# Computer Vision I

## Dense Stereo Correspondences

Anita Sellent

# Stereo

- Two Cameras
  - Overlapping field of view
  - Known transformation between cameras
  - From disparity compute depth



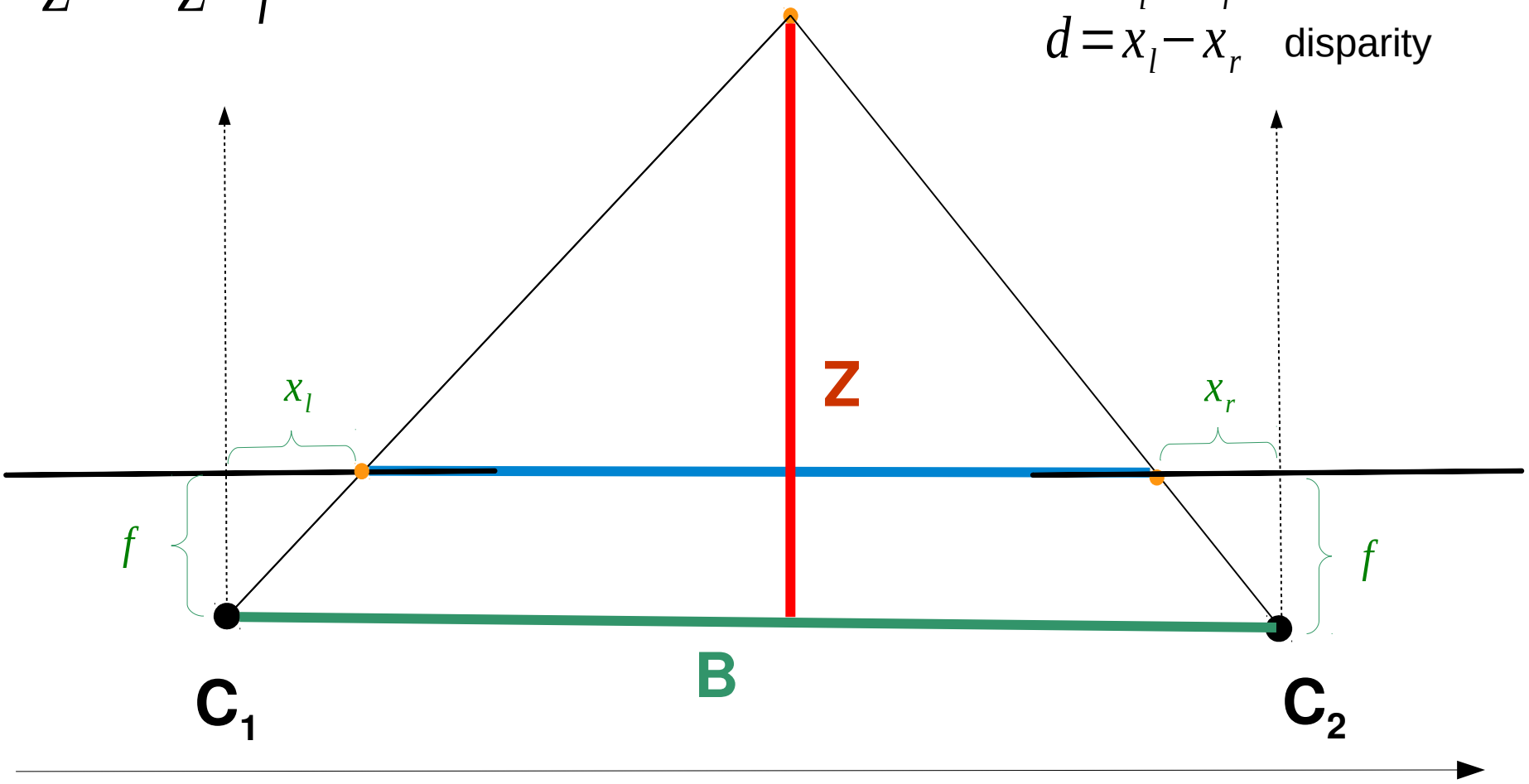
[ Bradski, Kaehler: Learning OpenCV, O'Reilly 2008]

# Triangulation: Top View

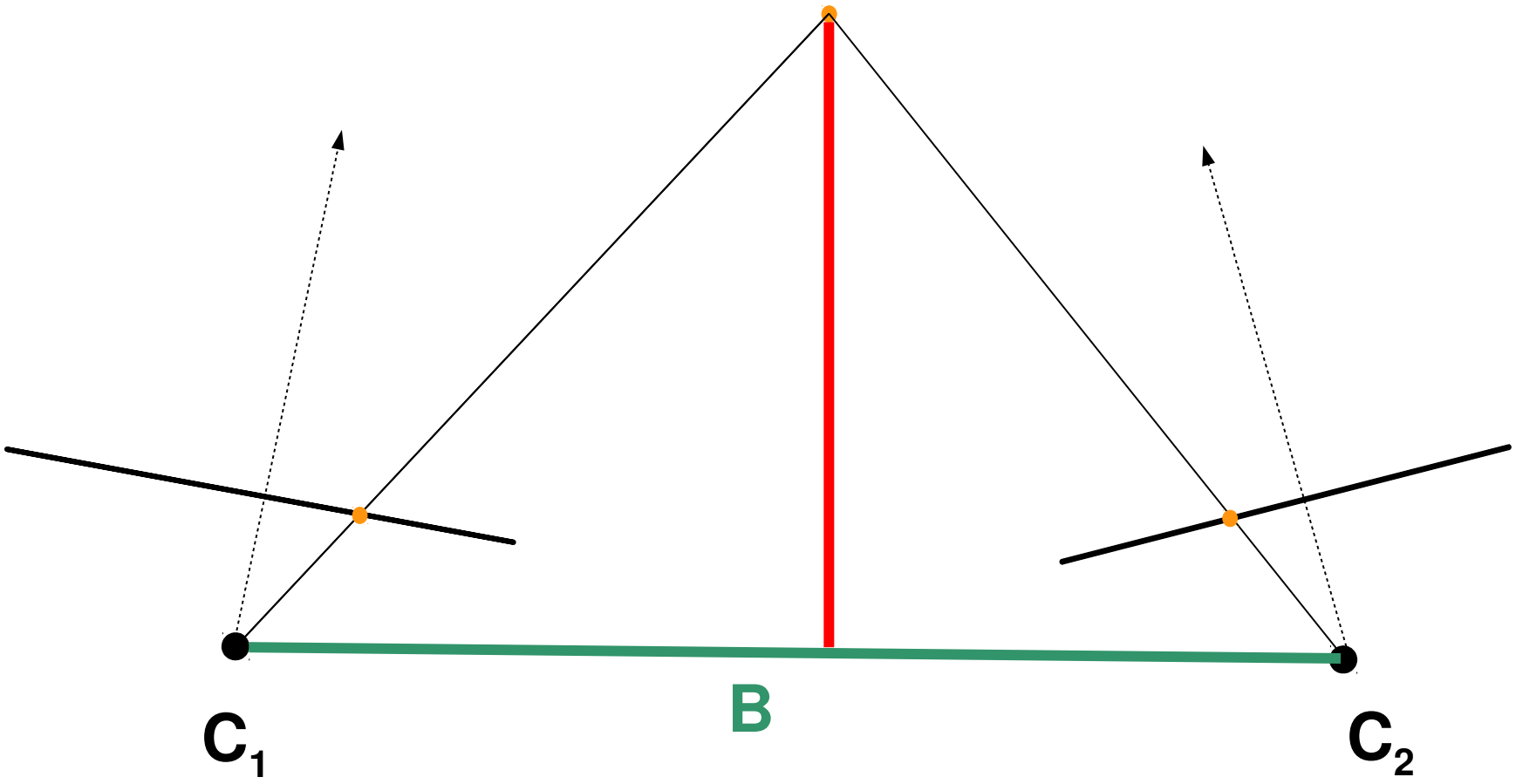
$$\frac{B}{Z} = \frac{B - x_l + x_r}{Z - f}$$

$$Z = \frac{fB}{x_l - x_r} = \frac{f\,B}{d}$$

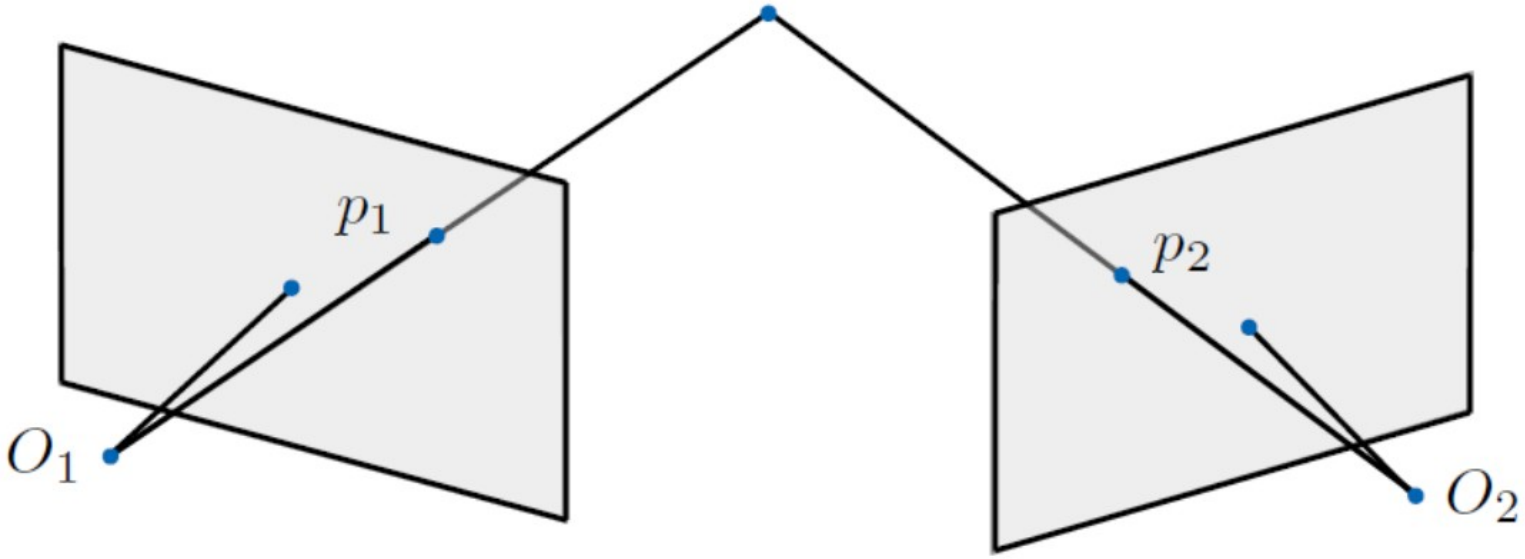
$$d = x_l - x_r \quad \text{disparity}$$



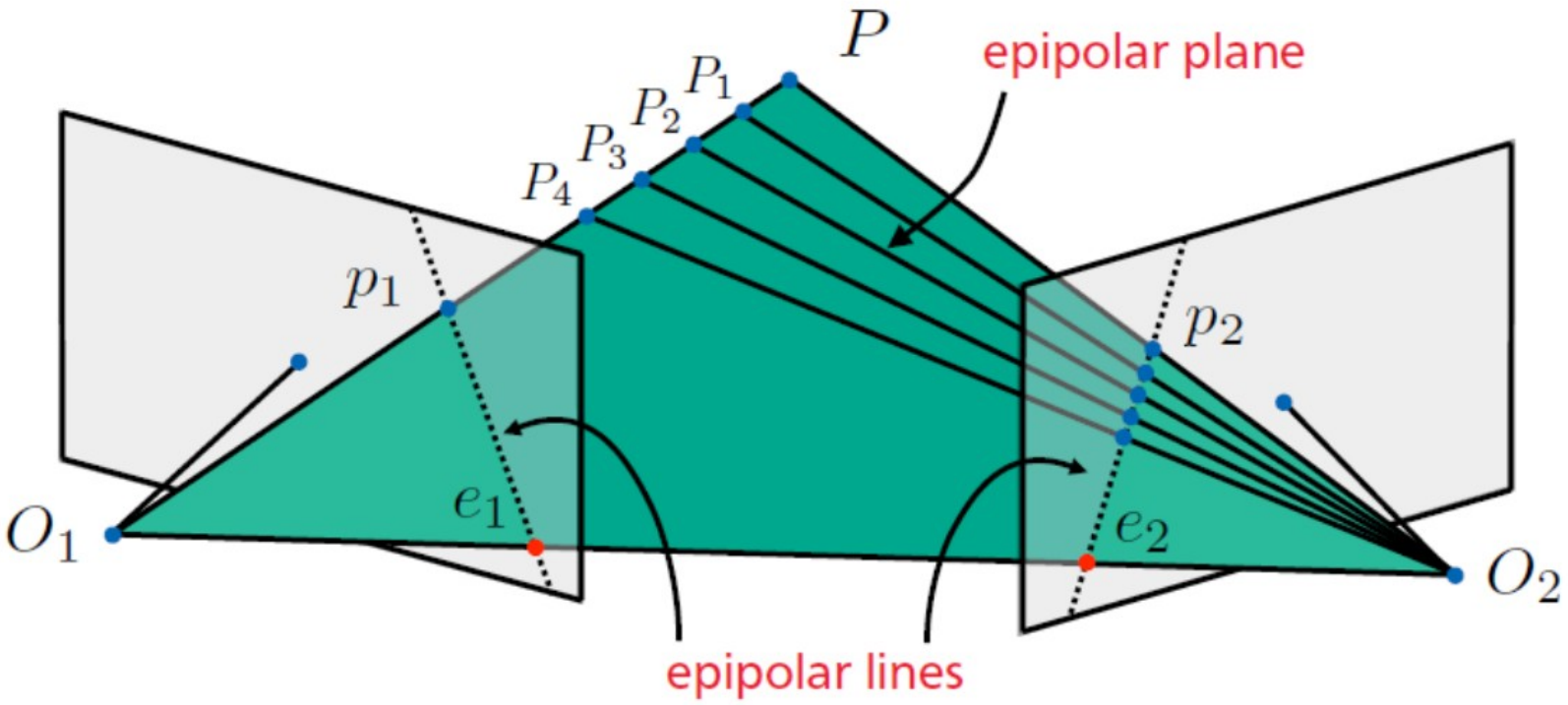
# Triangulation: Top View



# Epipolar Lines

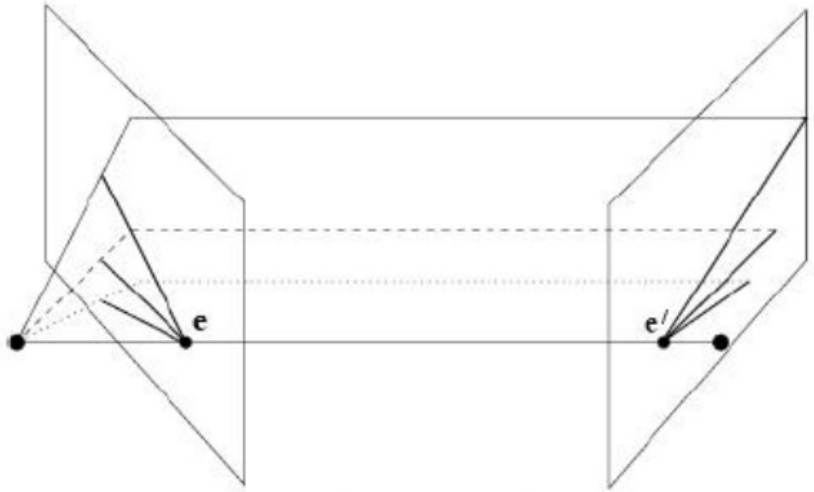


# Epipolar Lines

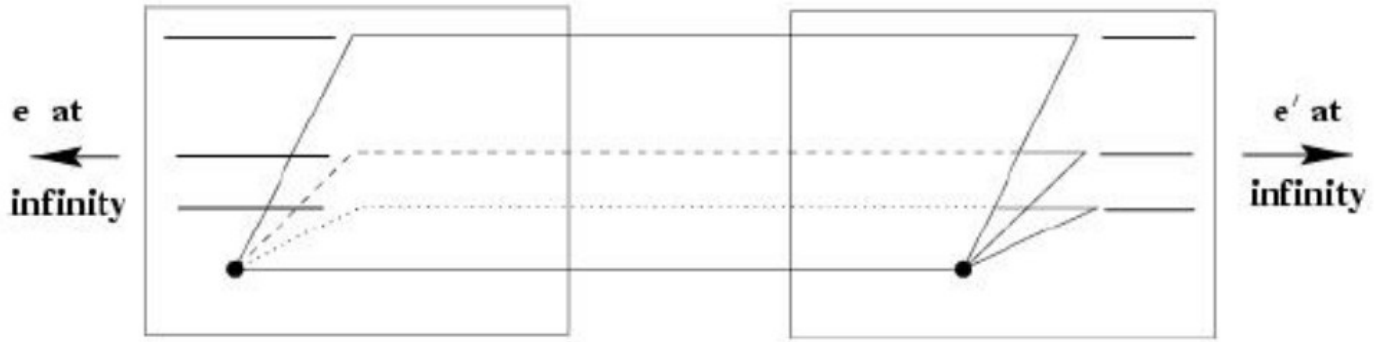



- Epipolar line: Constrains the location where a particular point (here $p_1$) from one view can be found in the other.

