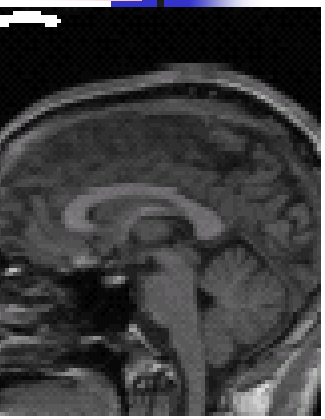
Image and Vision Computing
Review

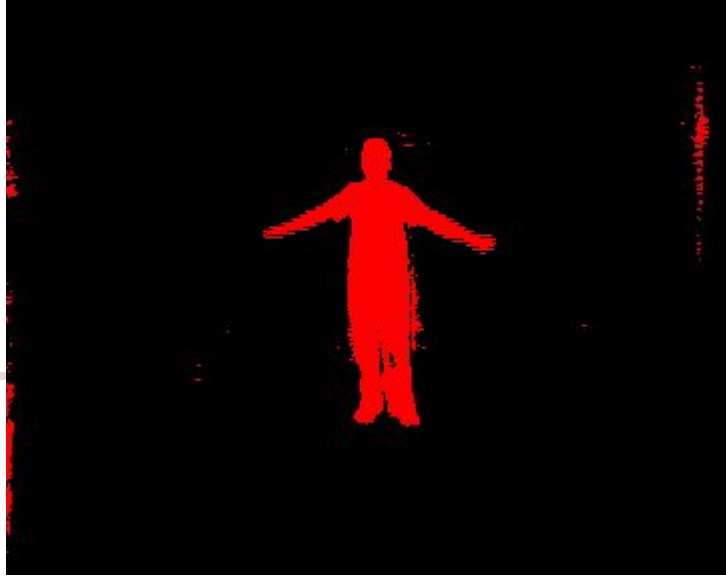
Instructor: Chen Yisong

HCI & Multimedia Lab, Peking University

chenys@graphics.pku.edu.cn

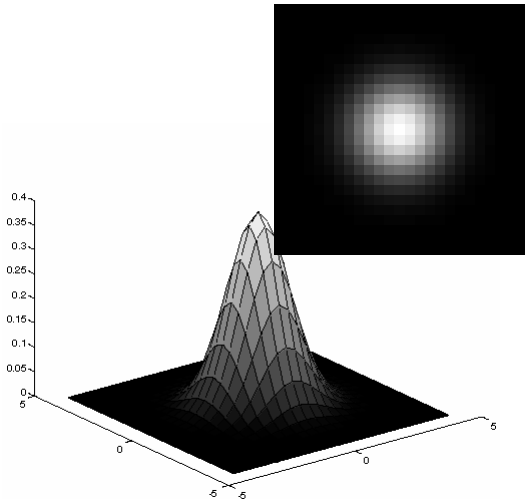
数字图像的表示



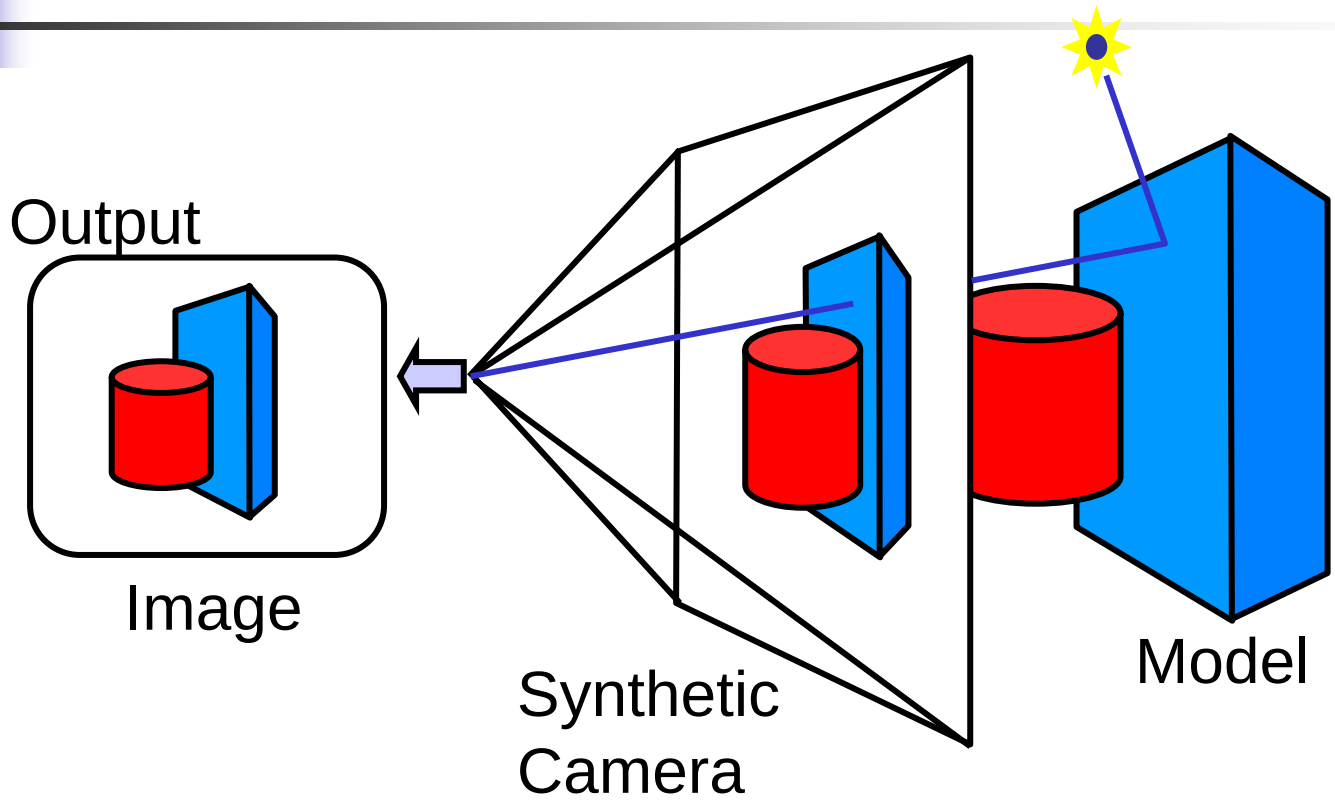


像素值的含义

- 物体表面的亮度(颜色)
- 生物体的吸收特性，如X光片
- 区域的温度，如红外照相
- 深度信息
- 任意二维函数的值

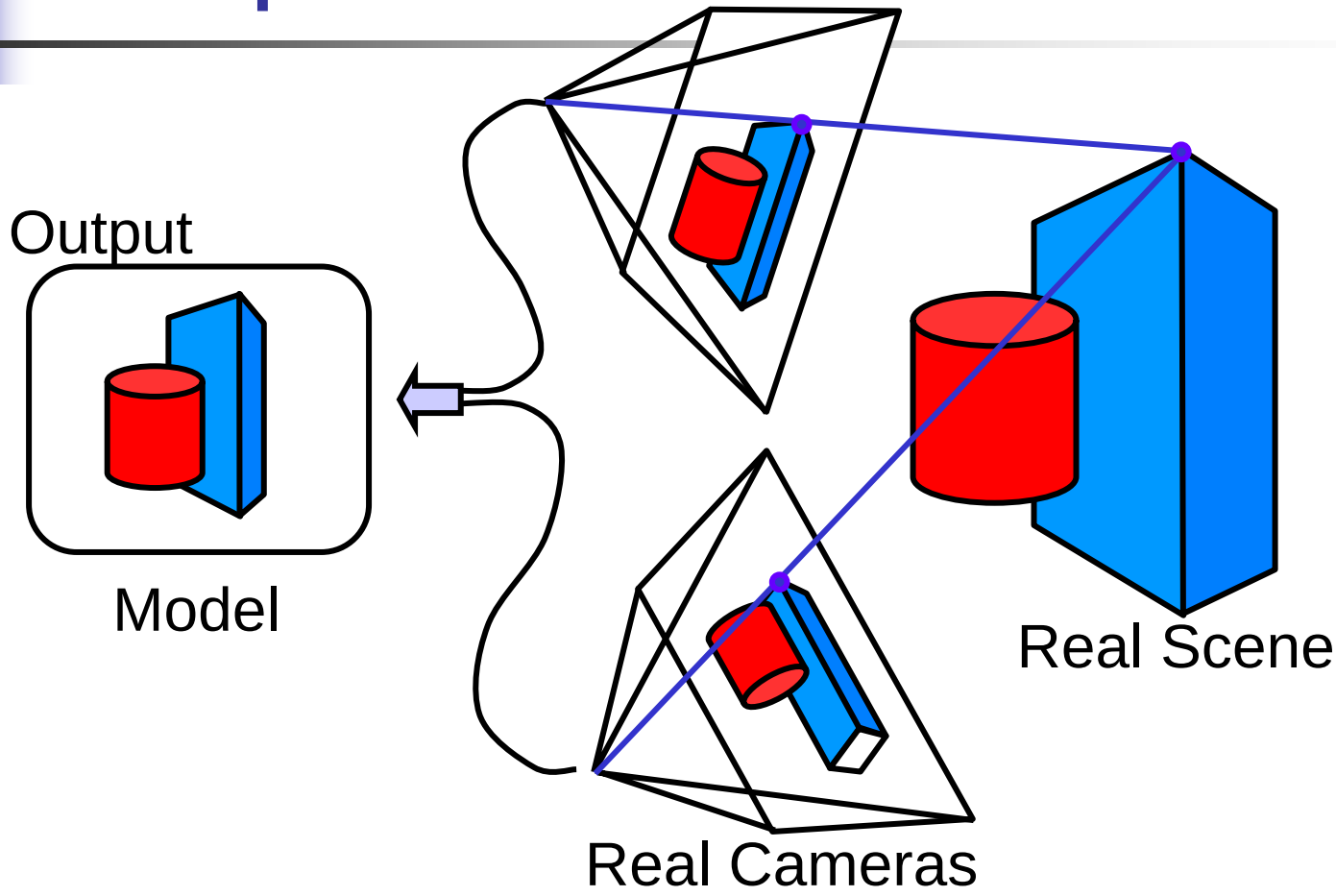


Computer Graphics



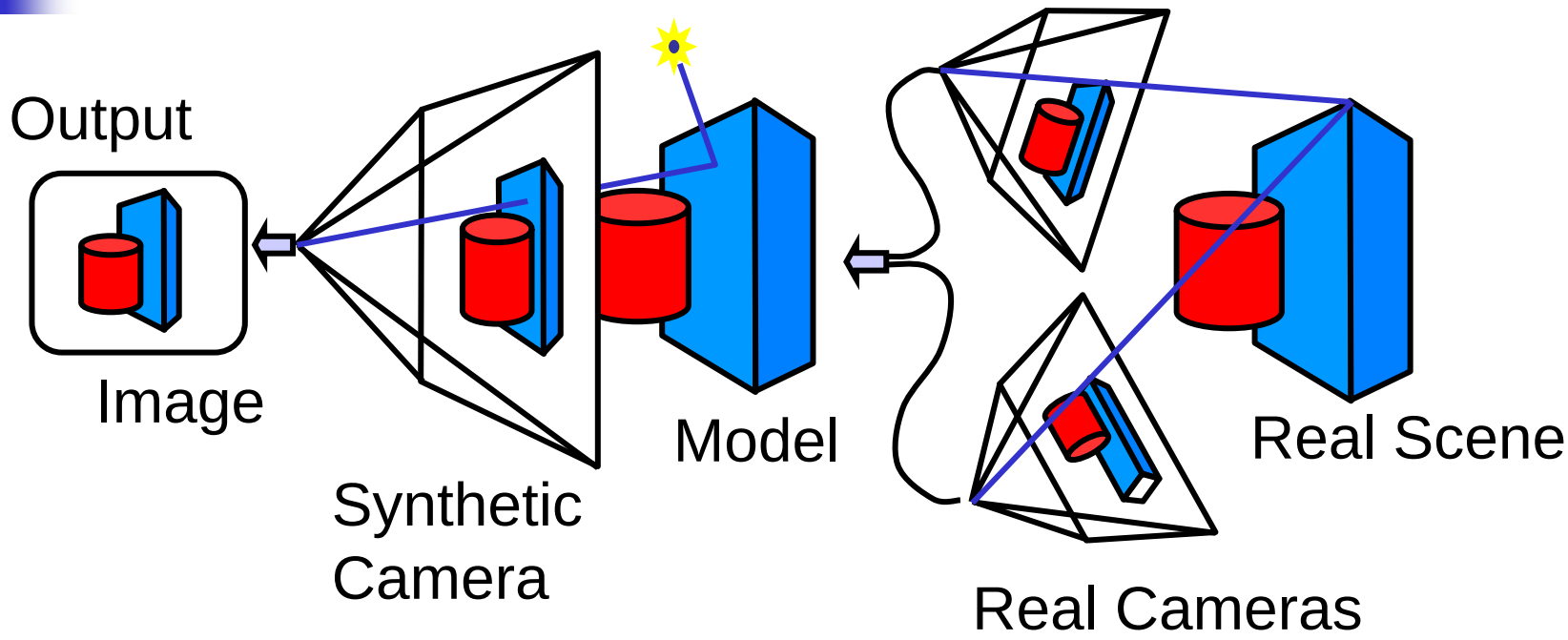
