} l'_a \\ l'_b \\ l'_c \end{bmatrix} = q_x l'_a + q_y l'_b + l'_c = 0$$

Fundamental matrix



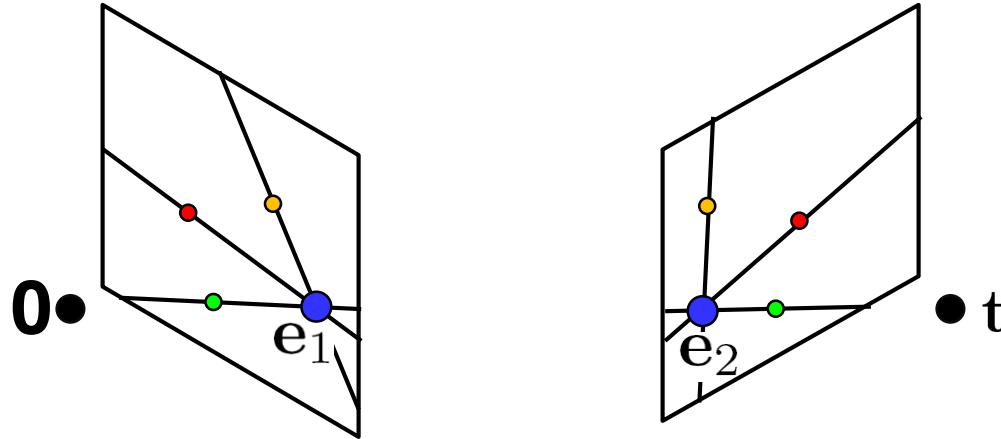
- Epipolar geometry, very special 3×3 fundamental matrix \mathbf{F}
- \mathbf{F} maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point p is: $\mathbf{F}^T \mathbf{F} \mathbf{p}$
- Epipolar constraint on corresponding points: $\mathbf{q}^T \mathbf{F} \mathbf{p} = 0$

Fundamental matrix



- Two Special points: e_1 and e_2 (the *epipoles*): projection of one camera into the other

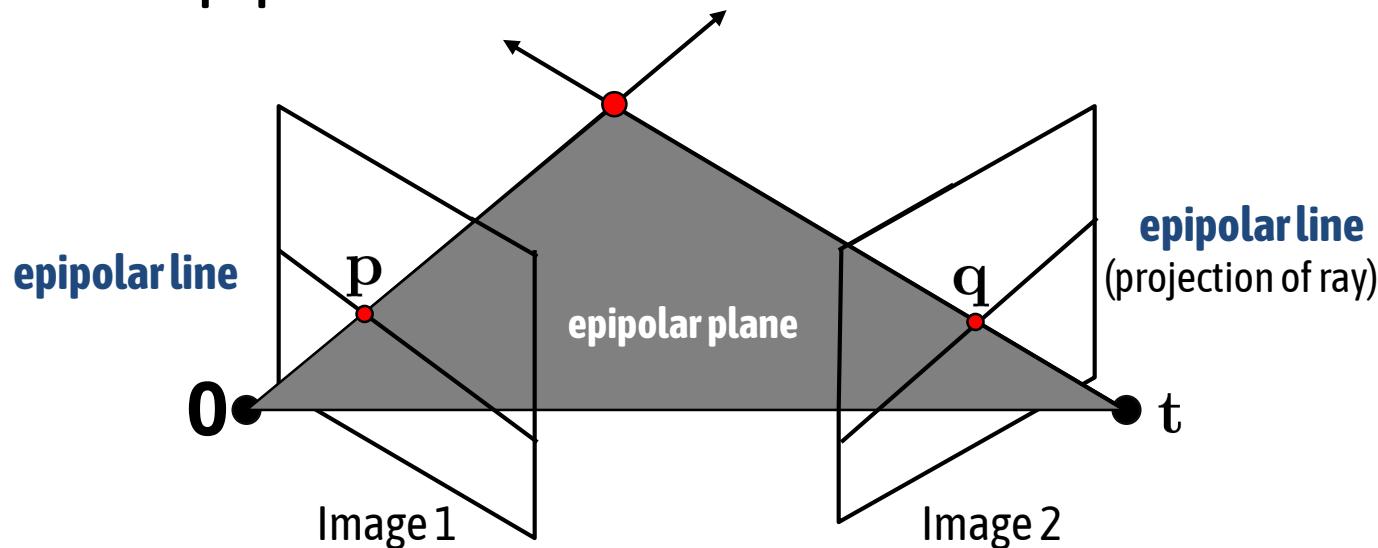
Fundamental matrix



- Two Special points: e_1 and e_2 (the *epipoles*): projection of one camera into the other
- All of the epipolar lines in an image pass through the epipole
- Epipoles may or may not be inside the image

Properties of the Fundamental Matrix

- Fp is the epipolar line associated with p
- $F^T q$ is the epipolar line associated with q



$$q^T F p = 0 \rightarrow (F^T q)^T p = 0$$

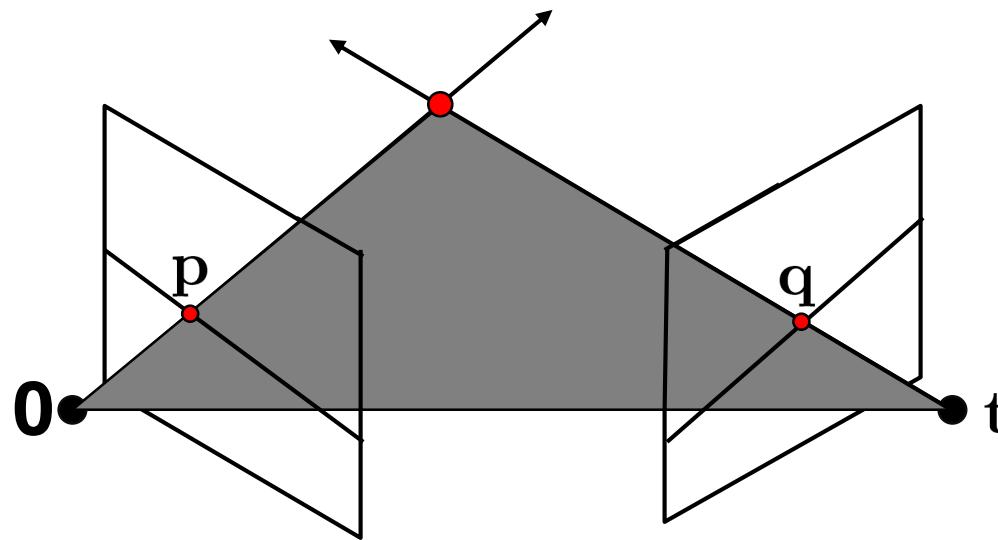
Example



Demo

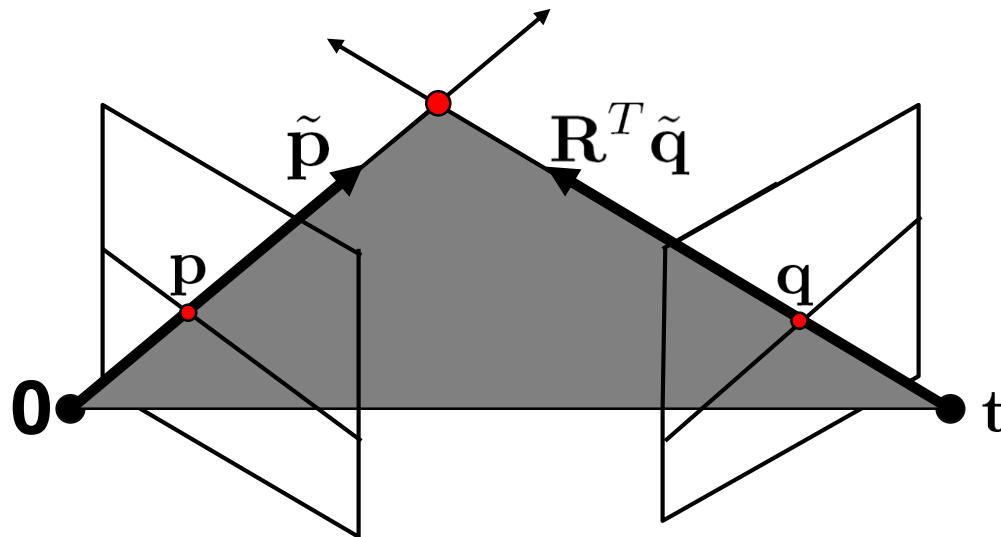
<https://www.cs.cornell.edu/courses/cs5670/2023sp/demos/FundamentalMatrix/?demo=demo1>

Fundamental matrix



- Why does \mathbf{F} exist?
- Let's derive it...

Fundamental matrix – calibrated case



\mathbf{K}_1 :intrinsic parameters of camera 1

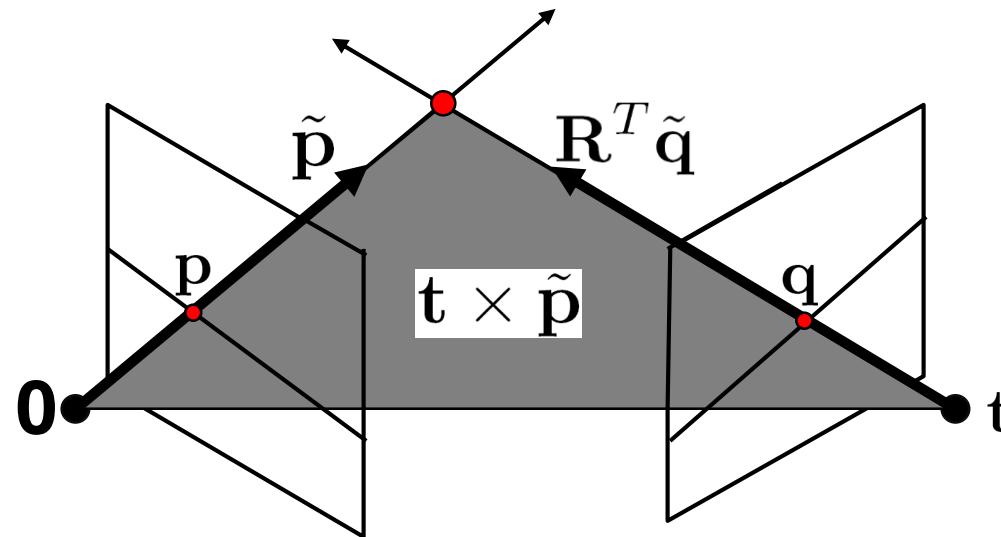
\mathbf{K}_2 :intrinsic parameters of camera 2

\mathbf{R} :rotation of image 2 w.r.t. camera 1

$\tilde{\mathbf{p}} = \mathbf{K}_1^{-1} \mathbf{p}$:ray through \mathbf{p} in camera 1's (and world) coordinate system

$\tilde{\mathbf{q}} = \mathbf{K}_2^{-1} \mathbf{q}$:ray through \mathbf{q} in camera 2's coordinate system

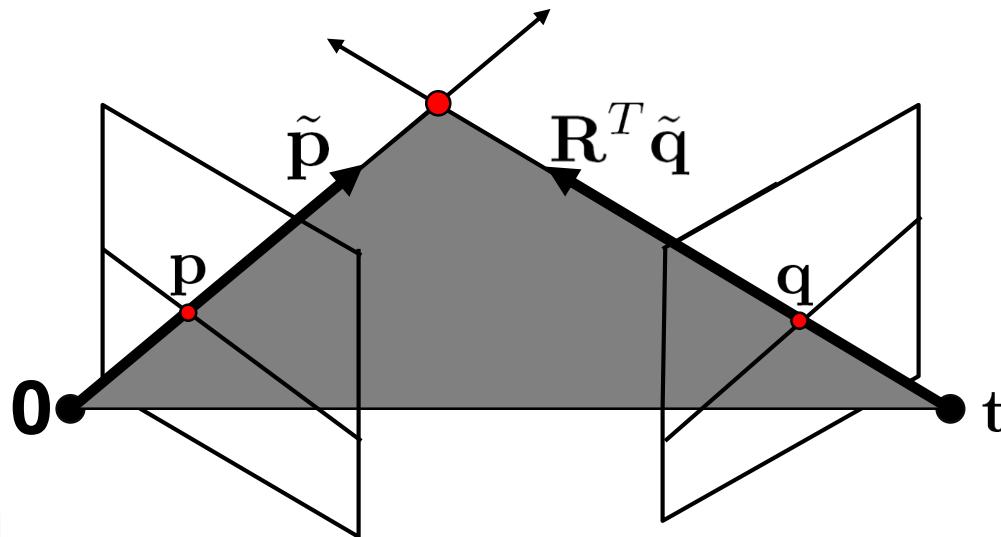
Fundamental matrix – calibrated case



- One more substitution:
 - Cross product with $\mathbf{t} = [t_x \ t_y \ t_z]$ (on left) can be represented as a 3×3 matrix

$$[\mathbf{t}]_{\times} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \quad \mathbf{t} \times \tilde{\mathbf{p}} = [\mathbf{t}]_{\times} \tilde{\mathbf{p}}$$

Fundamental matrix – calibrated case



$$\tilde{p} = K_1^{-1}p \text{ : ray through } p \text{ in camera 1's (and world) coordinate system}$$

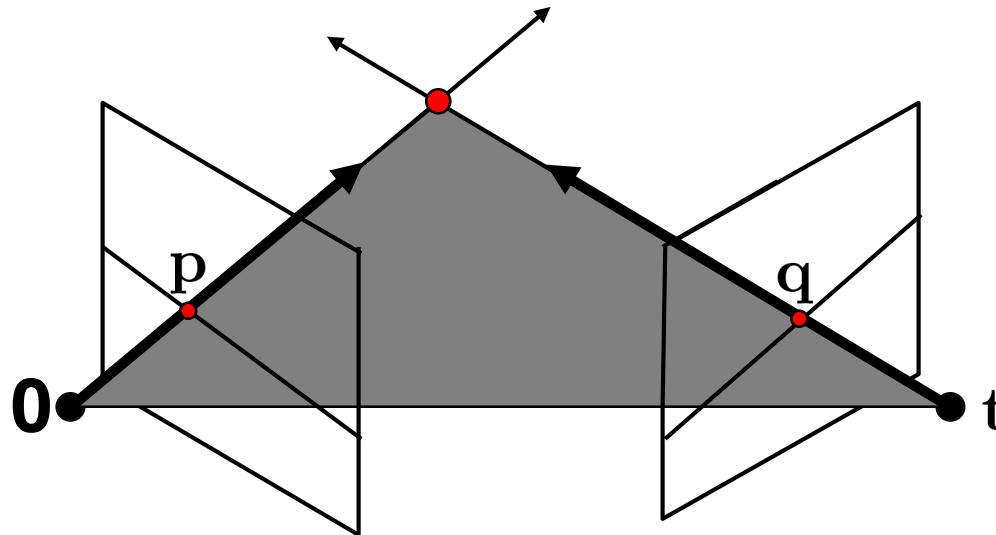
$$\tilde{q} = K_2^{-1}q \text{ : ray through } q \text{ in camera 2's coordinate system}$$

$$\tilde{q}^T R [t]_\times \tilde{p} = 0 \quad \tilde{q}^T E \tilde{p} = 0$$

$E \leftarrow$ the Essential matrix

$$q^T F p = 0$$

Fundamental matrix – uncalibrated case



\mathbf{K}_1 :intrinsic of camera 1

\mathbf{K}_2 :intrinsic of camera 2

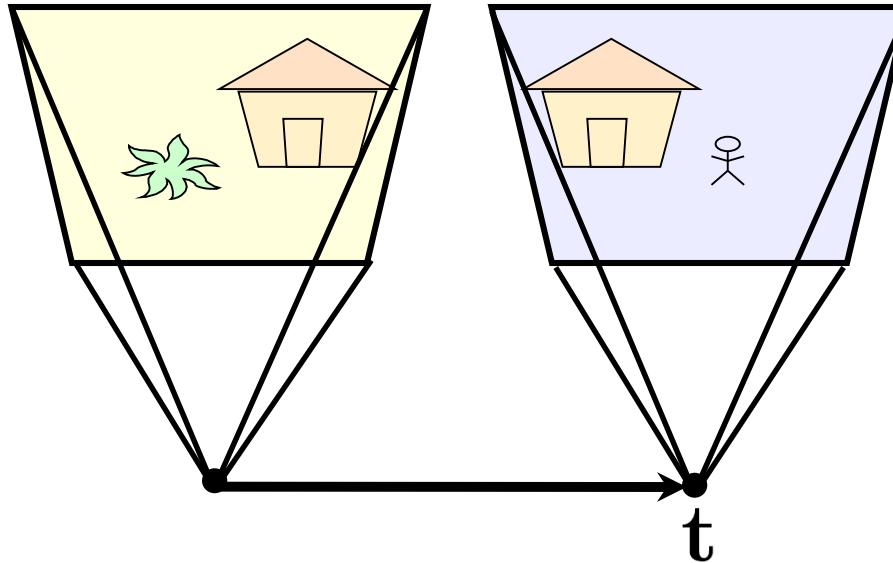
\mathbf{R} :rotation of image 2 w.r.t. camera 1

$$\mathbf{q}^T \underbrace{\mathbf{K}_2^{-T} \mathbf{R} [\mathbf{t}]_\times \mathbf{K}_1^{-1}}_{\mathbf{F}} \mathbf{p} = 0$$

$$\mathbf{q}^T \mathbf{F} \mathbf{p} = 0$$

$\mathbf{F} \leftarrow$ the Fundamental matrix

Rectified case



$$\mathbf{R} = \mathbf{I}_{3 \times 3}$$

$$\mathbf{t} = [1 \ 0 \ 0]^T$$

$$\mathbf{E} = \mathbf{R} [\mathbf{t}]_\times = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Working out the math

- For a point $[a, b, 1]^T$ in image 1:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ b \end{bmatrix}$$

- Its corresponding point $[x, y, 1]^T$ in image 2 must satisfy:

$$[x \ y \ 1] \cdot \begin{bmatrix} 0 \\ -1 \\ b \end{bmatrix} = 0 \implies y = b$$



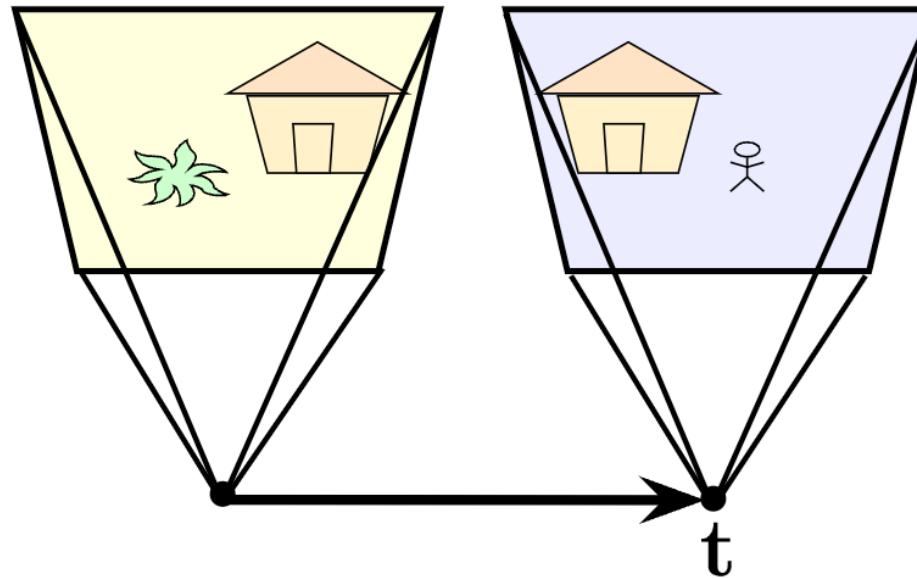
Original stereo pair



After rectification

$$y = b$$

Rectified case



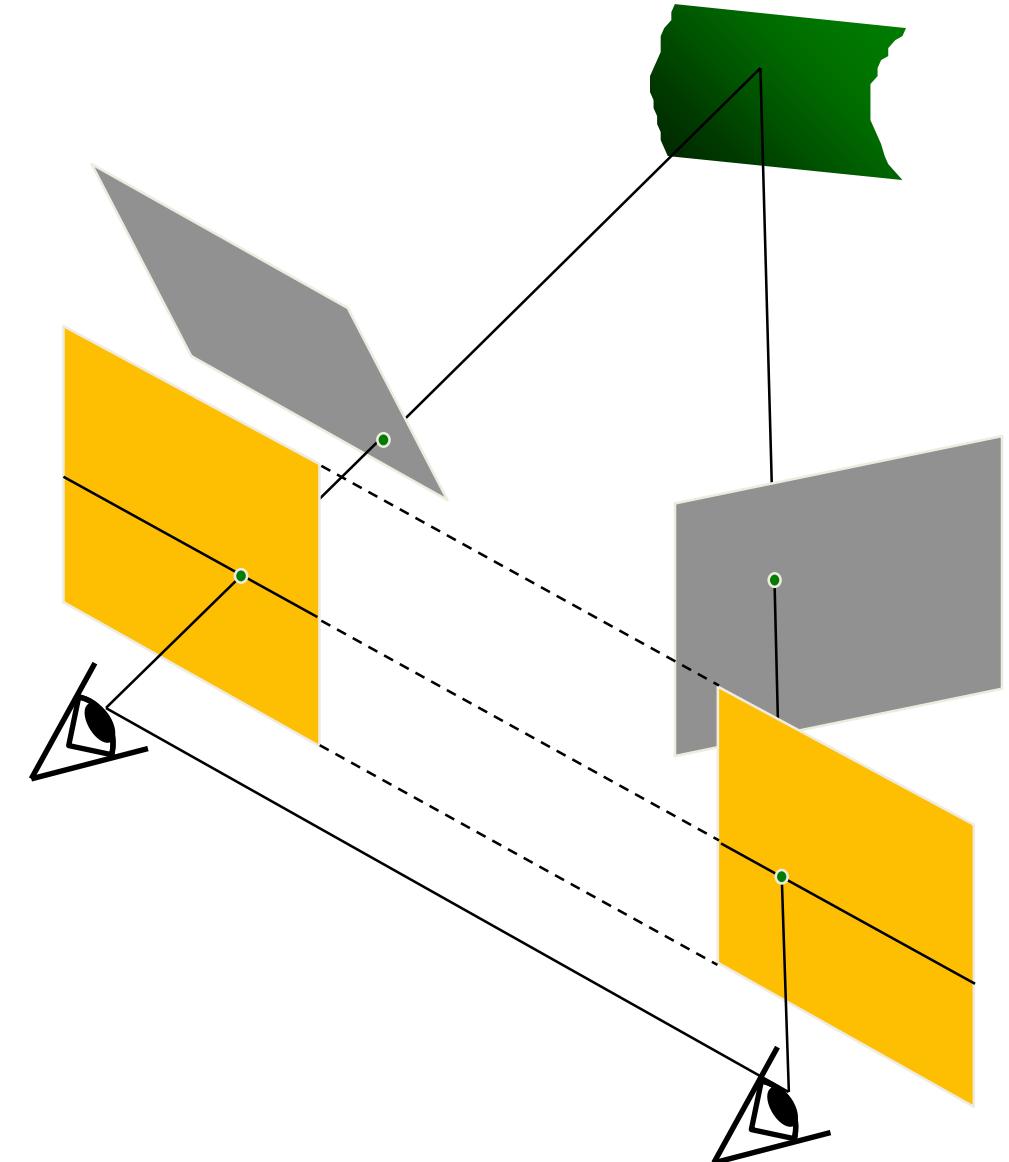
$$\mathbf{R} = \mathbf{I}_{3 \times 3}$$

$$\mathbf{t} = [1 \ 0 \ 0]^T$$

$$\mathbf{E} = \mathbf{R} [\mathbf{t}]_{\times} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Stereo image rectification

- Reproject image planes onto a common plane
 - Plane parallel to the line between optical centers
- Pixel motion is horizontal after this transformation
- Two homographies, one for each input image
 - C. Loop and Z. Zhang. [Computing Rectifying Homographies for Stereo Vision](#). CVPR 1999.



Fundamental matrix song

<http://danielwedge.com/fmatrix/>

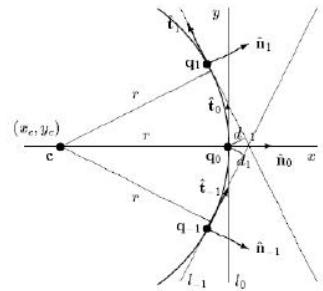
Questions?