# Epipolar Lines

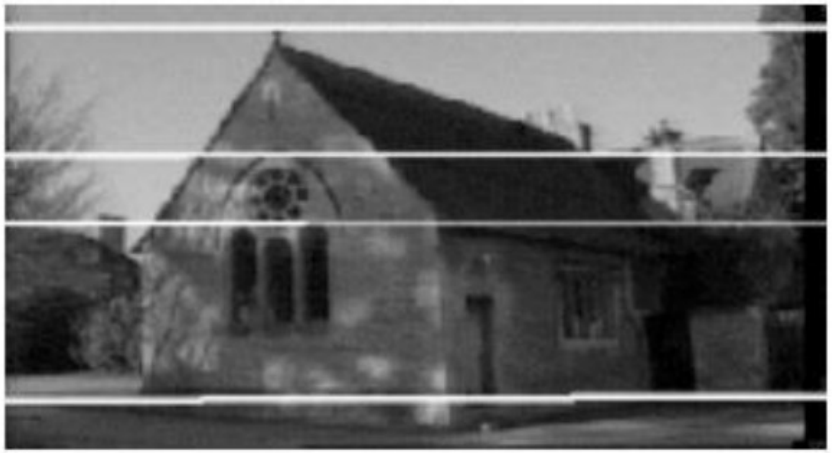


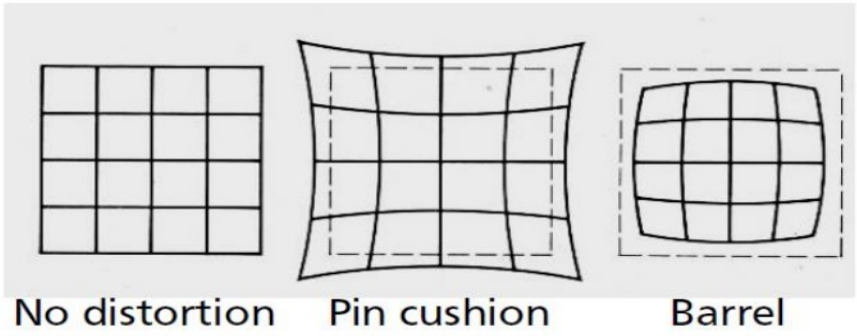

# Epipolar Lines




# Practically

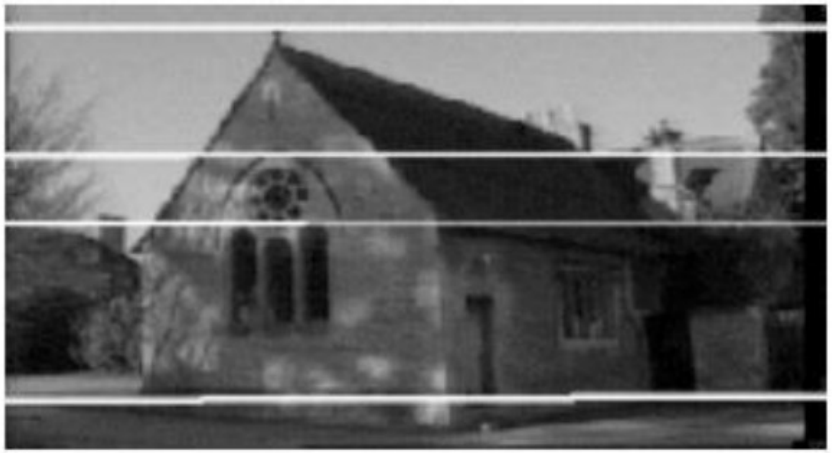
- Cameras are never perfectly aligned.
- The lenses of the cameras are never perfect.



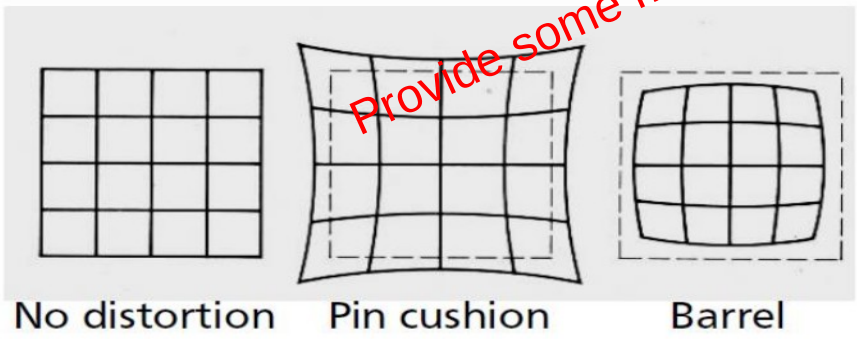



# Practically

- Cameras are never perfectly aligned.
- The lenses of the cameras are never perfect.



Provide some mathematical help!




# Practically

- Image undistortion
  - Remove radial and tangential lens distortion
- Stereo calibration
  - Compute the geometrical relationships between cameras in space
- Stereo rectification
  - Transform individual images so that they appear as if the had seem taken by two row aligned image planes

# Practically

- Image undistortion
  - Remove radial and tangential lens distortion

$$x = g_1(\hat{x}, \hat{y}) = \hat{x}(1 + K_1 r^2 + K_2 r^4 + \ldots) + (P_2(r^2 + 2\hat{x}^2) + 2P_1 \hat{x}\hat{y})(1 + K_1 r^2 + K_2 r^4 + \ldots)$$

$$y = g_2(\hat{x}, \hat{y})$$

# Practically

- Image undistortion
  - Remove radial and tangential lens distortion

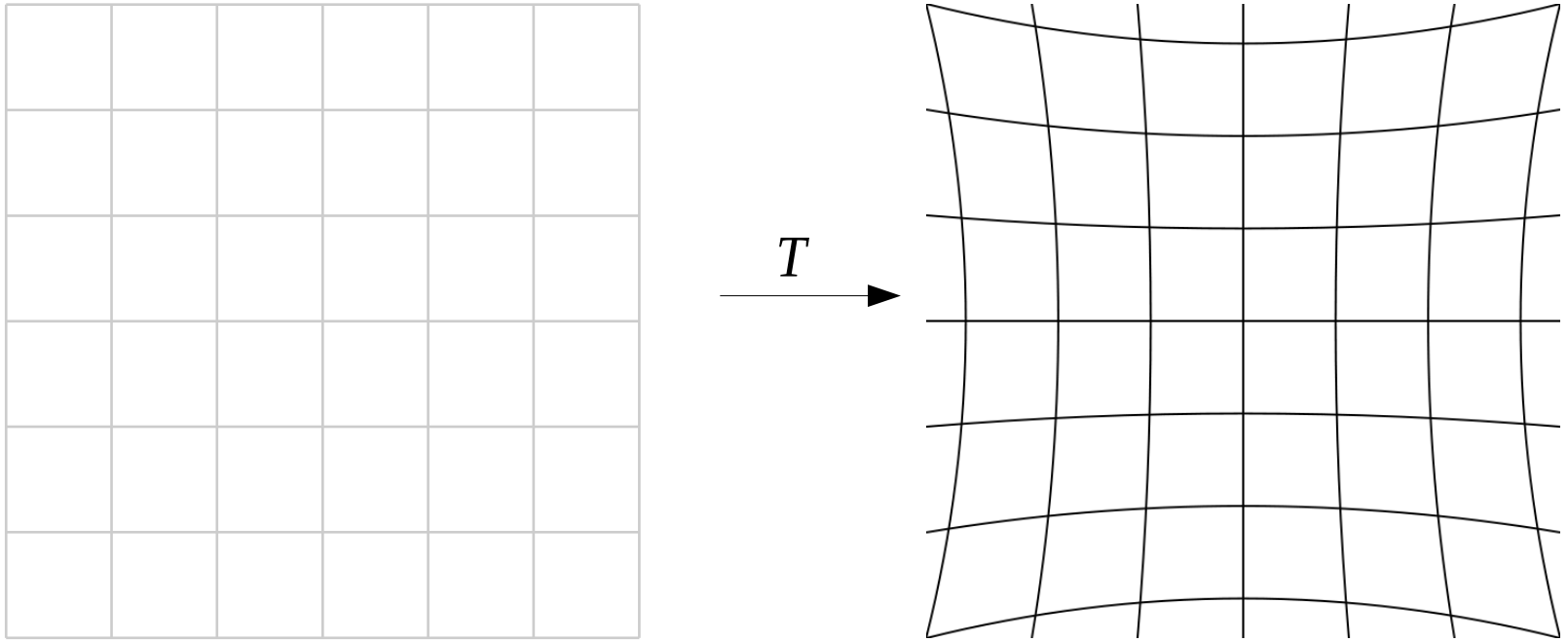
$$s = g_1(x, y) = x\left(1 + K_1 r^2 + K_2 r^4 + \ldots\right) + \left(P_2\left(r^2 + 2x^2\right) + 2P_1 x y\right)\left(1 + K_1 r^2 + K_2 r^4 + \ldots\right)$$

$$t = g_2(x, y)$$

Undistorted position

Distorted position

# Geometric Image Transformations



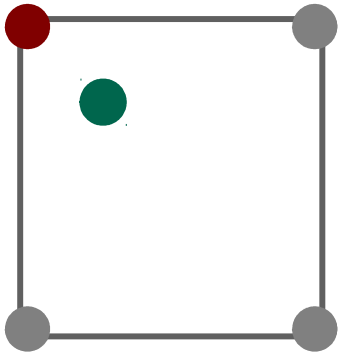

Coordinates: $(x, y)$ $T(x, y)$

Image: $I(s, t)$

# Image Interpolation Schemes

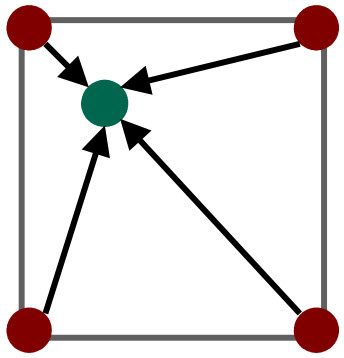
- Nearest Neighbor Approach

$(\hat{x}, \hat{y})$



- Bilinear Interpolation

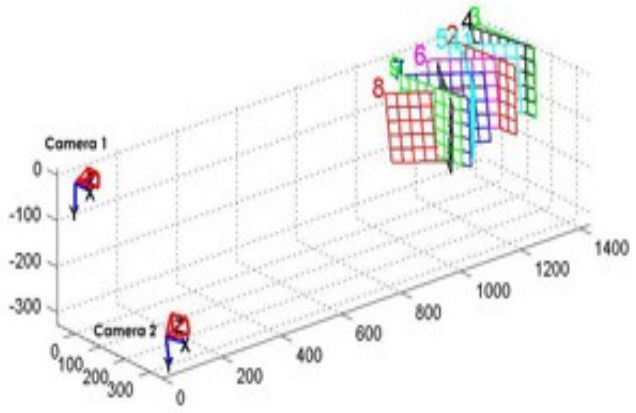
$(x, y)$

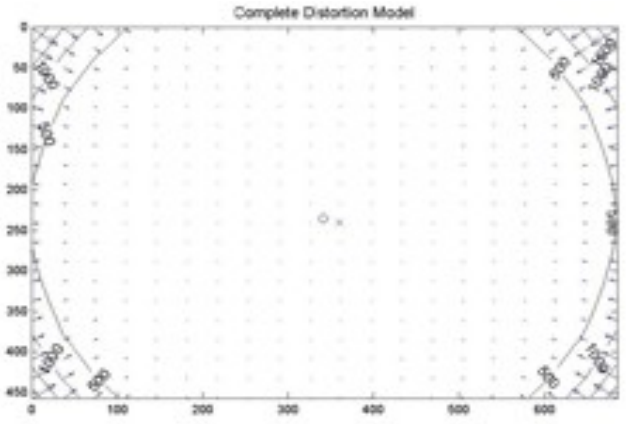


# Practically

- Image undistortion
- Stereo calibration
- Stereo rectification

Planar
Strong features



# A Stereo Vision System

# A Stereo Vision System

Calibration
(offline)