(slides courtesy of Michael Cohen)

Computer Vision



(slides courtesy of Michael Cohen)

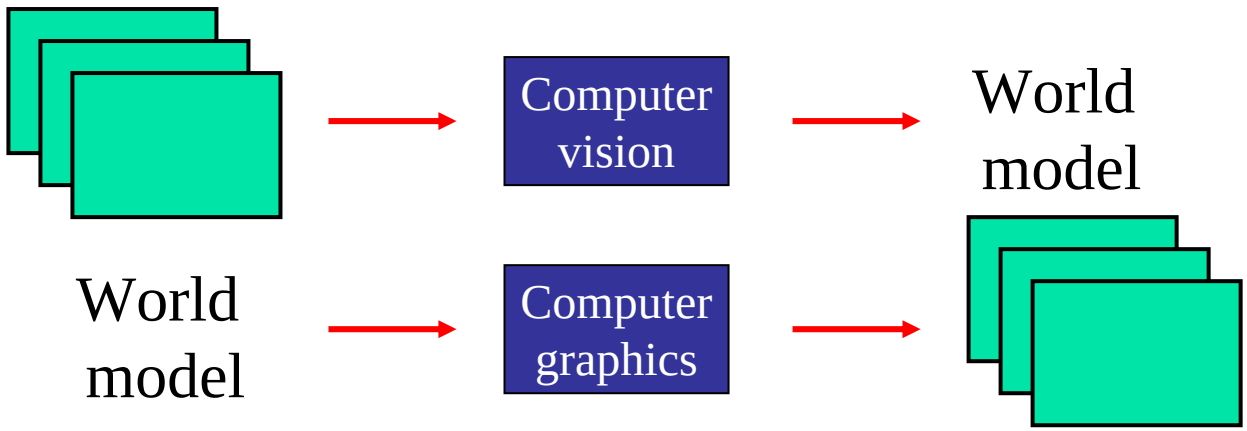
Combined



(slides courtesy of Michael Cohen)

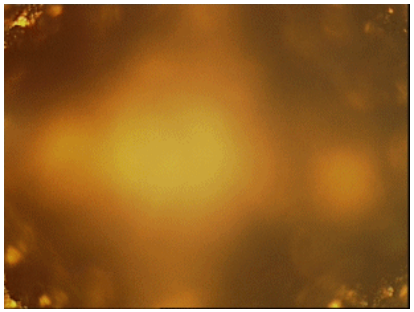
What is Computer Vision?

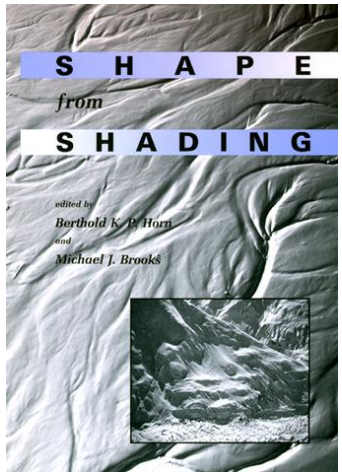
- Image Understanding (AI, behavior)
- A sensor modality for robotics
- Computer emulation of human vision
- Inverse of Computer Graphics



Shape from ...