Sparse correspondence

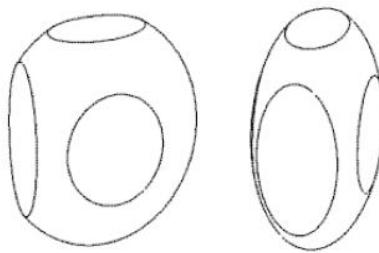
- Early stereo matching algorithms were **feature-based**, i.e., they first extracted a **set of potentially matchable image locations**, using either **interest operators** or **edge detectors**, and then **searched for corresponding locations** in other images using a patch-based metric
- More recent work in this area has focused on first **extracting highly reliable features** and then using these as **seeds to grow additional matches**

3D curves and profiles

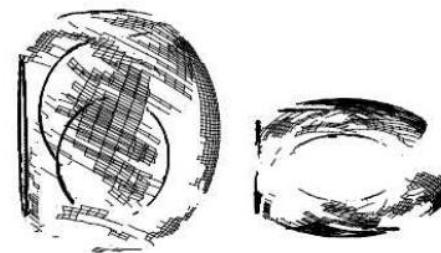
- Example of sparse correspondence is the matching of profile curves (or occluding contours), which occur at the boundaries of objects



(a)



(b)



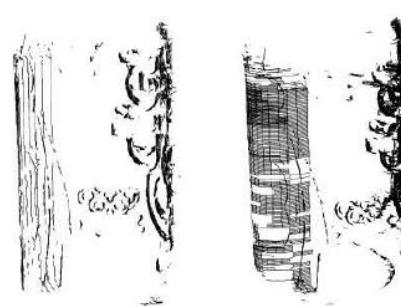
(c)



(d)



(e)



(f)



(g)

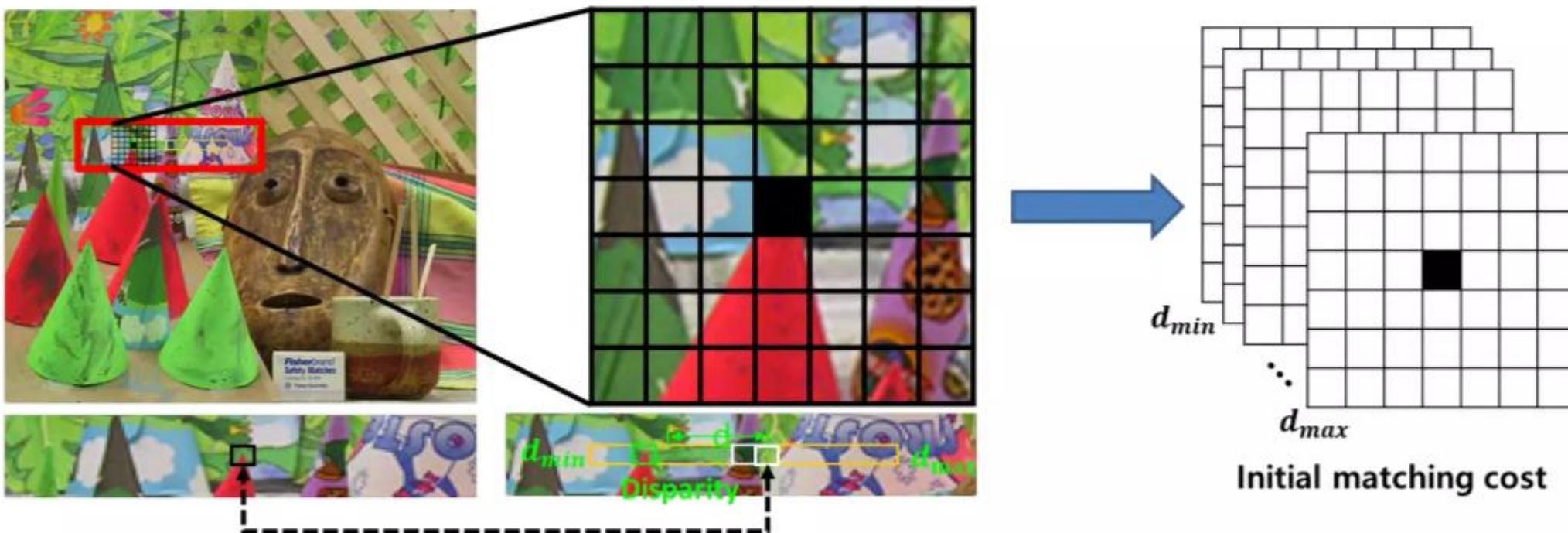
Dense correspondence

- Most stereo matching algorithms today focus on dense correspondence, as this is required for applications such as image-based rendering or modeling.
- 1. matching cost computation;
- 2. cost (support) aggregation;
- 3. disparity computation and optimization; and
- 4. disparity refinement

Traditional Stereo Matching

1. Matching cost computation

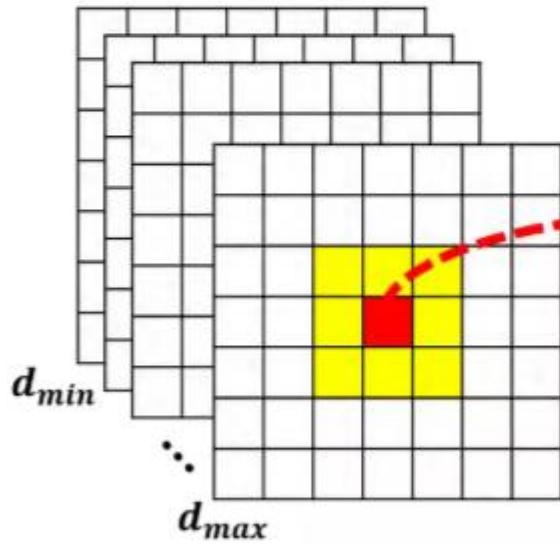
- Compare with neighborhoods around pixels on the epipolar line for the best match
- Consider its neighborhood defined by a square window



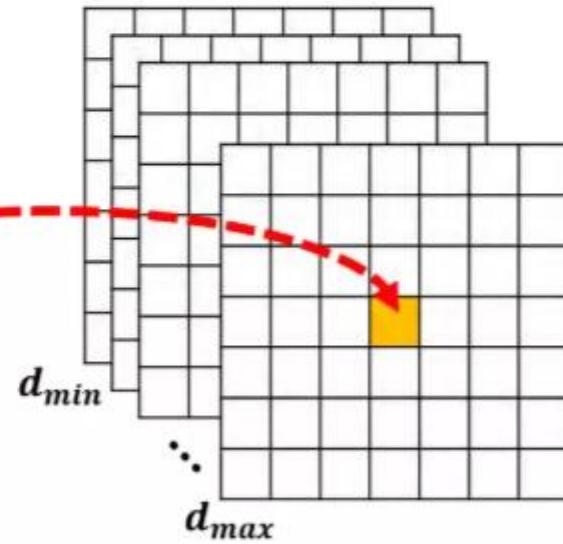
Traditional Stereo Matching

2. Cost aggregation

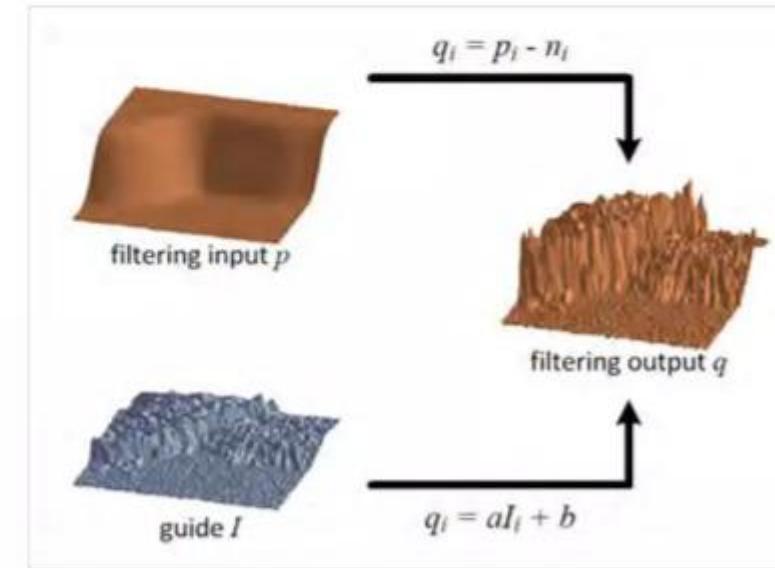
- Color differences and a variation exist in the depth discontinuous
- The variation in the disparity value is small between adjacent pixels



Initial matching cost



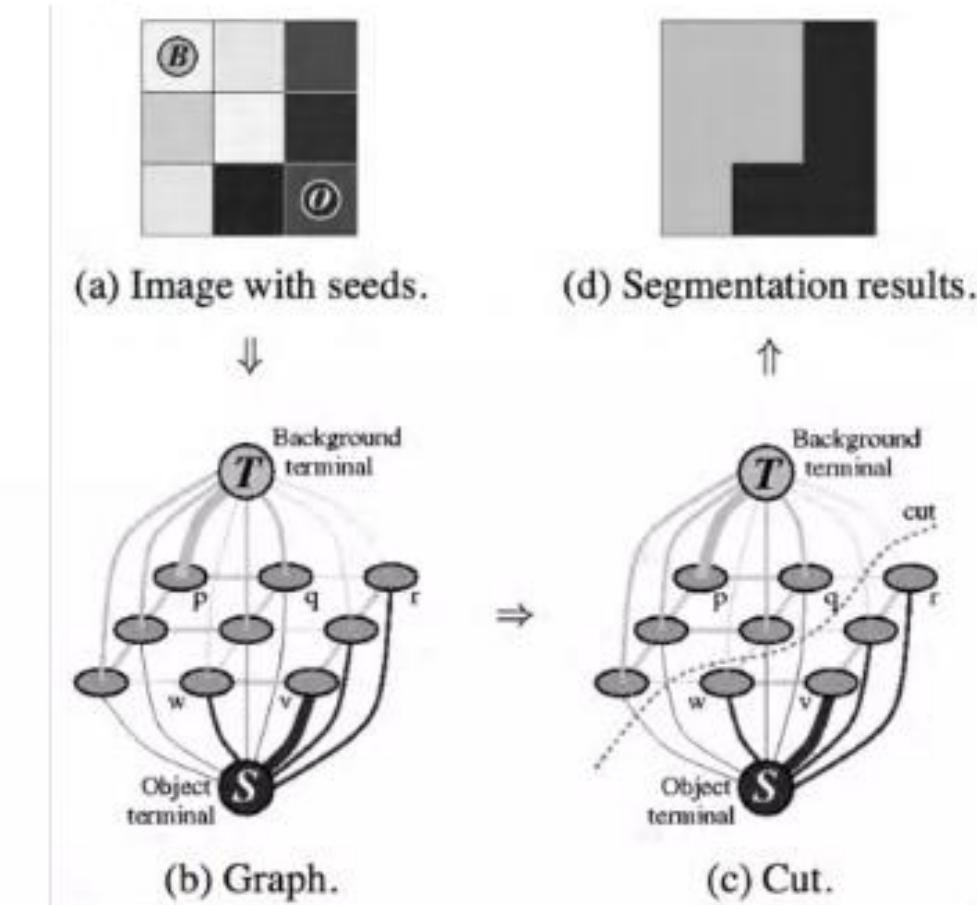
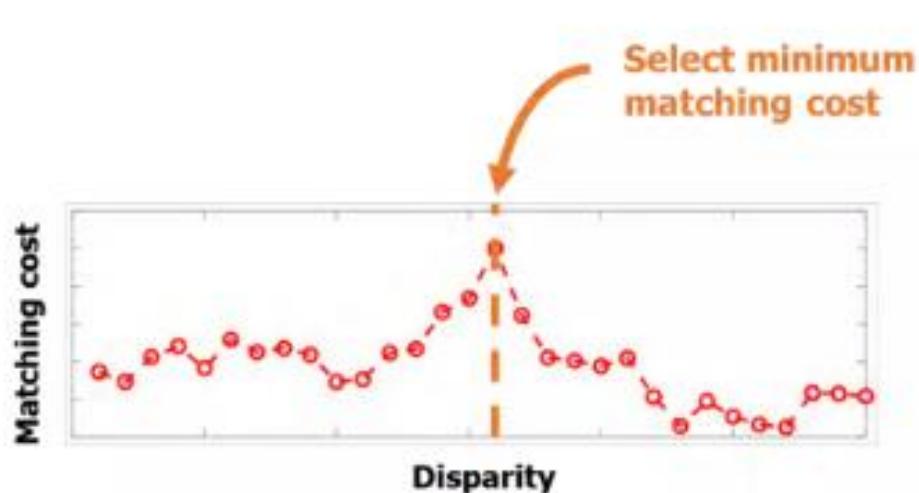
Aggregated matching cost



Ex) Guided filtering

Traditional Stereo Matching

- 3. Disparity computation/optimization
 - Winner-takes-all (WTA)
 - Dynamic programming
 - Graph-cut [1]

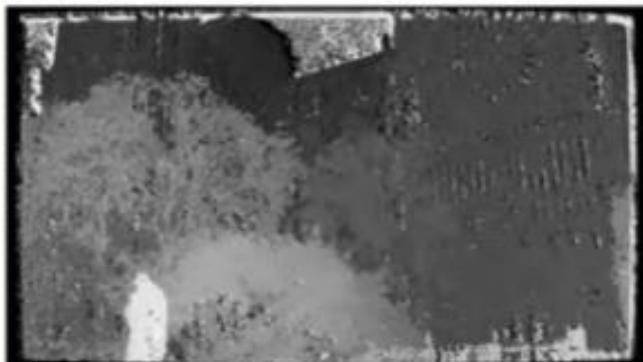


[1] Boykov, Yuri, Olga Veksler, and Ramin Zabih. "Fast approximate energy minimization via graph cuts." *IEEE Transactions on pattern analysis and machine intelligence* 23.11 (2001): 1222-1244.

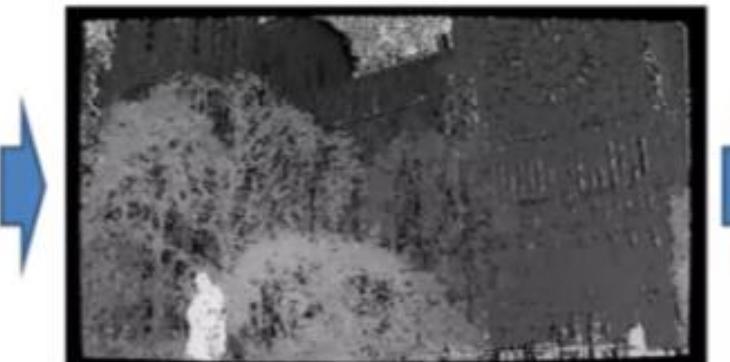
Traditional Stereo Matching

4. Disparity Refinement

- Left-Right Consistency Check
- Median filtering



Winner-takes-all (inverse) depth map



Confidence based outlier removal



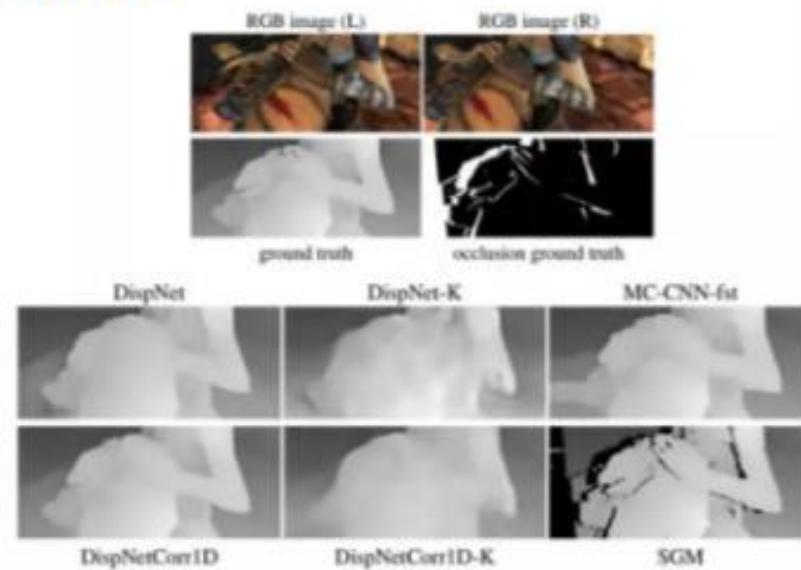
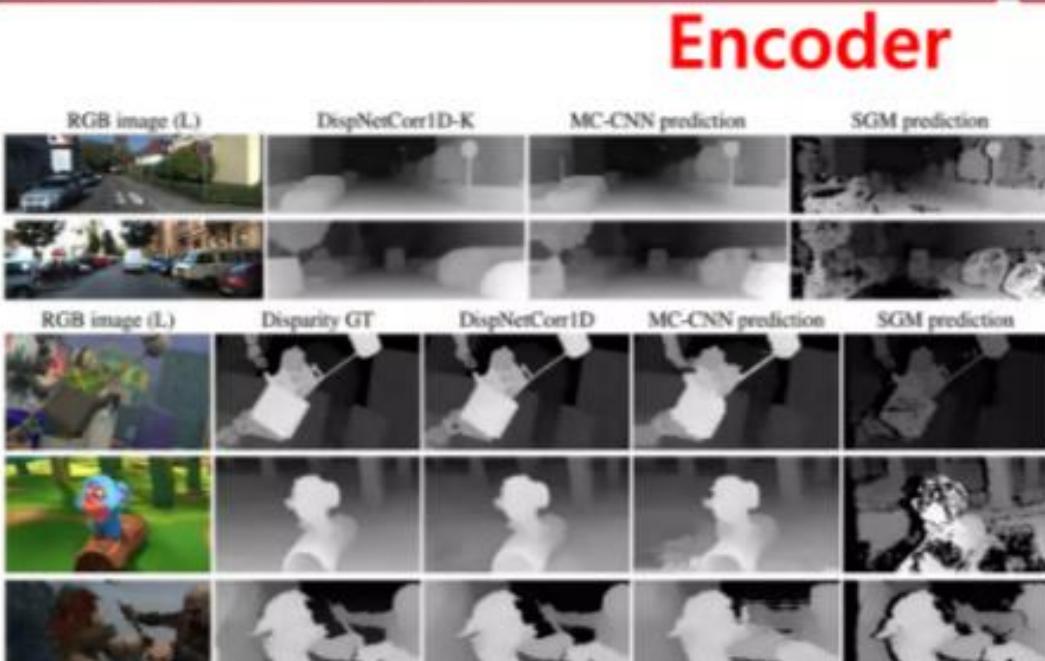
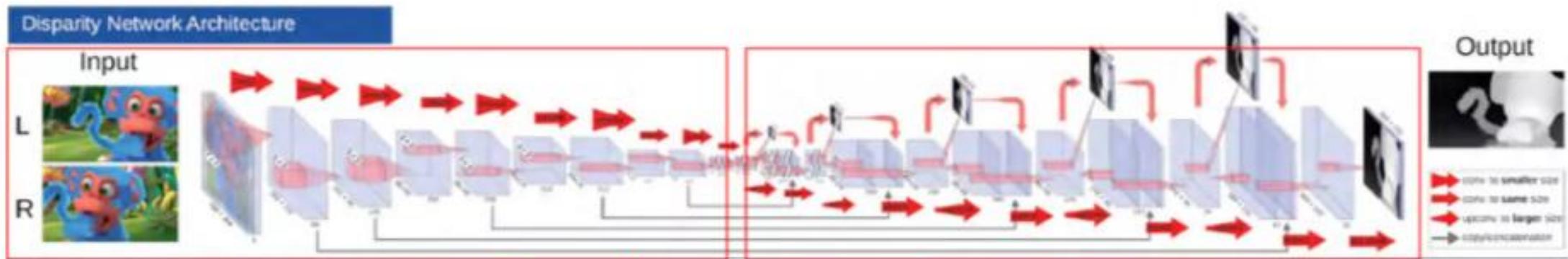
Depth refinement [1]

[1] Ha, Hyowon, et al. "High-quality depth from uncalibrated small motion clip." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

Deep Neural Network

2D and 3D Models

DispNet (FlowNet)

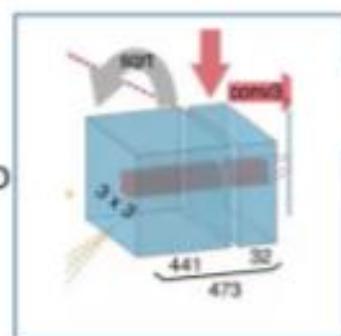


DispNet

End-to-end disparity estimation network (No need optimization)

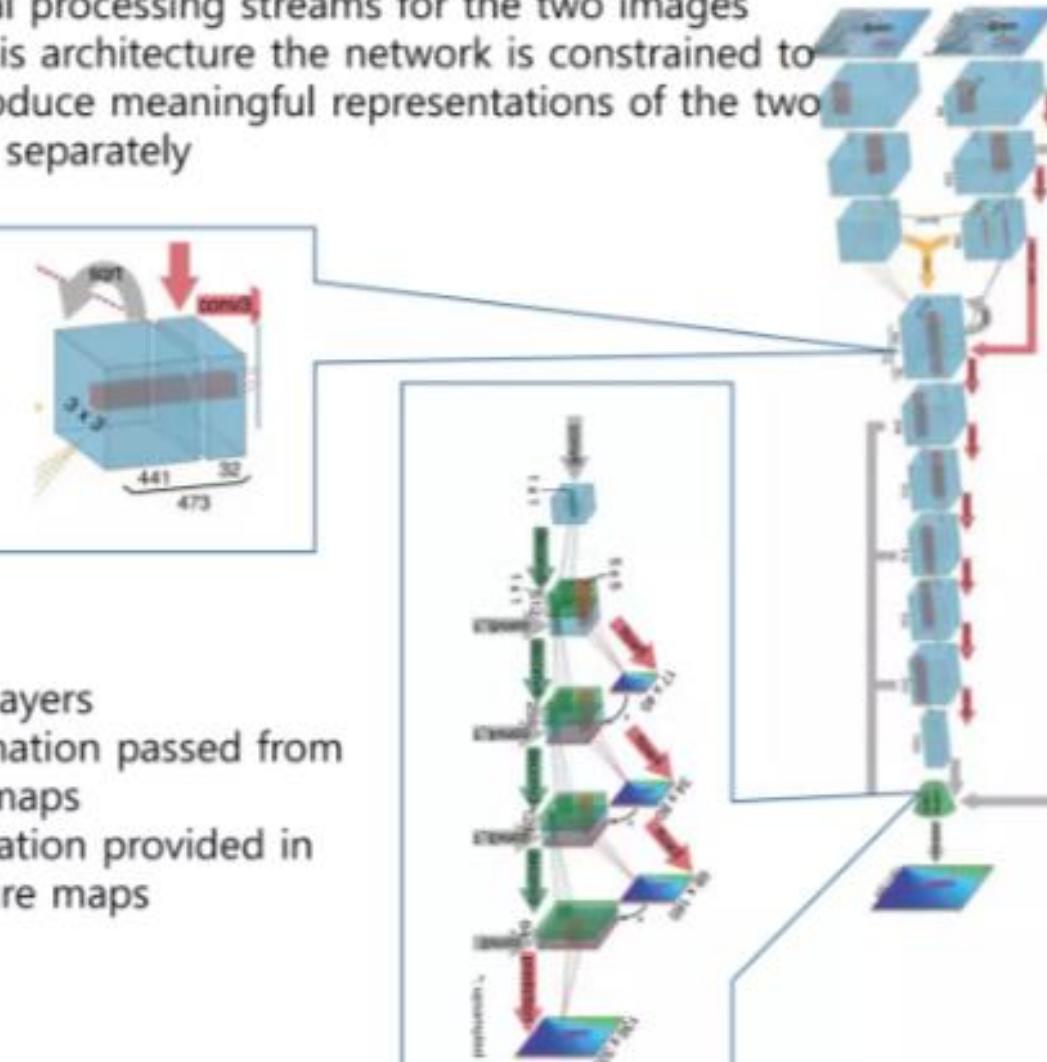
Convolution layer

- identical processing streams for the two images
- With this architecture the network is constrained to first produce meaningful representations of the two images separately



Correlation layer

- Multiplicative patch comparisons between two feature maps
- No trainable weights



Upconvolutional layers

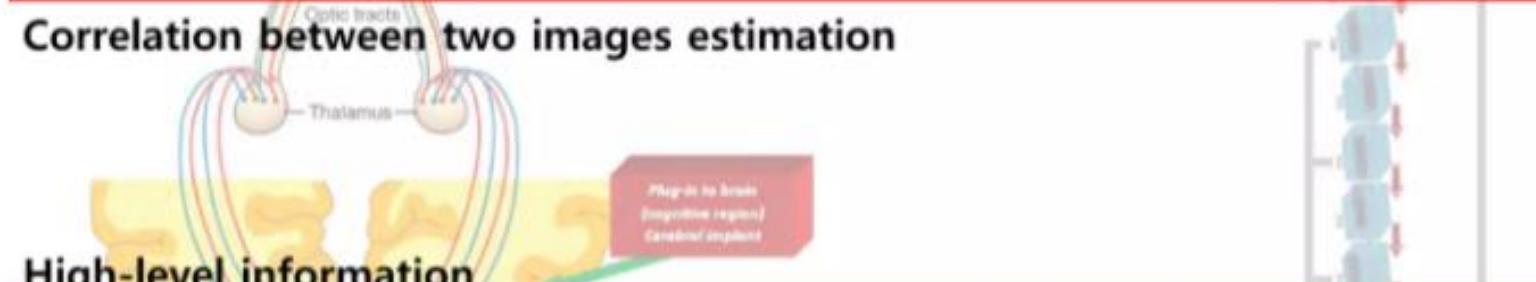
- high-level information passed from coarser feature maps
- fine local information provided in lower layer feature maps

Human Visual System and DispNet

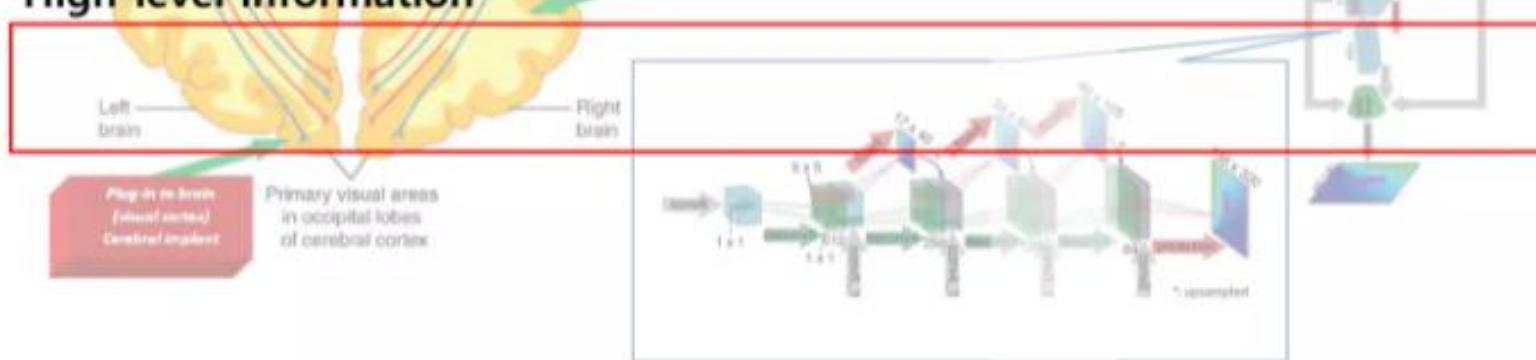
Encoding images



Correlation between two images estimation



High-level information

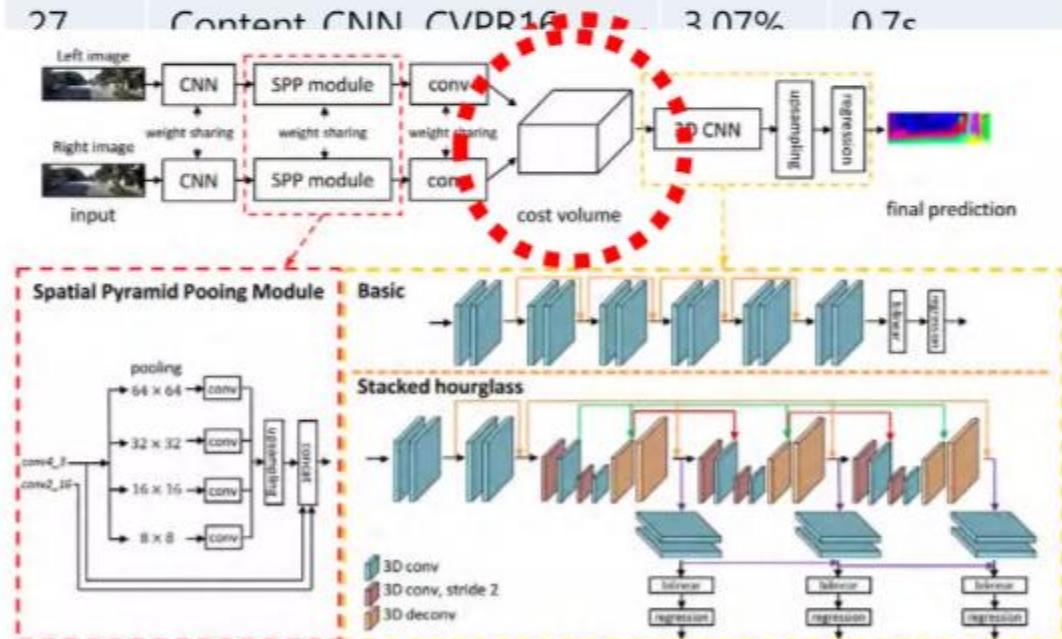


Is DispNet the best??