Rectification

Stereo Correspondences

Triangulation

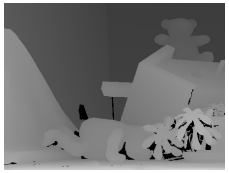
PC, FPGA

Intrinsic and extrinsic parameter

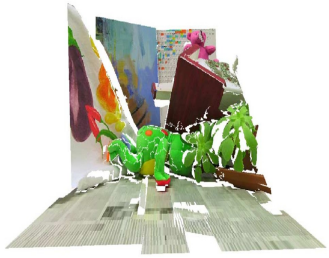
Stereo pair

Rectified Stereo pair

Disparity map

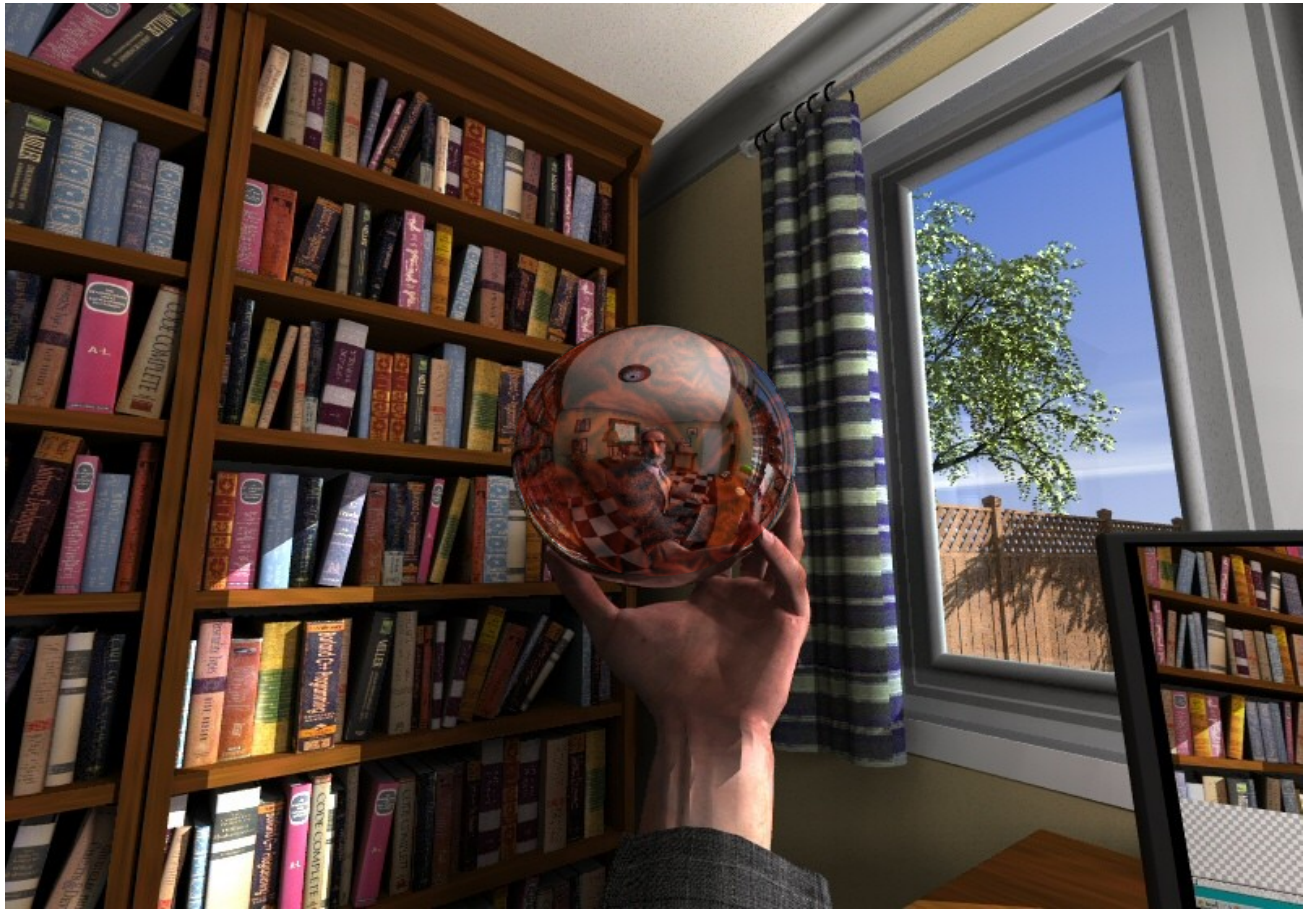


Depth map

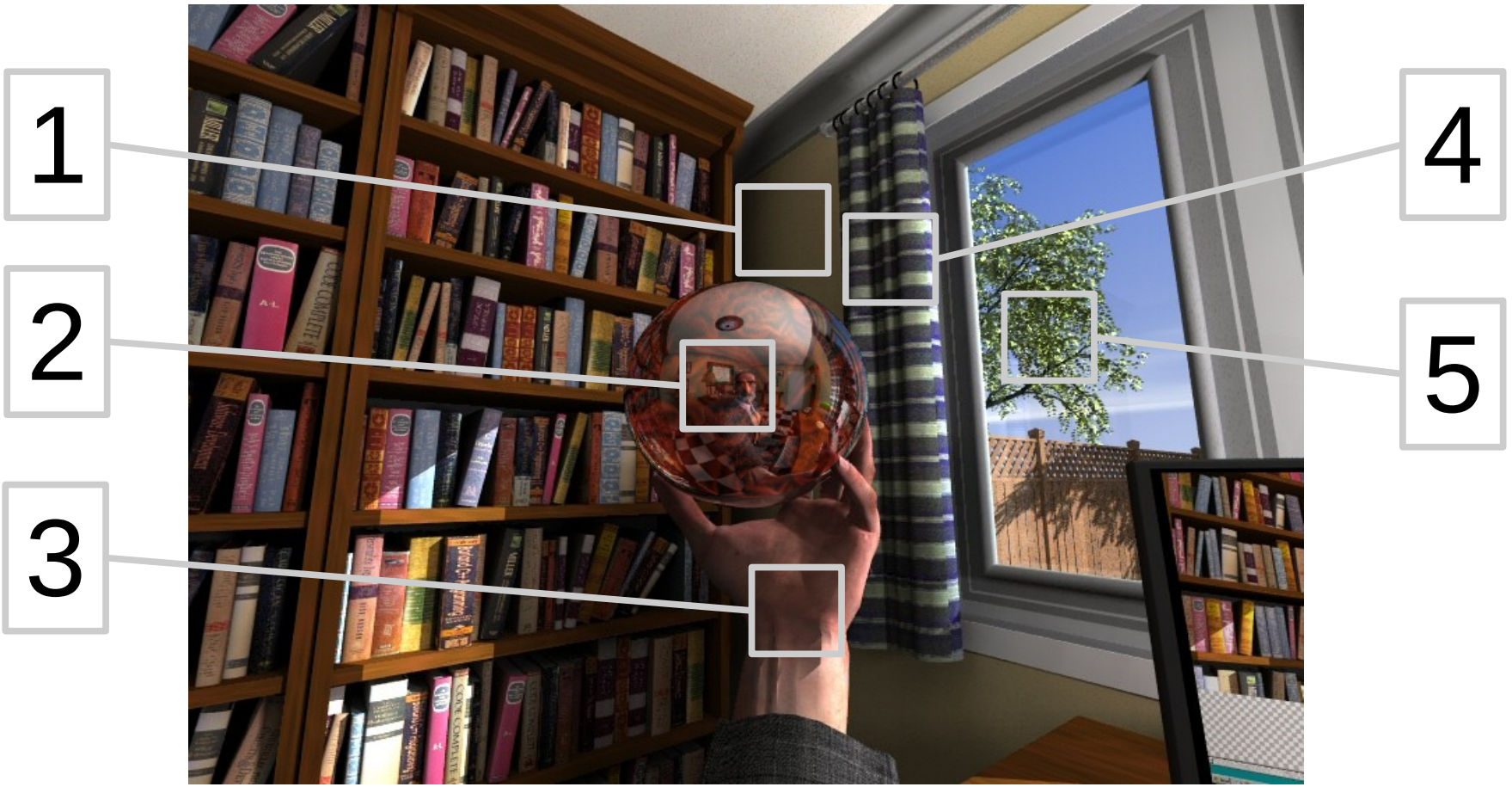


# Disparity Estimation: Assumption I

- The color of a pixel does not change with the viewpoint



# Quiz: Which Points Satisfy Color Constancy?




6 : None of the above

# Disparity Estimation: Assumption I

- The color of a pixel does not change with the viewpoint

Noise
Reflective Material
Transparent Material
Scattering Effects

Pixels that are visible only in one camera

# Disparity Estimation: Assumption I

- The color of a pixel does not change with the viewpoint

Noise
Reflective Material
Transparent Material
Scattering Effects

Pixels that are visible only in one camera

# Disparity Estimation: Assumption I

- The color of a pixel does not change with the viewpoint

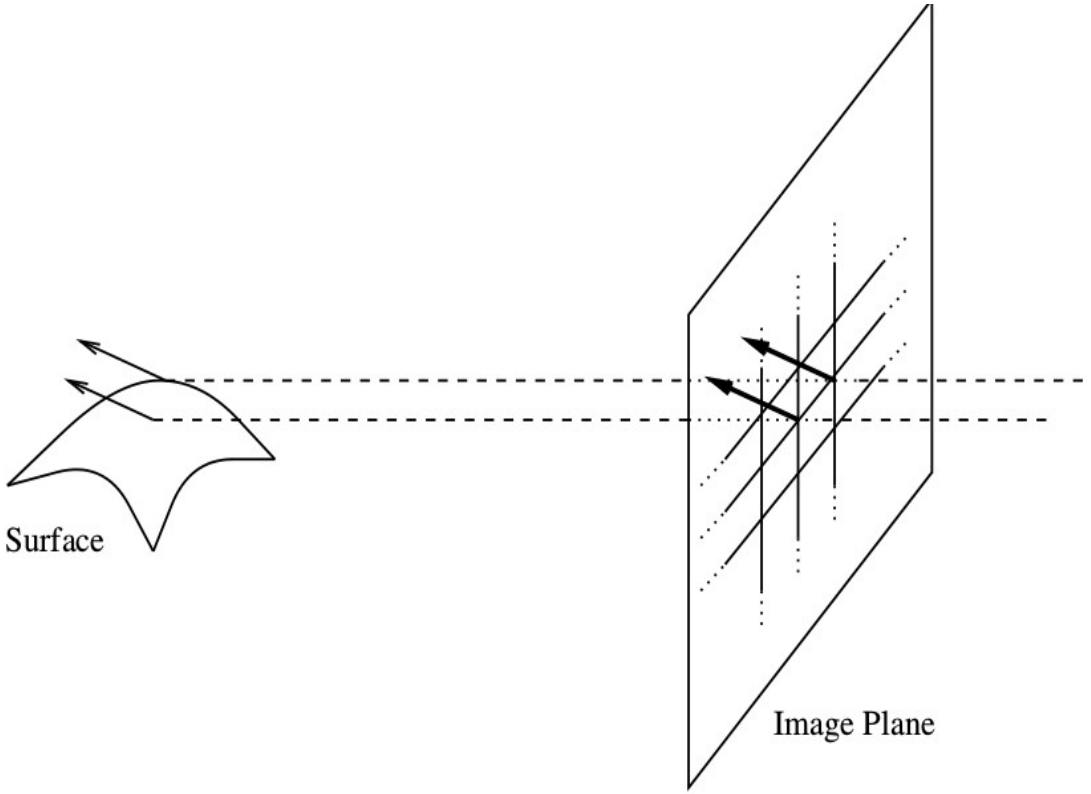
$$I_l(x,y) \approx I_r(x-d,y)$$

# Disparity Estimation: Assumption I

- There are 256 different gray values in an image.
- This image has 450 pixels per line.

# Disparity Estimation: Assumption II

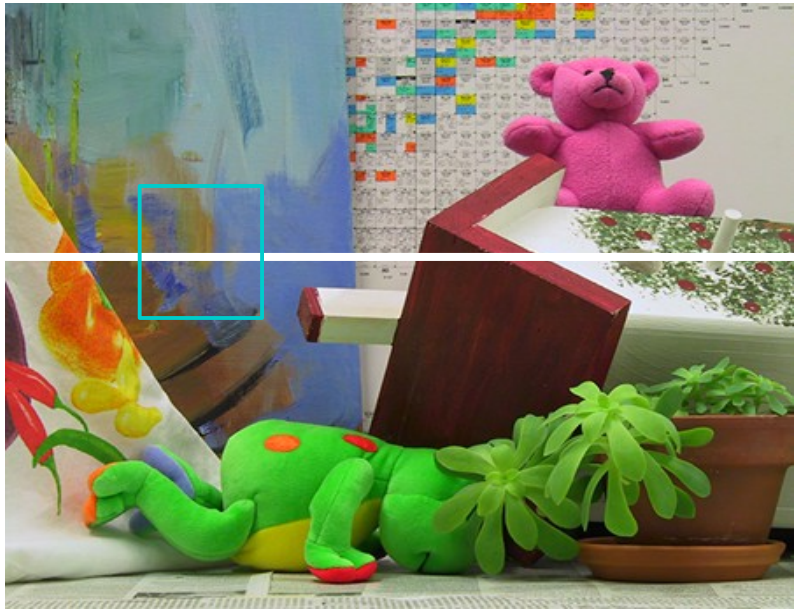
- Neighboring pixels belong to the same surface




$$d(x,y) \approx d(x+\epsilon_x, y+\epsilon_y)$$

# Disparity Estimation: Assumption II

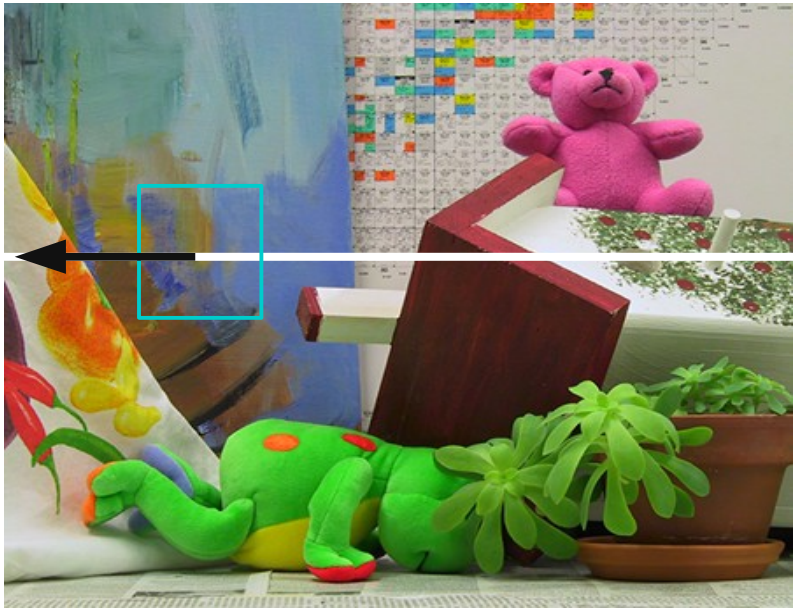
- Neighboring pixels belong to the same surface
- Neighboring pixels belong to a fronto-paralell plane



$$d(x,y)=d(x+\Delta x, y+\Delta y)$$

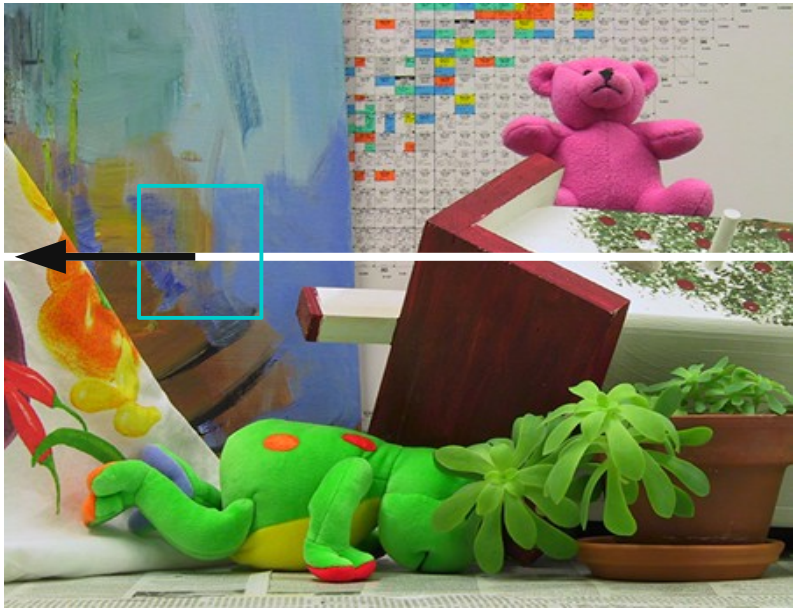
# A Basic Stereo Algorithm: Blockmatching

- For each pixel
  - For each disparity value
    - Compare patch similarity
    - Assign disparity with highest similarity



# A Basic Stereo Algorithm: Blockmatching

- For each pixel
  - For each disparity value
    - Compare patch similarity
    - Assign disparity with highest similarity

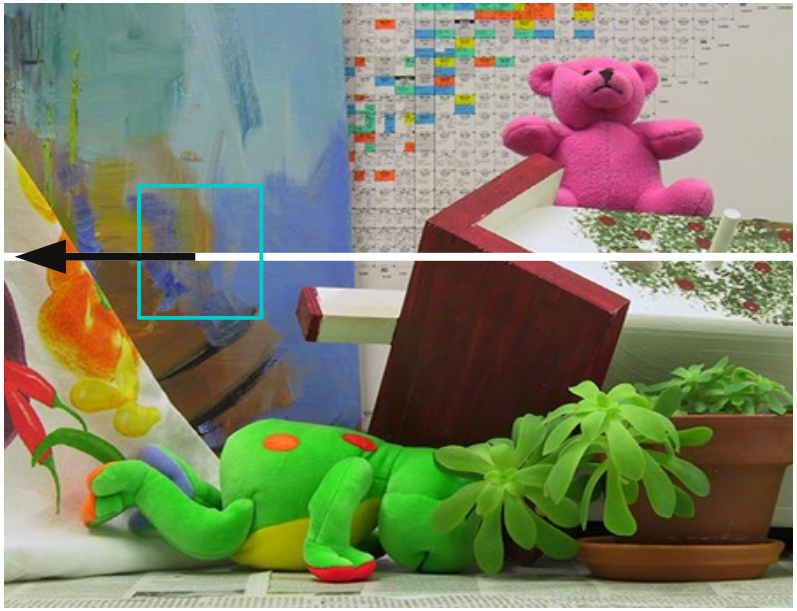


# Sparse vs. Dense Depth Estimation II

- Sparse Feature Matches (e.g. Harris corners )
  - Highly distinguishable texture
  - Region descriptor robust to scale, rotation, ...
- Dense Correspondences
  - For every pixel
  - Search along epipolar line

# A Basic Stereo Algorithm: Blockmatching

- For each pixel
  - For each disparity value
    - Compare patch similarity
    - Assign disparity with highest similarity