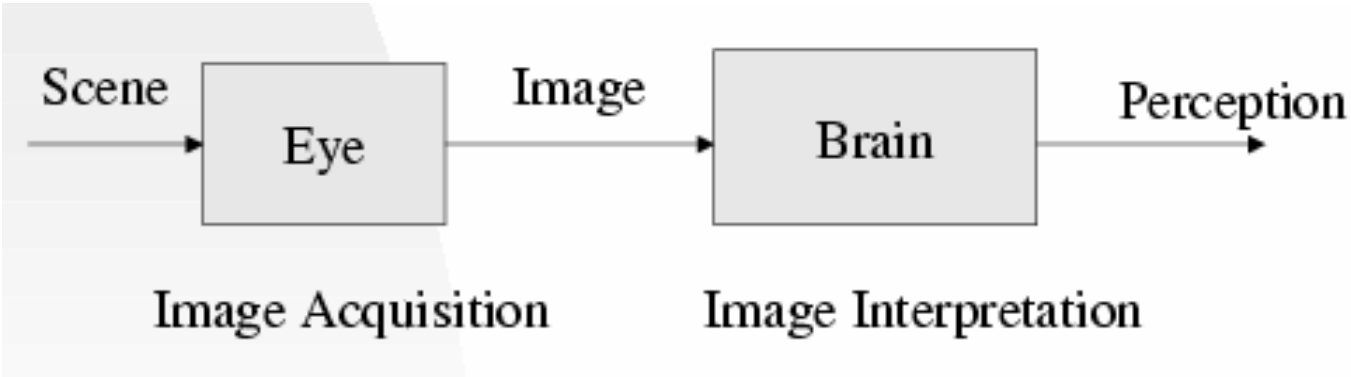
many different approaches/cues





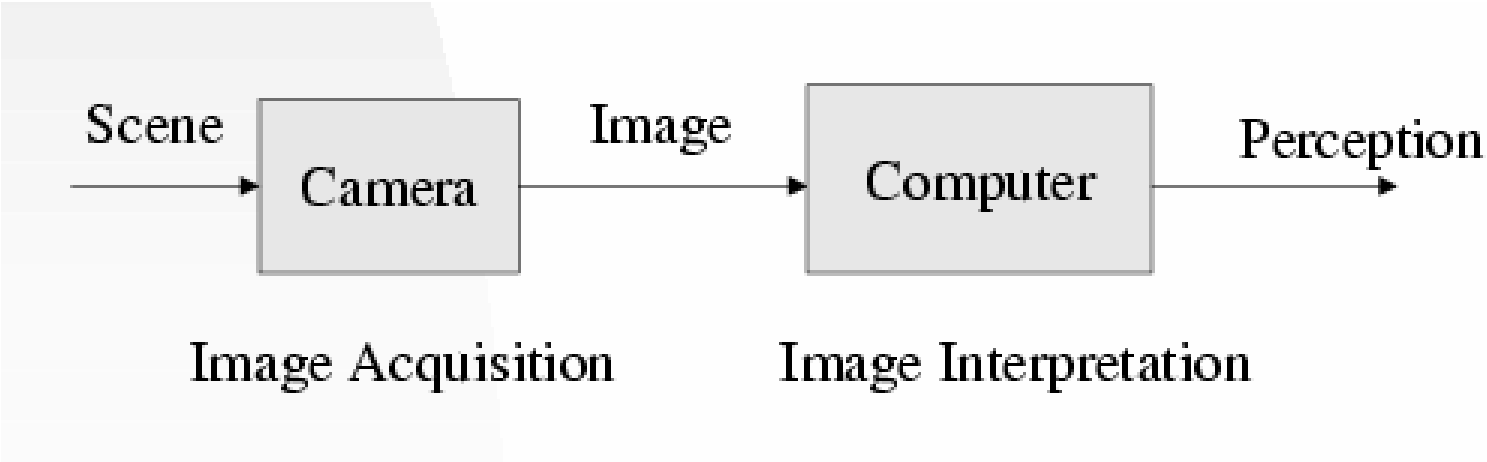
Vision

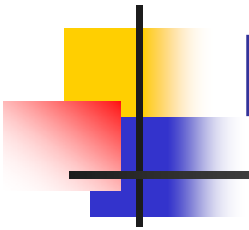
- Vision is the process of discovering what is present in the world and where it is by looking.



Computer Vision

- Computer Vision is the study of analysis of pictures and videos in order to achieve results similar to those as by men.





Main topics

- Shape (and motion) recovery

  “What is the 3D shape of what I see?”
- Segmentation

  “What belongs together?”
- Tracking

  “Where does something go?”
- Recognition

  “What is it that I see?”

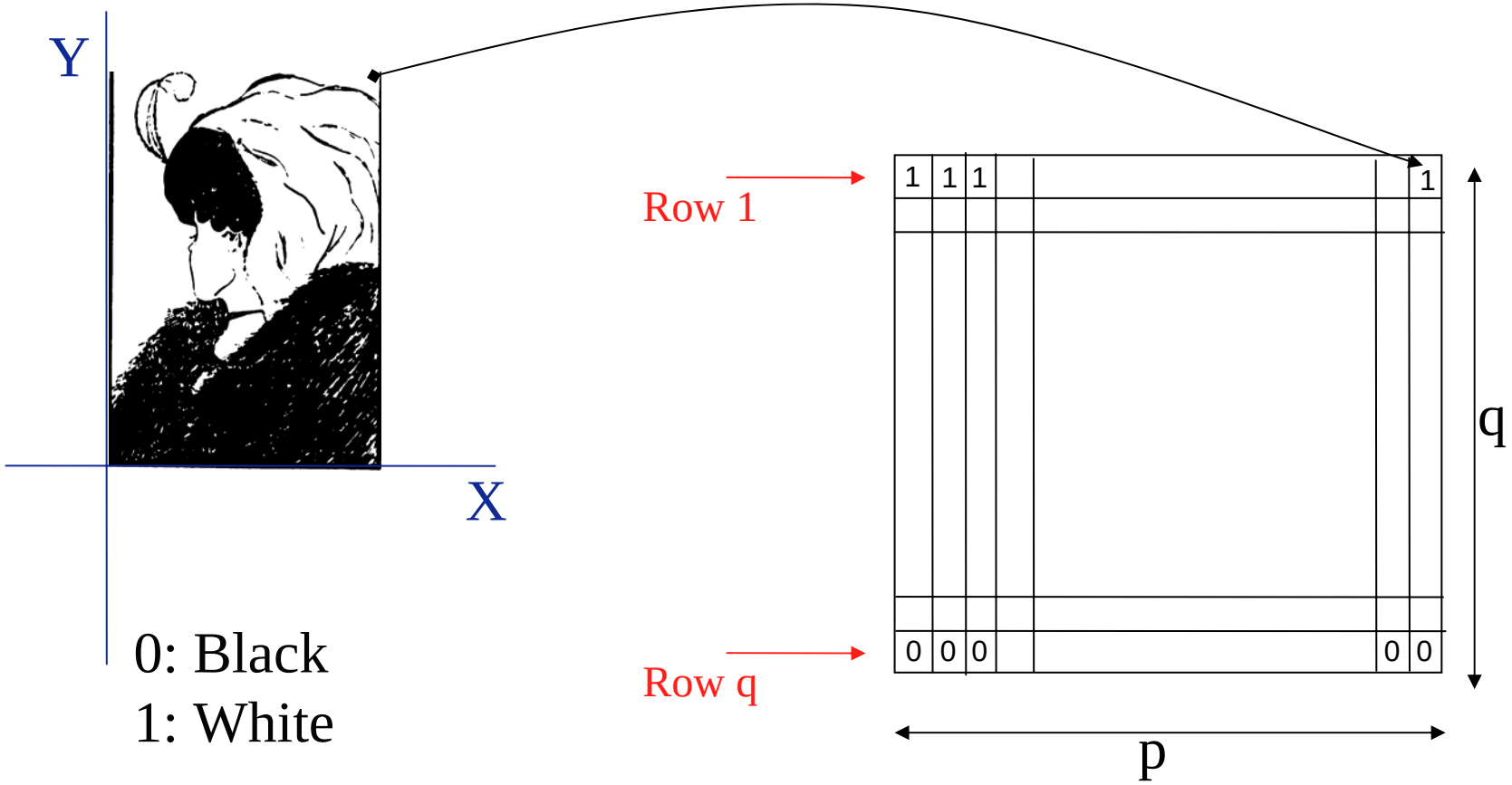
Main topics

- Camera & Light
  - Geometry, Radiometry, Color
- Digital images
  - Filters, edges, texture, optical flow
- Shape (and motion) recovery
  - Multi-view geometry
  - Stereo, motion, photometric stereo, ...
- Segmentation
  - Clustering, model fitting, probalistic
- Tracking
  - Linear dynamics, non-linear dynamics
- Recognition
  - templates, relations between templates