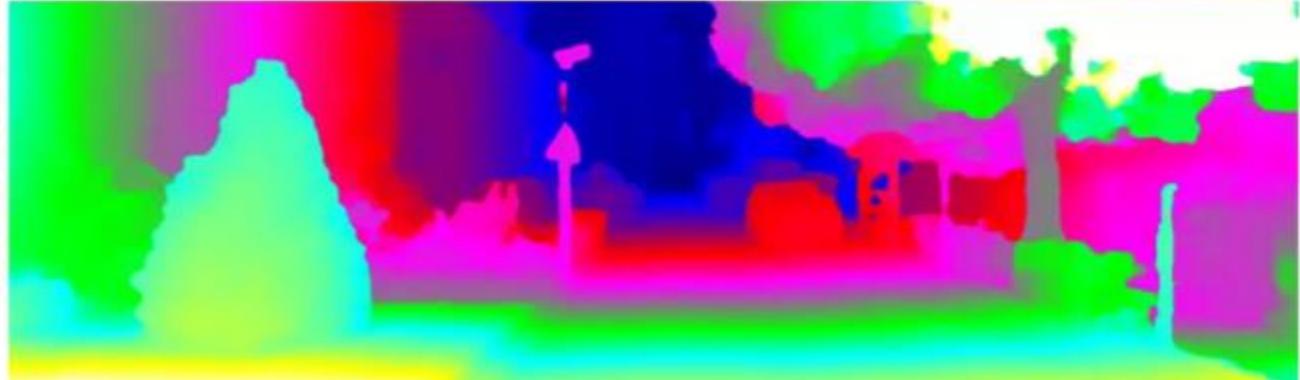
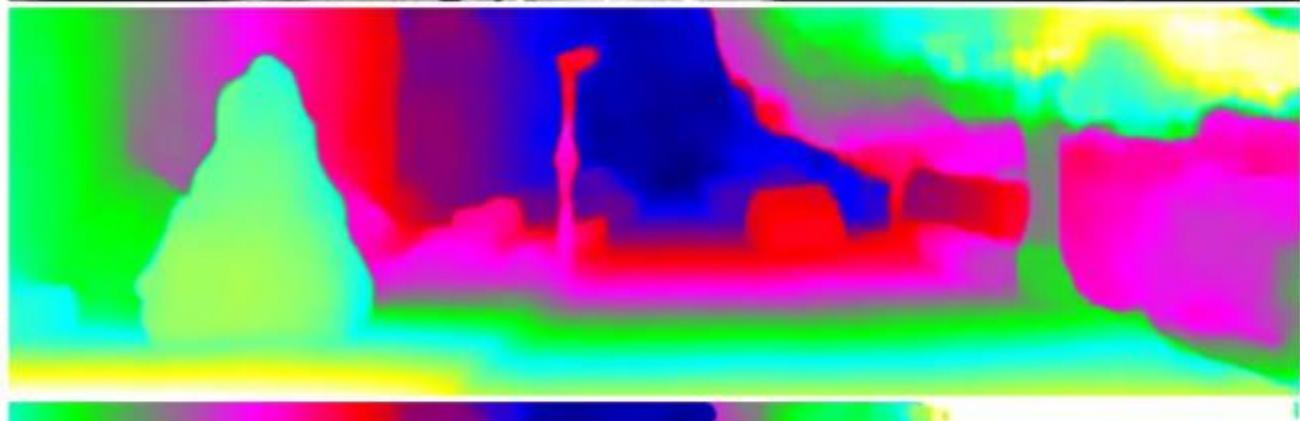
KITTI stereo evaluation 2012

2018/04/23

Rank	Method	Out-Noc	Runtime
1	PSMNet, CVPR18	1.49%	0.41s
2	iResNet-i2, CVPR18	1.71%	0.12s
15	MC-CNN-arct, JMLR16	2.43%	67s
27	Content CNN, CVPR18	2.07%	0.7s



PSMNet, CVPR 18



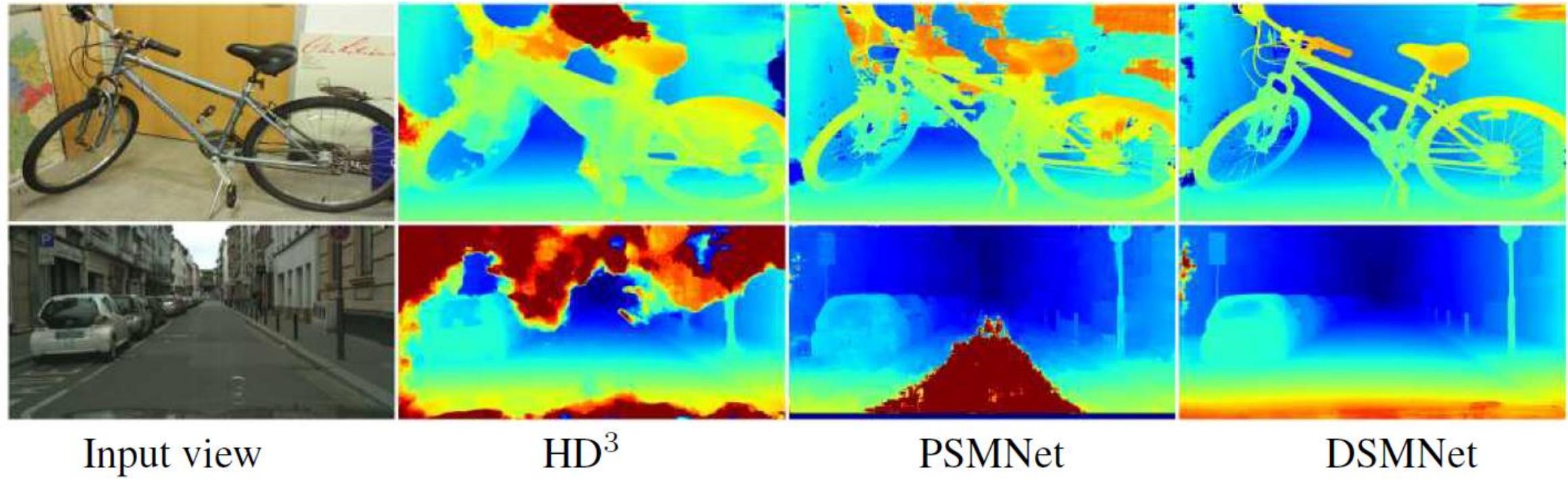
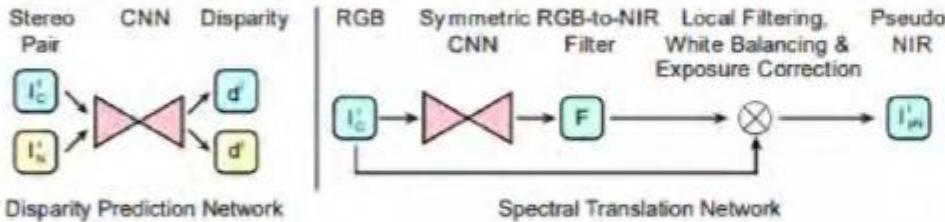


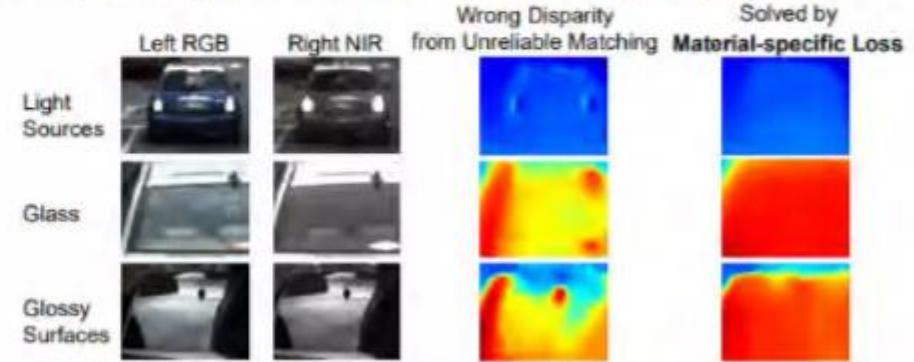
Figure 12.18 *Disparity maps computed by three different DNN stereo matchers trained on synthetic data and applied to real-world image pairs (Zhang, Qi et al. 2020) © 2020 Springer.*

Deep Material Stereo [CVPR'18]

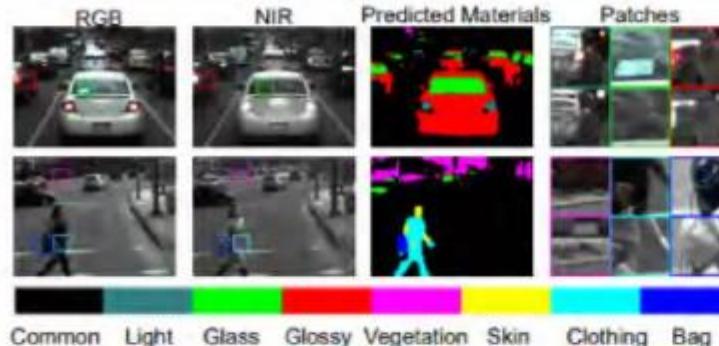
Simultaneous Disparity Prediction & Spectral Translation



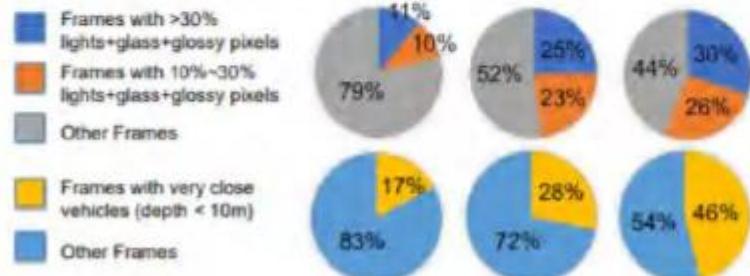
Incorporating Material Awareness into Disparity Prediction



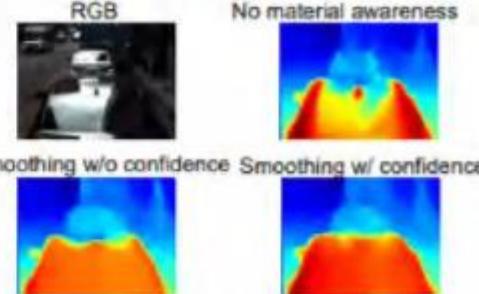
Materials with Large Appearance Variation



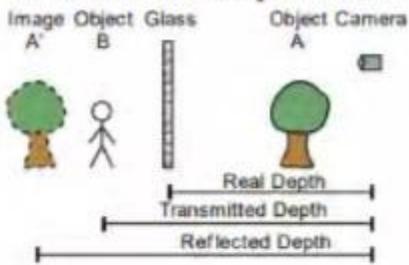
Material and Vehicle Stats



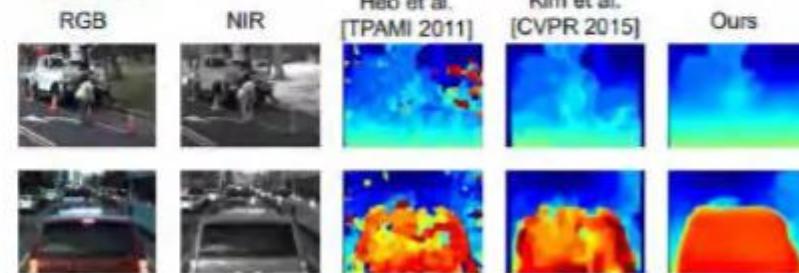
Confidence Weighted Smoothing



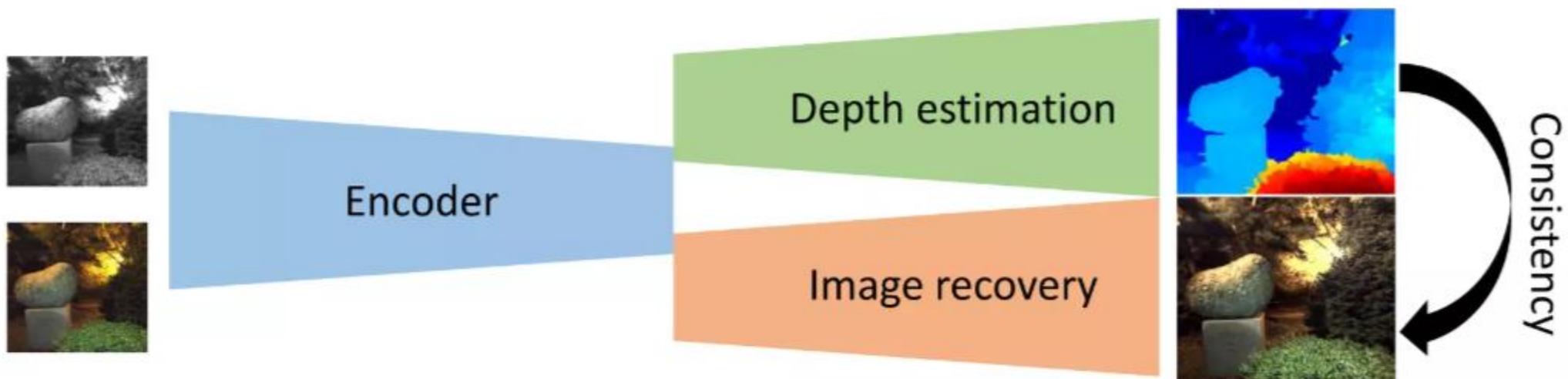
"Close Scene Prior" For Glass and Glossy Surfaces



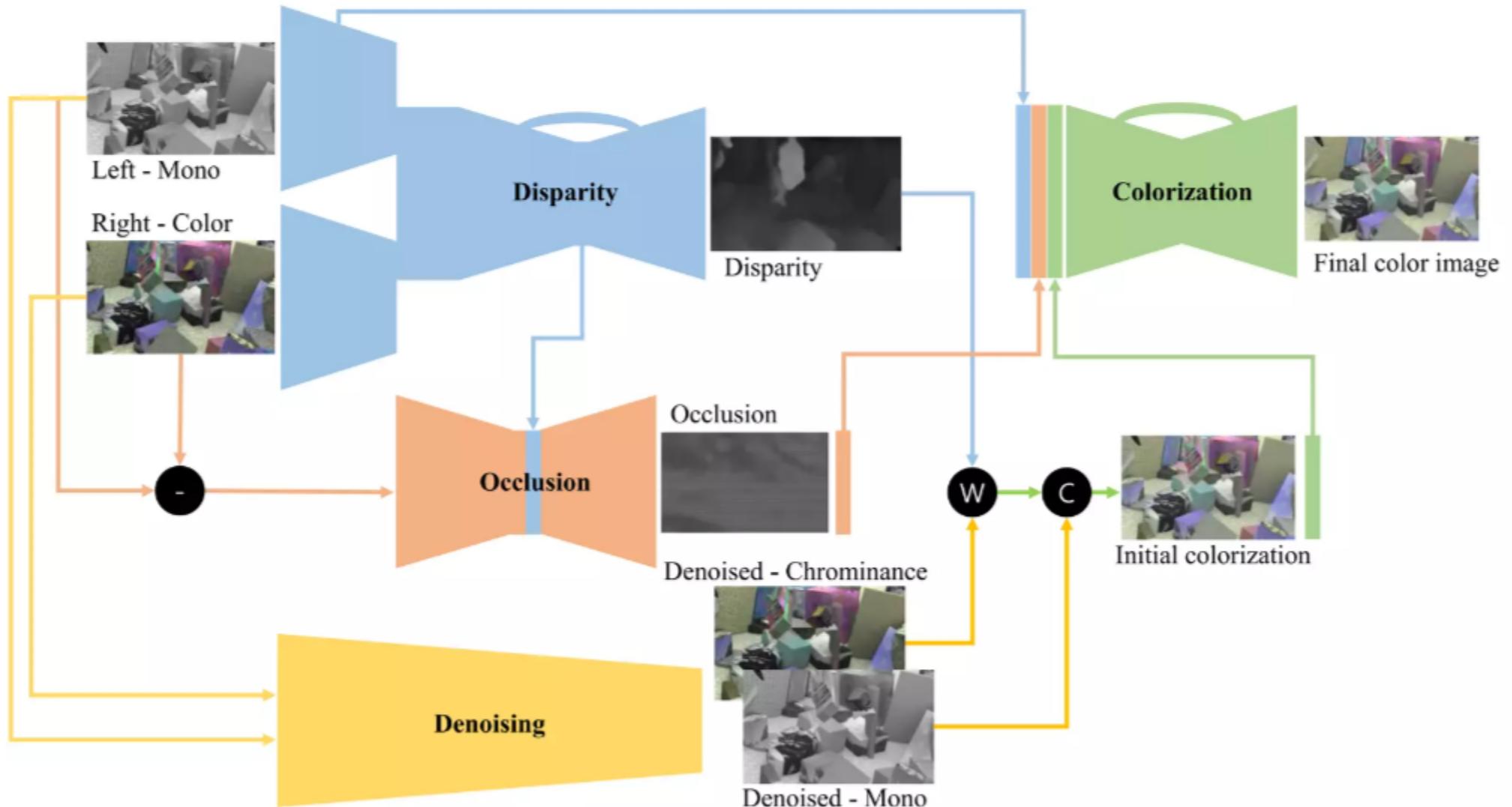
Results



CNN version of RGB-W Stereo

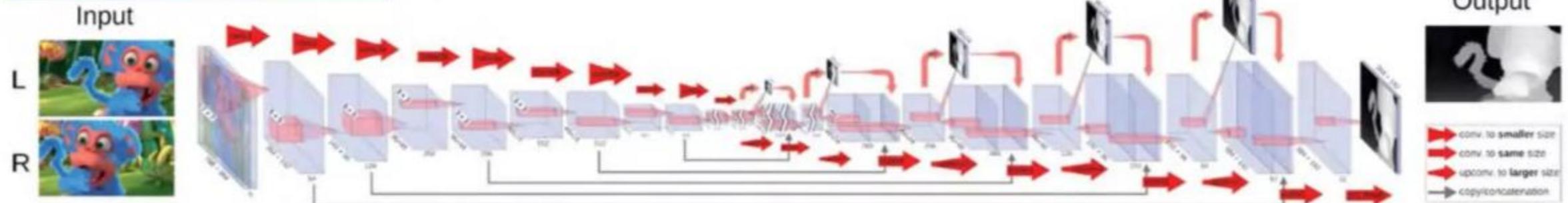


CNN version of RGB-W Stereo

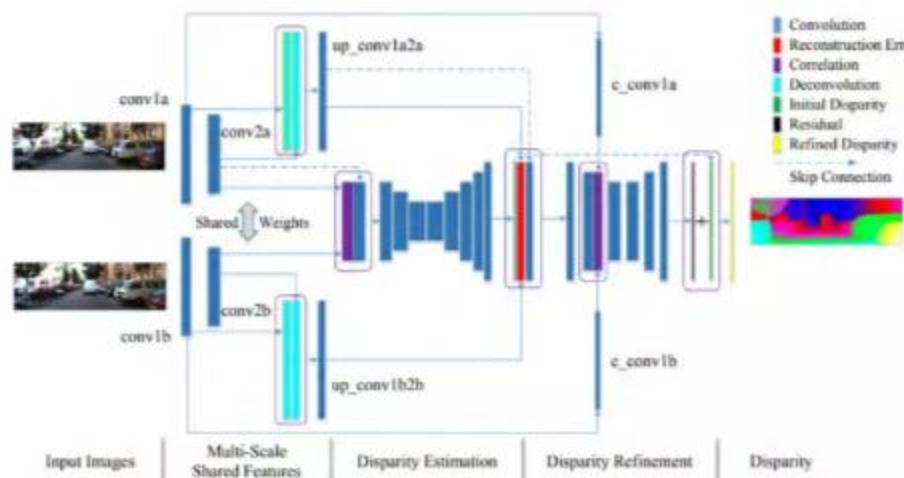


Convolutional Neural Network

Disparity Network Architecture

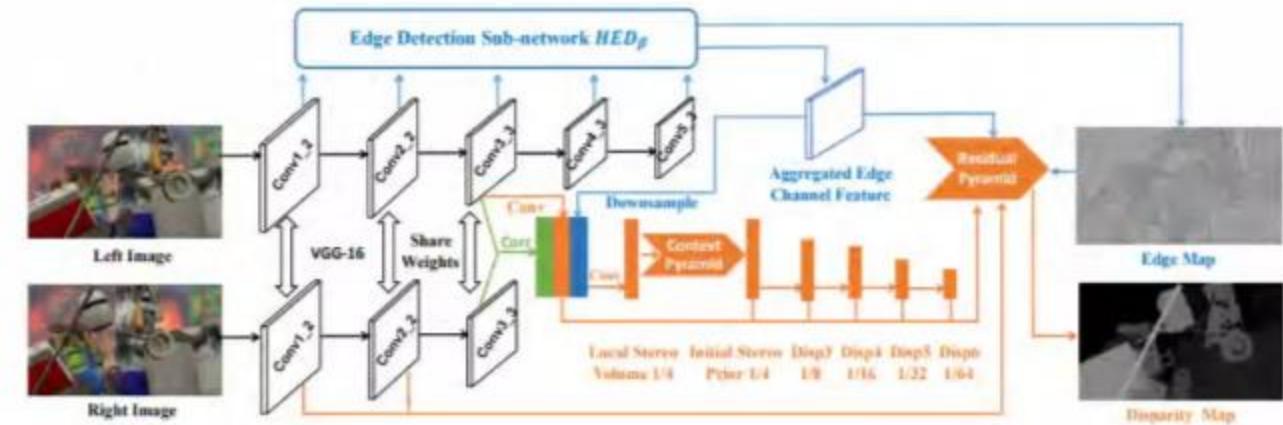


DispNet, CVPR 16



PSMNet, CVPR 18

EdgeStereo, ArXiv



EPINET [CVPR'18]

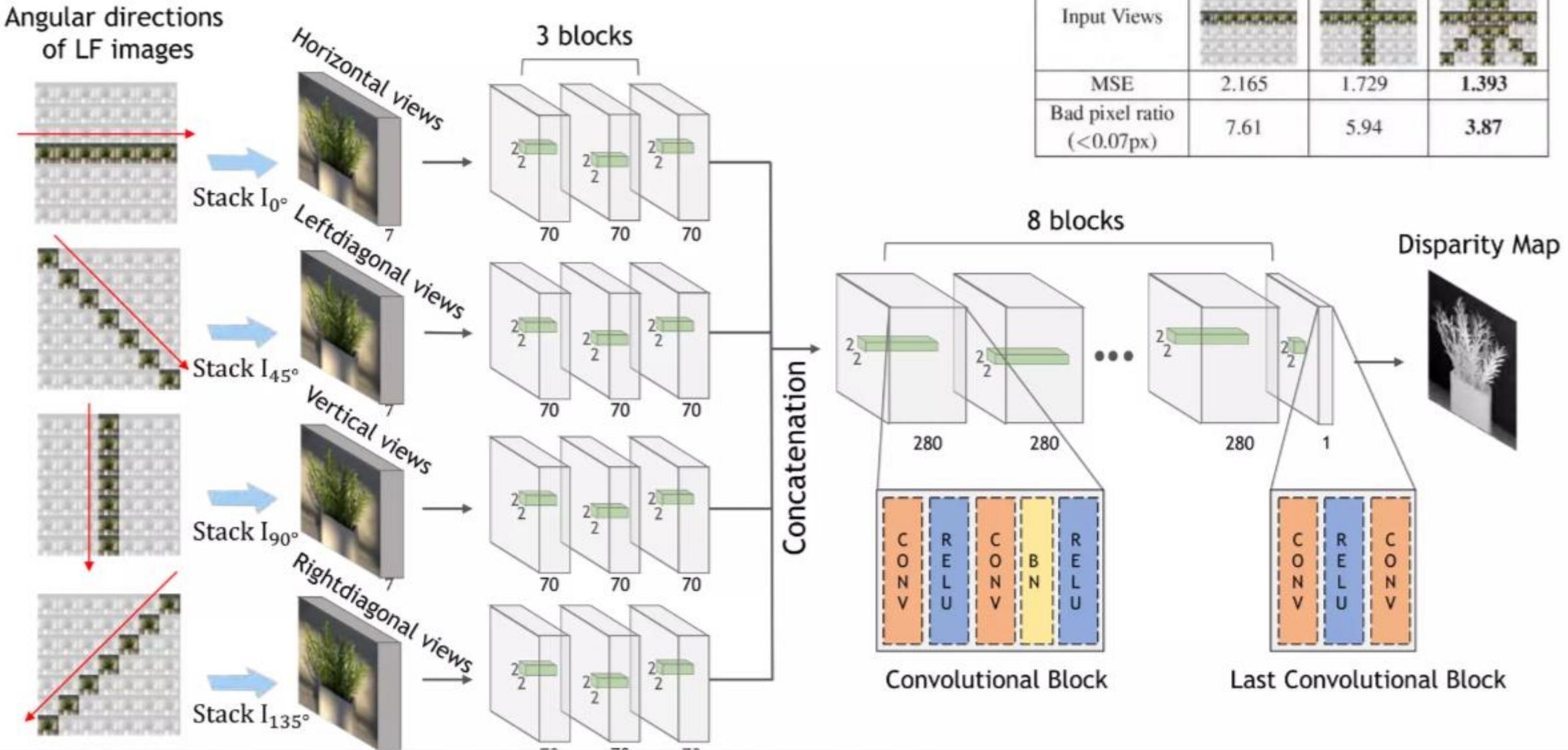


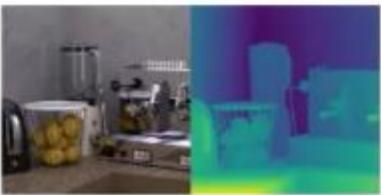
Table 1. The effect of the number of viewpoints on performance.