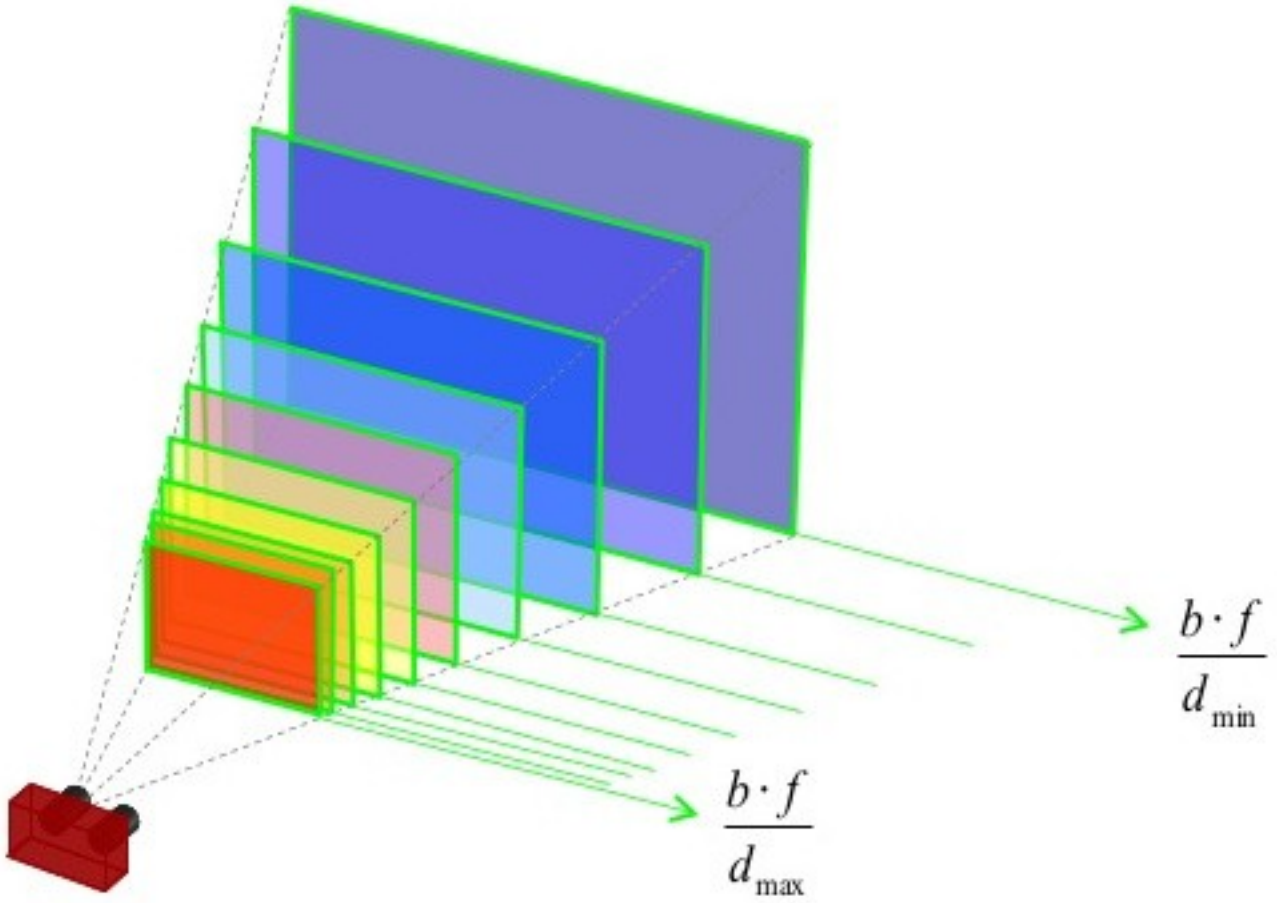


# Discretization of the Solution Space

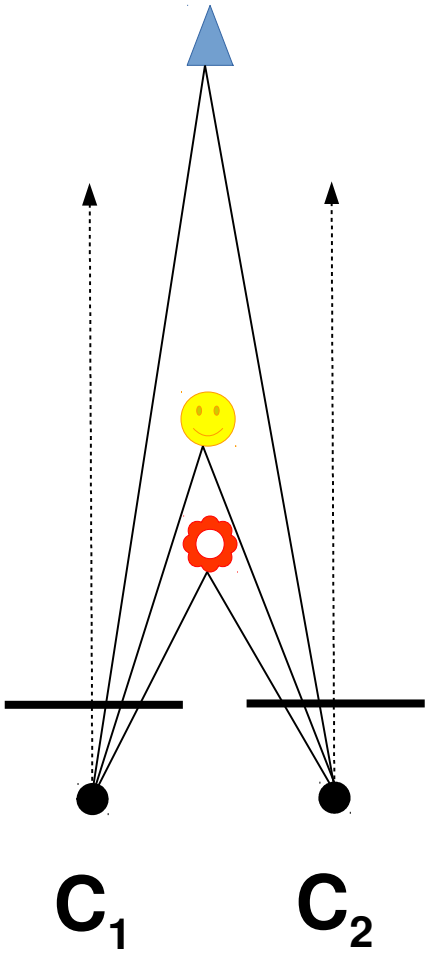
- Pros:
  - Can use fast optimization algorithms
  - Restrict need for image interpolation
  - More flexibility in similarity measures
- Cons:
  - Only certain distances become measurable

# Discretization of the Solution Space

- Assume limited range of discrete disparities

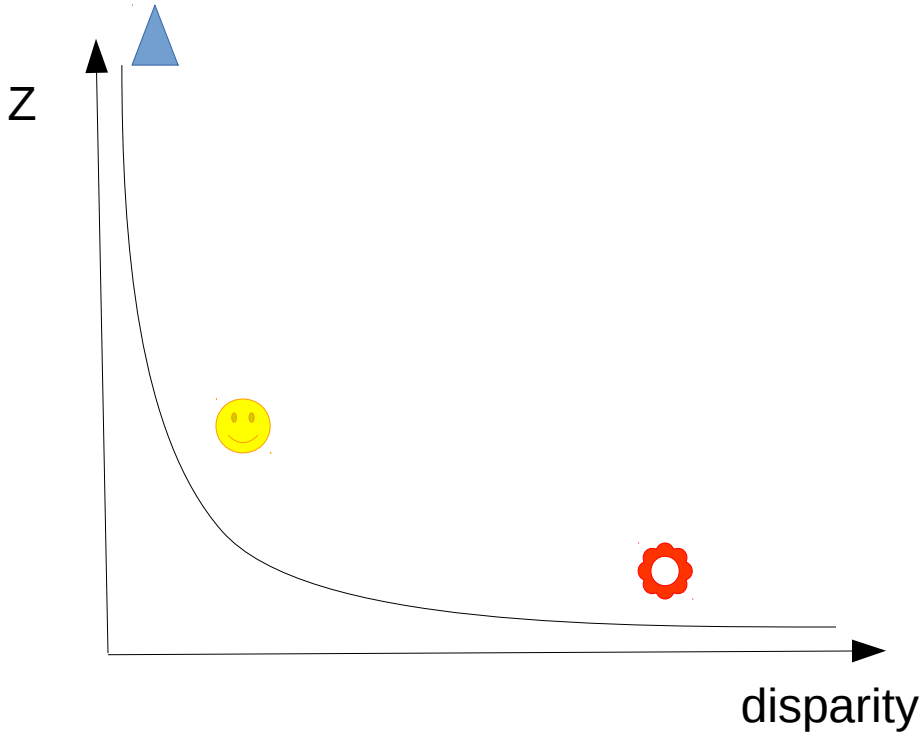


# Disparity




$$Z = \frac{fB}{x_l - x_r} = \frac{f\,B}{d}$$

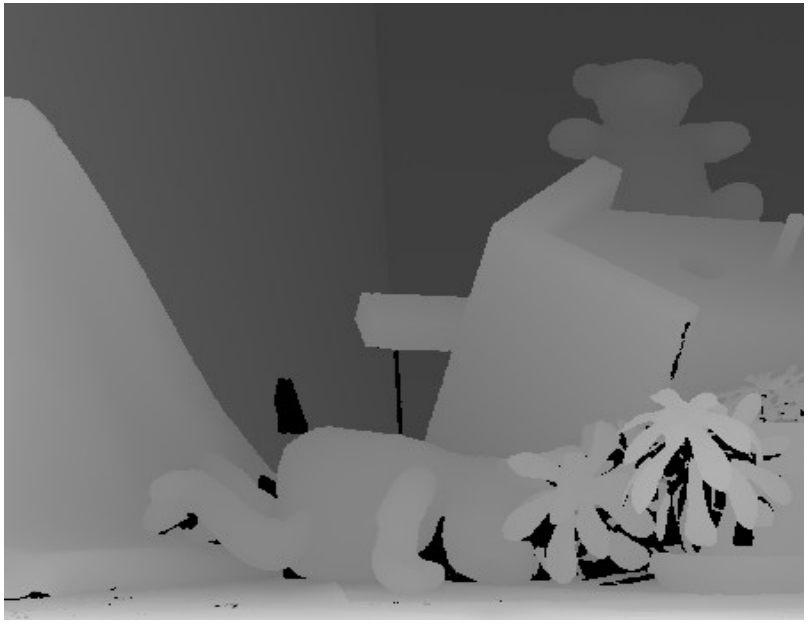
$$d = x_l - x_r \quad \text{disparity}$$



# Discretization of the Solution Space

- Assume limited range of discrete disparities

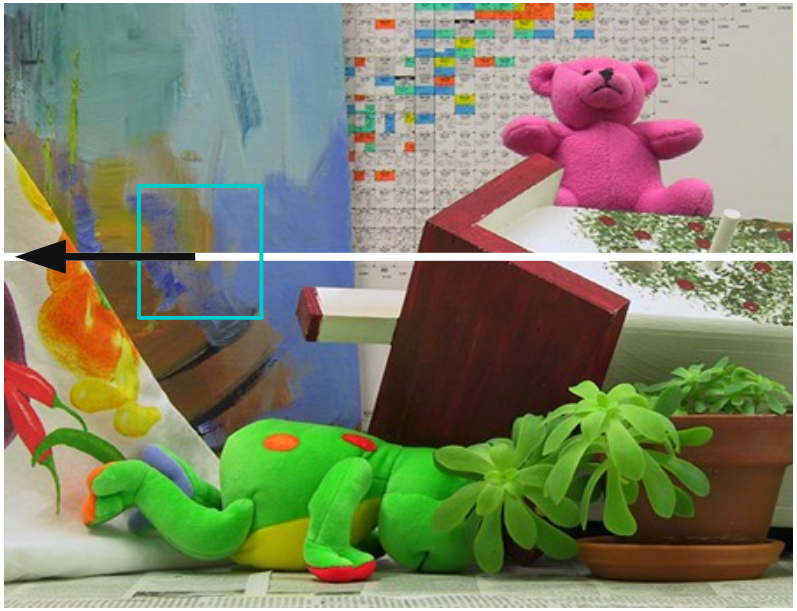
Discrete estimates



Ground truth

# A Basic Stereo Algorithm: Blockmatching

- For each pixel
  - For each disparity value
    - Compare patch similarity
    - Assign disparity with highest similarity



# What are good Similarity Measures?

- Sum of Squared differences ( on a window )
- Sum of Absolute differences ( on a window )
- Any robust function ( on a window )
- Any robust function on a brightness dependent window
- Cross Correlation (Normalized )
- Census Transform with Hamming distance
- ...

[ http://vision.middlebury.edu/stereo/ ]

# Image Noise

- Photon/ Shot noise
  - Photons arrive at the sensor at random times
- Dark noise
  - Thermal activity leads to random signals
- Read noise
  - Charge conversion introduces random error

$$\hat{I}(x,y) = I(x,y) + n(x,y)$$

$$n \sim N(0,\sigma)$$

# Gaussian Image Noise

- Noise for each pixel in each camera is independent

$$\hat{I}_l(x,y) = I_l(x,y) + n_l(x,y) \quad \hat{I}_r(x,y) = I_r(x,y) + n_r(x,y)$$

- Difference of images with disparity

$$\hat{I}_l(x,y) - \hat{I}_r(x-d,y) = I_l(x,y) - I_r(x-d,y) + n(x,y)$$

- Probability

$$Pr(I_l, I_r | \boldsymbol{d}, \theta) = \prod_{(x,y)\in N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-\left(I_l(\boldsymbol{x}) - I_r(\boldsymbol{x}-\boldsymbol{d})\right)^2}{2\sigma^2}\right)$$

- Negative logarithm

$$E = -\log\left(Pr(I_l, I_r | \boldsymbol{d}, \theta)\right) = \sum_{\boldsymbol{x}\in N} \frac{1}{2\sigma^2}\left(I_l(\boldsymbol{x}) - I_r(\boldsymbol{x}-\boldsymbol{d})\right)^2 + C$$

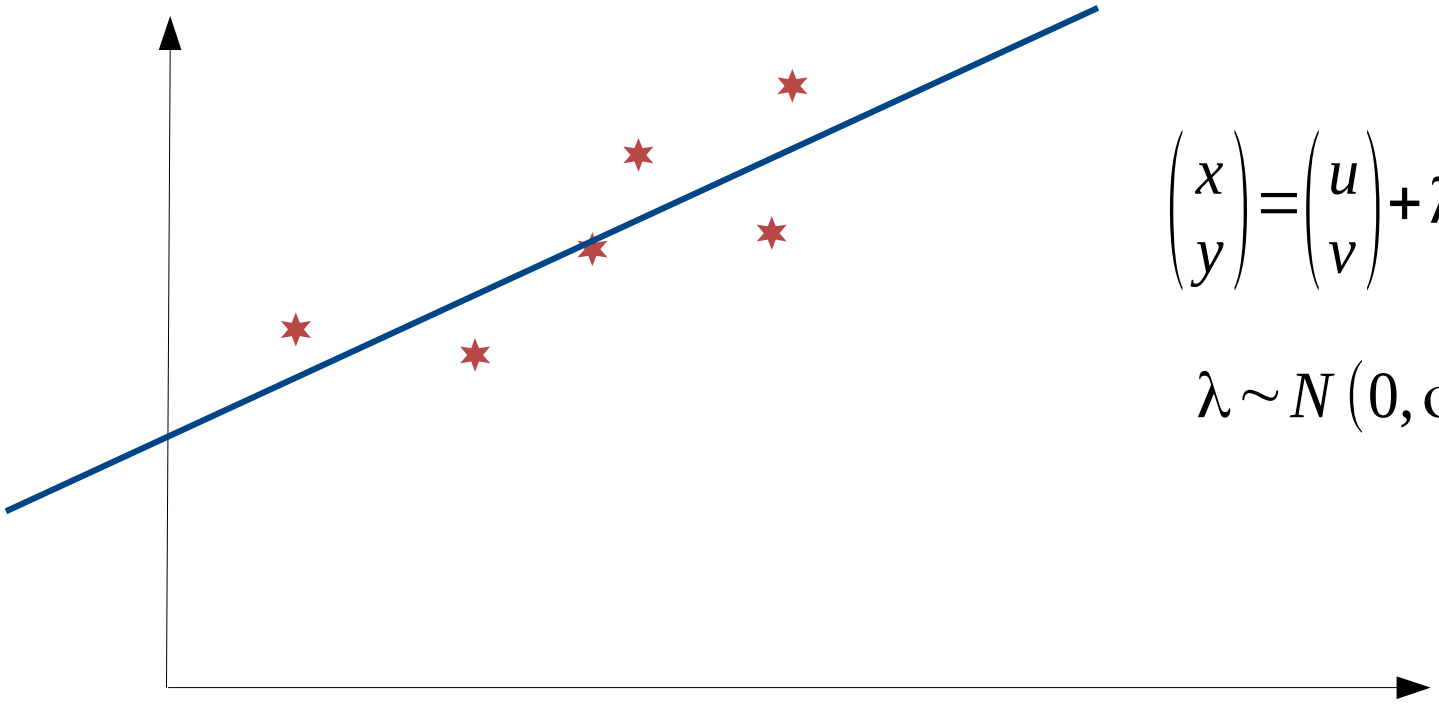
# Example: Line Fitting

- Given: a collection of data-points $\{(x_i, y_i)\}_{i=1\ldots N}$
- Task: fit a line to the data-points

[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Example: Line Fitting

- Given: a collection of data-points $\{(x_i, y_i)\}_{i=1\ldots N}$
- Task: fit a line to the data-points



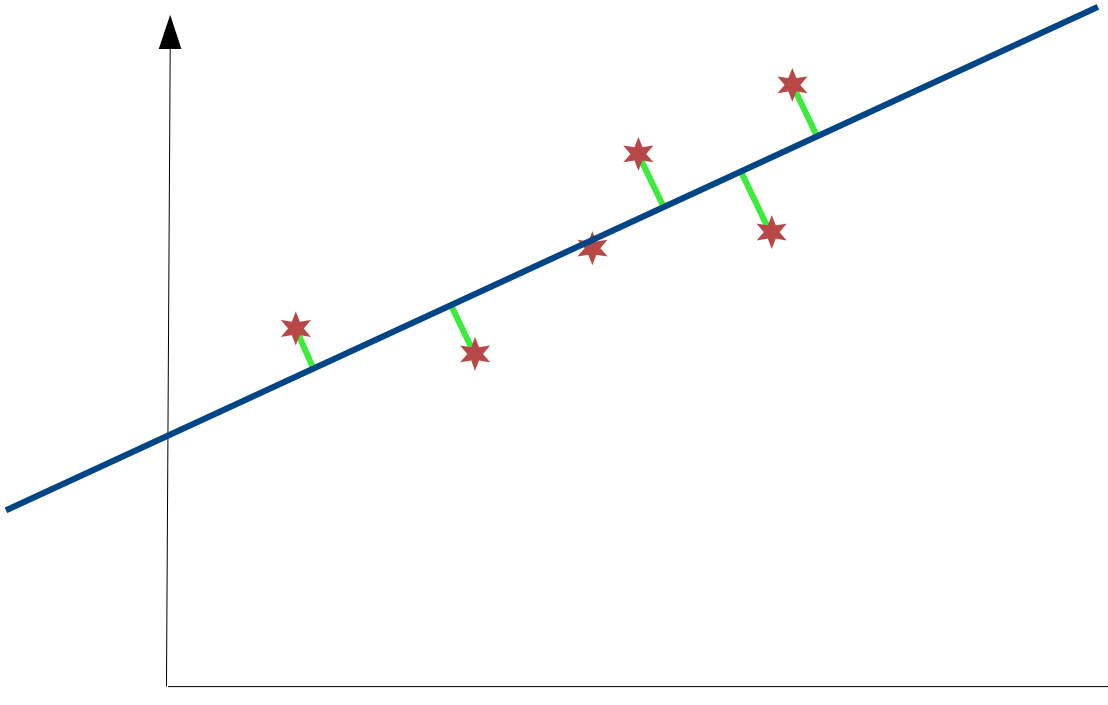
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \lambda \begin{pmatrix} n_x \\ n_y \end{pmatrix}$$

$$\lambda \sim N(0, \sigma)$$

[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Example: Line Fitting

- Given: a collection of data-points $\{(x_i, y_i)\}_{i=1\ldots N}$
- Task: fit a line to the data-points
- Approach: Gaussian noise in normal direction