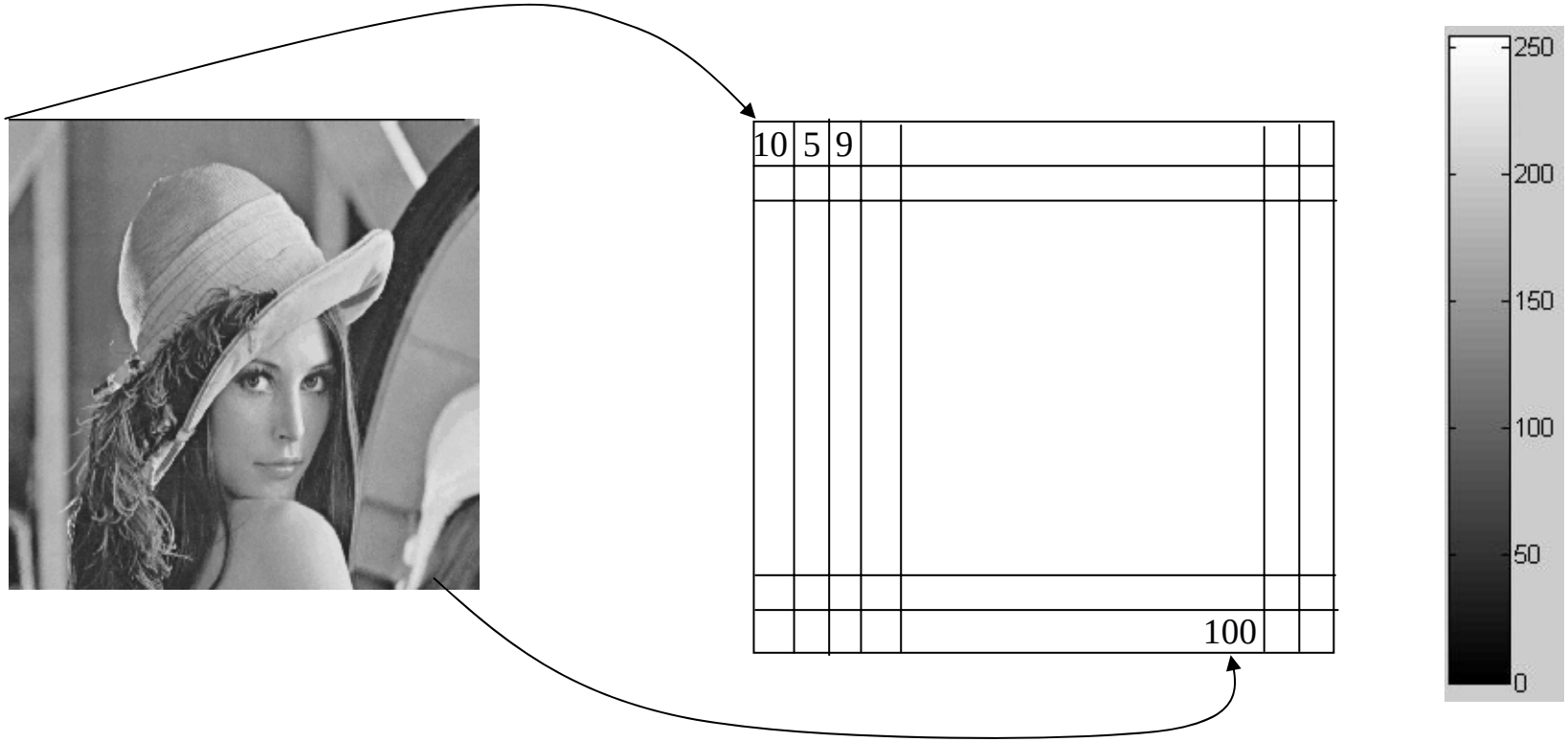
General

- Binary
- Gray Scale
- Color

Binary Images

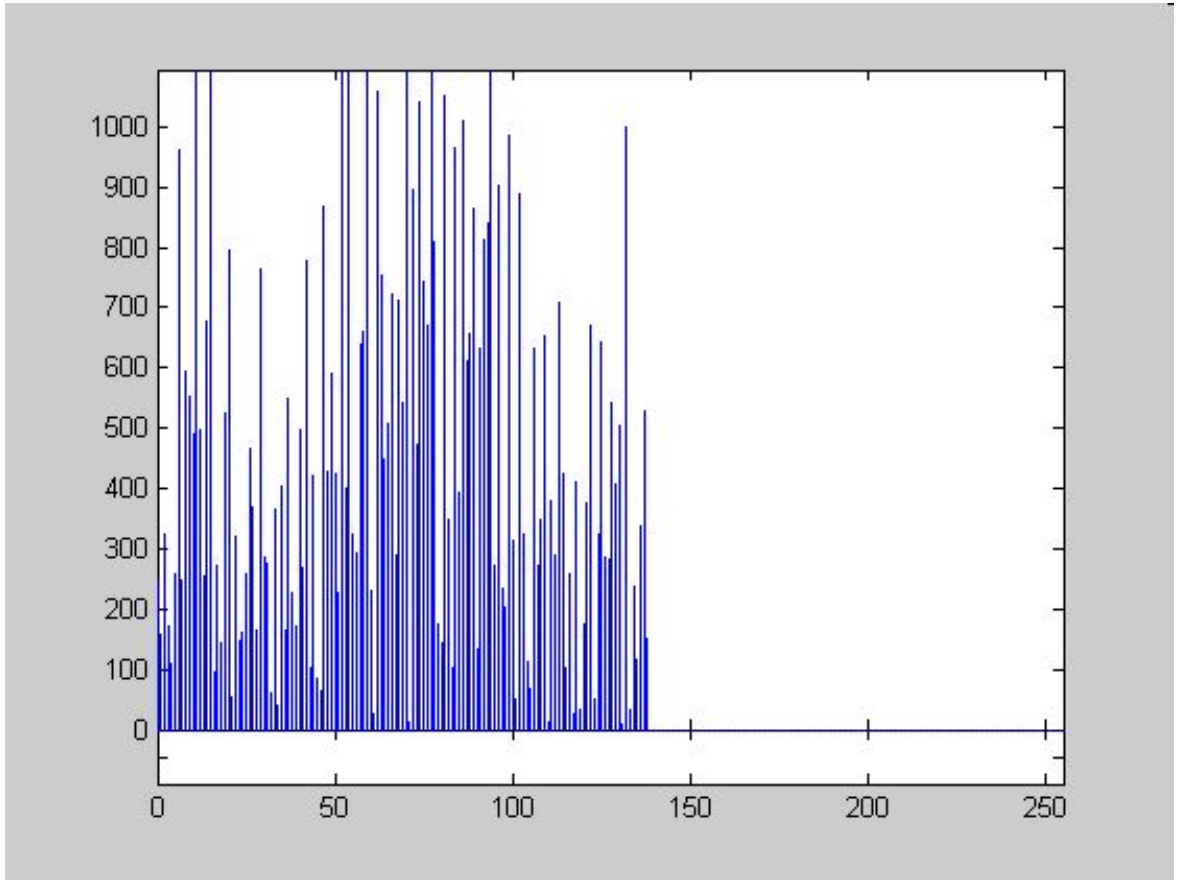


Gray Level Image



Color Image
Red, Green, Blue Channels

Image Histogram



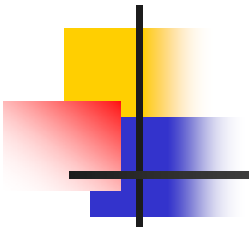


Image Noise

- Light Variations
- Camera Electronics
- Surface Reflectance
- Lens

Image Noise

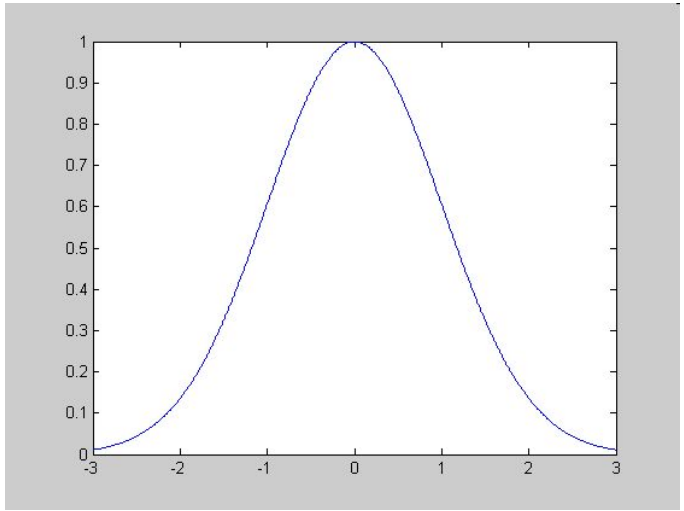
- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

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

➔

Gaussian Noise

$$n(x,y) = e^{\frac{-n^2}{2\sigma^2}}$$

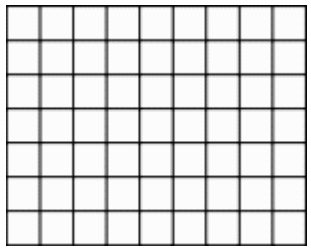


Salt & Pepper Noise

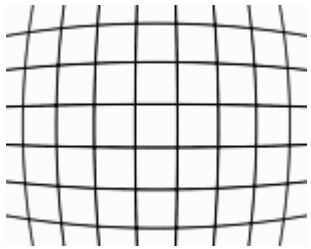
$$\hat{I}(x,y)=\begin{cases} I(x,y) & p<l \\ s_{\min}+r(s_{\max}-s_{\min}) & p\geq l \end{cases}$$

- p is uniformly distributed random variable
- l is threshold
- s_{min} and s_{min} are constant

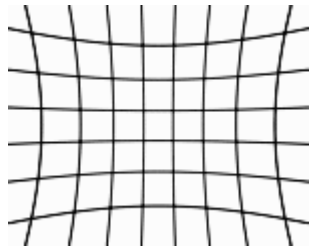
Correcting Lens distortion



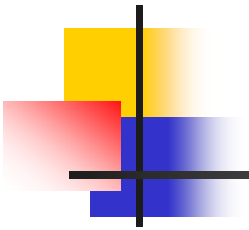
None



Barrel



Pincushion



图像和图像编码

- 衡量图像编码算法的三个主要指标
  - 压缩比（Compression ratio）
  - 失真度量（Distortion Measure）
  - 算法复杂度（Computational Costs）

图像和图像编码

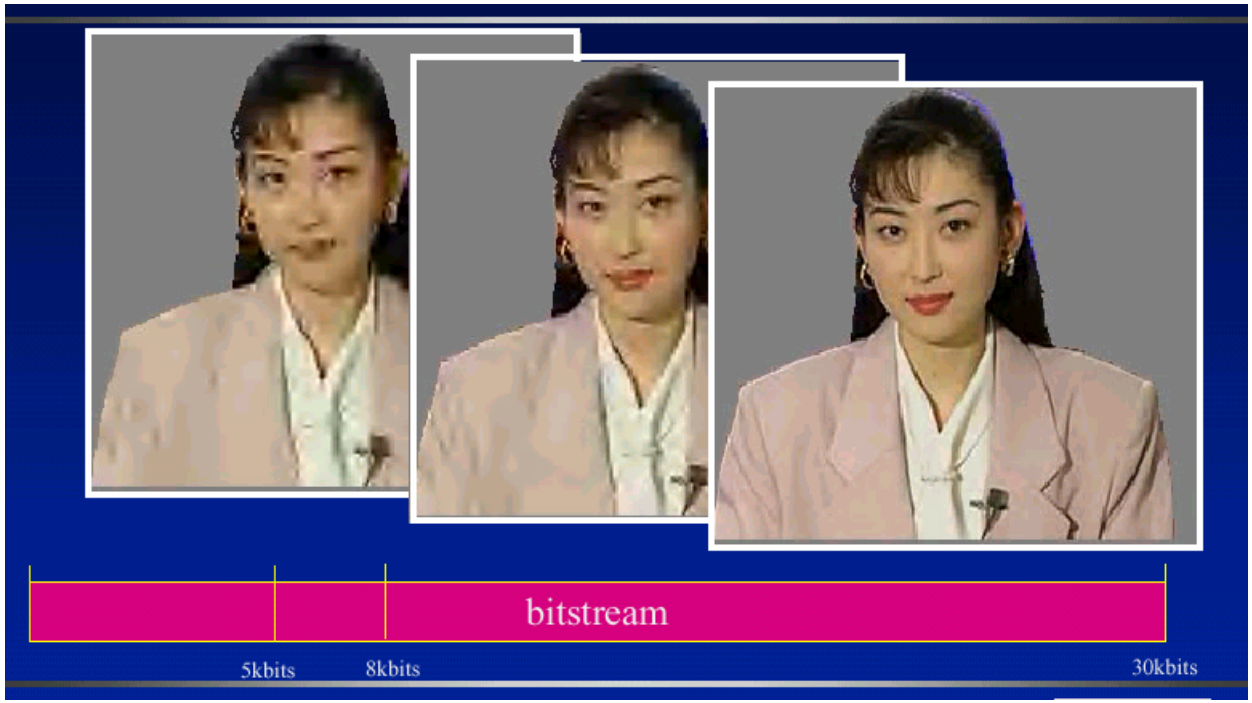
- 可分级性（Scalability）
  - 图像空间分辨率的可分级性
  - 图像解码重构质量的可分级性
  - 视频解码帧速率的可分级性