	1-stream	2-streams	4-streams
Input Views			
MSE	2.165	1.729	1.393
Bad pixel ratio (<0.07px)	7.61	5.94	3.87

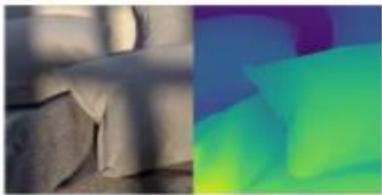
Lack of Data



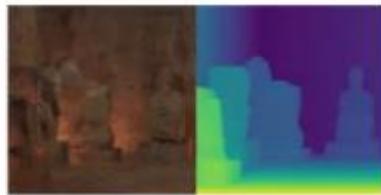
Antinous, Range: [-3.3, 2.8]



Kitchen, Range: [-1.6, 1.8]



Pillows, Range: [-1.7, 1.8]



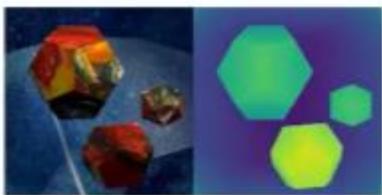
Tomb, Range: [-1.5, 1.9]



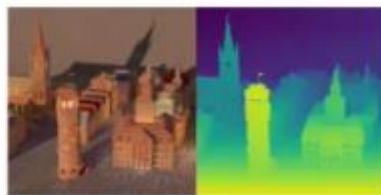
Boardgames, Range: [-1.8, 1.6]



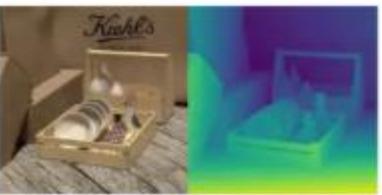
Medieval2, Range: [-1.7, 2.0]



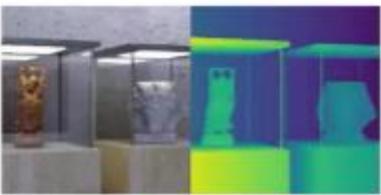
Platonic, Range: [-1.7, 1.5]



Tower, Range: [-3.6, 3.5]



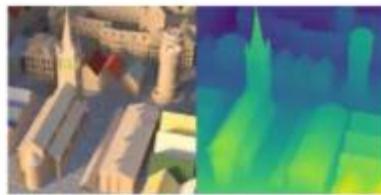
Dishes, Range: [-3.1, 3.5]



Museum, Range: [-1.5, 1.3]



Rosemary, Range: [-1.8, 1.8]



Town, Range: [-1.6, 1.6]



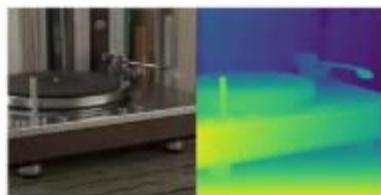
Greek, Range: [-3.5, 3.1]



Pens, Range: [-1.7, 2.0]



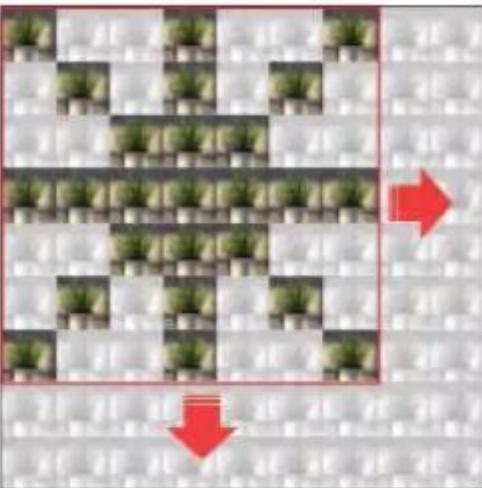
Table, Range: [-2.0, 1.6]



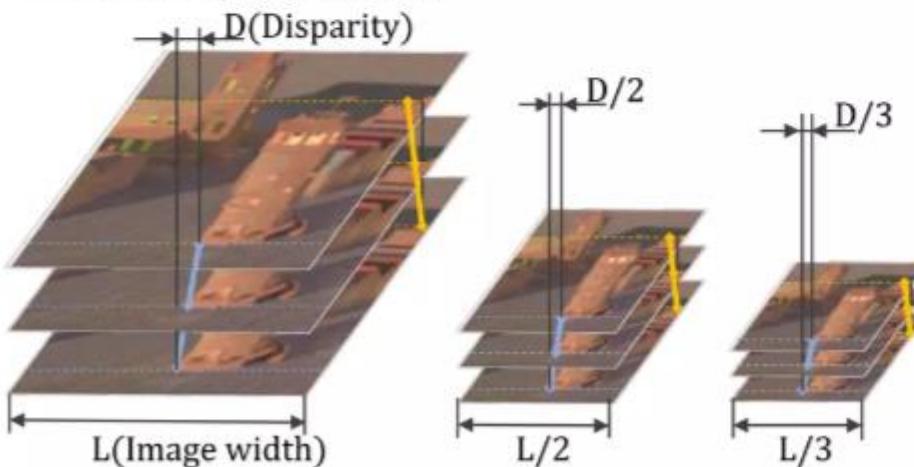
Vinyl, Range: [-1.6, 1.2]

Data Augmentation

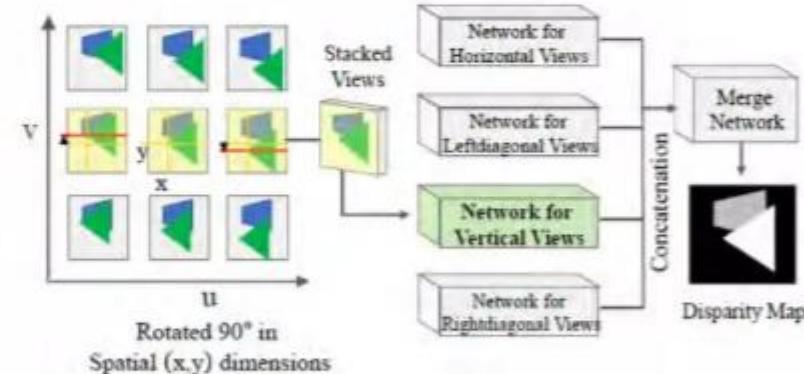
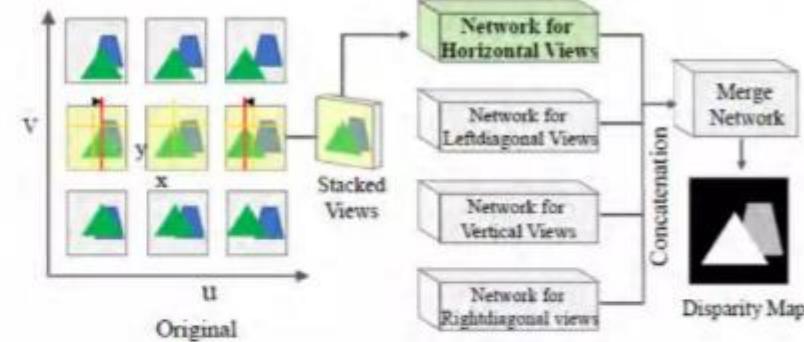
View-shift augmentation



Scale augmentation



Rotation augmentation



Angular resolution	3×3	5×5	7×7				9×9	
Augmentaion type	Full Aug	Full Aug	Color	Color + Viewshift	Color + Rotation	Color + scaling	Full Aug	Full Aug
Mean square error	1.568	1.475	2.799	2.564	1.685	2.33	1.434	1.461
Bad pixel ratio ($>0.07\text{px}$)	8.63	4.96	6.67	6.29	5.54	5.69	3.94	3.91

Questions?

Quiz 1

- Q1-What are the challenges when dealing with computer vision problems? (3)
- Q2-Suppose that we have a 1D image with values as (3, 2, 5, 8, 5, 2). Apply the average filter of size (1 x 3). What would be the value of last-secondd pixel. (3)
- Q3 Differentiate between Afine transformation and Projective transformation with respect to homographic planner perspective map. (4)

Quiz 1

- What are the challenges when dealing with computer vision problems?
 - Variation due to geometric change, photometric factors (illumination, appearance, noise), image occlusion etc
- Suppose that we have a 1D image with values as (3, 2, 5, 8, 5, 2). Apply the average filter of size (1 x 3). What would be the value of last-second pixel?
 - $(8+5+2)/3 = 5$
- Differentiate between Affine transformation and Projective transformation with respect to horizontal perspective map.
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Model fitting and optimization

Model fitting and optimization

- Some computer vision problems belong to this category.
- Given a set of incomplete data as input –we assume a model and seek an optimized output based on the model.

Input ---Model -----Output

Model fitting

- Scattered data interpolation
- Given $\{\mathbf{x}_k\}$ and $\{\mathbf{d}_k\}$, we want to find a function

$$\mathbf{f}(\mathbf{x}_k) = \mathbf{d}_k \quad (4.1)$$

- or an approximation

$$E_D = \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2 \quad (4.2)$$

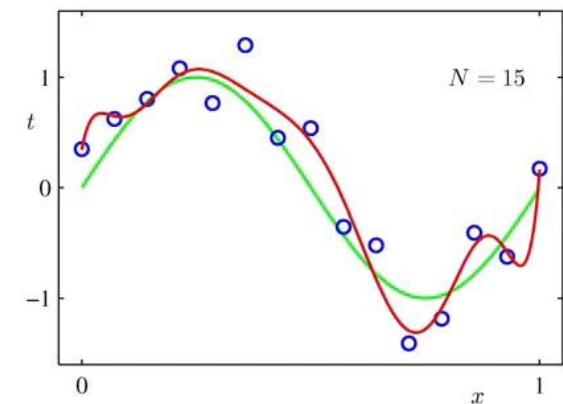
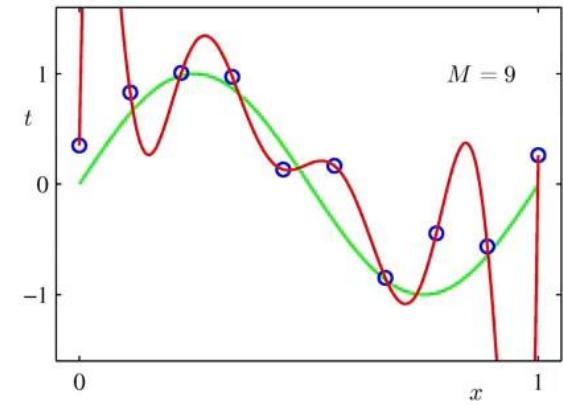


Figure 4.1(a)

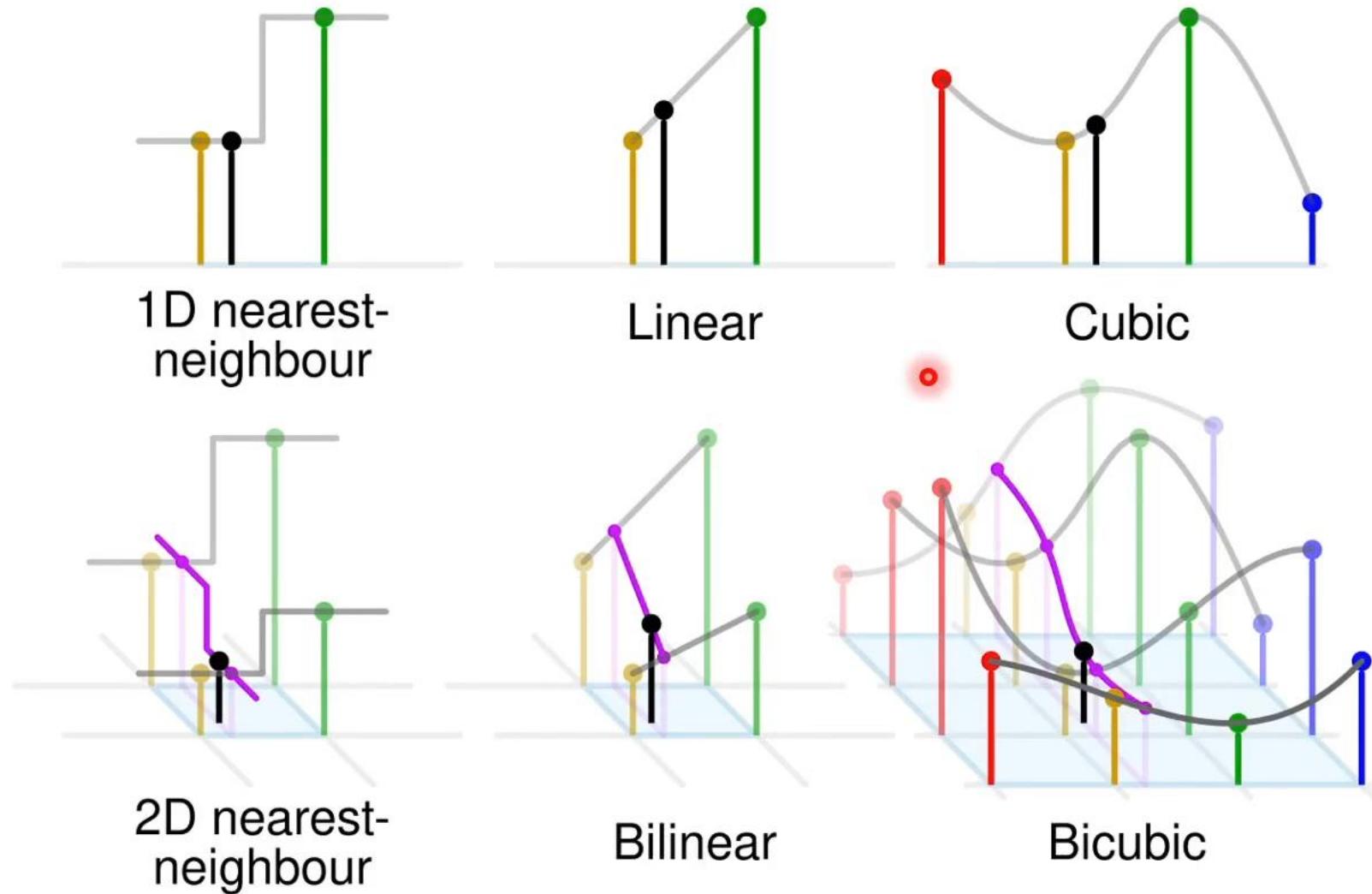
Pixels on an image

- regularly gridded



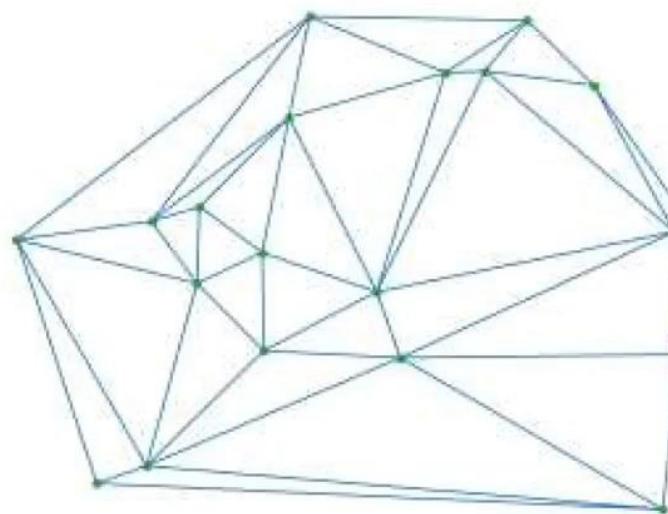
243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	10	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	7	51	137
23	32	33	148	168	203	179	43	27	17	12	8
17	26	12	160	255	255	109	22	26	19	35	24

Nearest/Linear/Cubic Interpolation

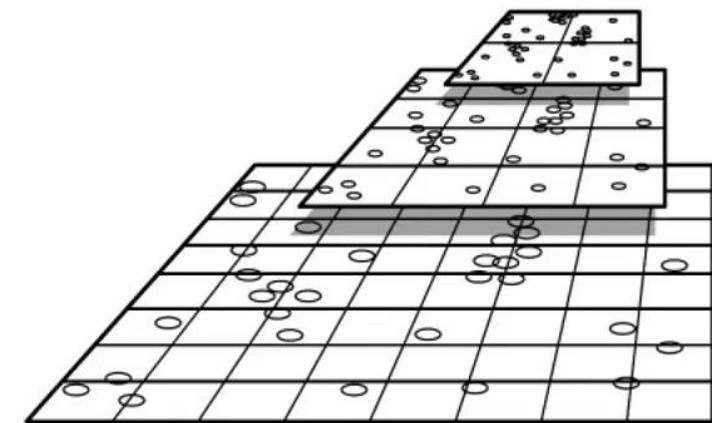


Scattered 2D Case

- Delaunay triangulation
- Pull-push algorithm



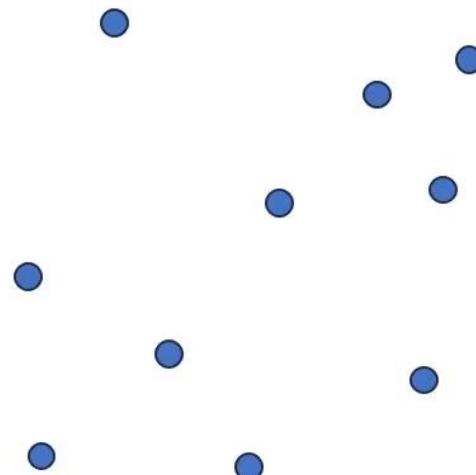
(a)



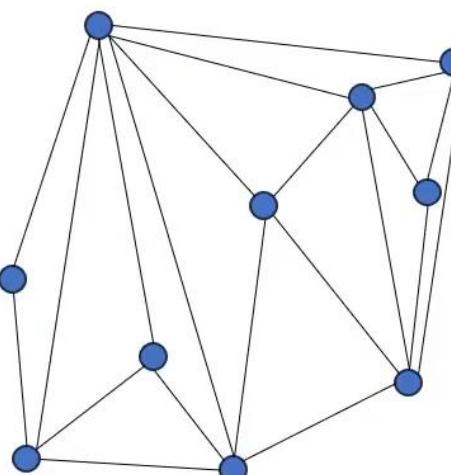
(b)

Figure 4.2

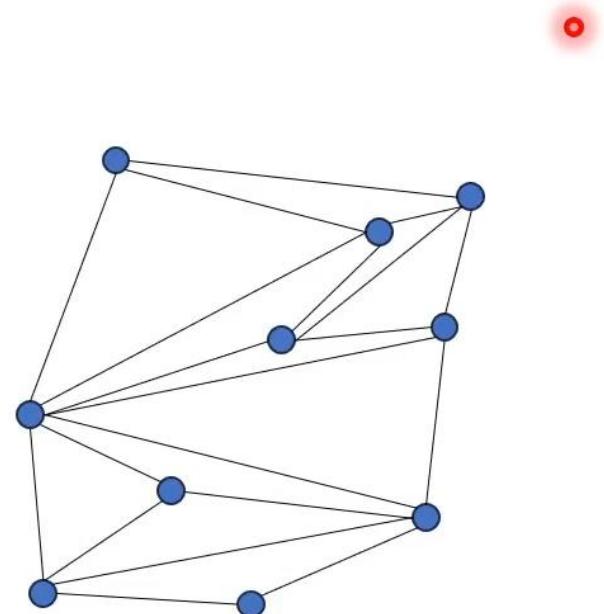
Delaunay Triangulation



Given a set of points, there are many possible meshes.

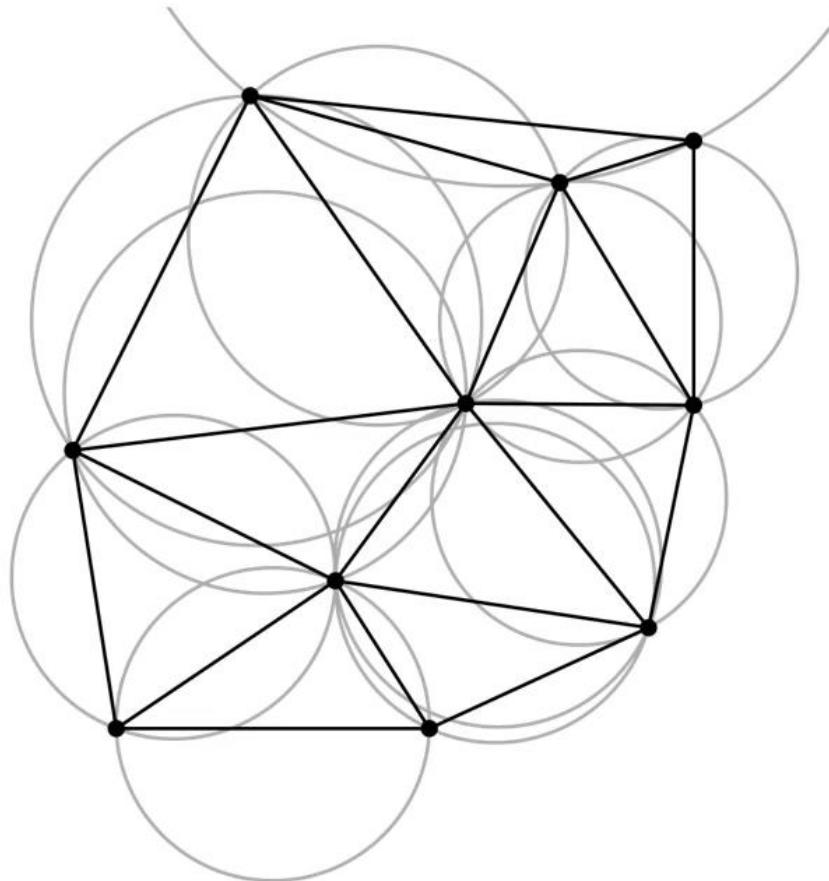


Case 1



Case 2

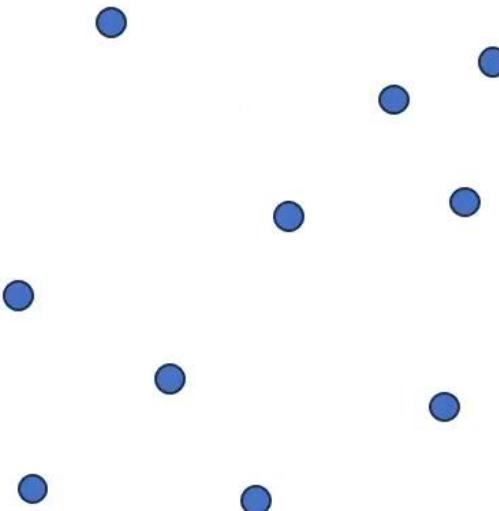
Definition of Delaunay Triangulation



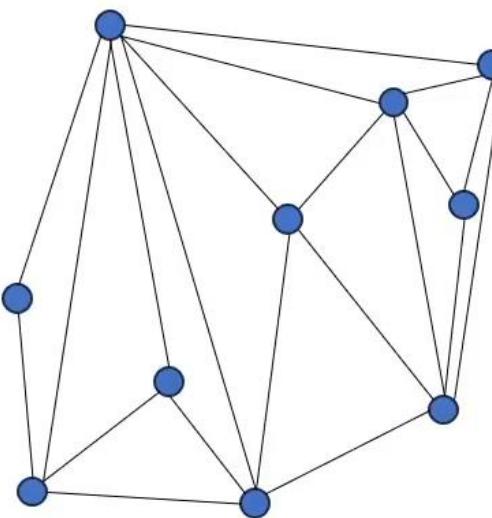
No vertex lies inside the circumcircle of any triangle.



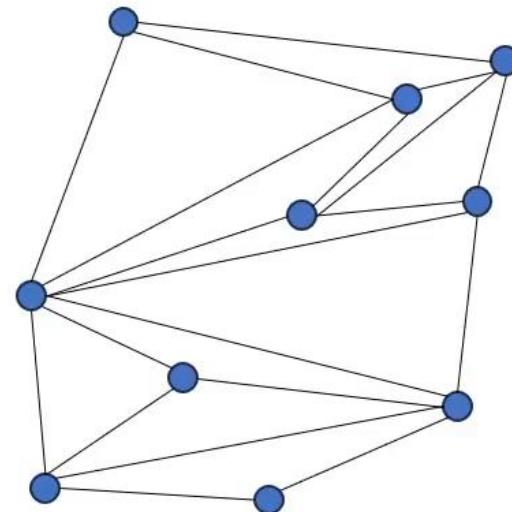
Delaunay Triangulation



Given a set of points, there are many possible meshes.



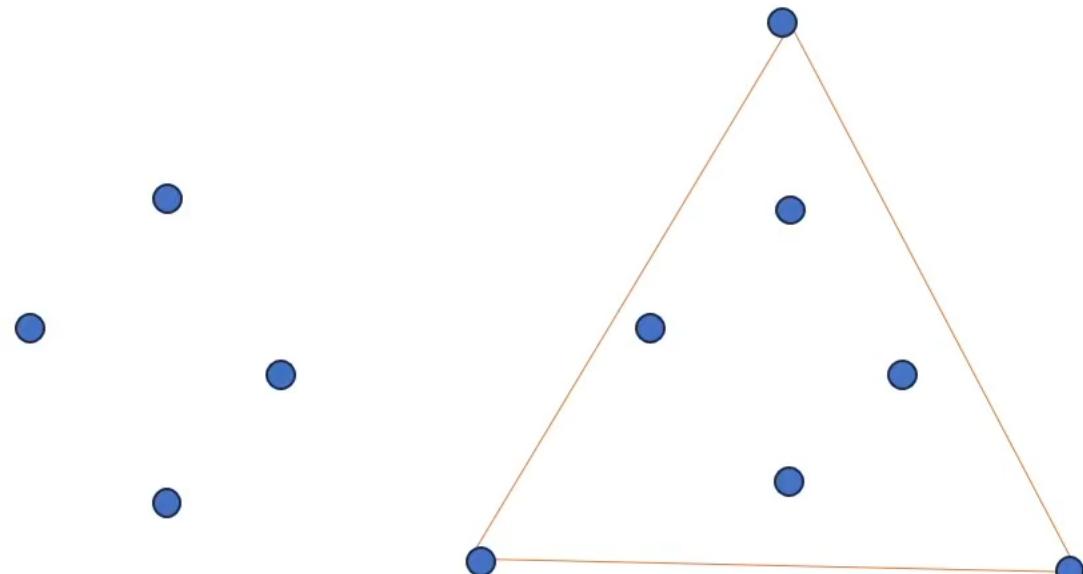
Case 1



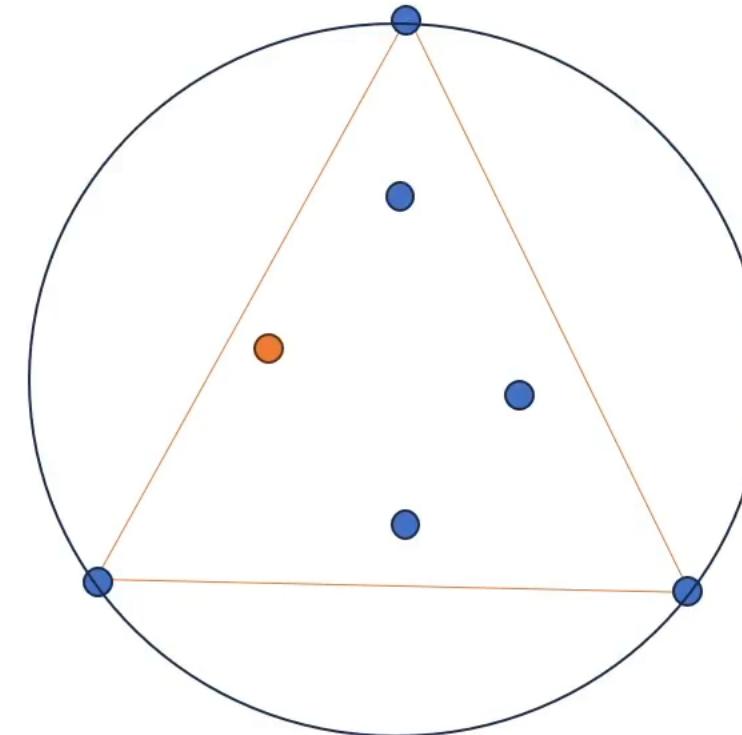
Case 2

Bowyer-Watson algorithm

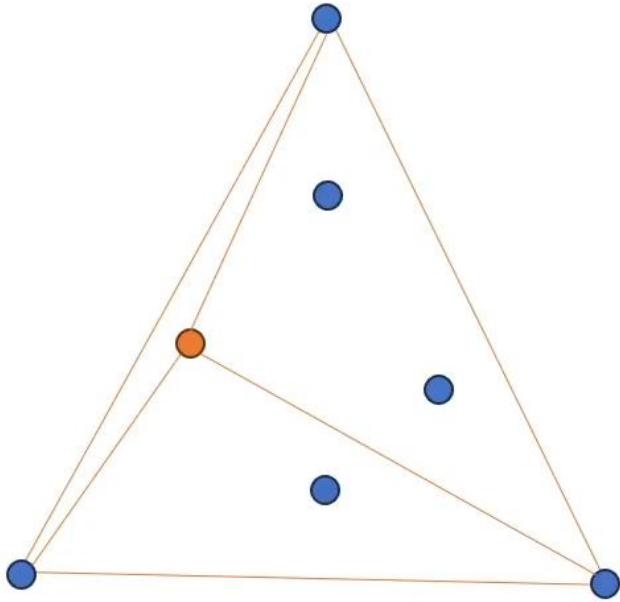
We use 4 points for illustration.



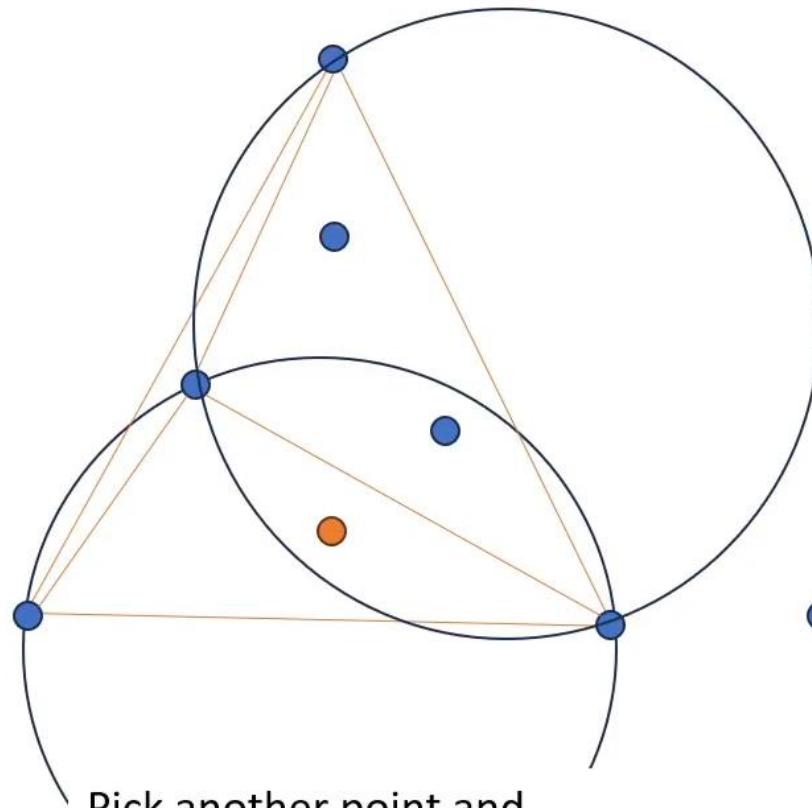
Create a super-triangle from three virtual vertices to enclose them.