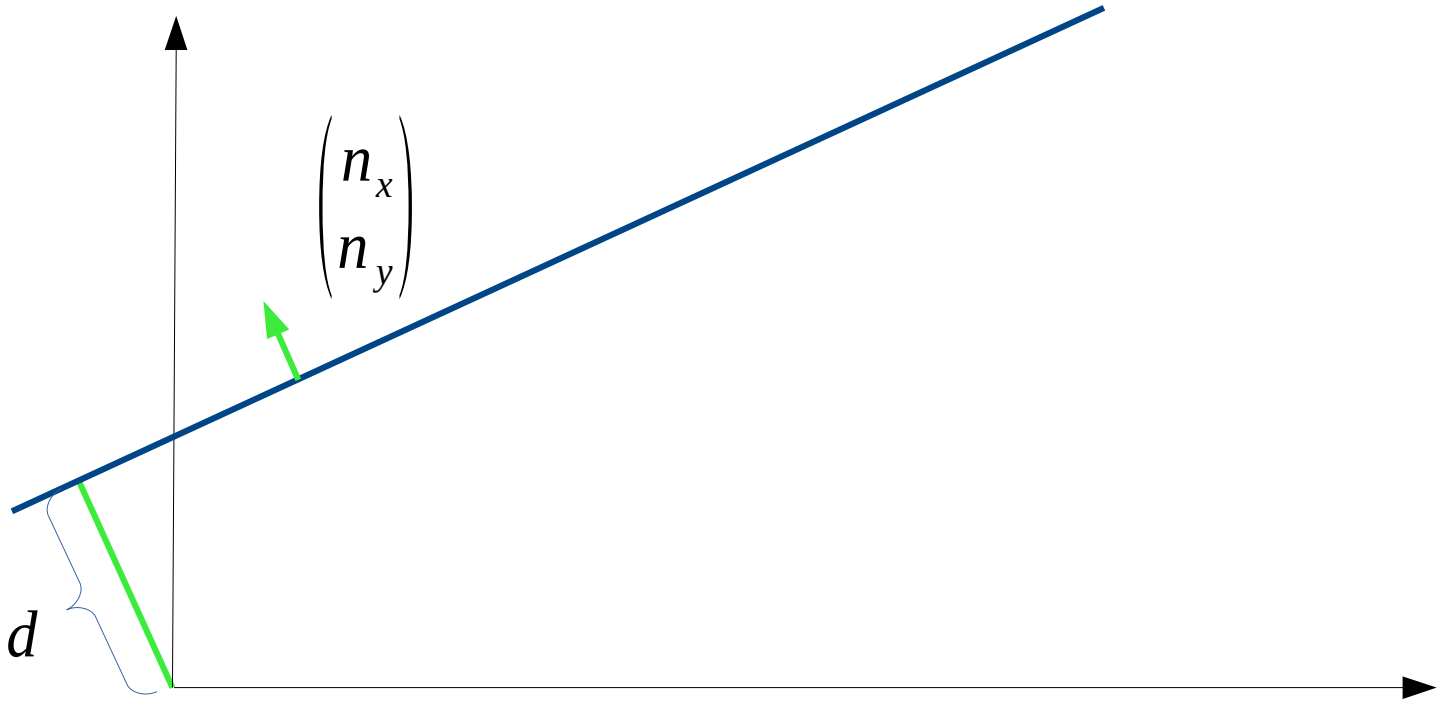
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \lambda \begin{pmatrix} n_x \\ n_y \end{pmatrix}$$

$$\lambda \sim N(0, \sigma)$$

[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Example: Line Fitting

- Reminder: Express lines as $L = \{(x, y) \in \mathbb{R}^2 | x n_x + y n_y = d\}$

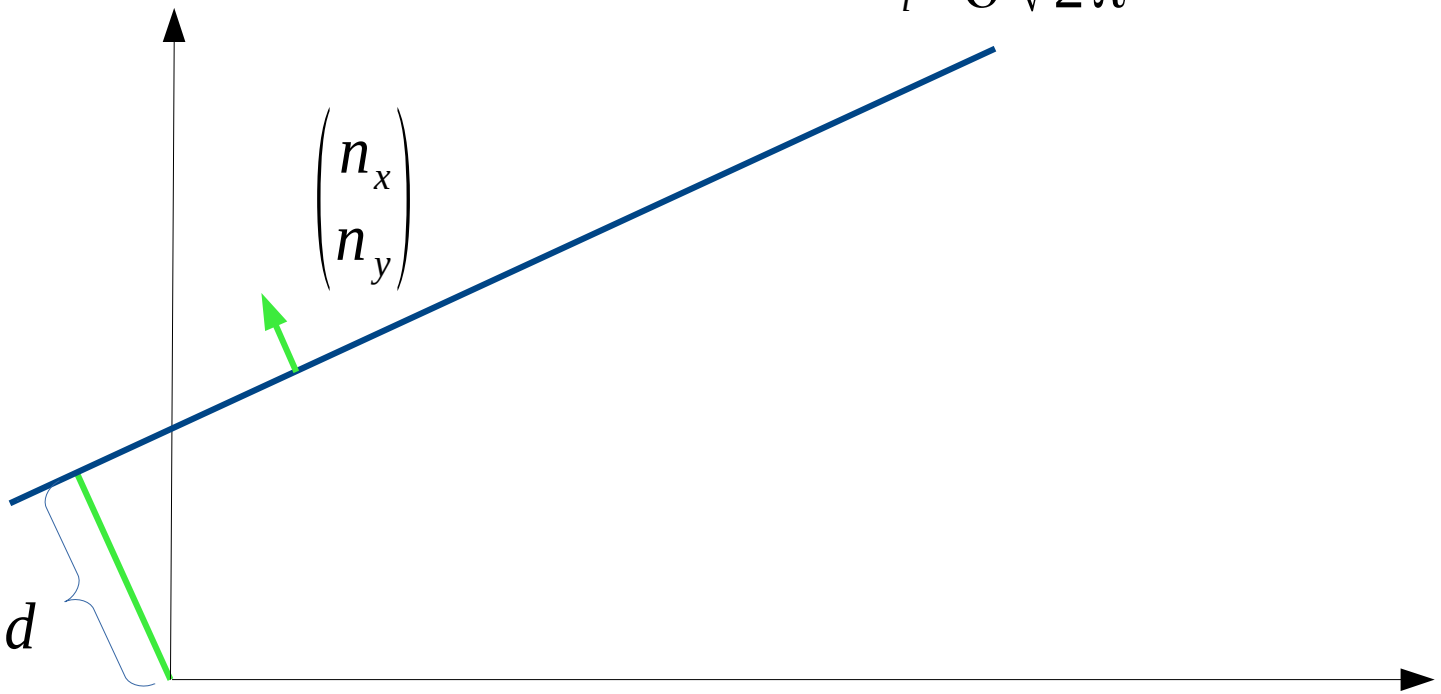



[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Example: Line Fitting

- Reminder: Express lines as $L=\{(x,y)\in\mathbb{R}^2 | x n_x + y n_y = d\}$
- Likelihood function $Pr(\mathbf{x}|n_x,n_y,d)=\prod_i Pr(x_i,y_i|n_x,n_y,d)$

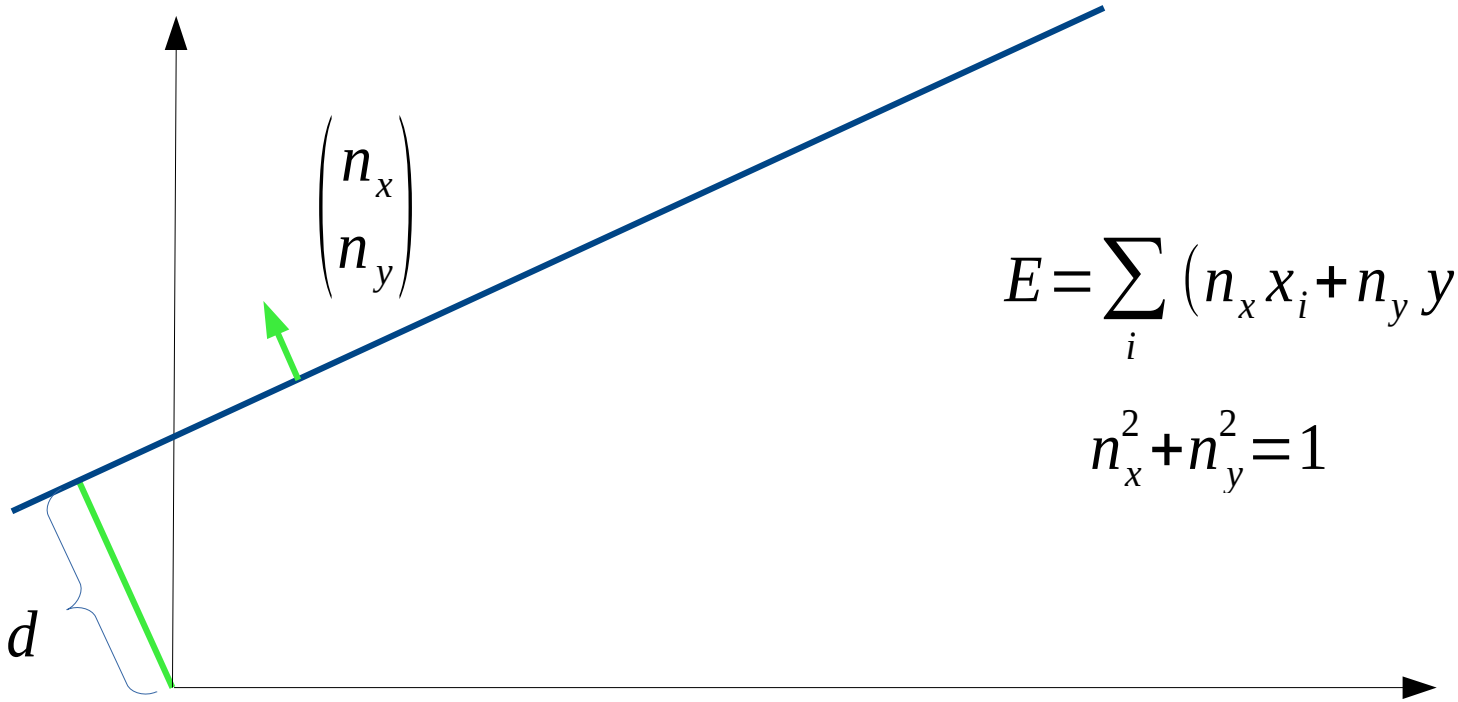
$$=\prod_i \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(n_x x_i + n_y y_i - d)^2}{2\sigma^2}\right)$$




[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Example: Line Fitting

- Reminder: Express lines as $L=\{(x,y)\in\mathbb{R}^2 | x n_x + y n_y = d\}$
- Likelihood function $Pr(\mathbf{x}|n_x,n_y,d)=\prod_i Pr(x_i,y_i|n_x,n_y,d)$
  - Max likelihood

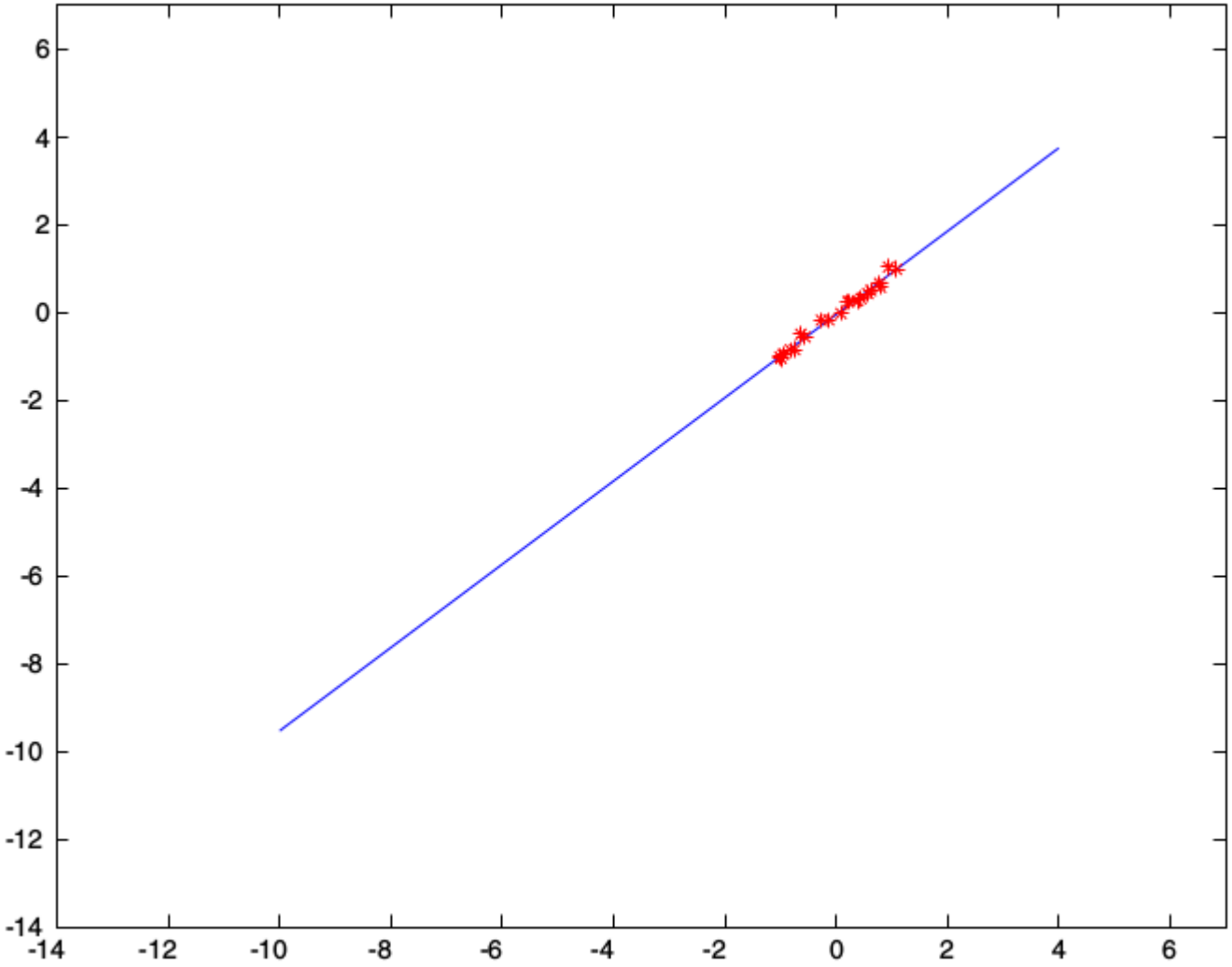



$$E=\sum_i (n_x x_i + n_y y_i - d)^2$$

$$n_x^2 + n_y^2 = 1$$

[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Example: Line Fitting




[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Similarity Measures

- Do brightness differences have a normal distribution?