图像和图像编码

- 图像空间分辨率可分级性（Resolution Scalability）

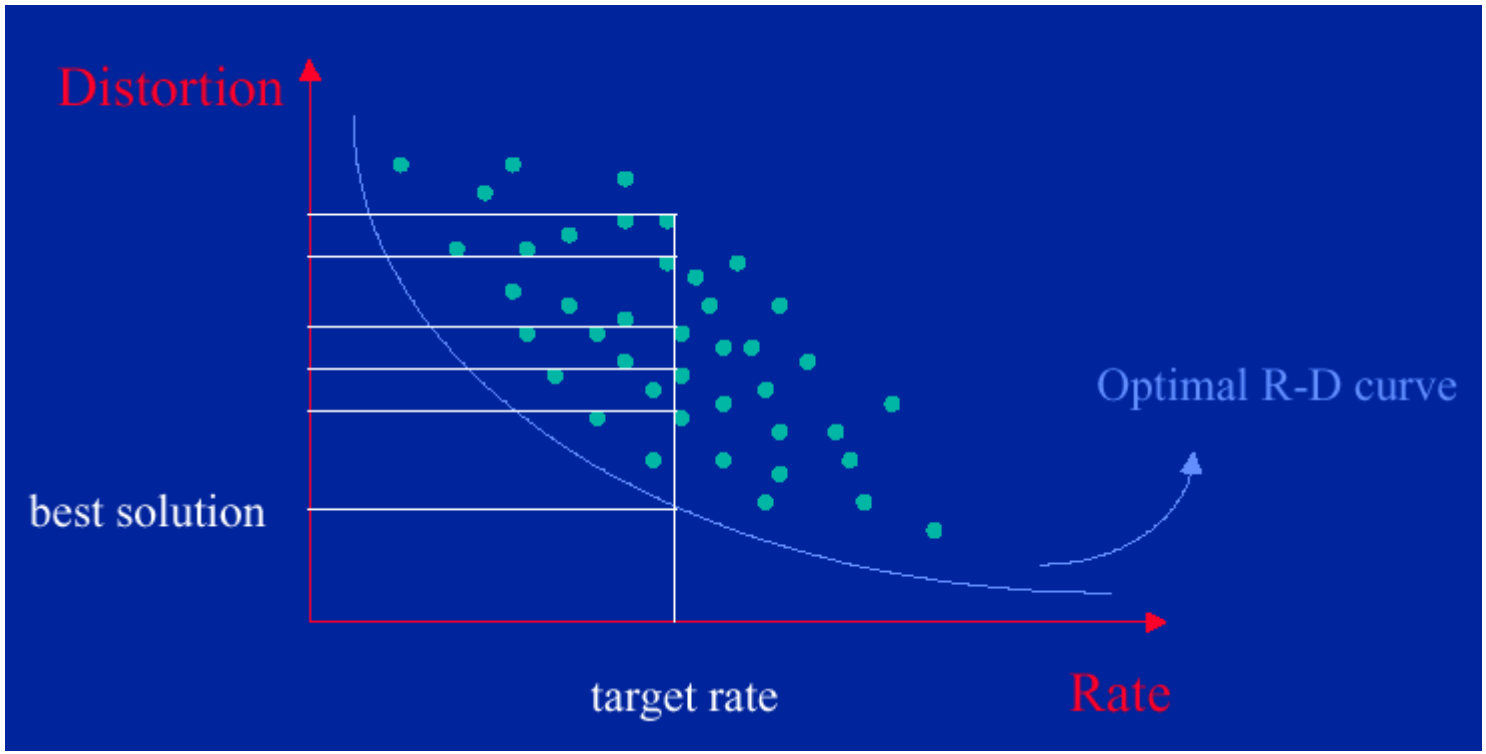
图像和图像编码

- 图像解码质量的可分级性（SNR Scalability）



率失真特性

- 一条典型的率失真特性曲线

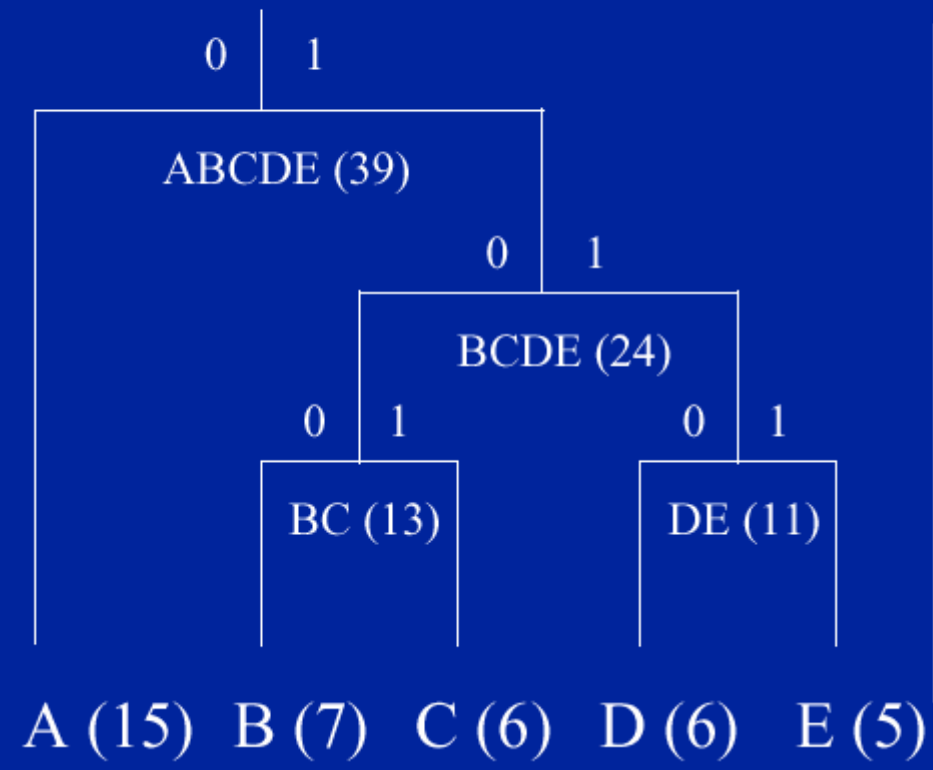


图像和图像编码

- 常用的图像编码方法
  - 预测编码（Motion estimation/compensation ）
  - 变换编码（Discrete cosine transform）
  - 统计编码/熵编码（Huffman/Arithmetic coding）
  - 向量量化编码（Vector quantization coding）
  - 子带编码（Subband coding）
  - 分形编码（Fractal image coding）
  - 小波编码（Wavelet-based image coding）

Huffman编码

Huffman coding (example)



symbol	code
A	0
B	100
C	101
D	110
E	111

分形图像编码

- 自相似性—分形蕨



分形图像编码

- Lena图像中的自相似性

分形图像编码

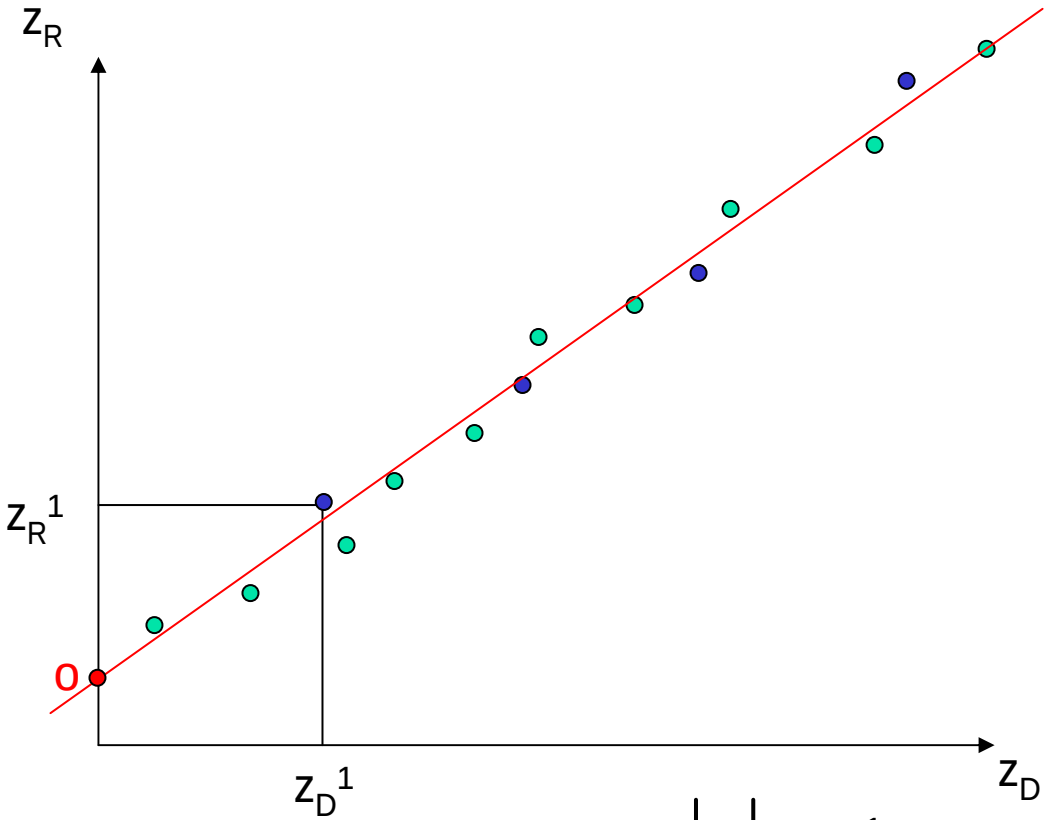
- 仿射变换（基于收缩映射）

$$\omega_i\left(\begin{bmatrix} x_D \\ y_D \\ z_D \end{bmatrix}\right) = \begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x_D \\ y_D \\ z_D \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}$$