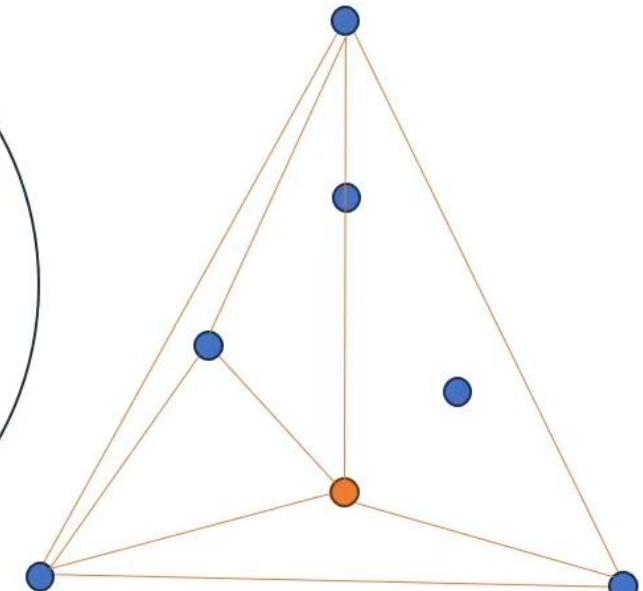
Select a point. Look for triangles whose circumcircles contain this point. It is the super-triangle this time. Thus, this is not a Delaunay triangle.



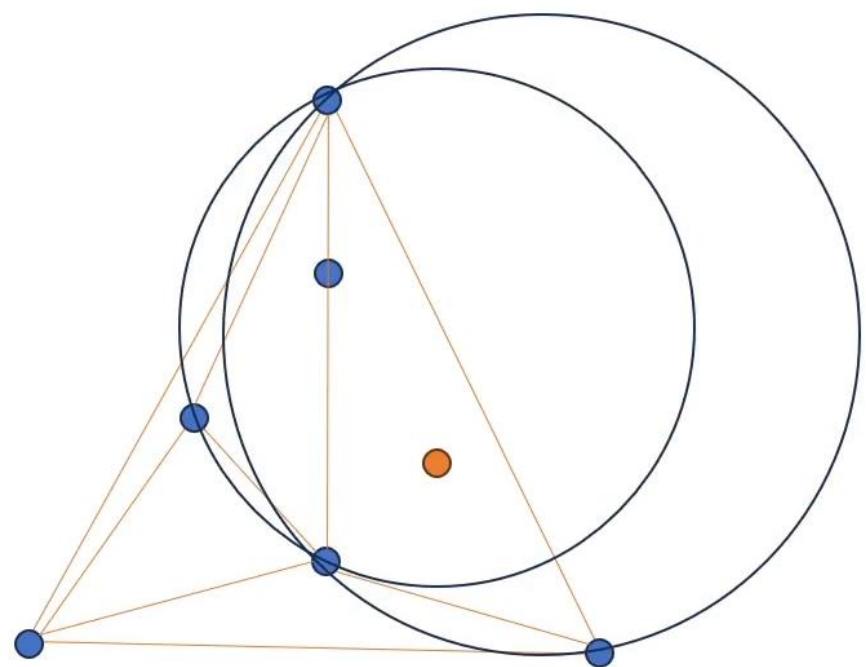
Connect this point to the three vertices to create new triangles.



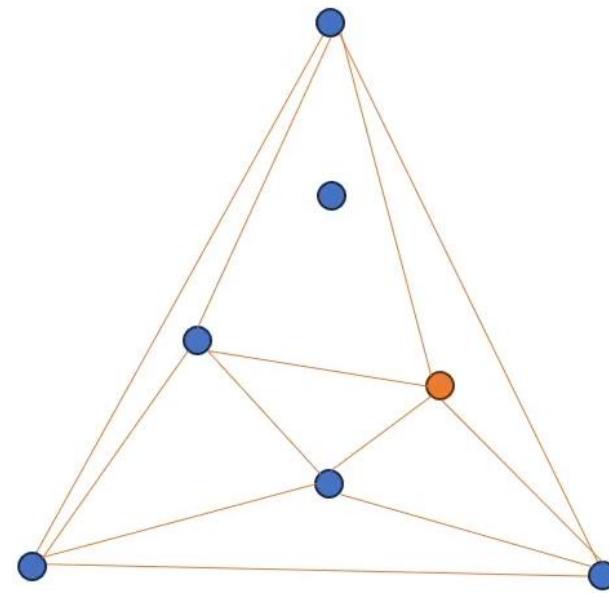
Pick another point and repeat. We have two triangles whose circumcircles contain this point.



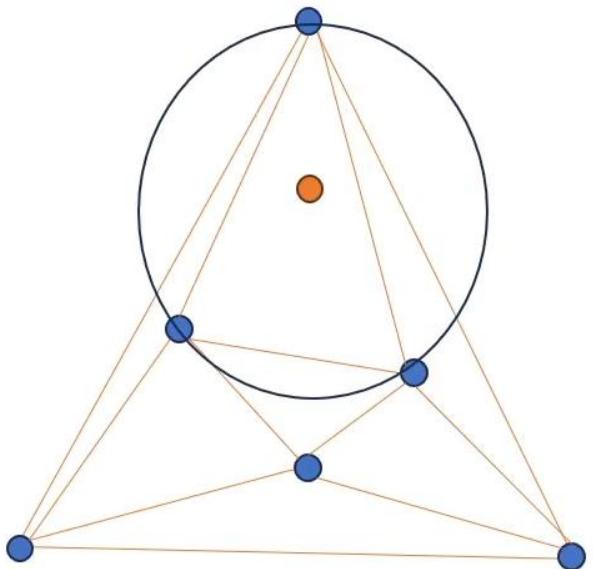
Remove their common edge and connect to their vertices.



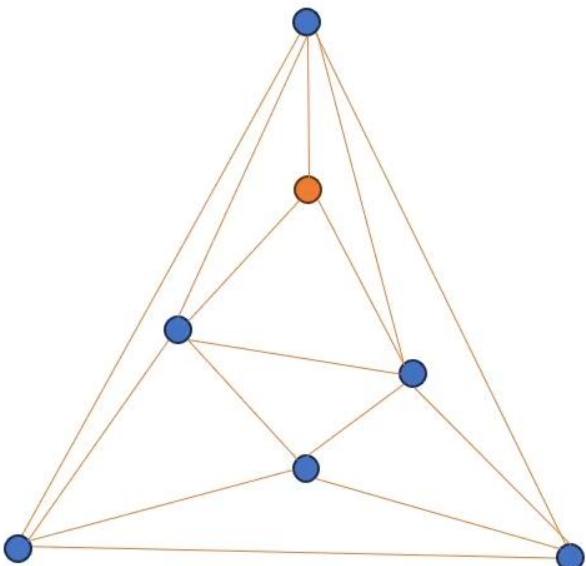
Pick another point and repeat. We have two triangles whose circumcircles cover this point.



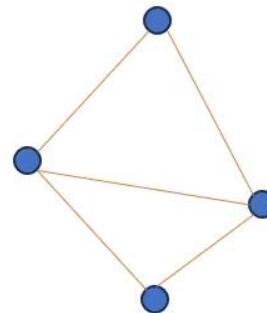
Remove their common edge and connect to their vertices.



No common edge to remove.



Connect the point to the triangle's vertices.



Finally, delete the super-triangle vertices and their edges.

Pull-push algorithm

Images from KRAUS, Martin. The pull-push algorithm revisited. *Proceedings GRAPP*, 2009, 2: 3.

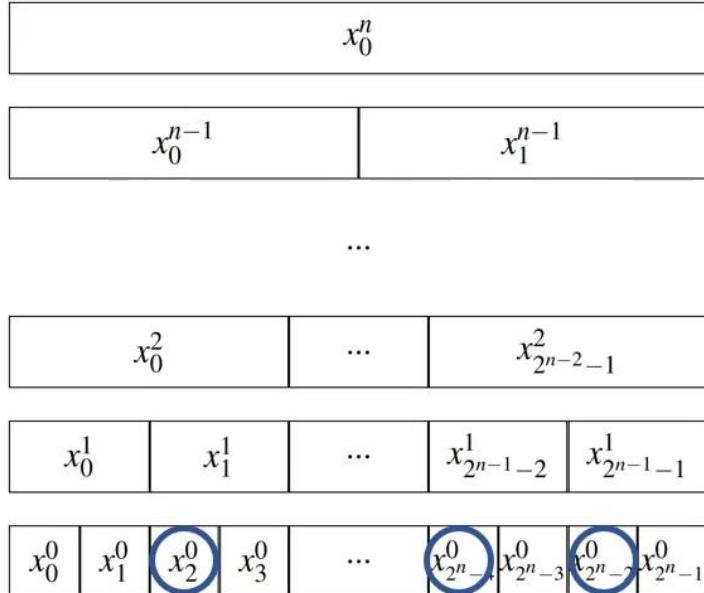
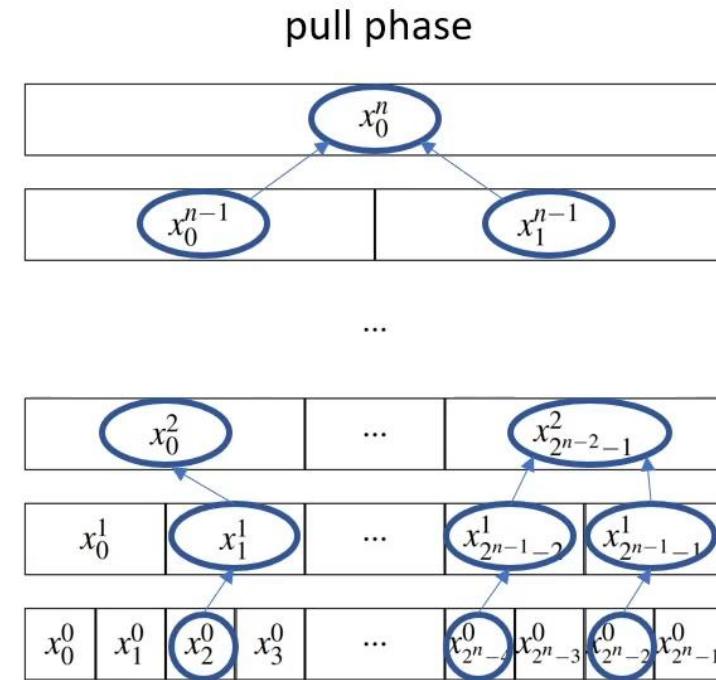
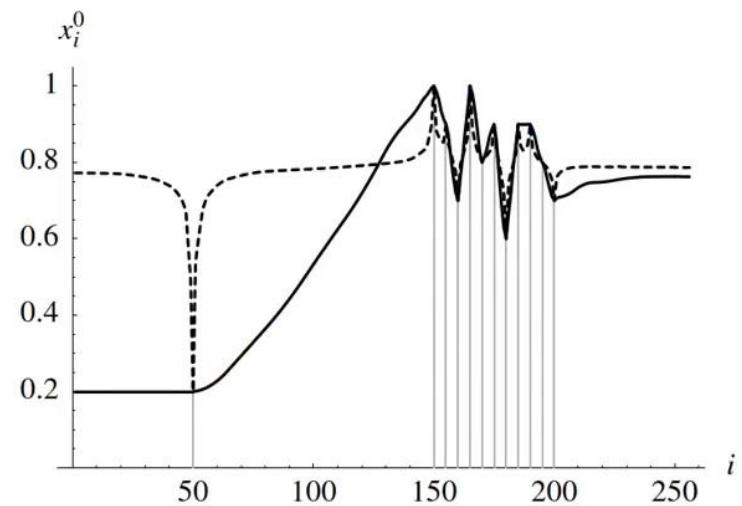
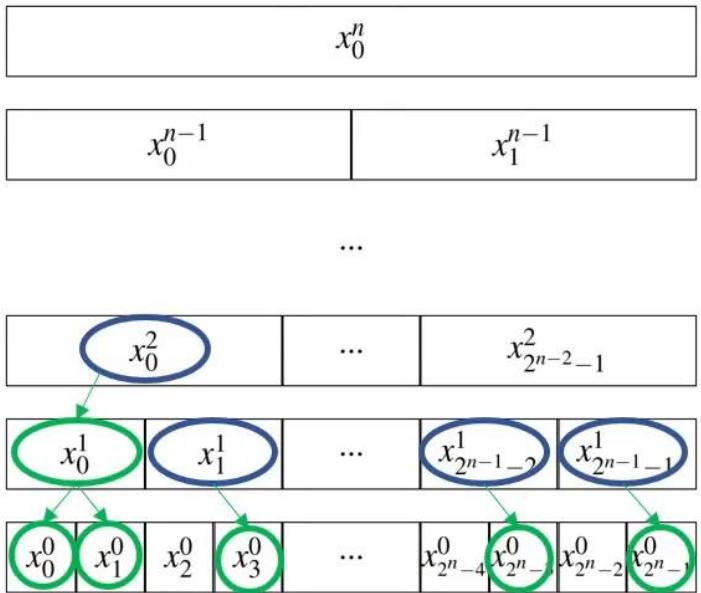


Illustration of the pixels x_i^r of an image pyramid for a one-dimensional input image x_i^0 of size 2^n .



push phase

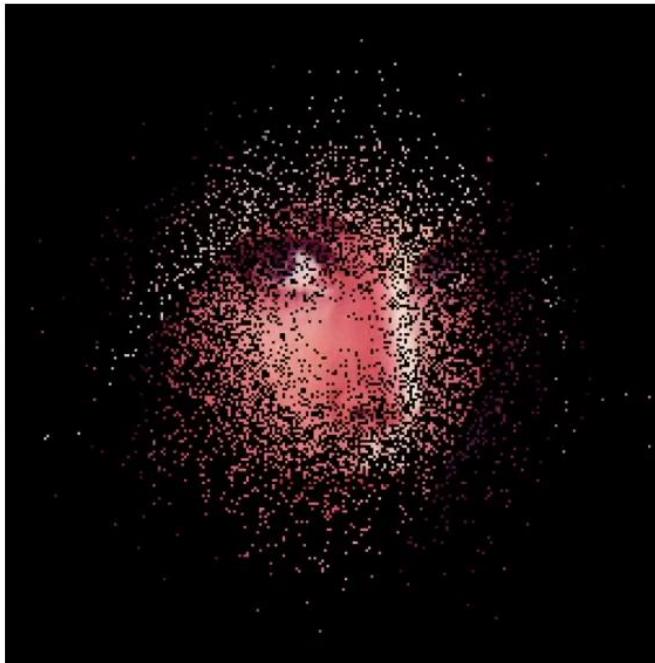


Interpolation of a set of scattered data (gray bars). Solid and gray lines are generated by different weight settings.

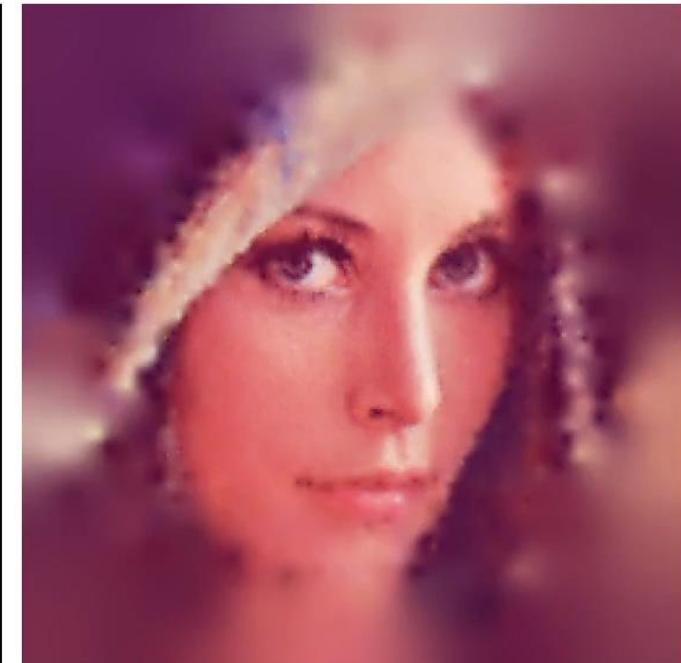
Image example



Lena
Original image as a reference



Input for the pull-push algorithm



Result

Images from KRAUS, Martin. The pull-push algorithm revisited. *Proceedings GRAPP*, 2009, 2: 3.

How to interpolate data in a high-dimensional space?


$$\text{kernel regression} \quad \mathbf{f}(\mathbf{x}) = \sum_k \mathbf{w}_k \phi(\|\mathbf{x} - \mathbf{x}_k\|) \quad (4.3)$$

$\phi(\cdot)$ is called a radial basis function (nickname kernel)

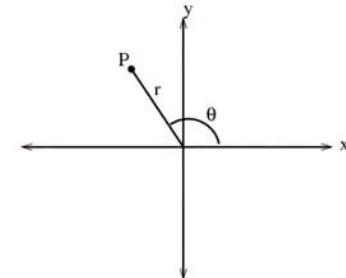


Image from mathinsight.org

Some widely used radial basis functions. It is called radial because it only takes distance into account.

Gaussian

$$\phi(r) = \exp(-r^2/c^2) \quad (4.4)$$

Hardy multiquadric

$$\phi(r) = \sqrt{(r^2 + c^2)} \quad (4.5)$$

Inverse multiquadric

$$\phi(r) = 1/\sqrt{(r^2 + c^2)} \quad (4.6)$$

Thin plate spline

$$\phi(r) = r^2 \log r. \quad (4.7)$$

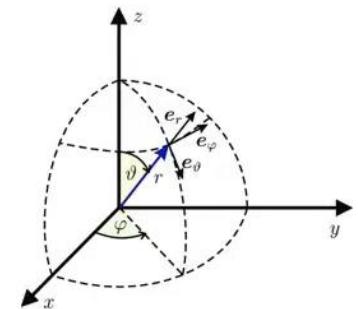
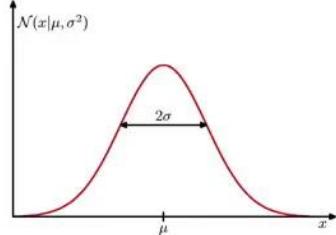


Image from Lotze, Jörg. (2006).
Efficient Polarimetric Antenna
Radiation Pattern Modeling.



$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2}(x - \mu)^2 \right\}$$

Gaussian distribution

Questions?

An example

The screenshot shows a web browser window displaying a Kaggle notebook. The title bar reads "The Boston Housing Dataset". The URL in the address bar is <https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset/notebook>. The page content is as follows:

The Boston Housing Dataset
Python · Boston House Prices

Notebook Input Output Logs Comments (24)

Run
22.9s Version 5 of 5

The Boston Housing Dataset

The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of [Boston MA](#). The following describes the dataset columns:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways

Feature dimension 14, number of instance 506



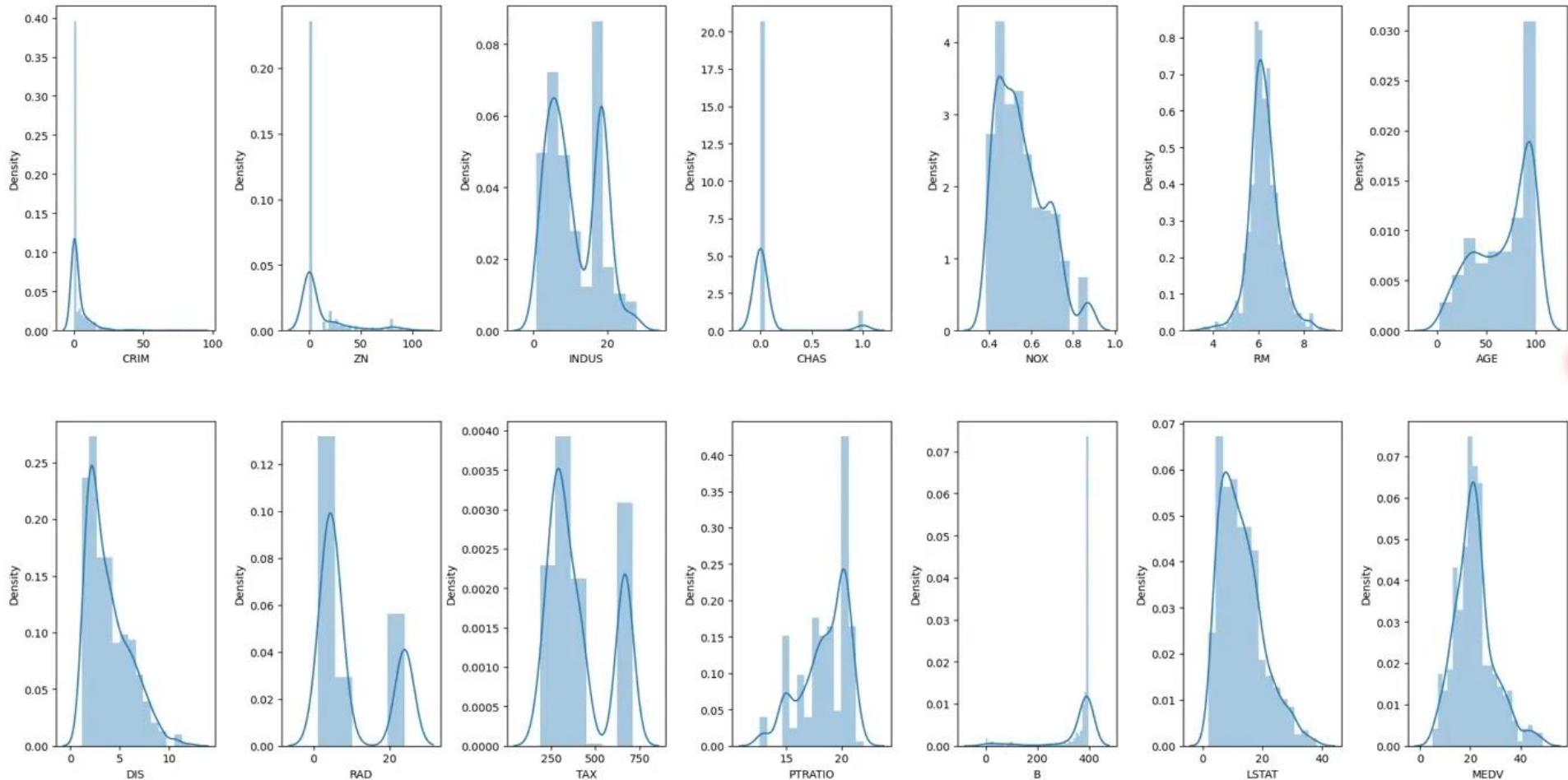
- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centers
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per \$10,000
- PTRATIO - pupil-teacher ratio by town
- B - $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town (This variable has an ethical problem. It is non-invertible and assumes that racial self-segregation had a positive impact on house prices.)
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in \$1000's (**House Price**)