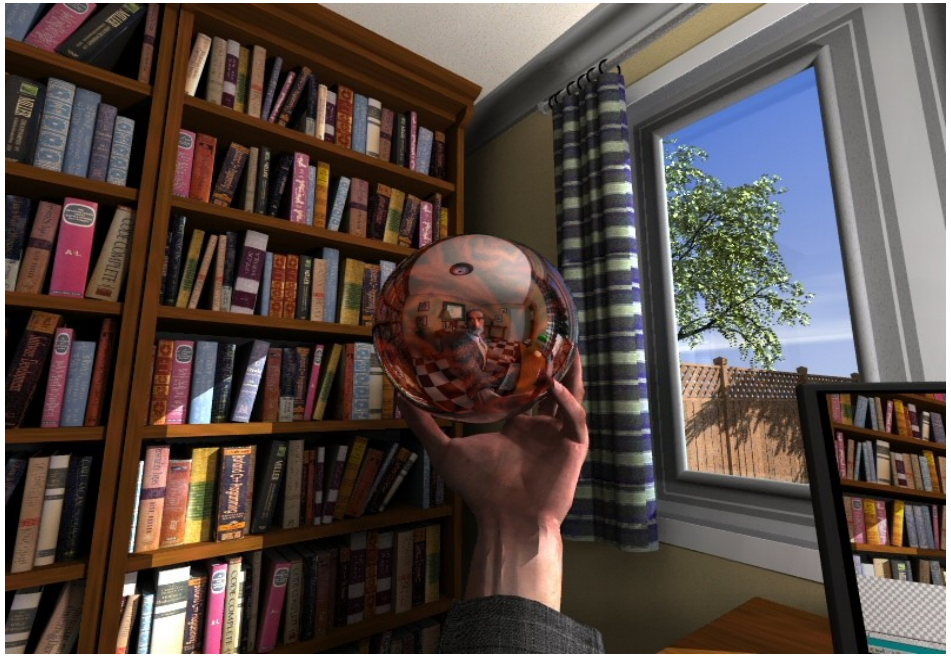
  (Is noise our only problem? )

# Similarity Measures

- Do brightness differences have a normal distribution?
  - Gaussian image noise $\hat{I}(x,y) = I(x,y) + n(x,y)$

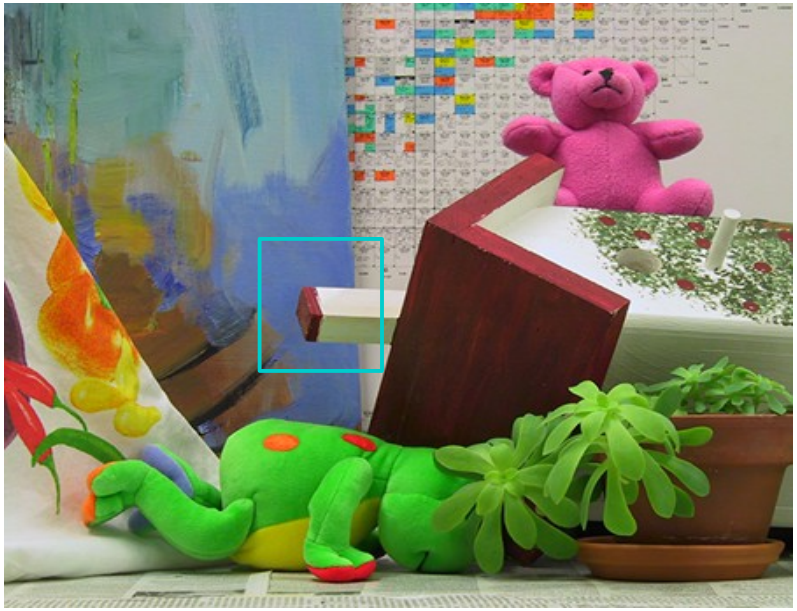
# Similarity Measures

- Do brightness differences have a normal distribution?
  - Gaussian image noise
  - Color constancy assumption

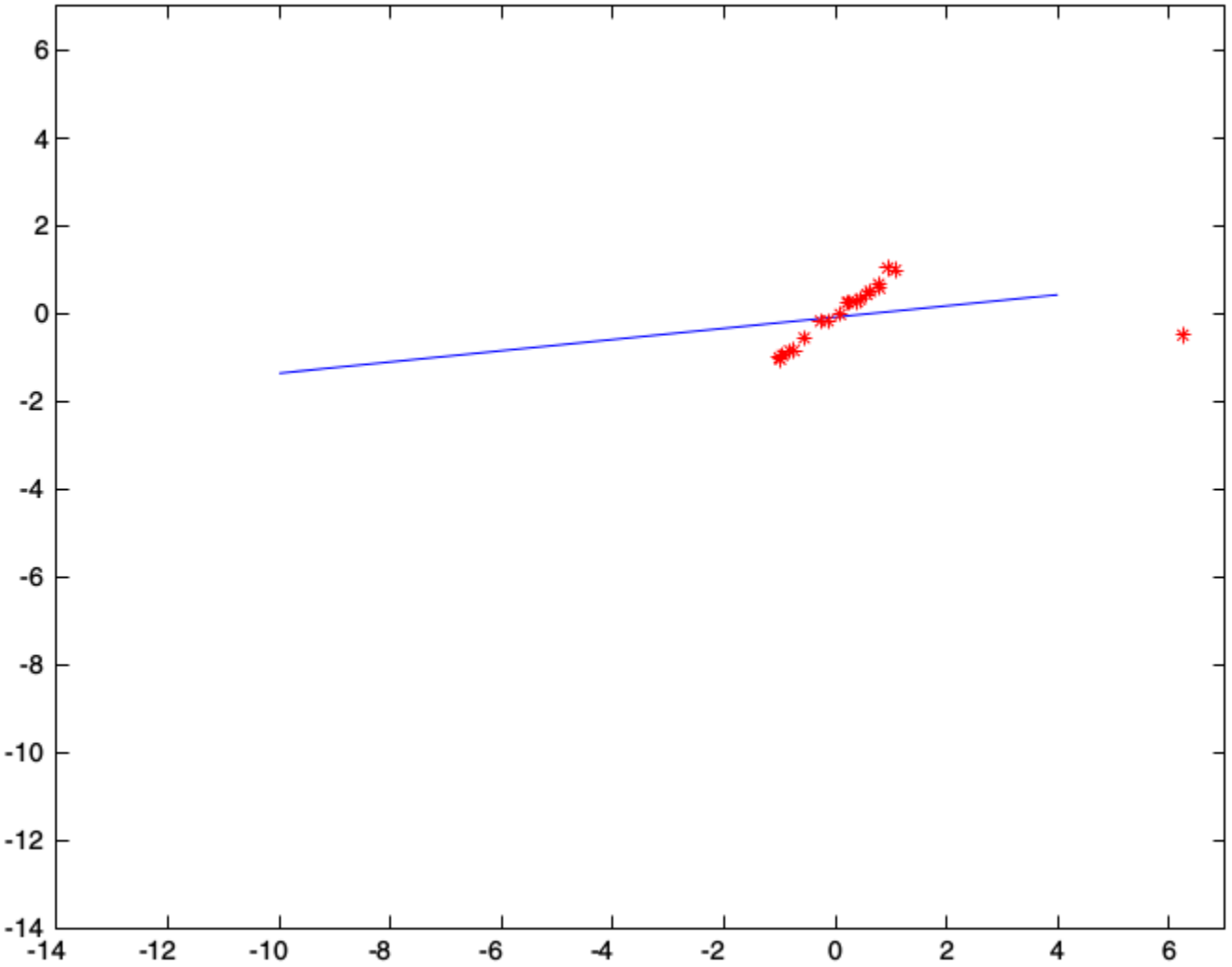


# Similarity Measures

- Do brightness differences have a normal distribution?
  - Gaussian image noise
  - Color constancy assumption
  - Inadequate window sizes & occlusion

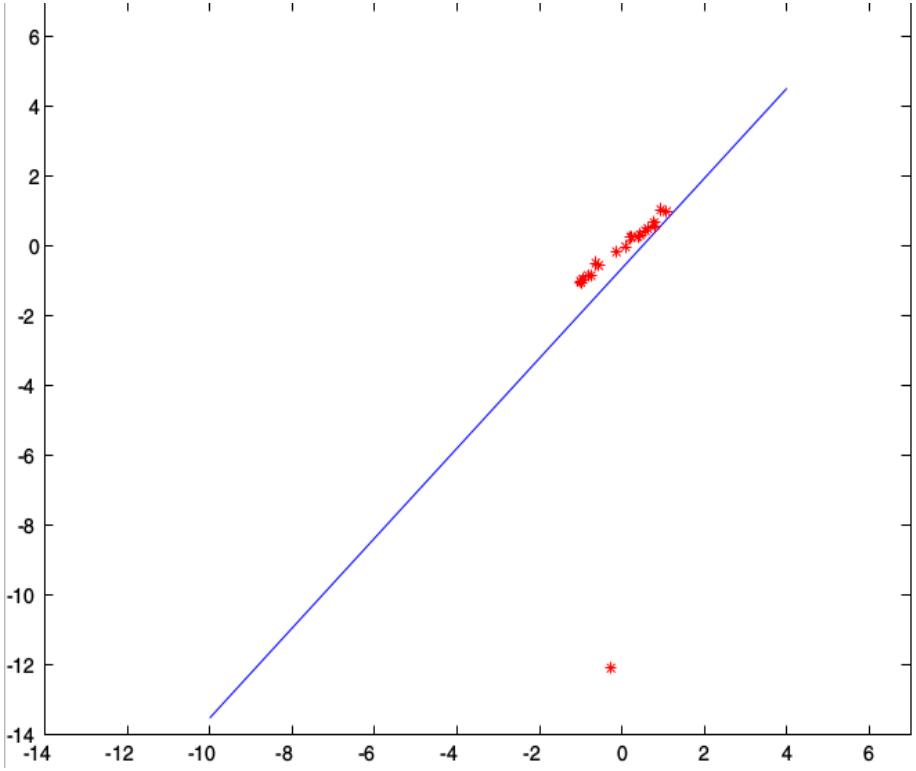


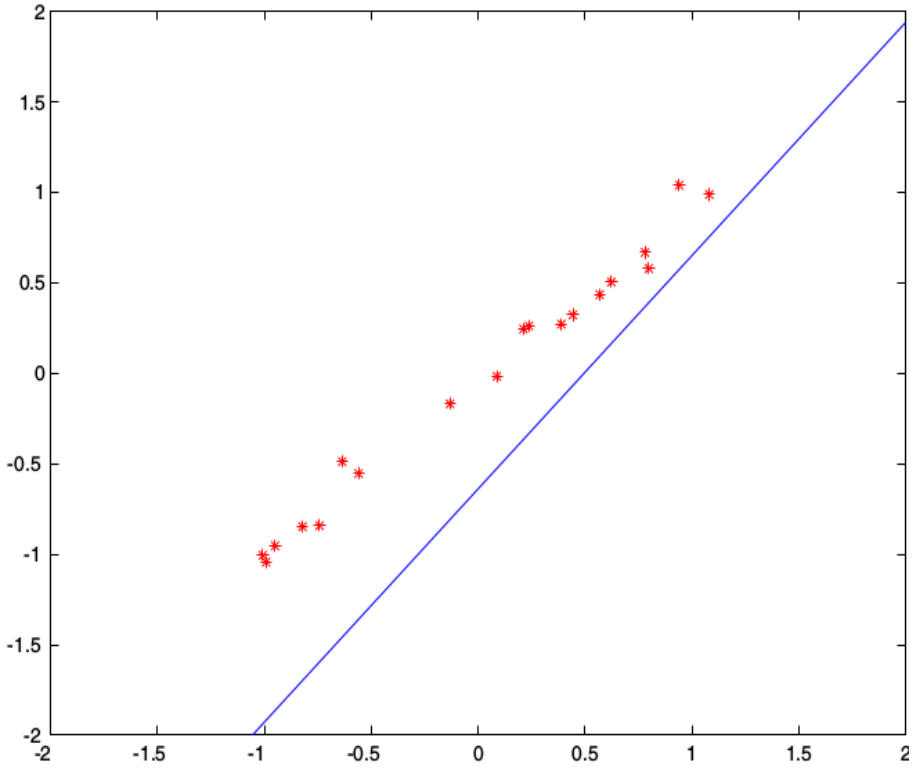
# Example: Line Fitting




[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Example: Line Fitting




Zoom in from left image

[ Forsyth, Ponce(2003): Computer Vision, Ch. 15]

# Example: Line Fitting

- How can we de-emphasize the influence of distant outliers?
  - Assume non-Gaussian distributions of error

    ↔ use non-quadratic error functions for the log-likelihood

$$Pr(\boldsymbol{x}|n_x, n_y, d) = \prod_i Pr(x_i, y_i|n_x, n_y, d)$$

$$= \prod_i \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(n_x x_i + n_y y_i - d)^2}{2\sigma^2}\right)$$

$$E = -\log(Pr(\boldsymbol{x}|n_x, n_y, d)) = C_1 \sum_i (n_x x_i + n_y y_i - d)^2 + C_2$$

# Example: Line Fitting

- How can we de-emphasize the influence of distant outliers?
  - Assume non-Gaussian distributions of error

    ↔ use non-quadratic error functions for the log-likelihood

$$Pr(\boldsymbol{x}|n_x, n_y, d) = \prod_i Pr(x_i, y_i|n_x, n_y, d)$$

$$= \prod_i \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(n_x x_i + n_y y_i - d)^2}{2\sigma^2}\right)$$

$$E = -\log(Pr(\boldsymbol{x}|n_x, n_y, d)) = C_1 \sum_i (n_x x_i + n_y y_i - d)^2 + C_2$$

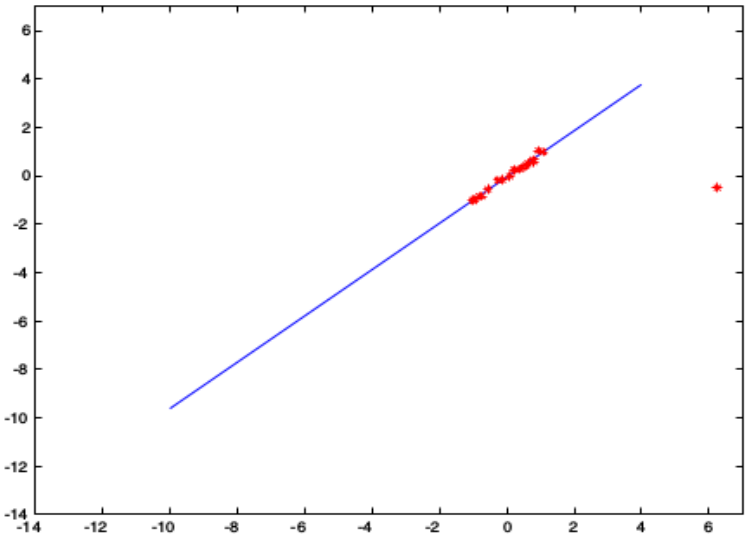
$$= C_1 \sum_i \rho(n_x x_i + n_y y_i - d) + C_2$$

# Robust Error Function

- Desirable properties
  - Inliers should be punished approximately quadratically

    ↔ they follow a Gaussian distribution
  - Outliers should have small/ no influence

    ↔ they are more probable than under Gaussian distribution



# Robust Error Function

- Influence function
  - Assume here differentiable error function
  - How much differs the energy with the location of data-point?

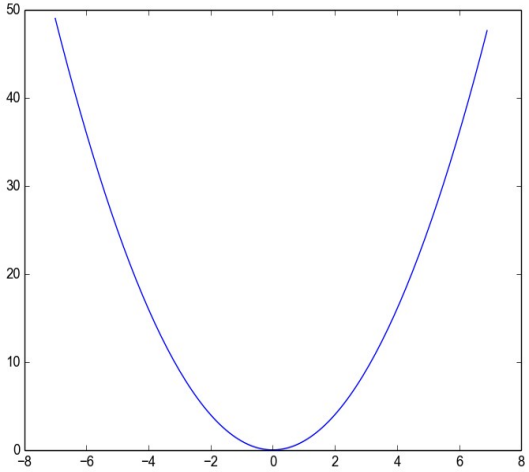
$$\hat{E}(\boldsymbol{x}) = \rho(n_x x + n_y y - d)$$

$$\varphi = \frac{\partial \rho}{\partial x}$$

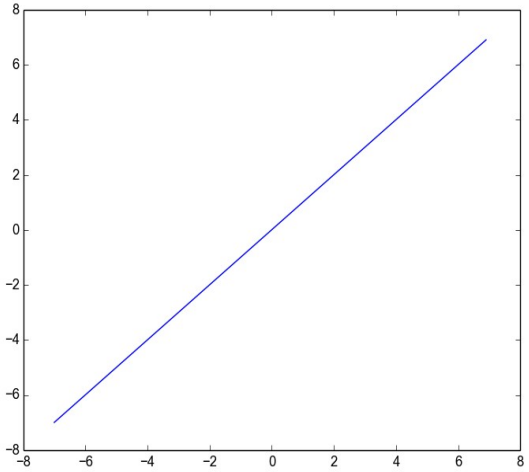
# Error Functions

- Quadratic

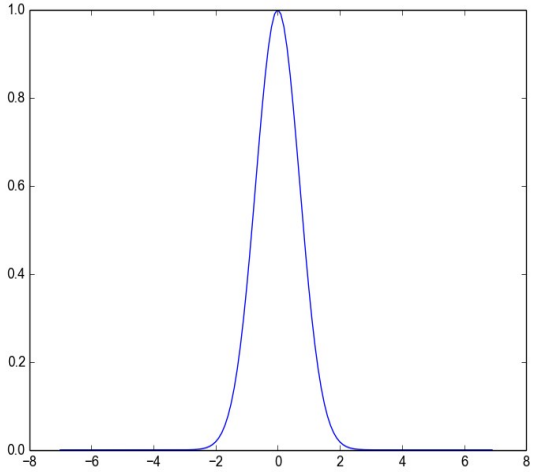
Error function




Influence function




Probability distribution




$$\rho(x|\sigma) = \frac{x^2}{2\sigma^2}$$

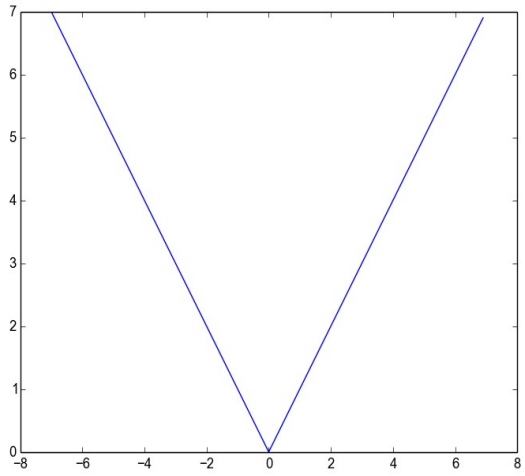
$$\varphi(x|\sigma) = \frac{x}{\sigma^2}$$