  - 位置变换：From (x_D,y_D) to (x_R,y_R)
  - 灰度变换

$$z_R = s \cdot z_D + o, |s| \le 1$$

Line Fitting

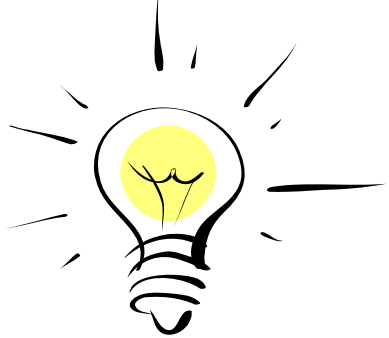


$$z_R = s \cdot z_D + o, |s| \leq 1$$

思考题

- 图像到自身存在完美的全等映射，全等映射显然是一个仿射变换……
- 看来只需要记录一个映射参数就可以完成高压缩比的分形编码……
- 这个理想方法可行吗？问题出在哪里？

答案



- “收缩”是IFS收敛到不动点的必要条件！

IFS	Fixed point
$X_{i+1}=X_i/3$	0
$X_{i+1}=X_i/2+1$	2
$X_{i+1}=sqrt(X_i)+2$	4
$X_{i+1}=X_i$?	Starting X

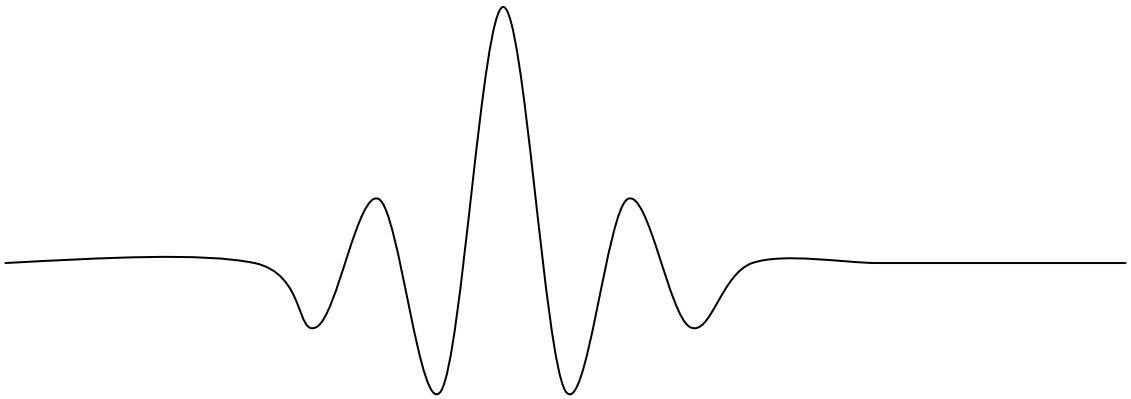
- 每一图像都是相应迭代函数系统的不动点
- 不同的图像对应不同的迭代函数系统

小波图像编码

- 频域表示的不足之处
  - 对非平稳信号用Fourier变换进行分析不能提供完全的信息，即利用Fourier变换虽然可以知道信号所含有的频率信息，但不能知道这些频率信号究竟出现在哪些时间段上。
- 小波函数的特点
  - 小—在时域都具有紧支集或近似紧支集。
  - 波动性—具有正负交替的震荡波形。

小波图像编码

- 一个典型的小波函数波形



小波图像编码

- 小波图像编码的数学理论奠基人是I. Daubechies et. al，广泛投入实用研究则起自S. Mallat于20世纪80年代末提出的倍频程信号分解算法。
- 信号的小波分析的基本原理：将某一尺度空间 分解为一个较粗尺度空间 与其正交补空间 的直和，进而，信号在就可以分解为一个近似描述（即它在粗尺度空间 上的投影）和一个细节描述（即它在正交补空间 上的投影）。信号的主要能量集中在它的近似描述中，细节描述则是对近似描述的补充表示。

小波图像编码

- 倍频程信号分解算法（S. G. Mallat）

$$A_j f(t) = D_{j-1} f(t) + A_{j-1} f(t)$$

$$= D_{j-1} f(t) + D_{j-2} f(t) + A_{j-2} f(t)$$

$$= D_{j-1} f(t) + D_{j-2} f(t) + D_{j-3} f(t) + A_{j-3} f(t)$$

$$= \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

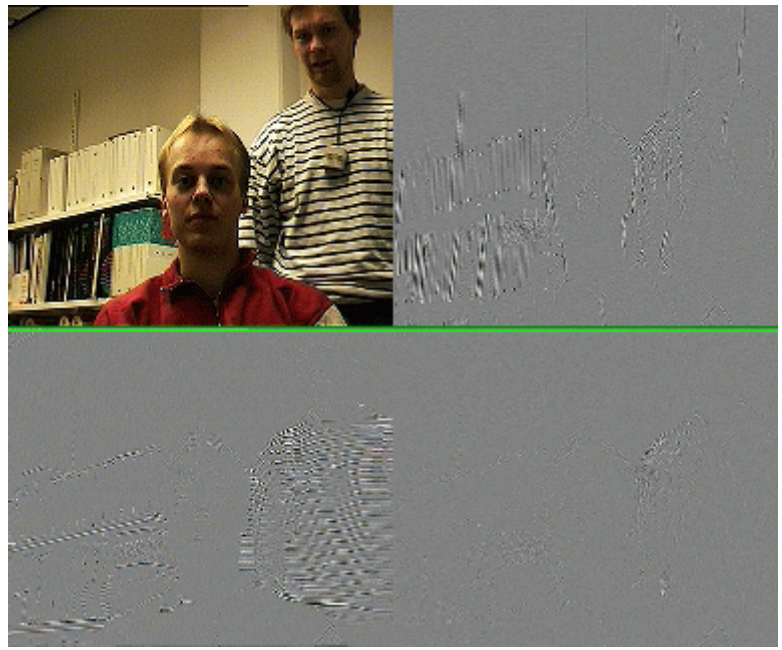
$$= D_{j-1} f(t) + D_{j-2} f(t) + D_{j-3} f(t) + \cdots + D_{j-n} f(t) + A_{j-n} f(t)$$

- 上式是S. Mallat建立的金字塔式离散小波变换算法的一个简略数学表达。其本质是信号的渐进分解和分而治之的思想。从上式式中体现的逐级分解形式可以直观地看出小波分解的潜在可分级性能。