Its data look like

```
['housing.csv']
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Histogram of the 14 features



Model the house price as a function of other variables

$$\mathbf{f}(\mathbf{x}) = \sum_k \mathbf{w}_k \phi(\|\mathbf{x} - \mathbf{x}_k\|) \quad (4.3)$$

For one-dimension output

$$PredictedHousePrice = \sum_{k=1}^{506} w_k \Phi(\|\mathbf{x} - \mathbf{x}_k\|)$$

$$\mathbf{x}_k = \begin{Bmatrix} CRIM \\ ZN \\ \vdots \\ LSTAT \end{Bmatrix}$$

For two-dimension output

$$\begin{bmatrix} PredictedHousePrice \\ PredictedLSTAT \end{bmatrix} = \sum_{k=1}^{506} \mathbf{w}_k \Phi(\|\mathbf{x} - \mathbf{x}_k\|)$$

$$\mathbf{x}_k = \begin{Bmatrix} CRIM \\ ZN \\ \vdots \\ B \end{Bmatrix}$$

Solve the \mathbf{w}_k parameters

- If we want our function to exactly interpolate the data values, we solve the linear system of equations

$$\mathbf{f}(\mathbf{x}_k) = \sum_l \mathbf{w}_l \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) = \mathbf{d}_k \quad (4.8)$$

$$Price_1 = \sum_{k=1}^{506} w_k \phi(\|\mathbf{x}_1 - \mathbf{x}_k\|)$$

$$Price_2 = \sum_{k=1}^{506} w_k \phi(\|\mathbf{x}_2 - \mathbf{x}_k\|)$$

⋮

$$Price_{506} = \sum_{k=1}^{506} w_k \phi(\|\mathbf{x}_{506} - \mathbf{x}_k\|)$$

data point as
(4.1)

III-conditioned

$$\begin{bmatrix} Price_1 \\ Price_1 \\ \vdots \\ Price_{506} \end{bmatrix} = \begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_{506}\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{x}_{506}\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_{506} - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_{506} - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_{506} - \mathbf{x}_{506}\|) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{506} \end{bmatrix}$$

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{506} \end{bmatrix} = \begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_1 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_{506}\|) \\ \phi(\|\mathbf{x}_2 - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_2 - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_2 - \mathbf{x}_{506}\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_{506} - \mathbf{x}_1\|) & \phi(\|\mathbf{x}_{506} - \mathbf{x}_2\|) & \cdots & \phi(\|\mathbf{x}_{506} - \mathbf{x}_{506}\|) \end{bmatrix}^{-1} \begin{bmatrix} Price_1 \\ Price_1 \\ \vdots \\ Price_{506} \end{bmatrix}$$

III-conditioned: a small change in a Price variable will cause huge changes of the output values of the **w** vector.

Regularizer

- Regularized data approximation problem

$$E_D = \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2 \quad (4.2)$$

$$E_W = \sum_k \|\mathbf{w}_k\|^p \quad (4.9)$$

Given $\mathbf{d}_k, \mathbf{x}_k, \lambda$, $E(\{\mathbf{w}_k\}) = E_D + \lambda E_W$
minimize

$$= \sum_k \left\| \sum_l \mathbf{w}_l \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) - \mathbf{d}_k \right\|^2 + \lambda \sum_k \|\mathbf{w}_k\|^p \quad (4.11)$$



Image from stock.adobe.com

P	Machine Learning	Neural Network
2	Ridge regression	Weight decay (encourage weights to decay to zero)
1	Lasso (least absolute shrinkage and selection operator)	

Ridge



Image from YouTube *Stoos Ridge Hike | Switzerland* by Alex Kay Travels

As λ decreases, w_i increases.
ridge-like shape

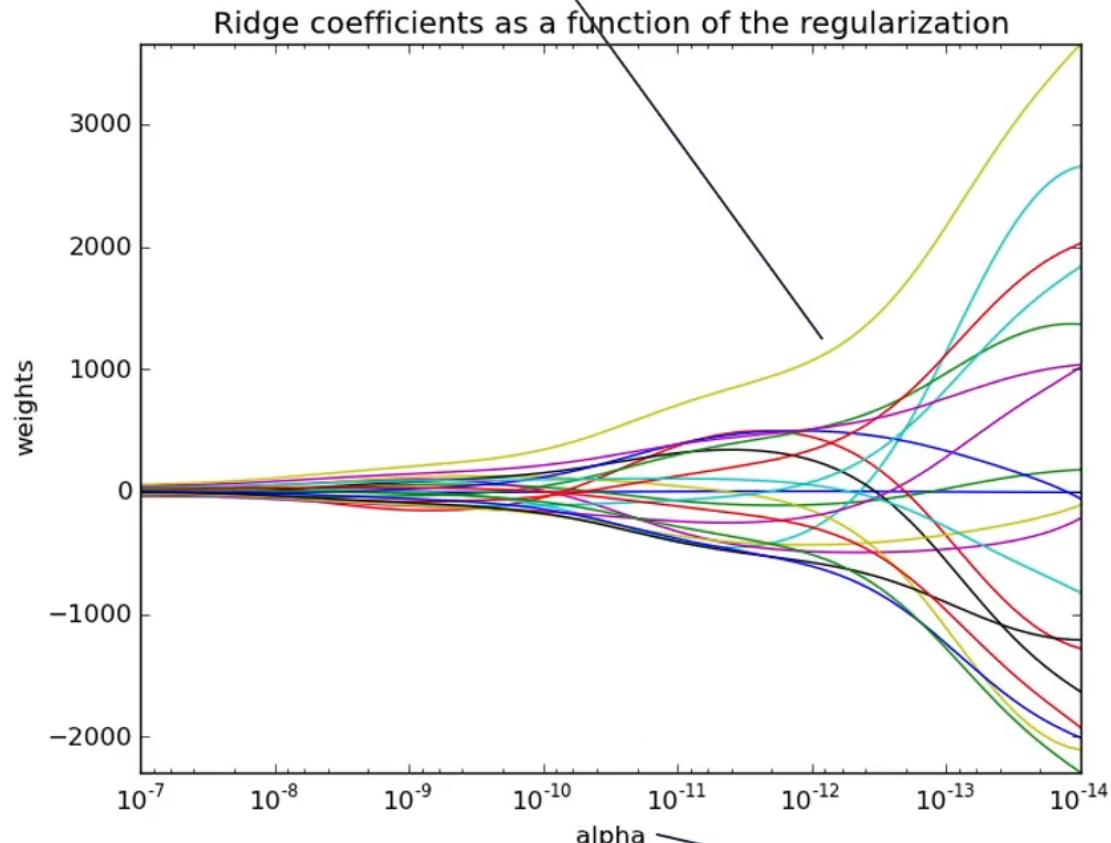


Image from fa.bianp.net

same as λ in
our textbook

Kernel regression

- If we just want to interpolate a set of data, we can use

$$\mathbf{f}(\mathbf{x}) = \frac{\sum_k \mathbf{d}_k \phi(\|\mathbf{x} - \mathbf{x}_k\|)}{\sum_l \phi(\|\mathbf{x} - \mathbf{x}_l\|)} \quad (4.12)$$

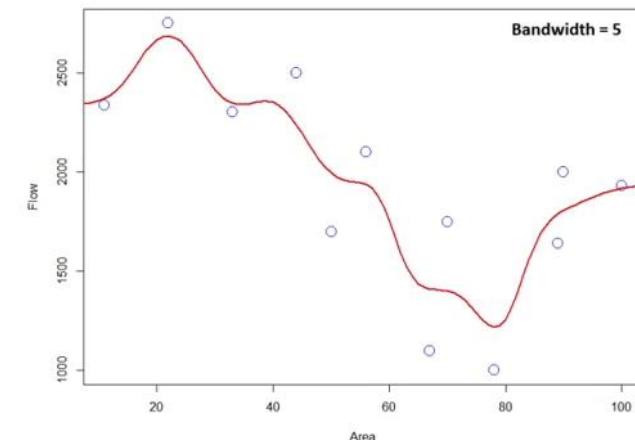
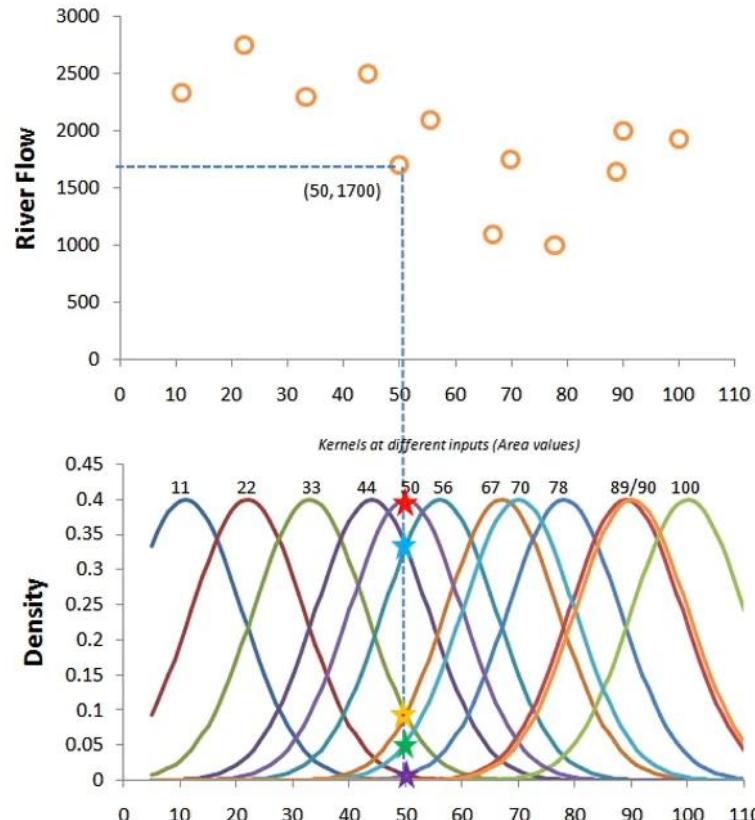
- There is no parameter to tune. It is just the weighted average of the data \mathbf{d}_k , and weights are decided by the radial basis function (kernel).

$$\phi'_k(\mathbf{x}) = \frac{\phi(\|\mathbf{x} - \mathbf{x}_k\|)}{\sum_l \phi(\|\mathbf{x} - \mathbf{x}_l\|)} \quad (4.13)$$

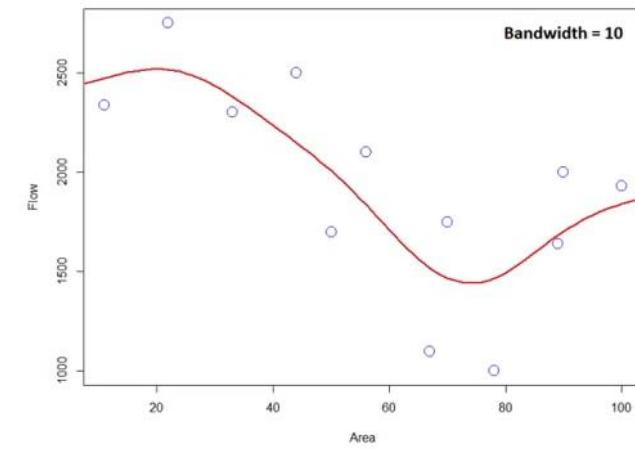
Example of Gaussian Kernel Regression

Gaussian

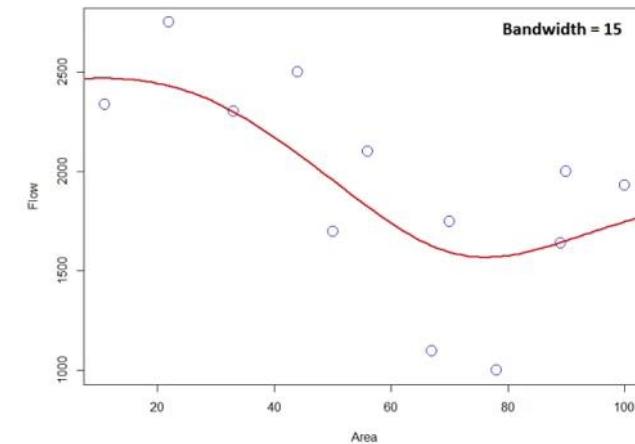
$$\phi(r) = \exp(-r^2/c^2) \quad (4.4)$$



small c

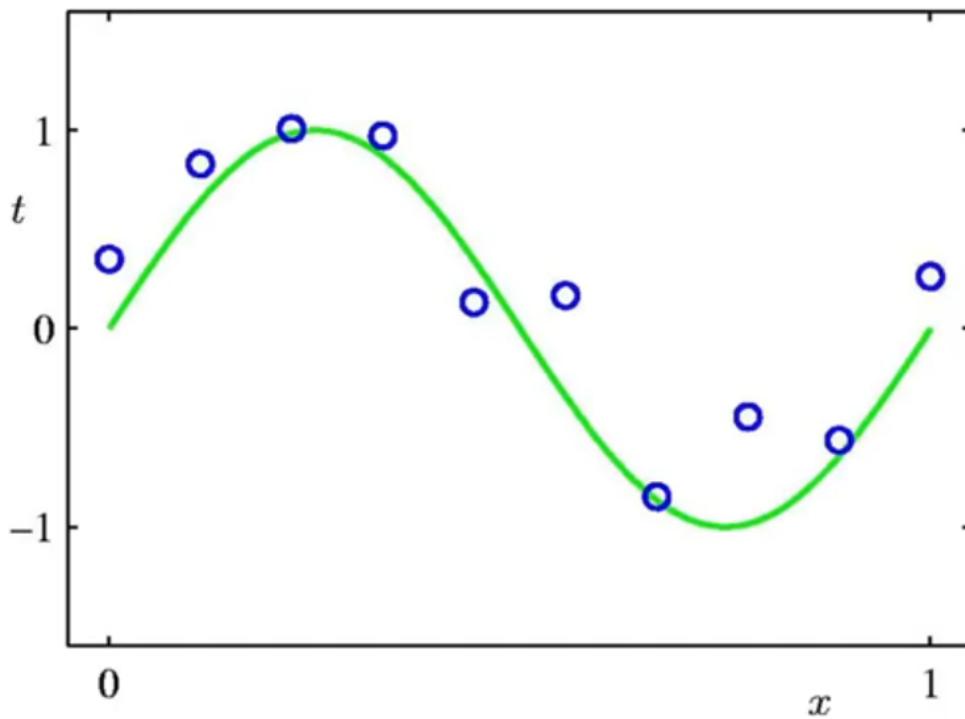


medium c



large c

Polynomial Curve Fitting

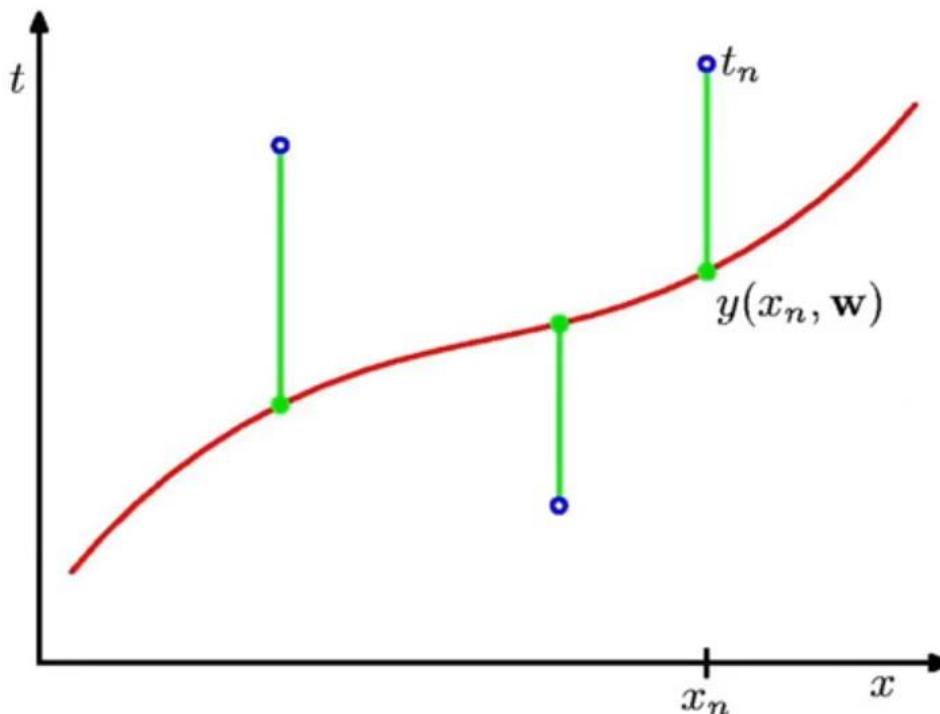


Green: the ground truth curve
 $\sin(2\pi x)$
blue points: observation values with noise
number of observations: 10

We use a polynomial function to fit the data

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$$

Sum-of-Squares Error Function



To determine the weight vector \mathbf{w} , we minimize the error function.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

0^{th} Order Polynomial ($M=0, N=10$)

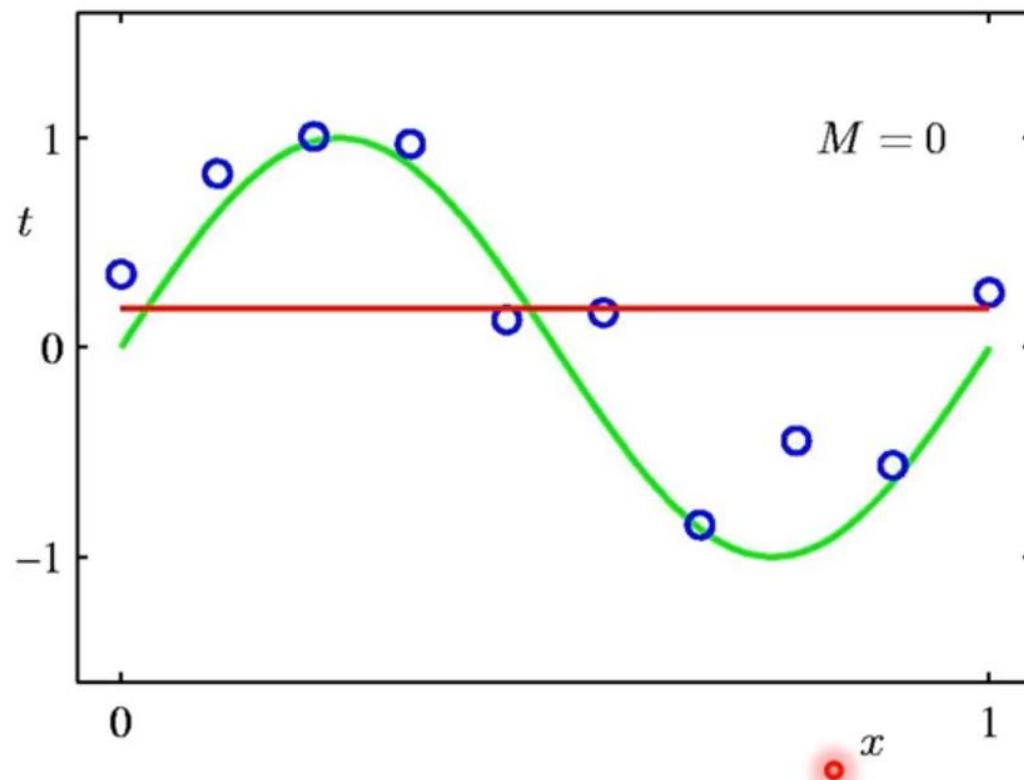
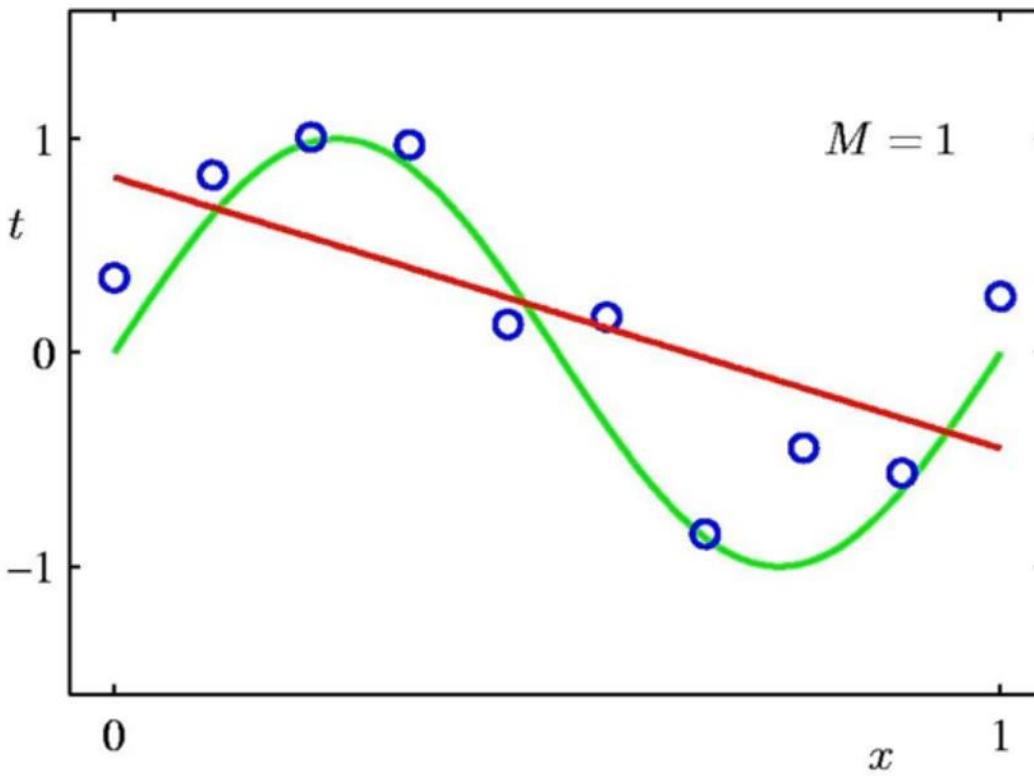


Figure 4.3

1st Order Polynomial ($M=1$, $N=10$)