$$Pr(x|\sigma) = \frac{1}{Z}\exp\left(-\frac{x^2}{2\sigma^2}\right)$$

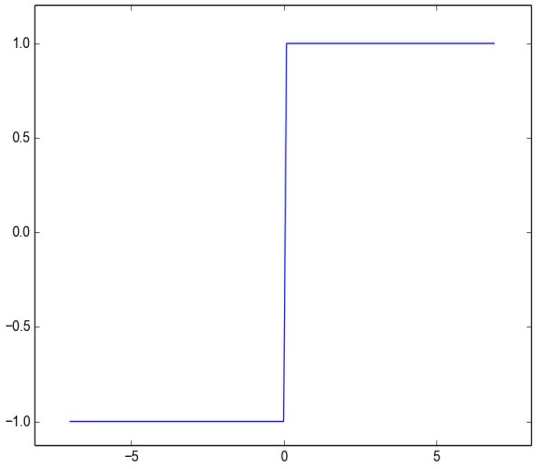
# Error Functions

- Linear

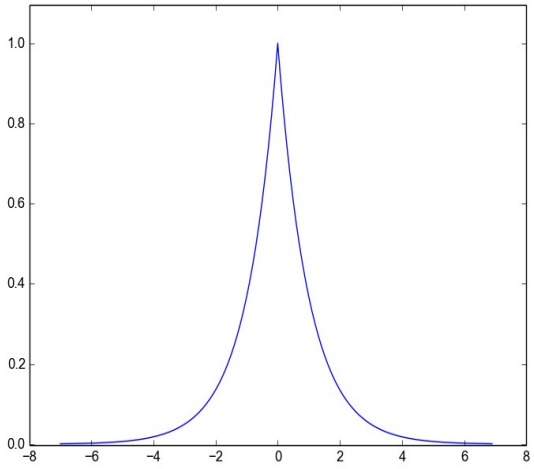
Error function




Influence function




Probability distribution




$$\rho(x|\sigma) = |x|$$

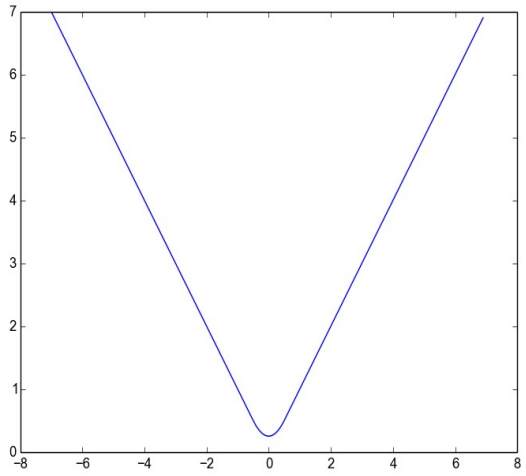
$$\varphi(x|\sigma) = sign(x)$$

$$Pr(x|\sigma) = \frac{1}{Z}\exp(-\rho)$$

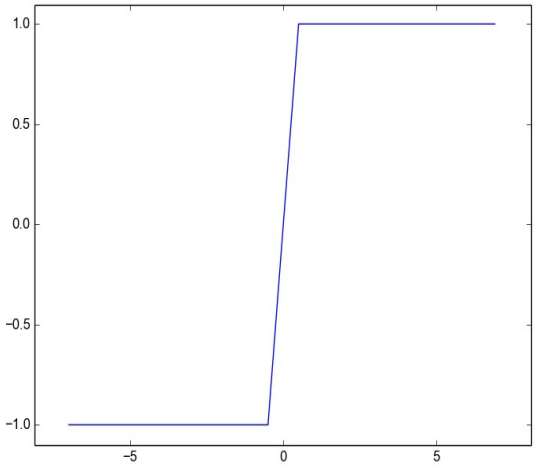
# Error Functions

- Huber

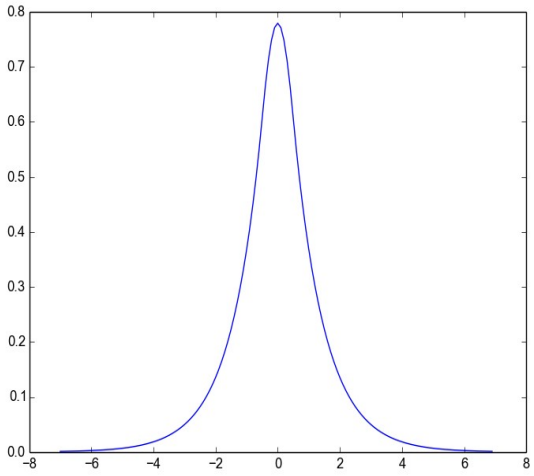
Error function



Influence function



Probability distribution



$$\rho(x|\sigma)=\begin{cases}\frac{x^2}{2\epsilon}+\frac{\epsilon}{2} & |x|\le\epsilon\\ |x| & \text{else}\end{cases}$$

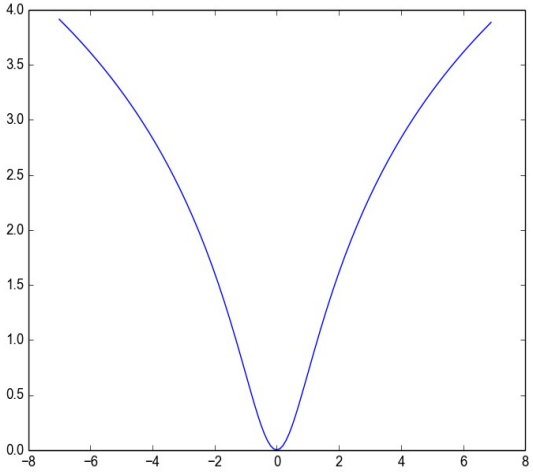
$$\varphi(x|\sigma)=\begin{cases}\frac{x}{\epsilon} & |x|\le\epsilon\\ sign(x) & \text{else}\end{cases}$$