小波图像编码

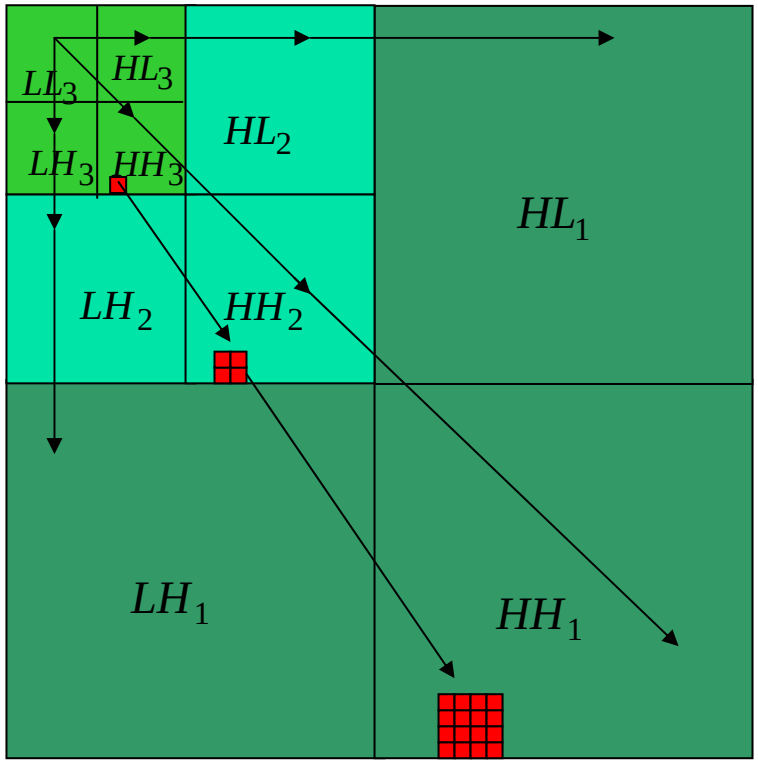
- 小波分解的时频特征

LL	*HL*
LH	*HH*



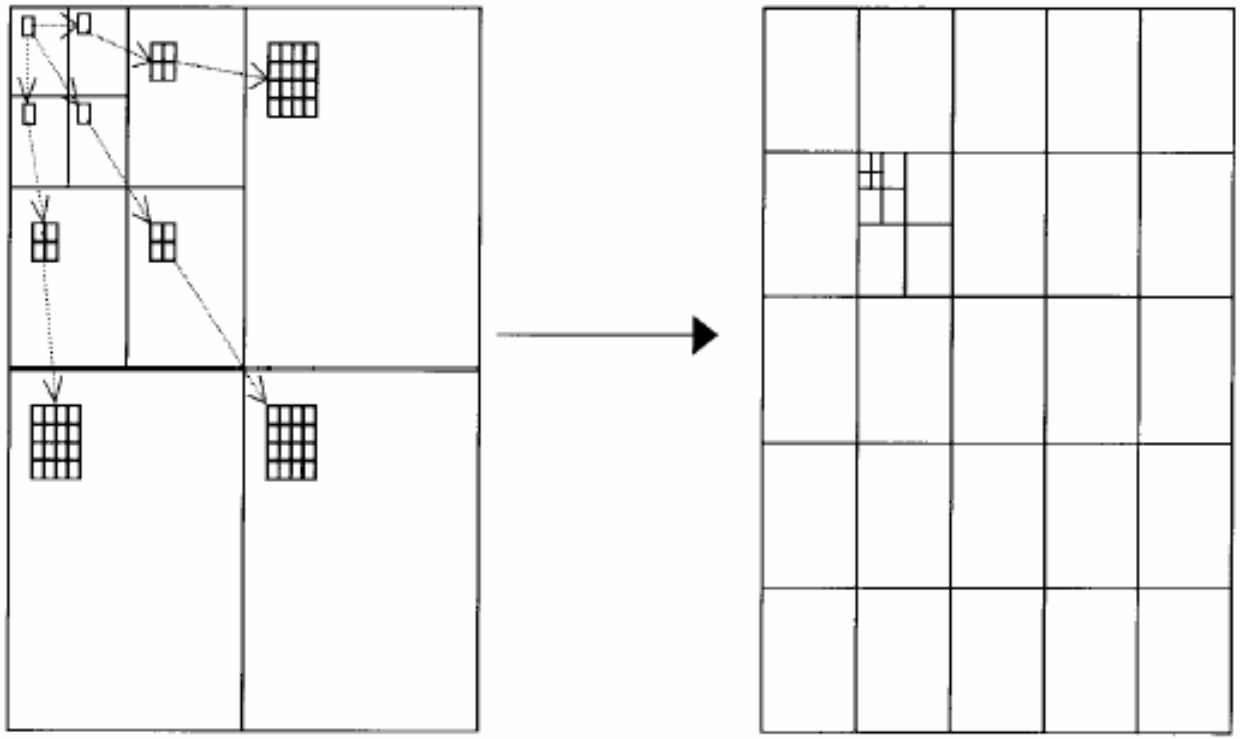
小波图像编码

- 小波分解图像中系数分布的零树结构



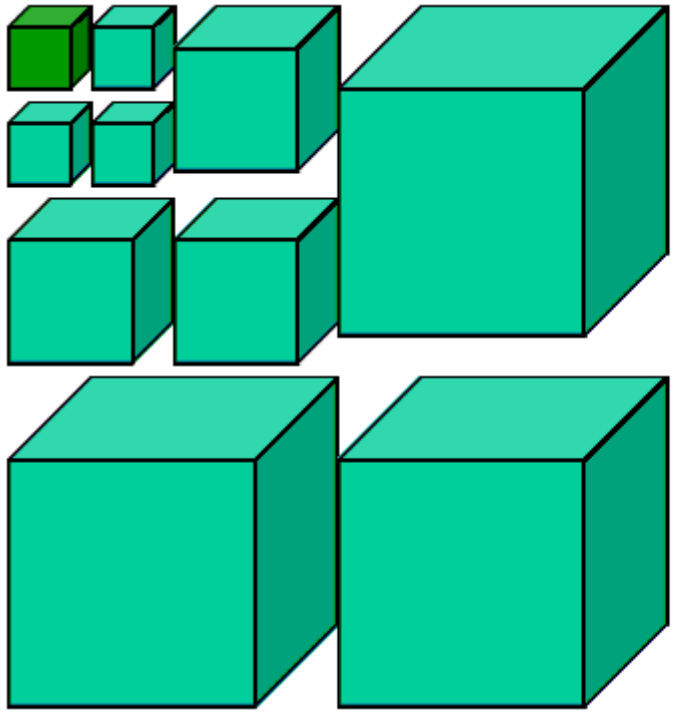
小波图像编码

- 小波分解图像中系数分布的零树结构（续）



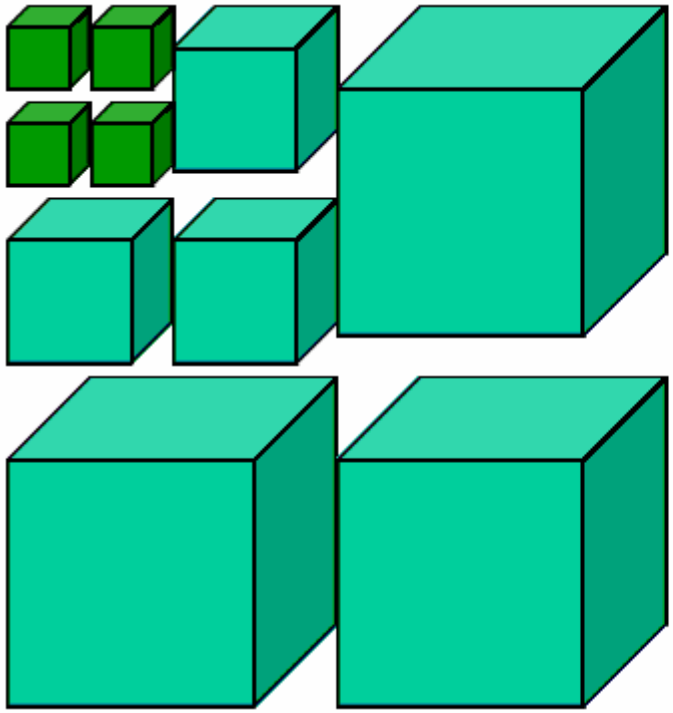
JPEG2000

- Scalability-Progressive by resolution



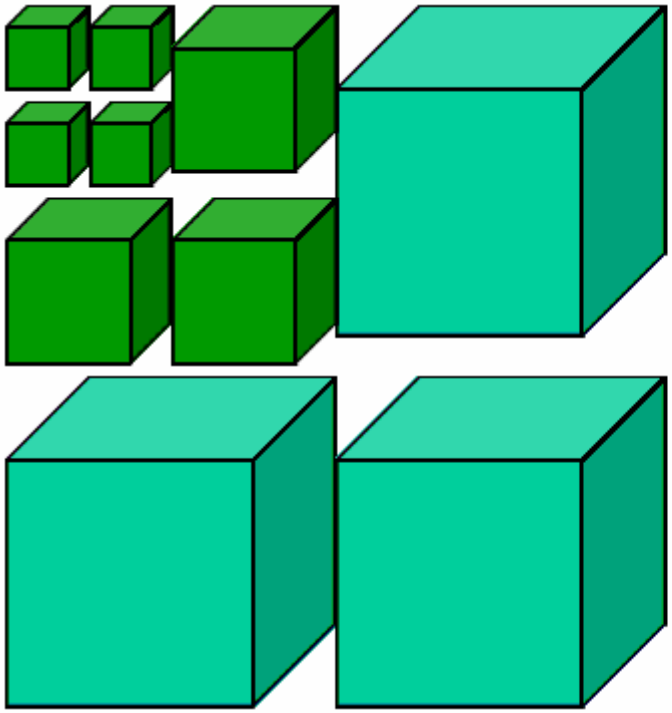
JPEG2000

- Scalability-Progressive by resolution



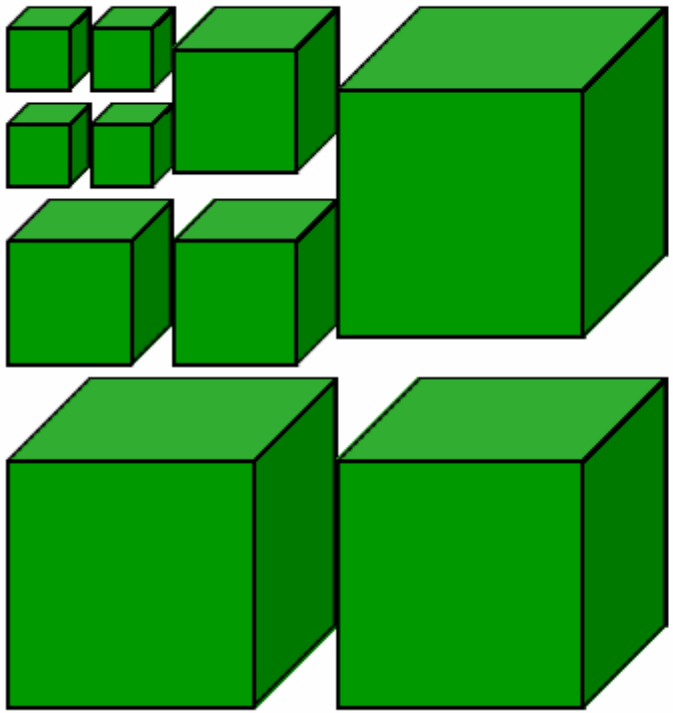
JPEG2000

- Scalability-Progressive by resolution



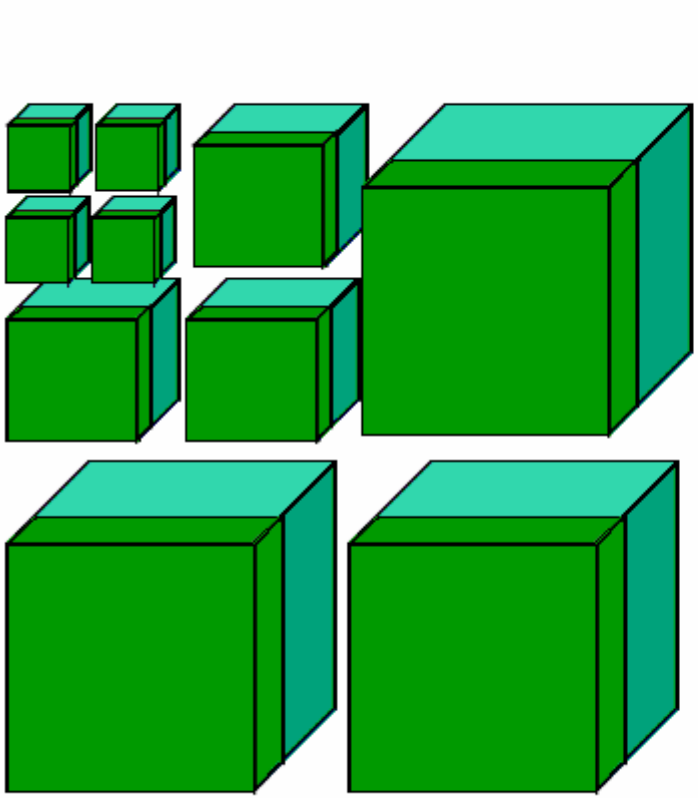
JPEG2000

- Scalability-Progressive by resolution



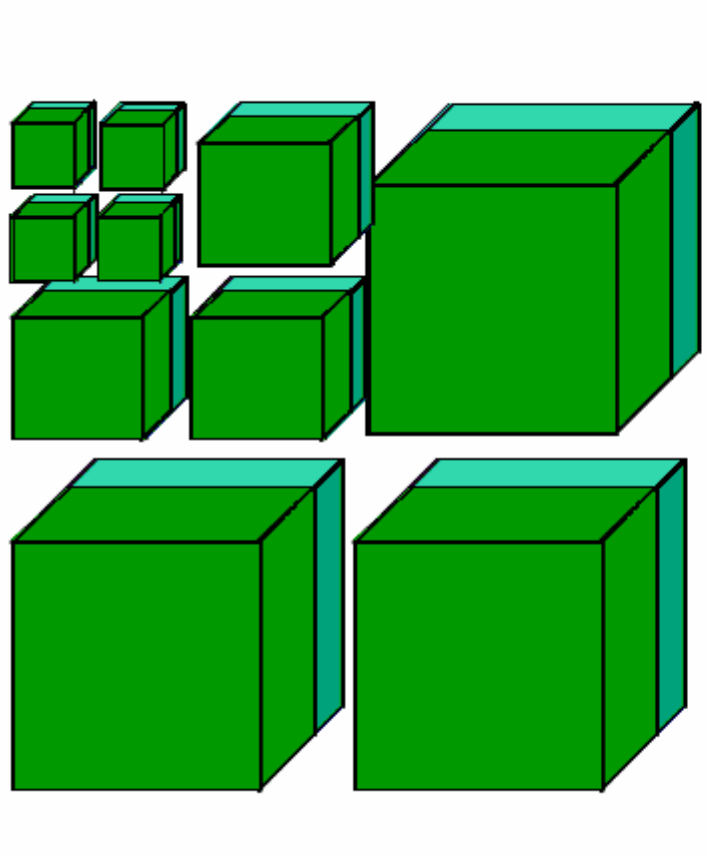
JPEG2000

- Scalability-Progressive by accuracy



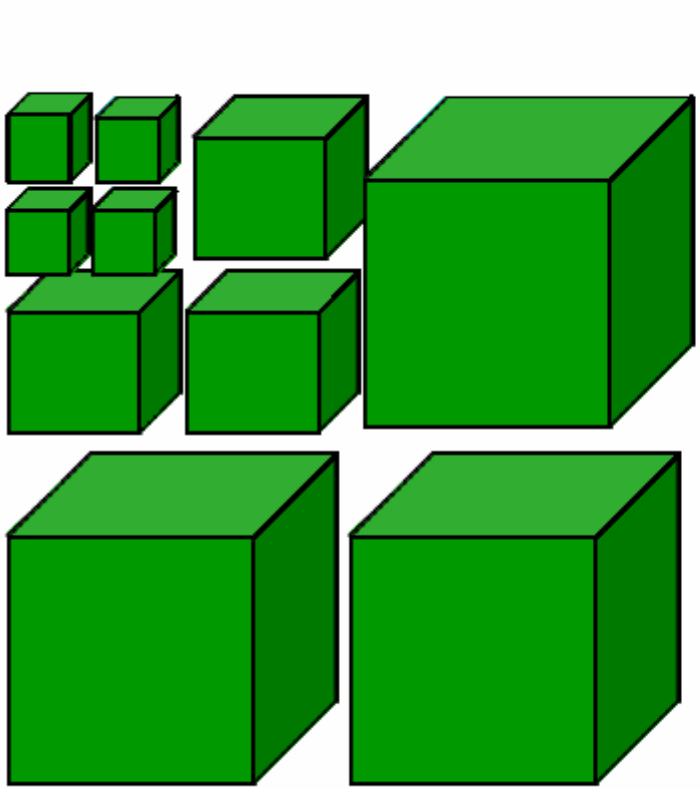
JPEG2000

- Scalability-Progressive by accuracy



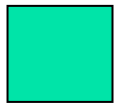
JPEG2000

- Scalability-Progressive by accuracy

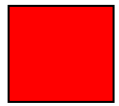


Error Resilience

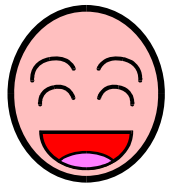
- More Examples