When $M=0$ or 1 , we get large training error values because our model does not fit the ground truth model.

Figure 4.3

3rd Order Polynomial ($M=3, N=10$)

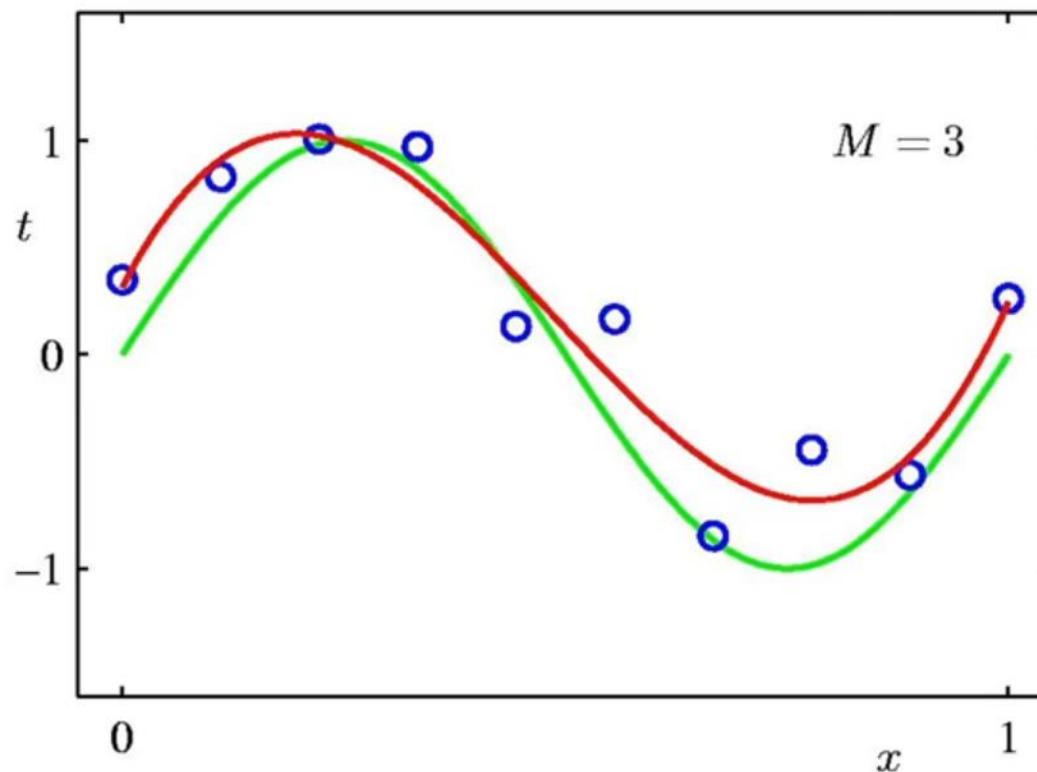
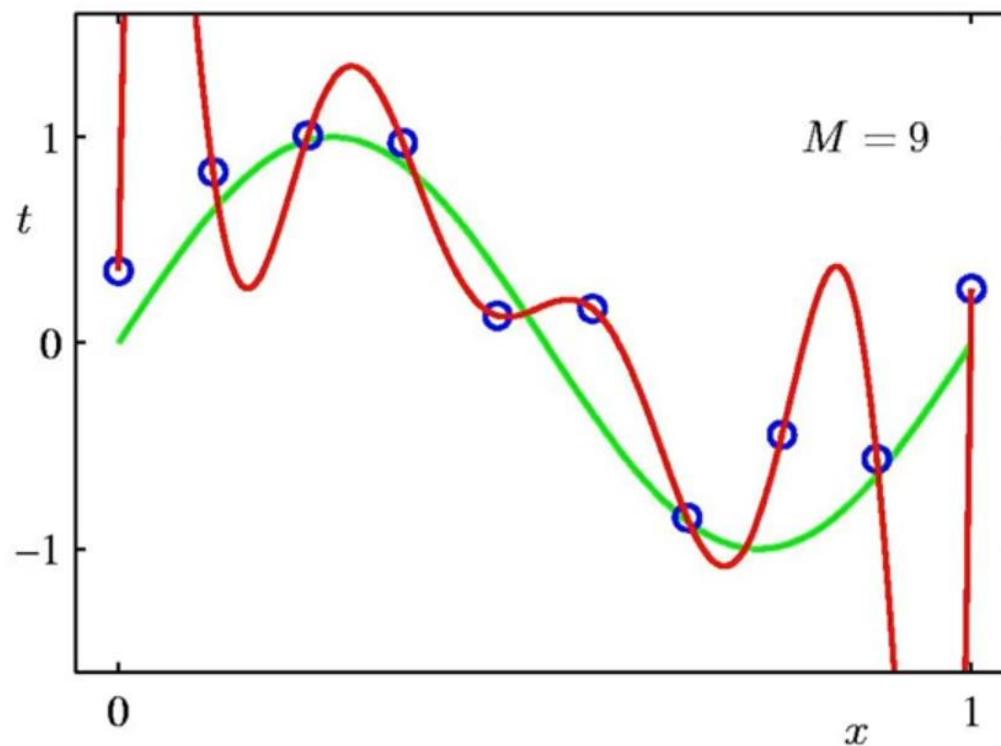


Figure 4.3

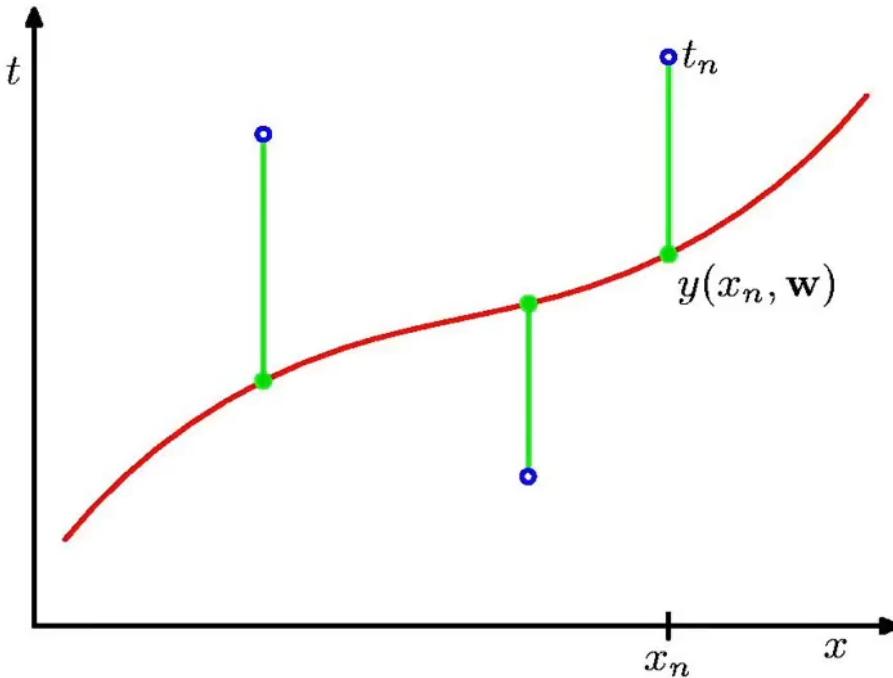
9th Order Polynomial ($M=9, N=10$)



When $M = 9$, our training error reduces to 0, but the learned model is highly unlike the ground truth model.

Figure 4.3

Sum-of-Squares Error Function



To determine the weight vector \mathbf{w} , we minimize the error function.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$

Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

Those parameters are
too large to make sense.

Overfitting and Underfitting

- If we split our dataset into two subsets: training and validation.
- We have a model containing hyperparameters (M, λ) and parameters (w_0, w_1, \dots, w_M).
- Under various hyperparameter values, we use the training set to learn the optimized parameters (w_0, w_1, \dots, w_M).

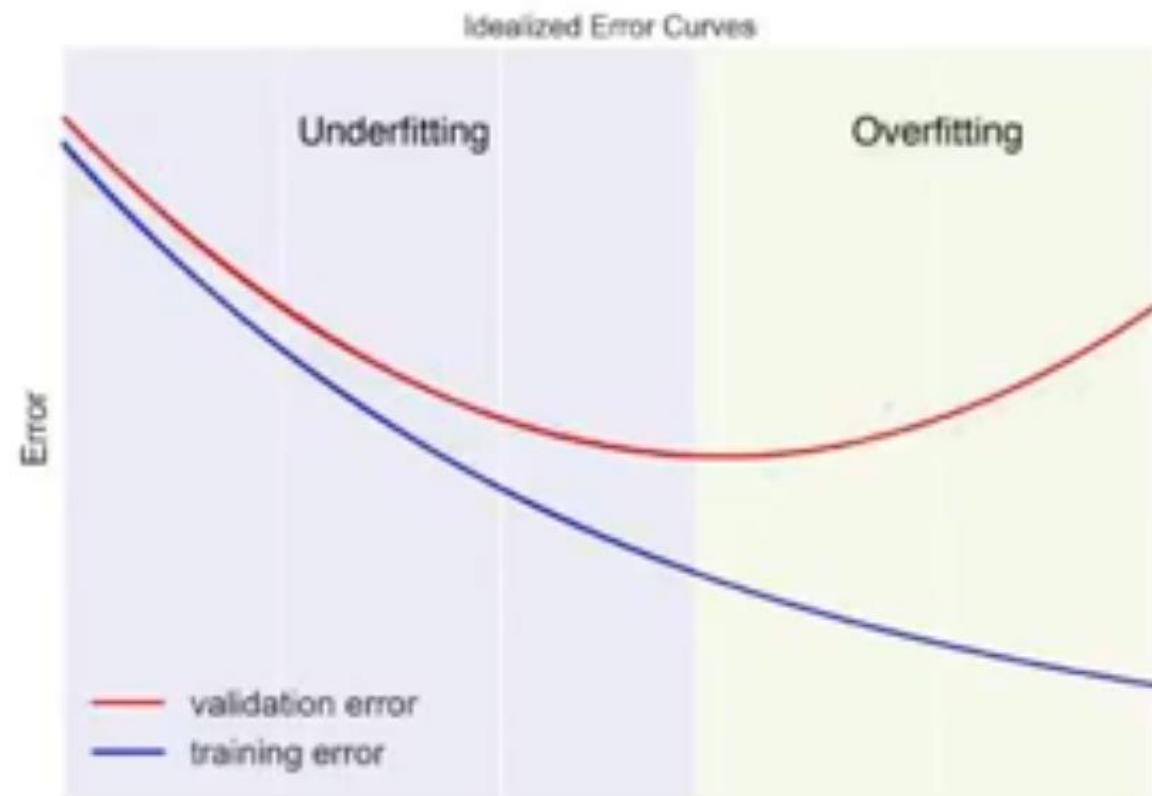
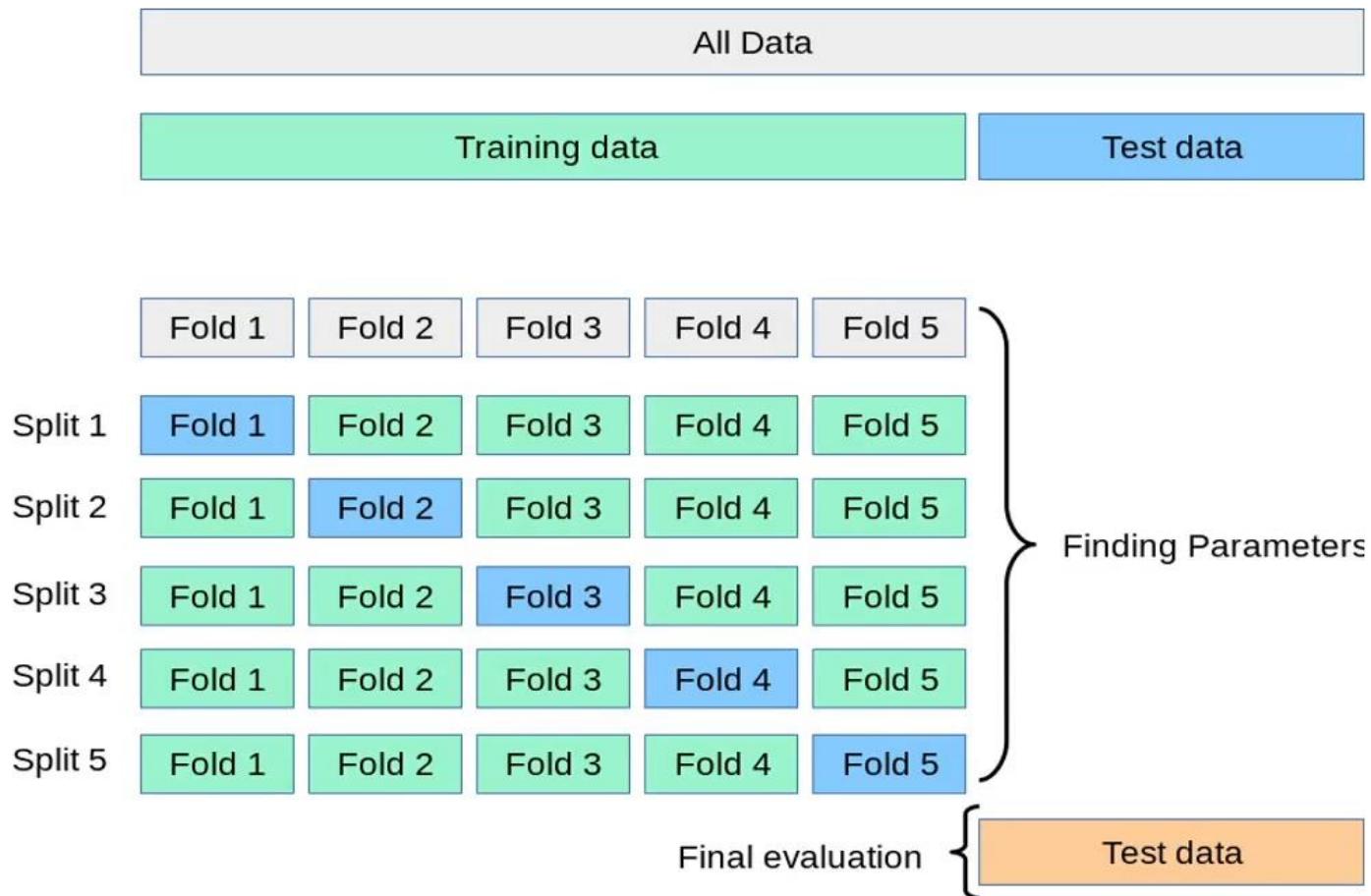


Figure 4.5

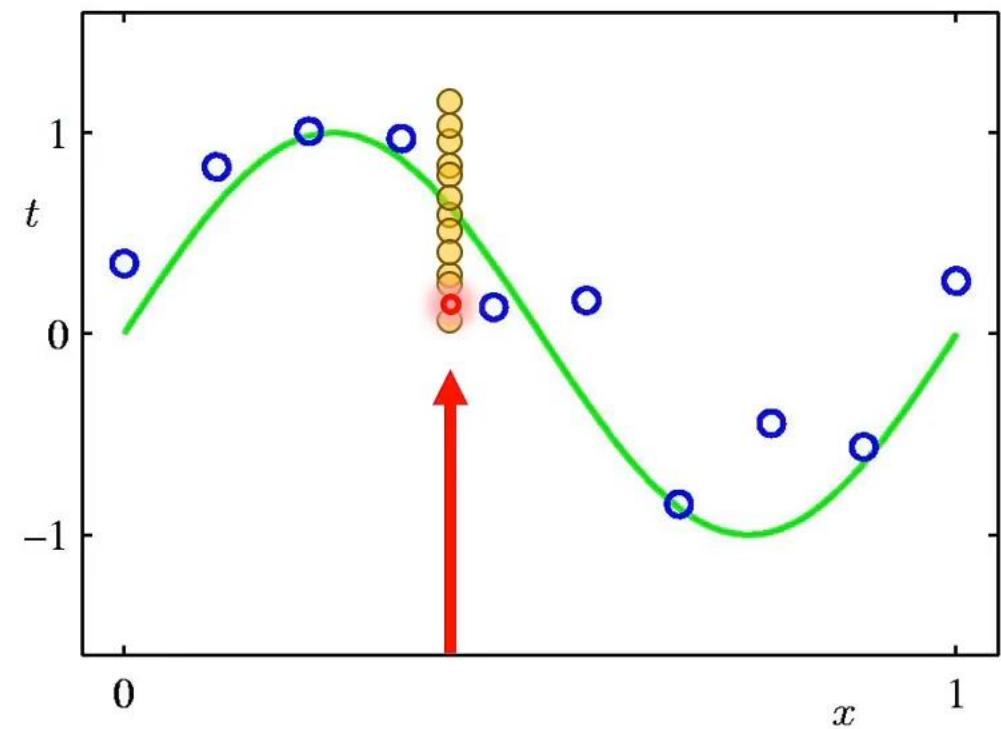
Cross-Validation



Bias-Variance tradeoff

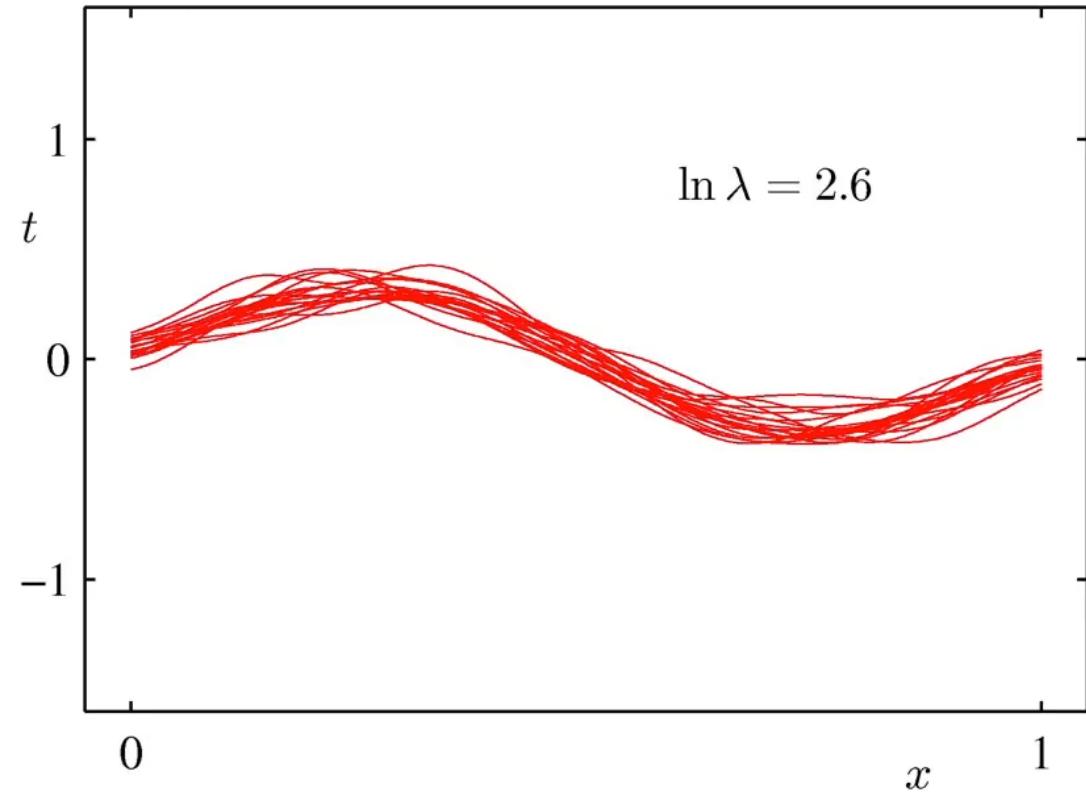
- \mathbf{x} : input vector
- t : observed value, which has a distribution based on \mathbf{x}
- $h(\mathbf{x})$: Expectation of t when input is \mathbf{x}

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x}) dt \quad (3.36) \text{ Bishop's book}$$



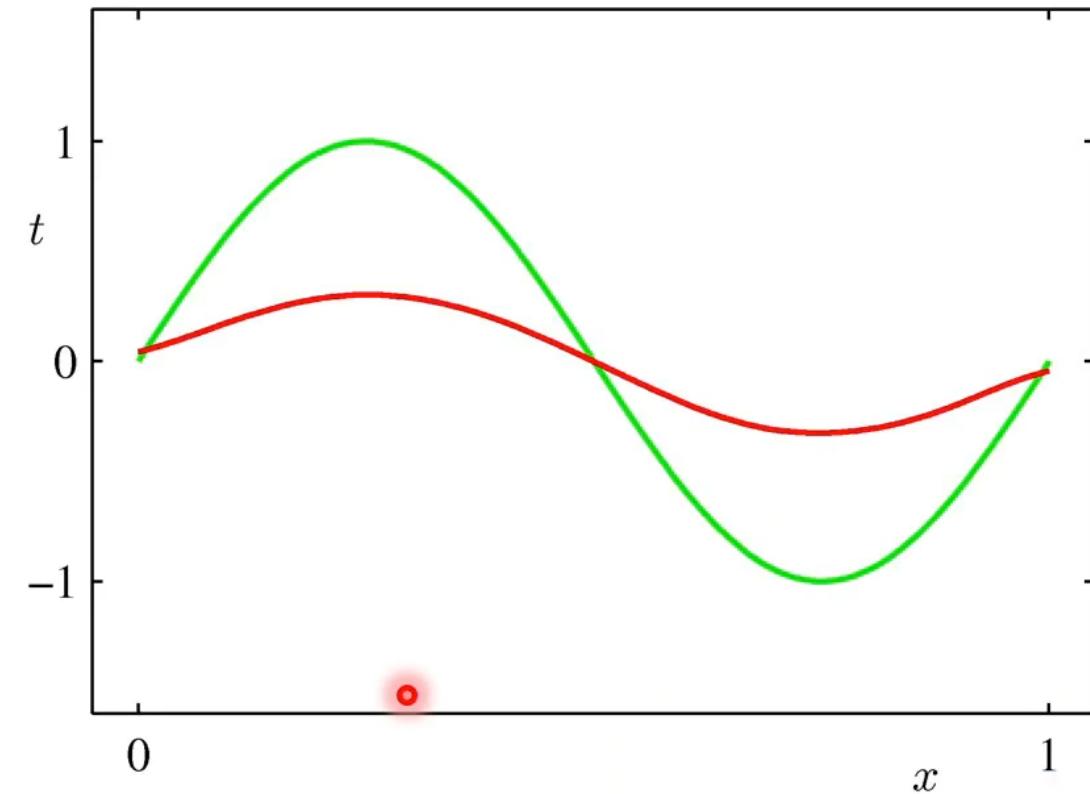
Gaussian basis function, 100 data sets, 25 points per set, $\lambda = e^{2.6}$

only 20 sets are shown for clarity



Low variance

mean of the 100 $y(x)$ functions



High bias

Figure 4.6⁵³

- Definition of an average loss

$$\mathbb{E}[L] = \iint L(t, y(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt \quad (1.86) \text{ Bishop}$$

- If the loss function is a squared term $L(t, y(\mathbf{x})) = \{y(\mathbf{x}) - t\}^2$, the average loss becomes

$$\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt \quad (1.87) \text{ Bishop}$$

- We want to choose $y(\mathbf{x})$ to minimize $\mathbb{E}[L]$, which occurs when its derivative is 0

$$\frac{\delta \mathbb{E}[L]}{\delta y(\mathbf{x})} = 2 \int \{y(\mathbf{x}) - t\} p(\mathbf{x}, t) dt = 0 \quad (1.88) \text{ Bishop}$$

‘Old Faithful’ data set

- 272 measurements of the eruption of the Old Faithful geyser at Yellowstone National Park in the USA



image from tw.traveleredge.com

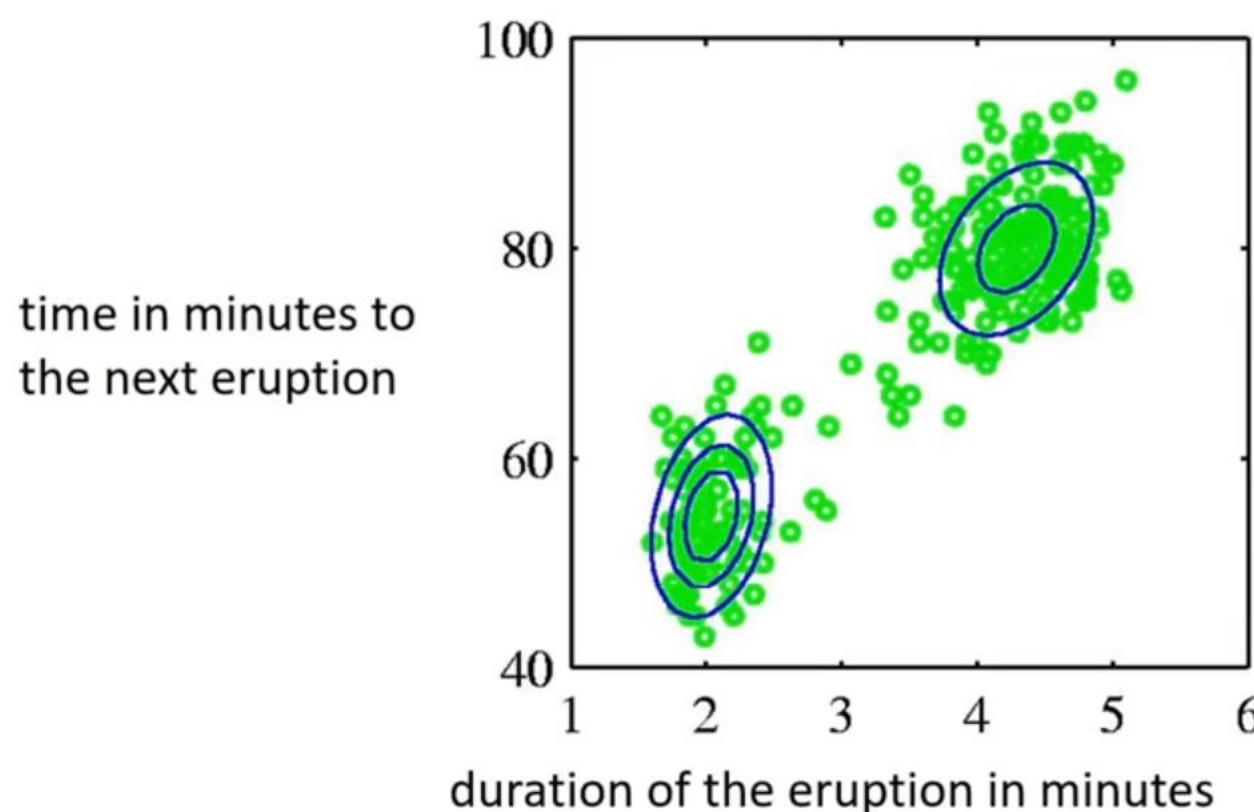


Figure 2.21
Bishop

Illustration

- If the loss function is a squared term $L(t, y(\mathbf{x})) = \{y(\mathbf{x}) - t\}^2$, the value of $y(\mathbf{x})$ should be the mean of t under the conditional distribution $p(t|\mathbf{x})$.

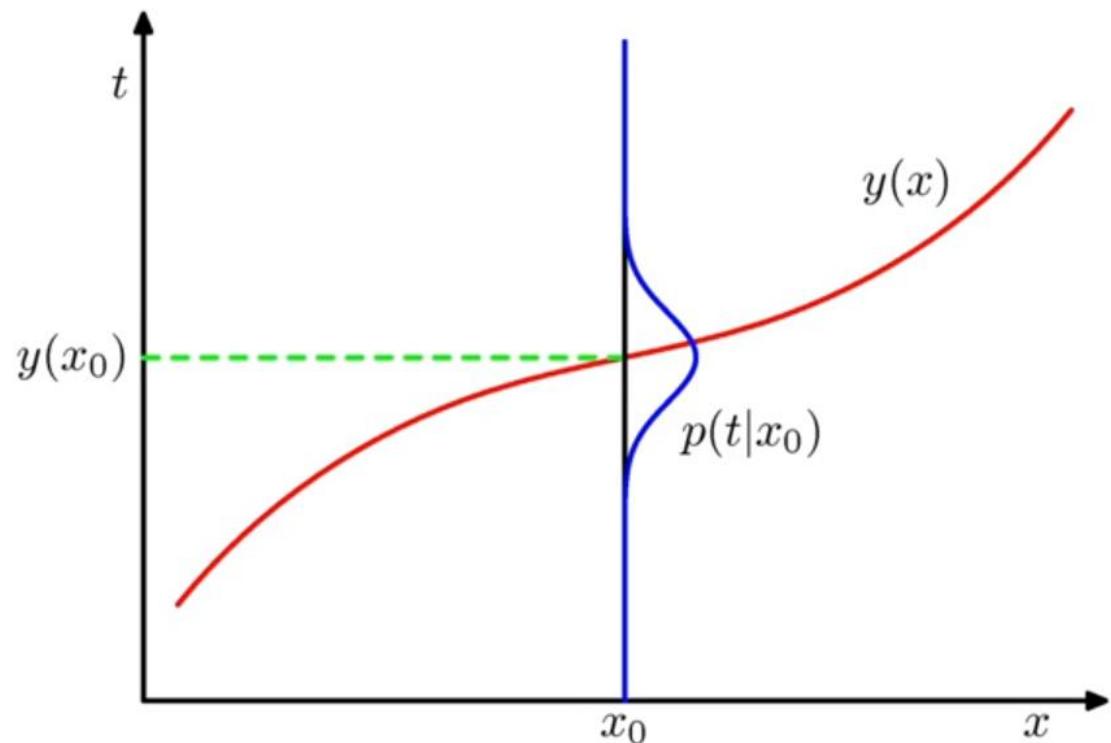


Figure 1.28
Bishop

3rd Order Polynomial ($M=3, N=10$)

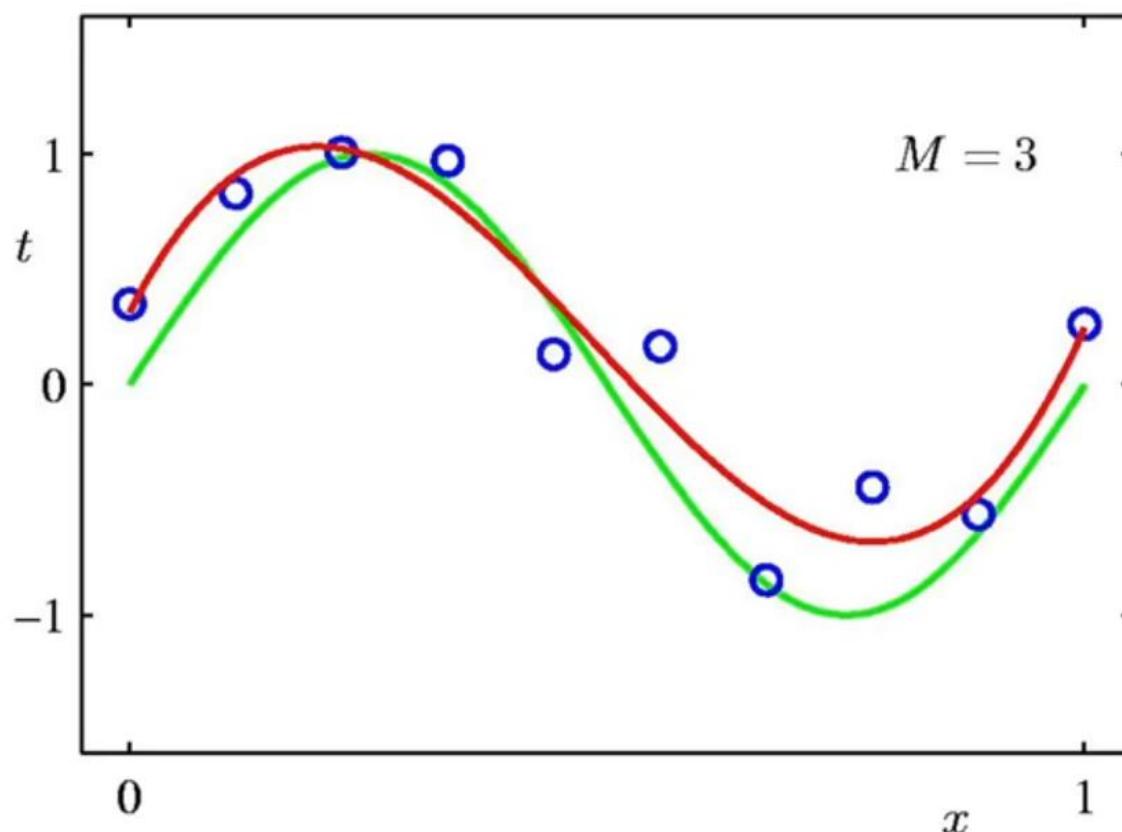
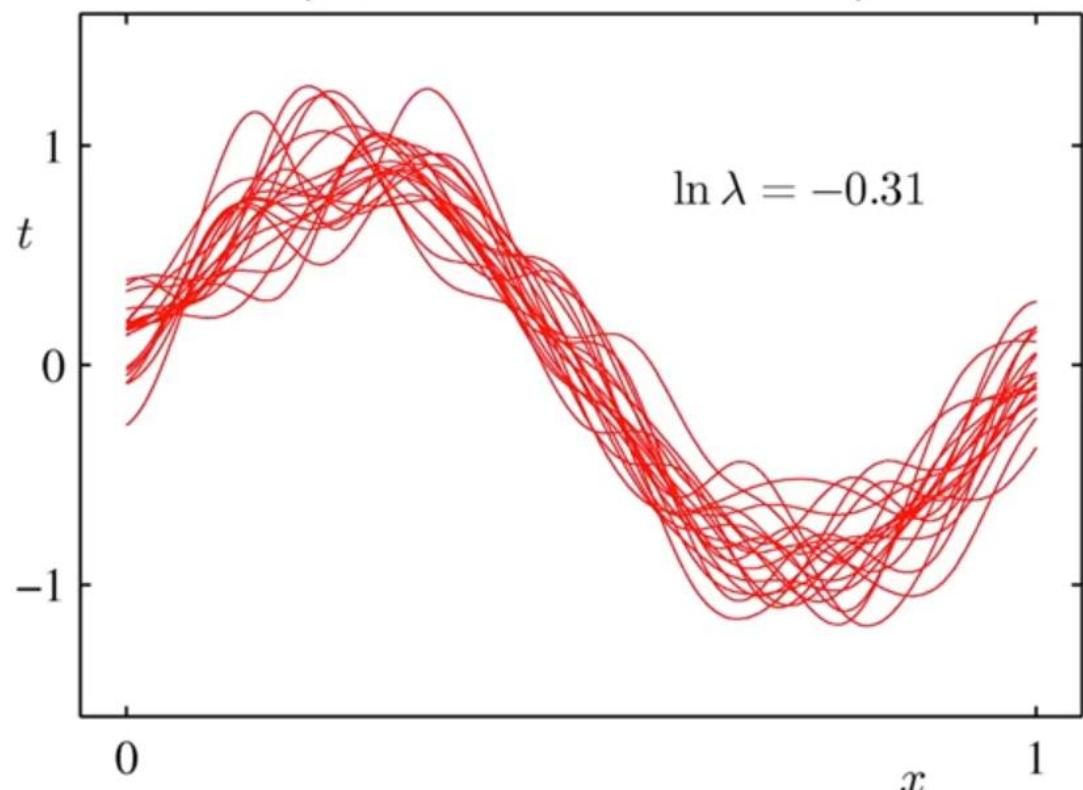


Figure 4.3

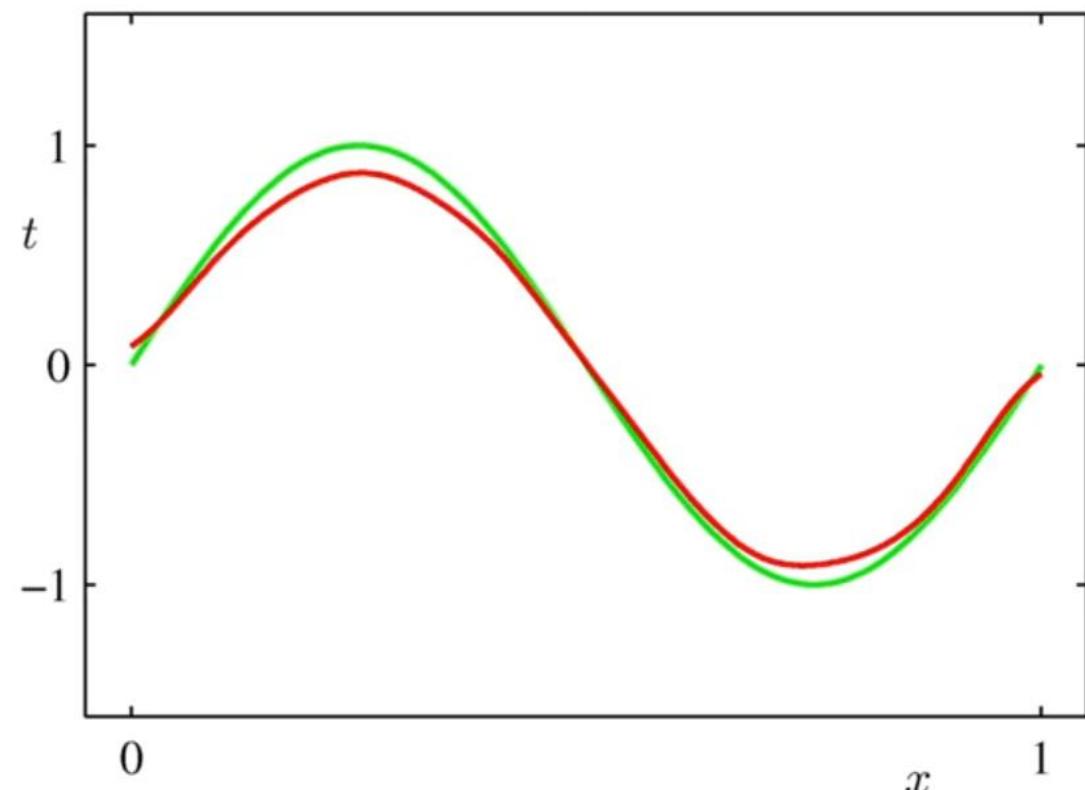
$$\lambda = e^{-0.31}$$

only 20 sets are shown for clarity



Medium variance

mean of the 100 $y(x)$ functions



Medium bias

Figure 3.5
Bishop

Regularizer

- Regularized data approximation problem

$$E_D = \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2 \quad (4.2)$$

$$E_W = \sum_k \|\mathbf{w}_k\|^p \quad (4.9)$$

Given $\mathbf{d}_k, \mathbf{x}_k, \lambda$, $E(\{\mathbf{w}_k\}) = E_D + \lambda E_W$
minimize

$$= \sum_k \left\| \sum_l \mathbf{w}_l \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) - \mathbf{d}_k \right\|^2 + \lambda \sum_k \|\mathbf{w}_k\|^p \quad (4.11)$$



Image from stock.adobe.com

P	Machine Learning	Neural Network
2	Ridge regression	Weight decay (encourage weights to decay to zero)
1	Lasso (least absolute shrinkage and selection operator)	

Robust Loss

- In (4.9) $p = 1$ and 2 are widely used. Any other choice? Barron proposes

$$f(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha} \left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1 \right)^{\alpha/2} - 1 \right) \quad (1) \text{ Barron}$$

$$\lim_{\alpha \rightarrow 2} f(x, \alpha, c) = \frac{1}{2} (x/c)^2 \quad \text{L2 loss} \quad (2) \text{ Barron}$$

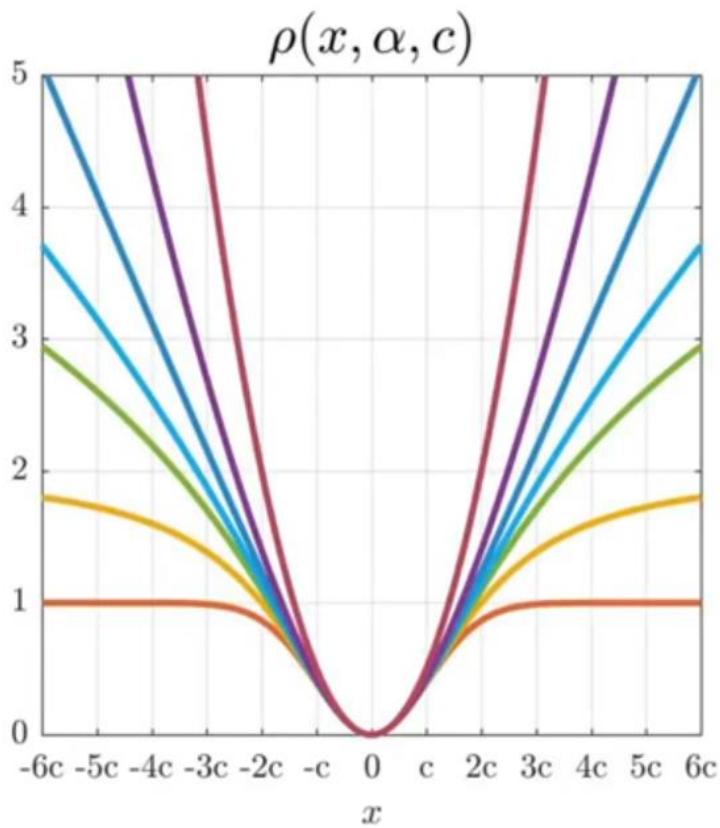
$$f(x, 1, c) = \sqrt{(x/c)^2 + 1} - 1 \quad \text{L1 loss} \quad (3) \text{ Barron}$$

$$\lim_{\alpha \rightarrow 0} f(x, \alpha, c) = \log \left(\frac{1}{2} (x/c)^2 + 1 \right) \quad \text{Cauchy loss} \quad (5) \text{ Barron}$$

$$f(x, -2, c) = \frac{2 (x/c)^2}{(x/c)^2 + 4} \quad \text{Geman-McClure loss} \quad (6) \text{ Barron}$$

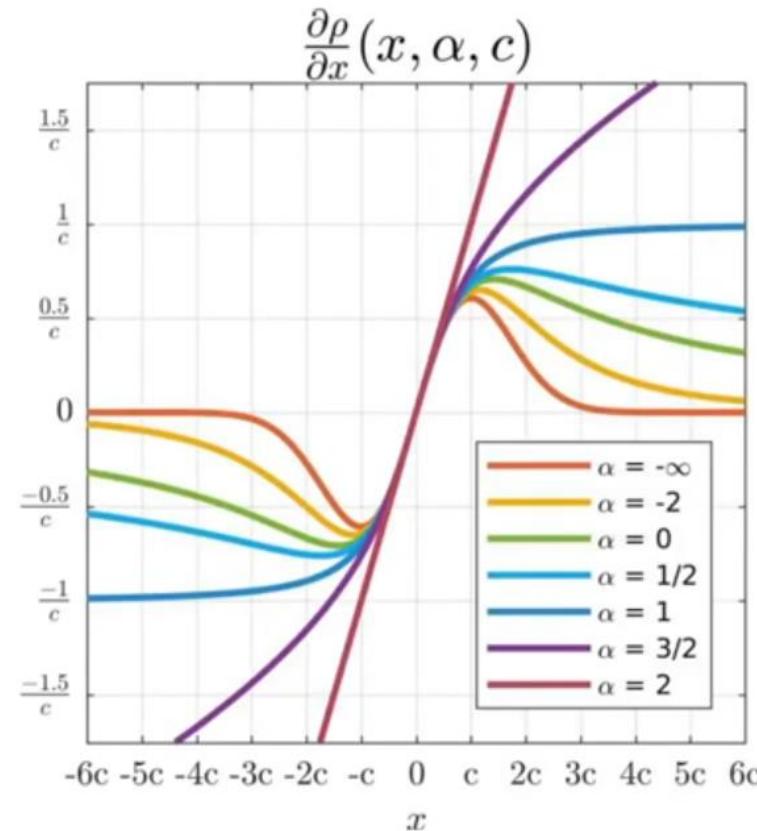
$$\lim_{\alpha \rightarrow -\infty} f(x, \alpha, c) = 1 - \exp \left(-\frac{1}{2} (x/c)^2 \right) \quad \text{Welsch loss} \quad (7) \text{ Barron}$$

BARRON, Jonathan. A General and Adaptive Robust Loss Function. CVPR 2019.