$$Pr(x|\sigma)=\frac{1}{Z}\exp(-\rho)$$

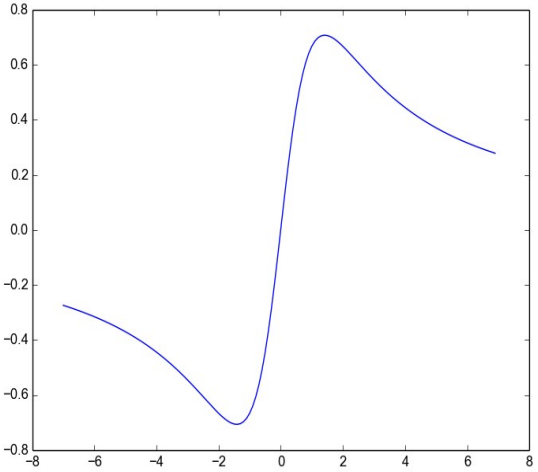
# Error Functions

- Lorentian ↔ Student's t distribution

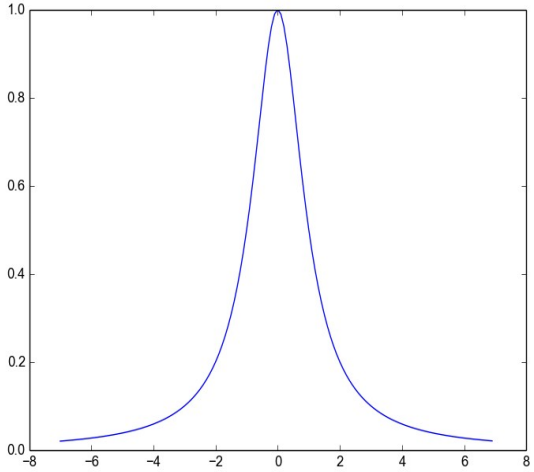
Error function




Influence function




Probability distribution

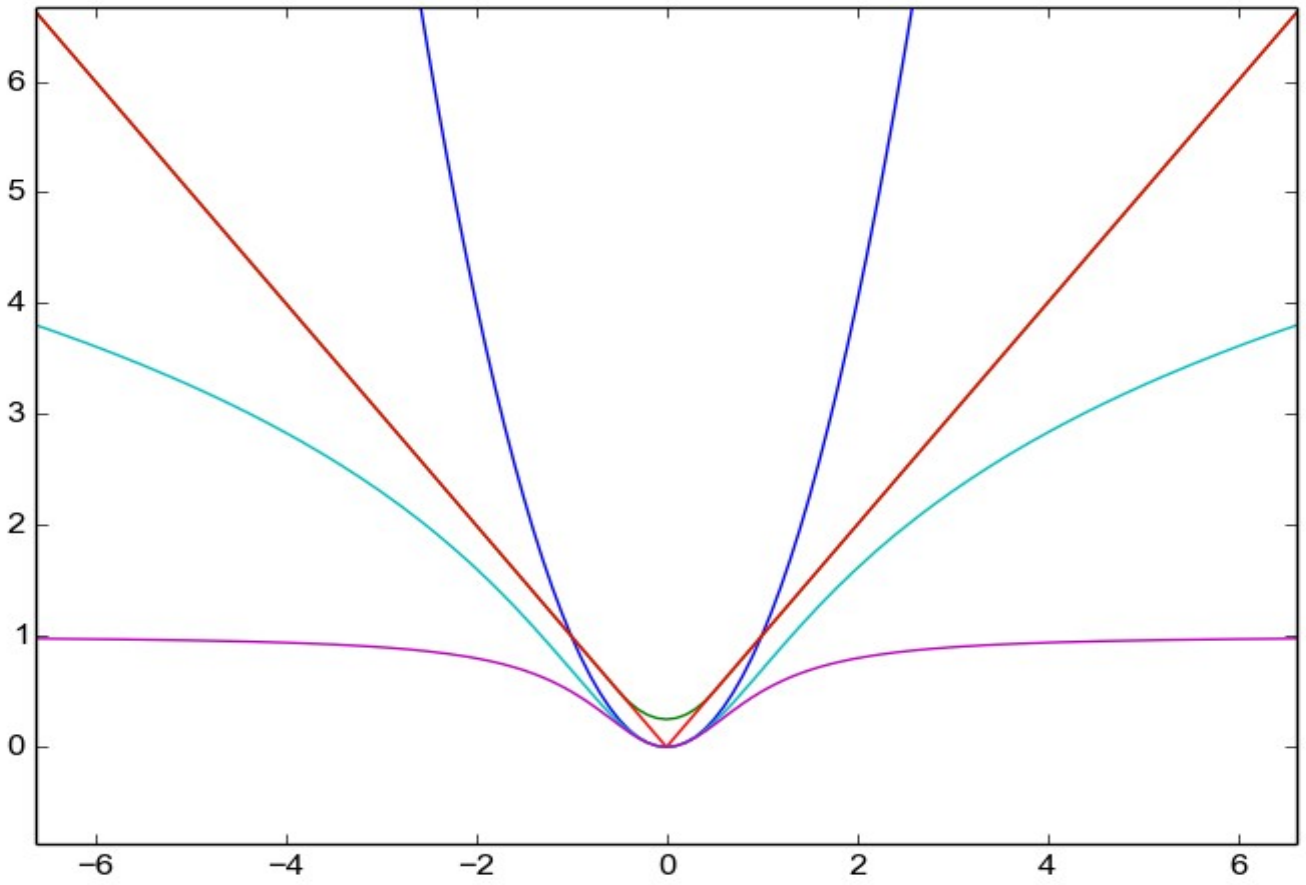



$$\rho(x|\sigma) = \log\left(1 + \frac{1}{2\sigma^2}x^2\right)$$

$$\varphi(x|\sigma) = \frac{2x}{2\sigma^2 + x^2}$$

$$Pr(x|\sigma) = \left(1 + \frac{x^2}{2\sigma^2}\right)^{-\alpha}$$

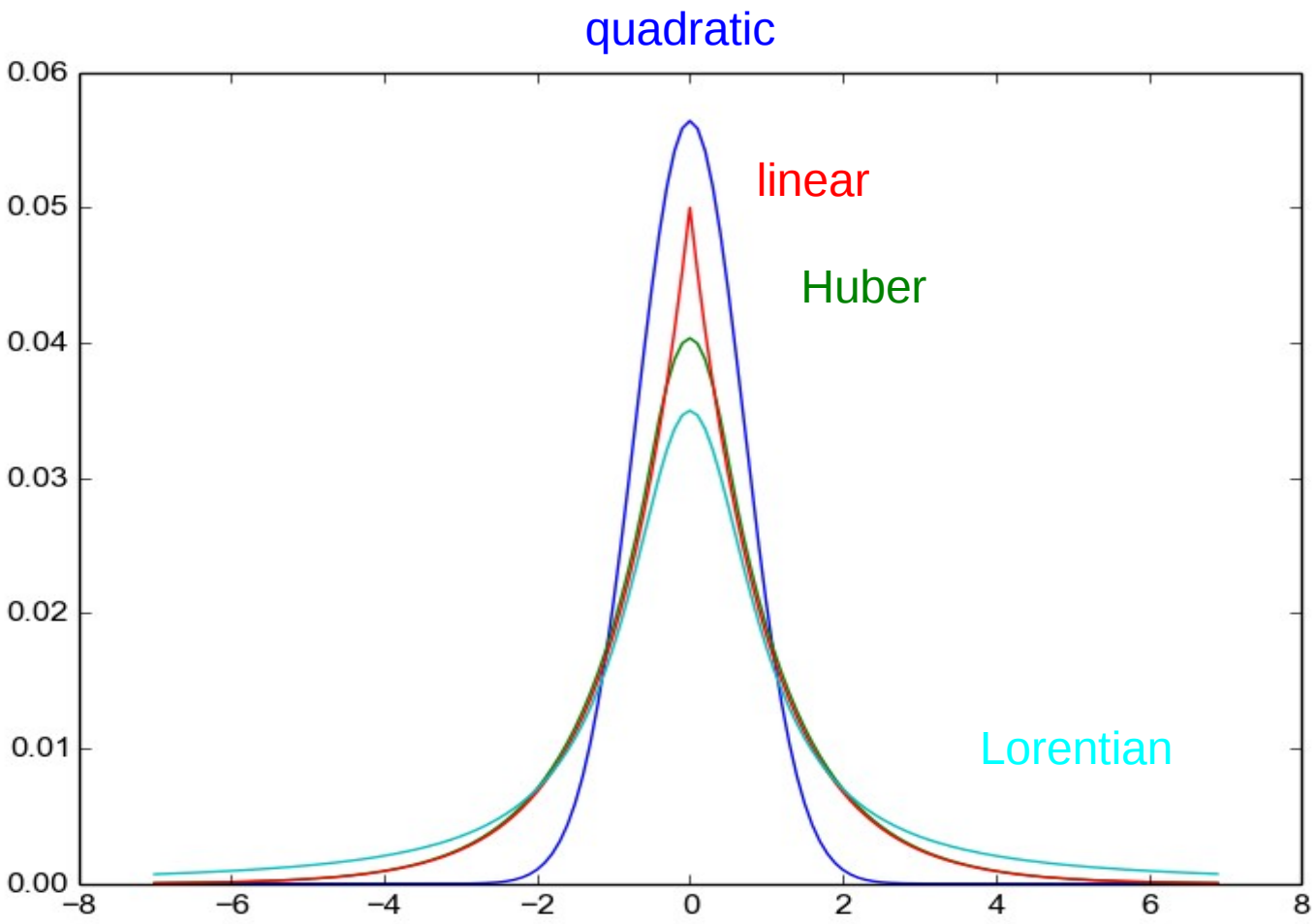
# Error Functions: Comparisons




$$\rho(x|\sigma) = \frac{x^2}{\sigma^2 + x^2}$$

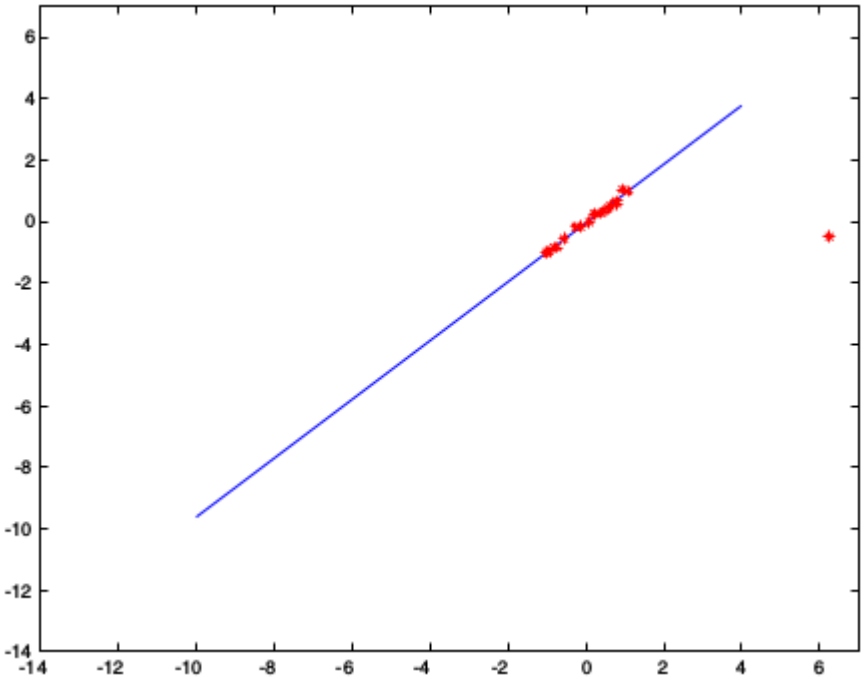
Geman McClure

# Error Distributions: Comparisons



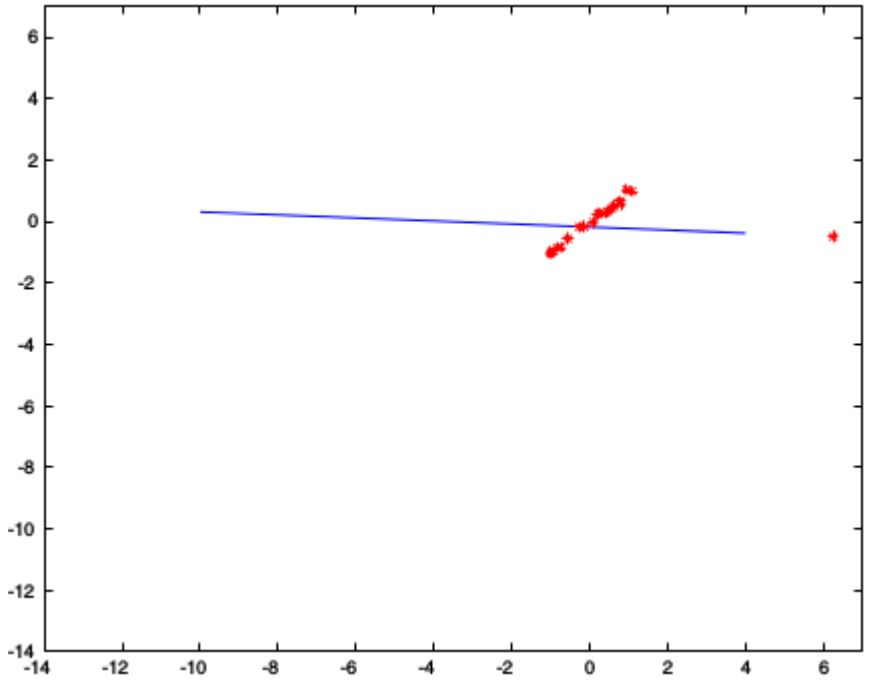
# Example: Line Fitting

- Correct choice of scale parameter $\sigma$



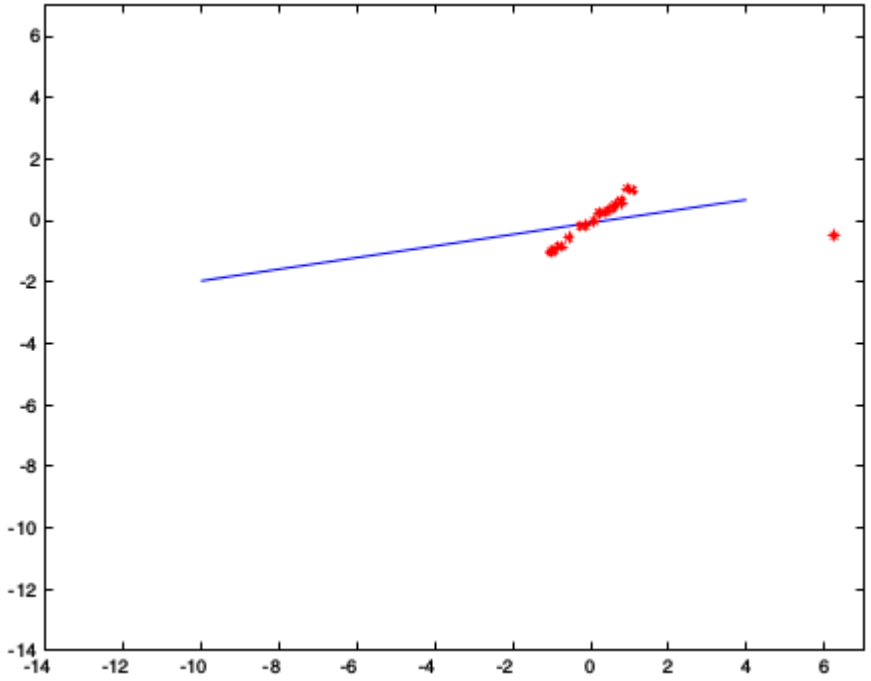
# Example: Line Fitting

- $\sigma$ too small
  - all data points are weighted down



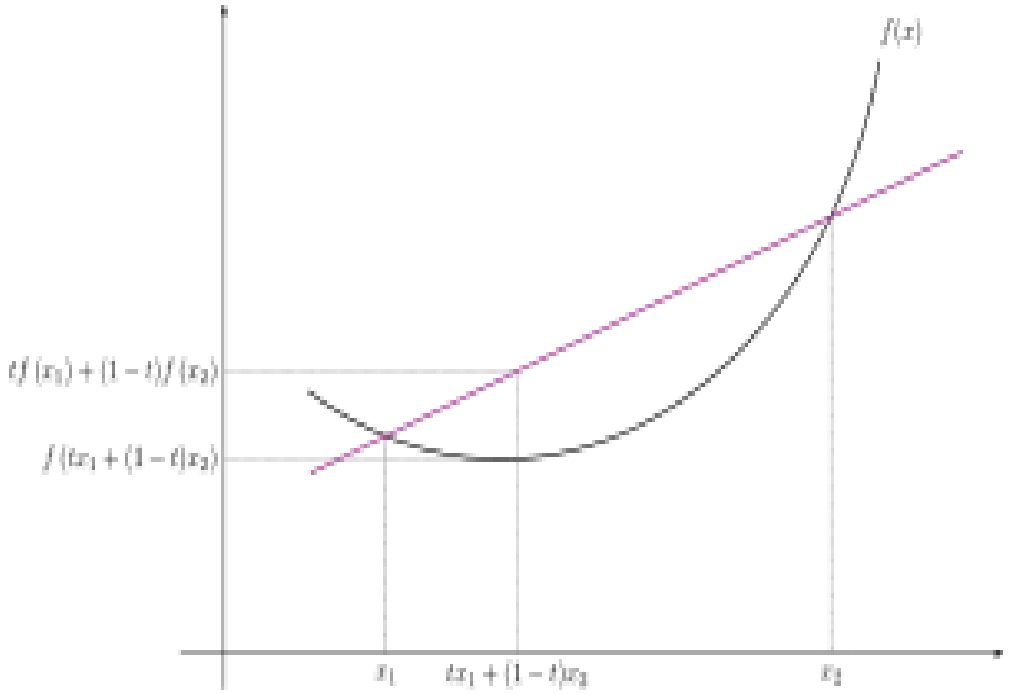
# Example: Line Fitting

- $\sigma$ too large
  - all data points are considered



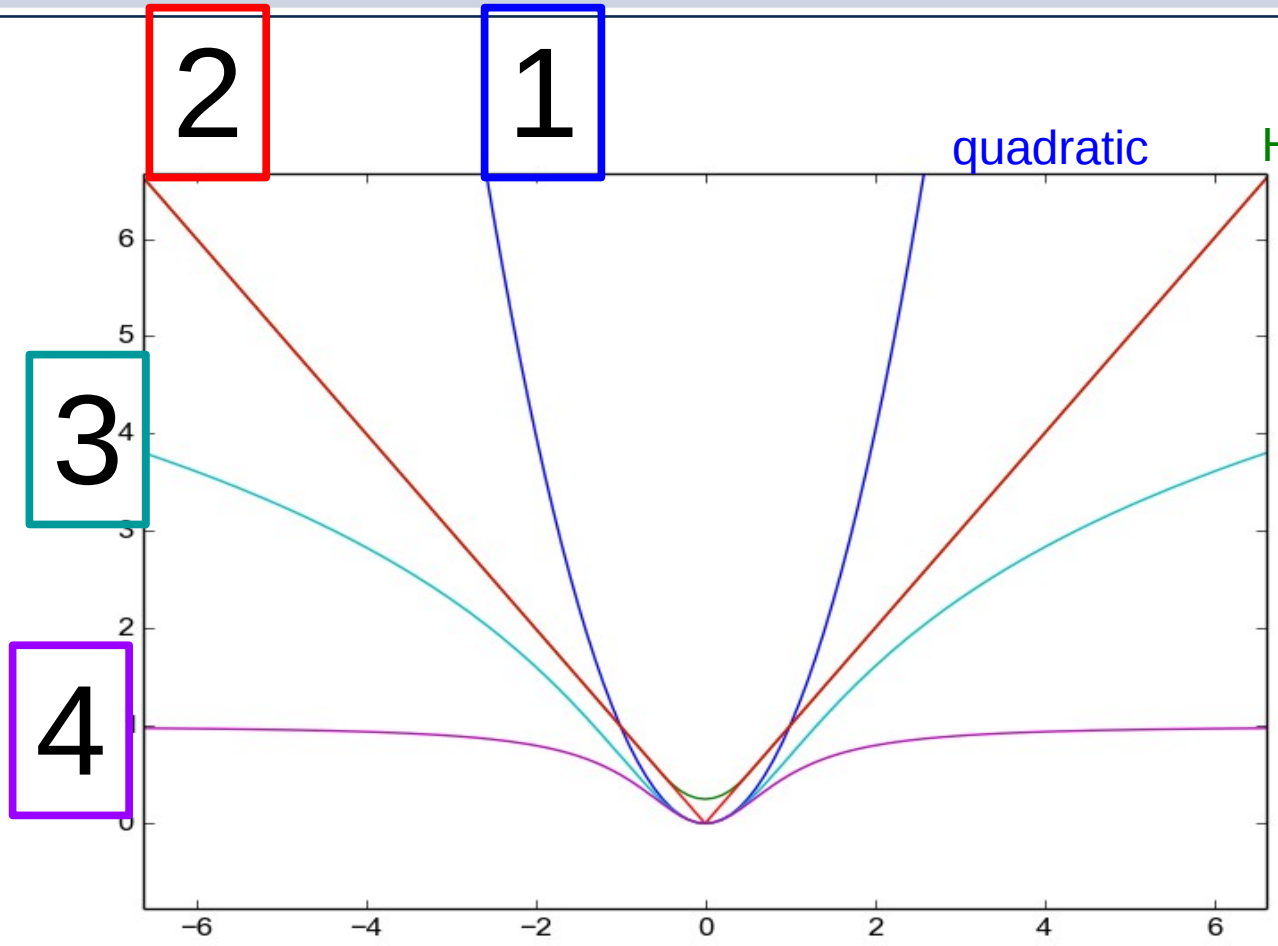
# Convex Functions

- $\forall x_1, x_2 \in X, \forall t \in [0,1]: \quad f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$




- Any local minimum of a convex function is a global minimum

# Quiz: Which Error Function is Convex?




linear

$$\rho(x|\sigma)=\log\left(1+\frac{1}{2\sigma^2}x^2\right)$$

Lorentian

$$\rho(x|\sigma)=\frac{x^2}{\sigma^2+x^2}$$

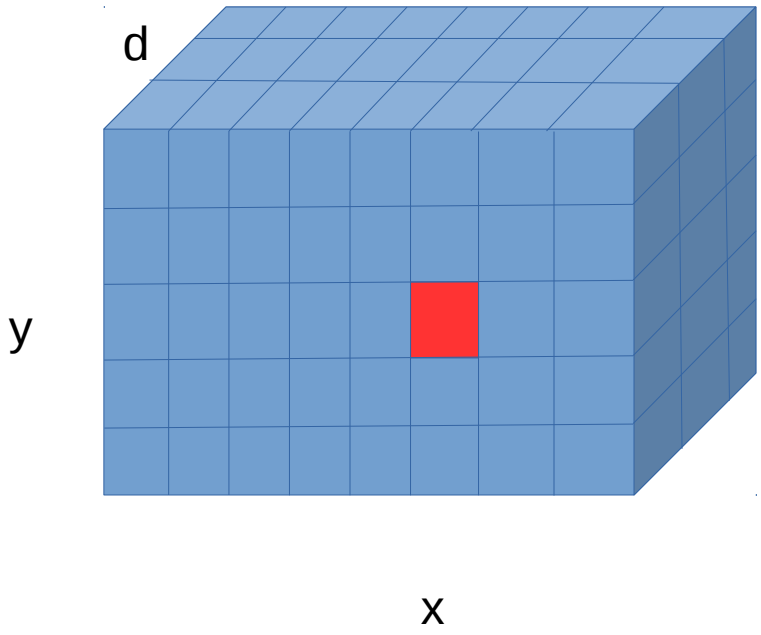
Geman McClure

5 : None

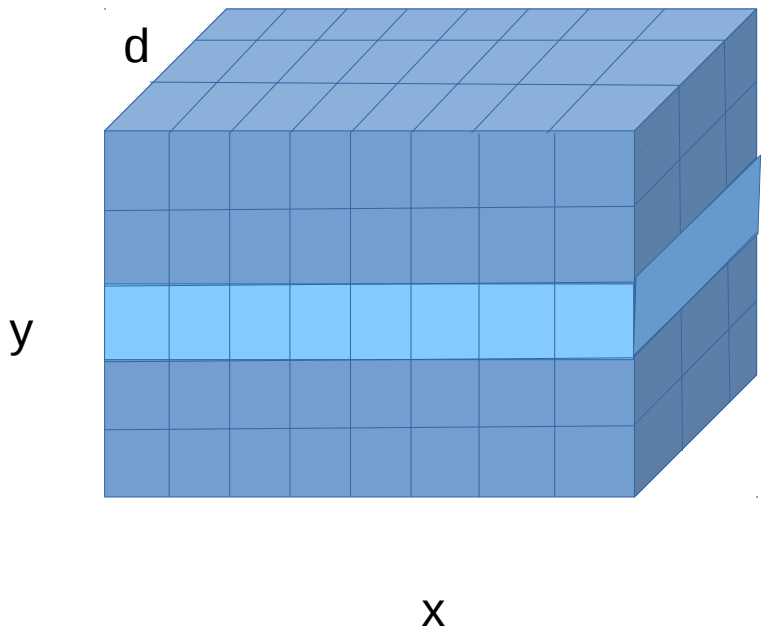
6 : all

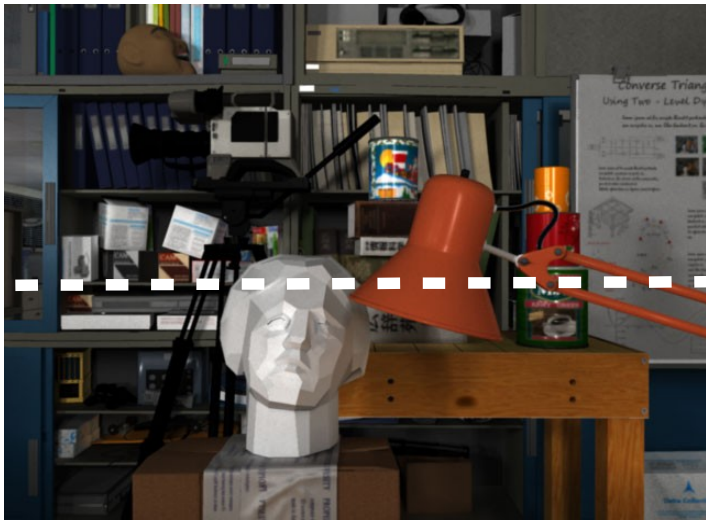
# Disparity Space Image

$$C(x, y, d)$$



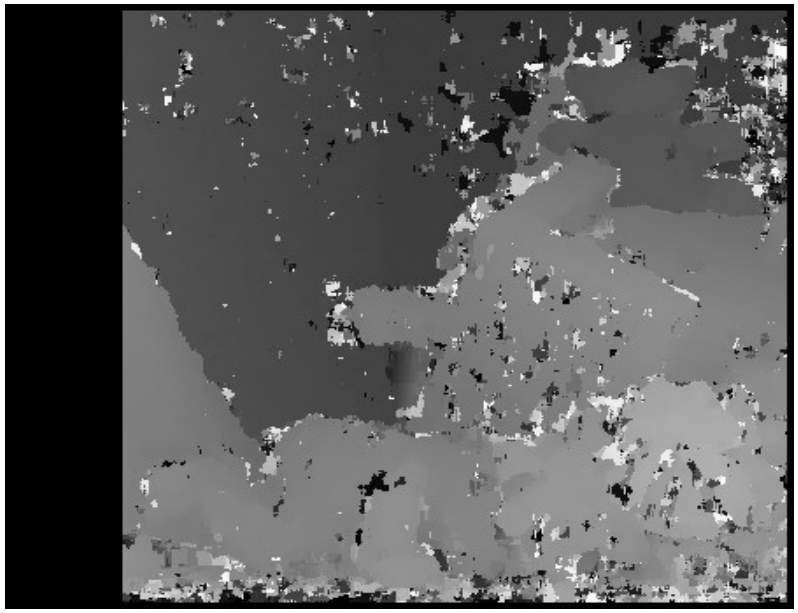
# Disparity Space Image



# A Basic Stereo Algorithm: Blockmatching

- For each pixel
  - For each disparity value
    - Compare patch similarity
    - Assign disparity with highest similarity



Result: Noisy, no smooth surfaces