$$\rho(x, \alpha, c) = \begin{cases} \frac{1}{2} \left(\frac{x}{c}\right)^2 & \text{if } \alpha = 2 \\ \log\left(\frac{1}{2} \left(\frac{x}{c}\right)^2 + 1\right) & \text{if } \alpha = 0 \\ 1 - \exp\left(-\frac{1}{2} \left(\frac{x}{c}\right)^2\right) & \text{if } \alpha = -\infty \\ \frac{|\alpha-2|}{\alpha} \left(\left(\frac{(x/c)^2}{|\alpha-2|} + 1\right)^{\alpha/2} - 1\right) & \text{otherwise} \end{cases}$$

(8) Barron



$$\frac{\partial \rho}{\partial x}(x, \alpha, c) = \begin{cases} \frac{x}{c^2} & \text{if } \alpha = 2 \\ \frac{2x}{x^2+2c^2} & \text{if } \alpha = 0 \\ \frac{x}{c^2} \exp\left(-\frac{1}{2} \left(\frac{x}{c}\right)^2\right) & \text{if } \alpha = -\infty \\ \frac{x}{c^2} \left(\left(\frac{(x/c)^2}{|\alpha-2|} + 1\right)^{(\alpha/2-1)}\right) & \text{otherwise} \end{cases}$$

(9) Barron

Figure 4.7⁵⁷

Comparison with Student's t-distribution

	Parameters	Normal (Gaussian)	Cauchy
Student's t-distribution	μ, λ, ν	$\nu \rightarrow \infty$	$\nu = 1$
Barron's distribution	μ, α, c	$\alpha = 2$	$\alpha = 0$

Barron's experiment: create a variational autoencoder

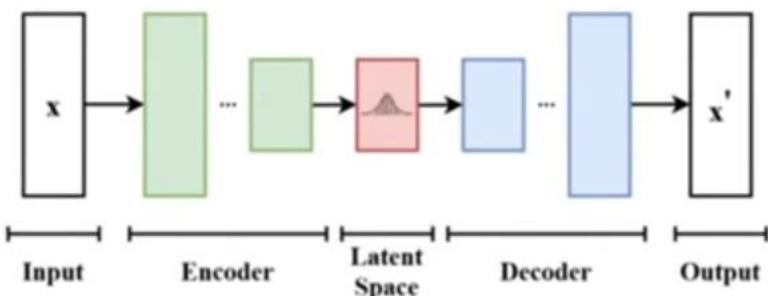


Figure from wikipedia

Large-scale CelebFaces Attributes (CelebA) Dataset



Image from mmlab.ie.cuhk.edu.hk

10,177 number of identities,
202,599 number of face images, and
5 landmark locations,
40 binary attributes annotations per image.

Results

Table 1
Barron

	Normal	Cauchy	t-dist.	Ours
Pixels + RGB	8,662	9,602	10,177	10,240
DCT + YUV	31,837	31,295	32,804	32,806
Wavelets + YUV	31,505	35,779	36,373	36,316

Validation set evidence lower bound (ELBO),
higher is better.



Figure 3
Barron
60

Colorization task

Scattered data interpolation problem



(a)

Input grayscale image, and
color scribbles



(b)

Output result



(c)

Ground truth image

Figure 4.10

YUV Color Space

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

$Y \in [0, 255]$
 $U \in [0, 255]$
 $V \in [0, 255]$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -0.00093 & 1.401687 \\ 1 & -0.3437 & -0.71417 \\ 1 & 1.77216 & 0.00099 \end{bmatrix} \begin{bmatrix} Y \\ U - 128 \\ V - 128 \end{bmatrix}$$

Equations from wikipedia 

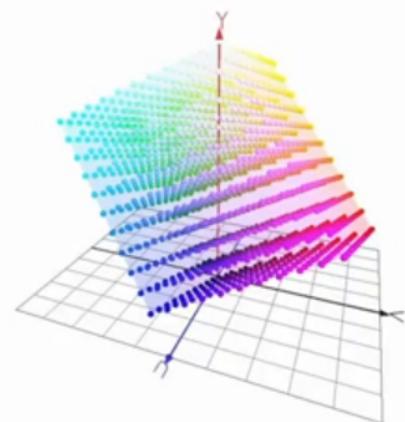


image from Philippe. "Color Space Transformations." (2006). 62

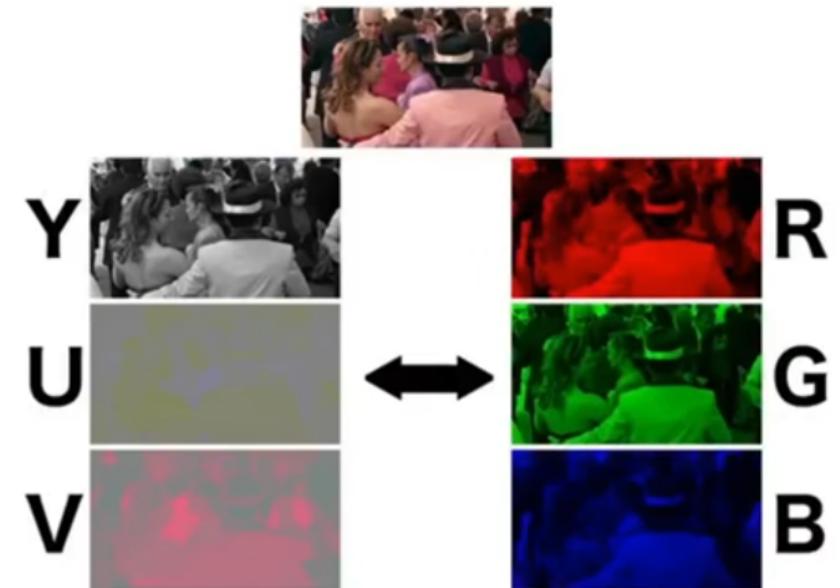


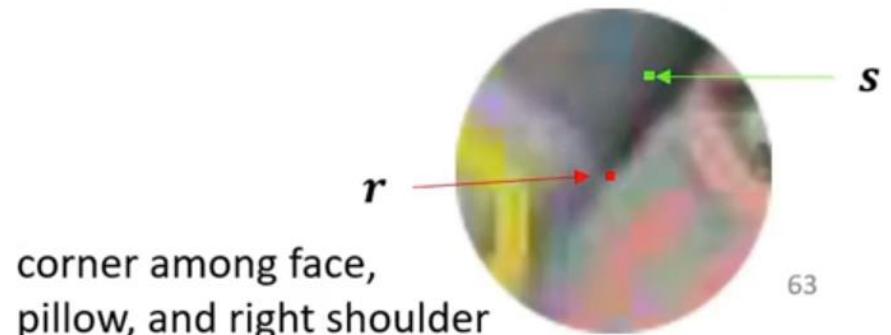
image from www.retrosix.wiki

Colorization using Optimization (Levin et al. ACM TOG 2004)

- Let the input grayscale image be Y , and the color images be U, V .
- Let \mathbf{r}, \mathbf{s} denote pixels on the image.
- The given scribbles assign some values of U, V .
- The remaining U value are assigned by minimizing

$$J(U) = \sum_{\mathbf{r}} \left(U(\mathbf{r}) - \sum_{\mathbf{s} \in N(\mathbf{r})} w_{rs} U(\mathbf{s}) \right)^2 \quad \bullet \quad (1) \text{ Levin}$$

$N(\mathbf{r})$ means a small neighborhood of \mathbf{r}
 w_{rs} is a weight coefficient



Weight Coefficient

- $Y(\mathbf{r})$ and $Y(\mathbf{s})$ are similar, $U(\mathbf{r})$ and $U(\mathbf{s})$ should be similar in terms of their
 - raw values

$$w_{\mathbf{rs}} \propto e^{-(Y(\mathbf{r}) - Y(\mathbf{s}))^2 / 2\sigma_r^2} \quad (2) \text{ Levin}$$

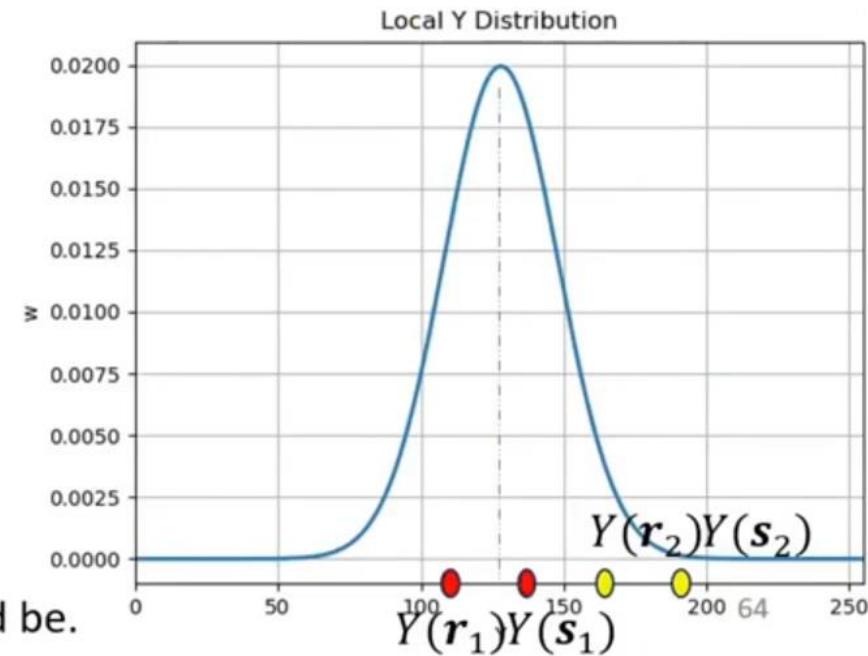
- and normalized values

$$w_{\mathbf{rs}} \propto 1 + \frac{1}{\sigma_r^2} (Y(\mathbf{r}) - \mu_r)(Y(\mathbf{s}) - \mu_r) \quad (3) \text{ Levin}$$

Consider this case: their difference is the same

$$|Y(\mathbf{r}_1) - Y(\mathbf{s}_1)| = |Y(\mathbf{r}_2) - Y(\mathbf{s}_2)|,$$

but $Y(\mathbf{r}_1)$ $Y(\mathbf{s}_1)$ are on the opposite side of the distribution and $Y(\mathbf{r}_2)$ $Y(\mathbf{s}_2)$ are on the same side. $U(\mathbf{r}_2)$ $U(\mathbf{s}_2)$ should be more similar. The further they are from the μ_r , the more similar they should be.



Optimization

- View the variables of the U image as a very long vector \mathbf{x} , and the U values of those scribbles as a given vector \mathbf{b} .

$$J(U) = \sum_{\mathbf{r}} \left(U(\mathbf{r}) - \sum_{\mathbf{s} \in N(\mathbf{r})} w_{\mathbf{rs}} U(\mathbf{s}) \right)^2 \quad (1) \text{ Levin}$$

becomes $\underset{\mathbf{x}}{\operatorname{argmin}} (\mathbf{Ax} - \mathbf{y})^2 \quad (4.28)$

where \mathbf{A} is a sparse matrix calculated from Y using (2)(3) Levin.

Additional Results

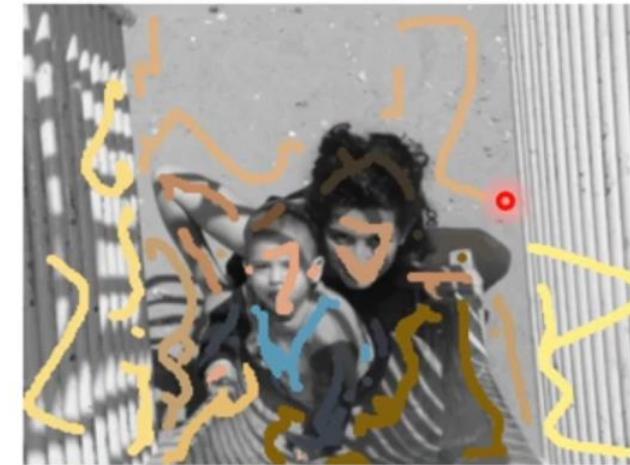
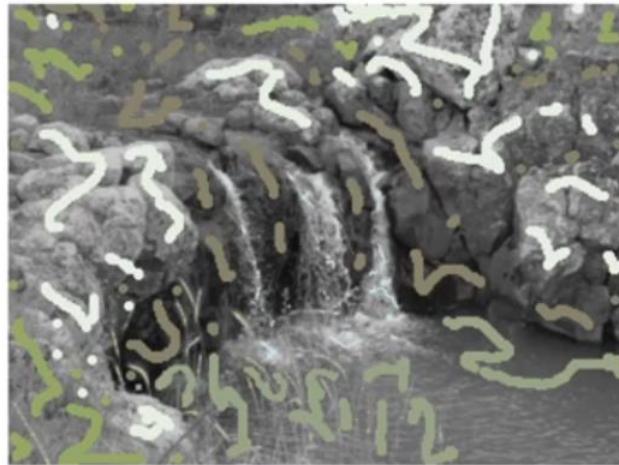


Figure 2
Levin
66

Interactive colorization

- (a1) initial scribbles
- (a2) artifacts
- (b1) additional scribbles
- (b2) result
- (c1) additional scribbles
- (c2) artifacts

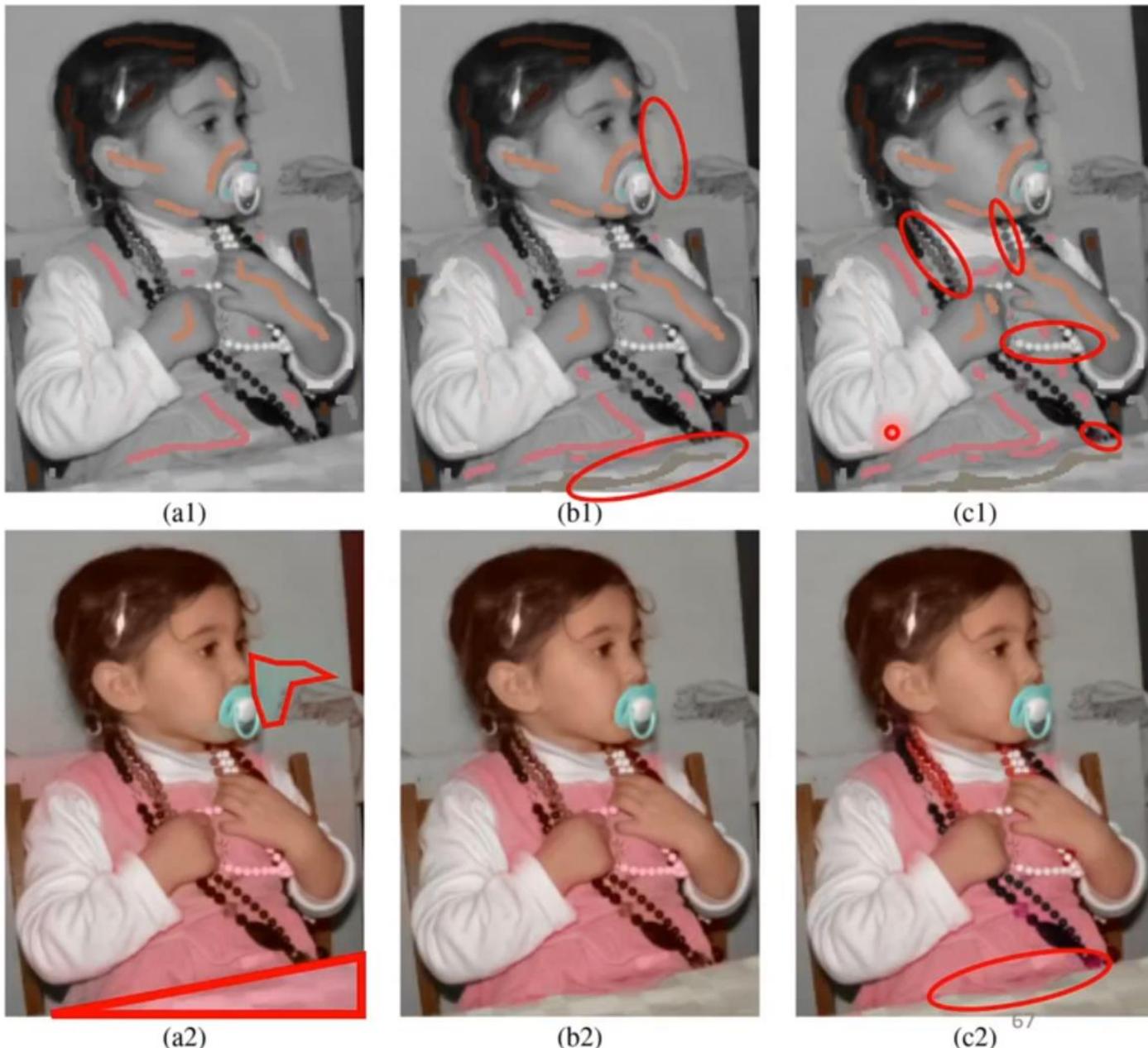
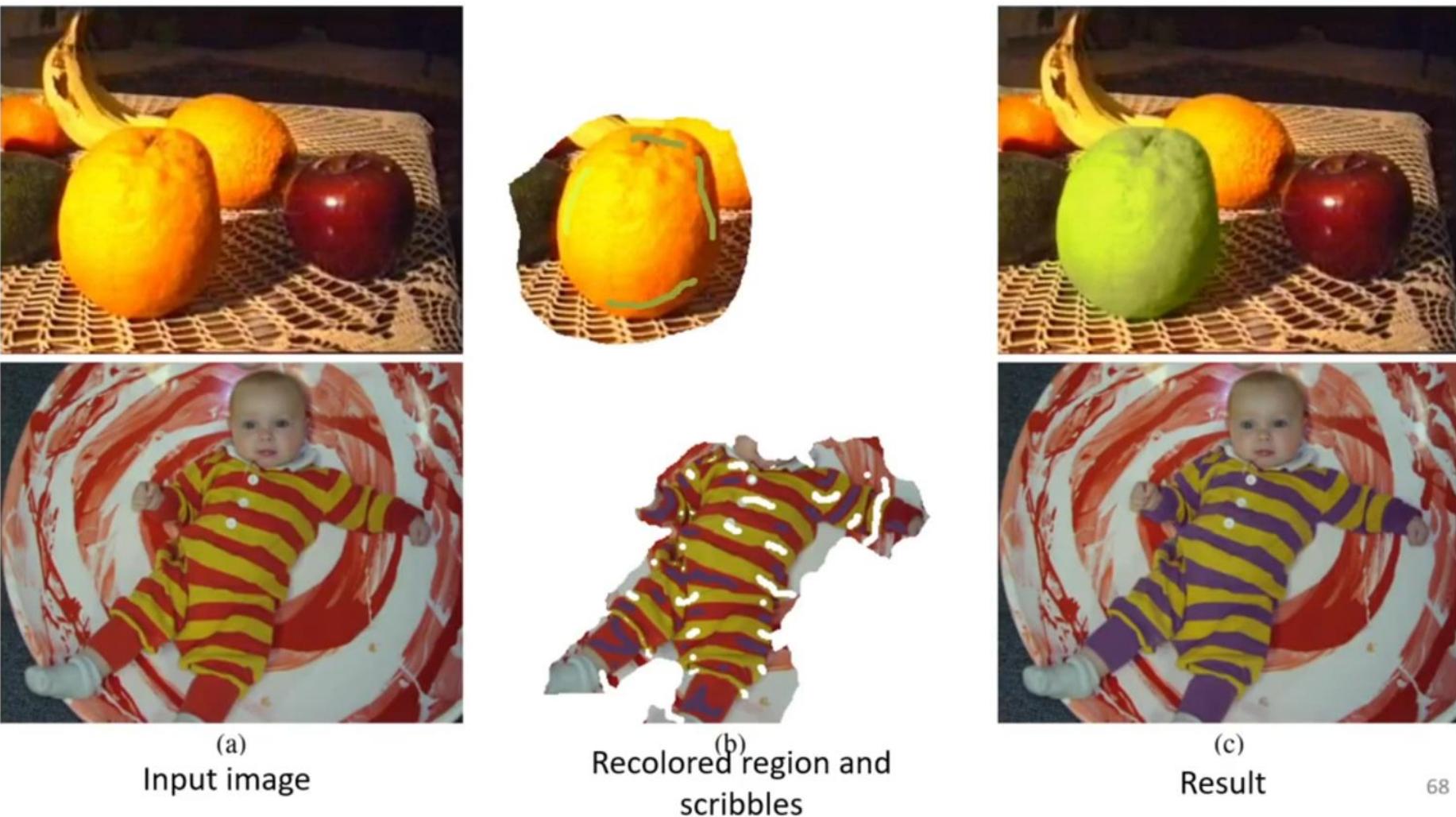


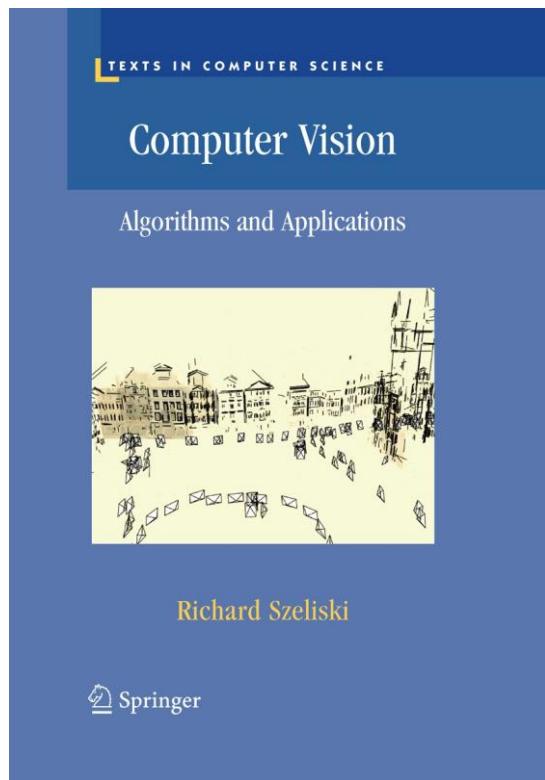
Figure 3
Levin

Recolor



Questions?

- Reference



Pattern Recognition and Machine Learning by Christopher M. Bishop (2006)

- Full free PDF book downloadable from the author's official website ([Link](#))