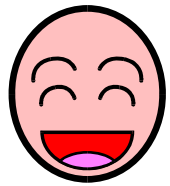


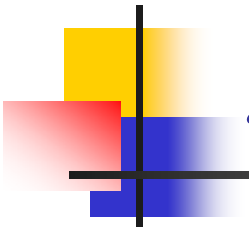
:Data packet,



: Protecting packet





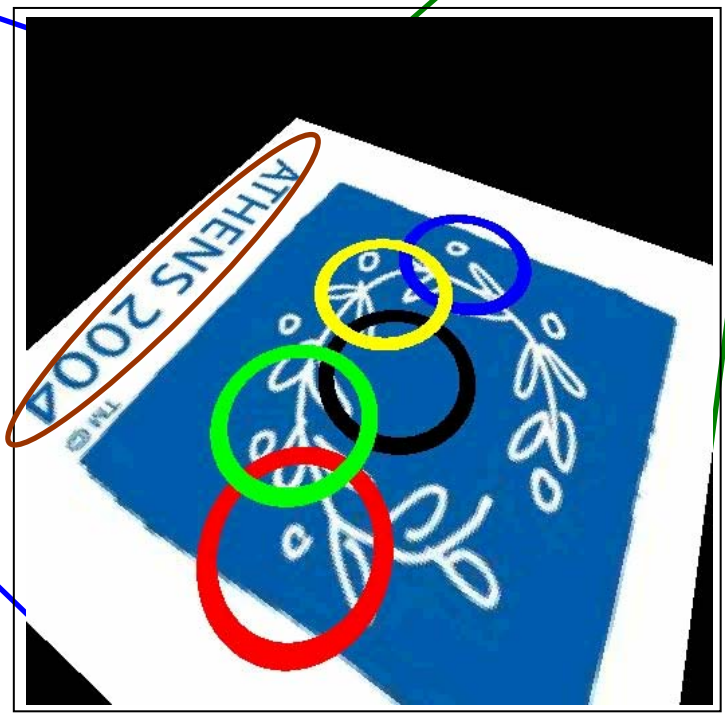


总结

- 几点最重要的思想
  - 分而治之的思想（Divide and Conquer）
  - 逐步求精的思想（Scalability & Iteration）
  - 动态规划的思想（Dynamic Programming）

- 请提问

What can we do with an image?



Object Detection/Recognition

Curve Detection/Fitting

Line Detection/Fitting

Key Feature Estimation

Scene Editing/Augmentation

Image Processing

- Define a new image *g* in terms of an existing image *f*
  - We can transform either the domain or the range of *f*
- Range transformation:

$$g(x, y) = t(f(x, y))$$

What kinds of operations can this perform?

Smoothing, Enhancing, Denoising, Binarizing……

Image Processing

- Some operations preserve the range but change the domain of f :

$$g(x, y) = f(t_x(x, y), t_y(x, y))$$

- What kinds of operations can this perform?

Translation, Rotation, Scaling……

Image Processing

- Still other operations operate on both the domain *and* the range of f .

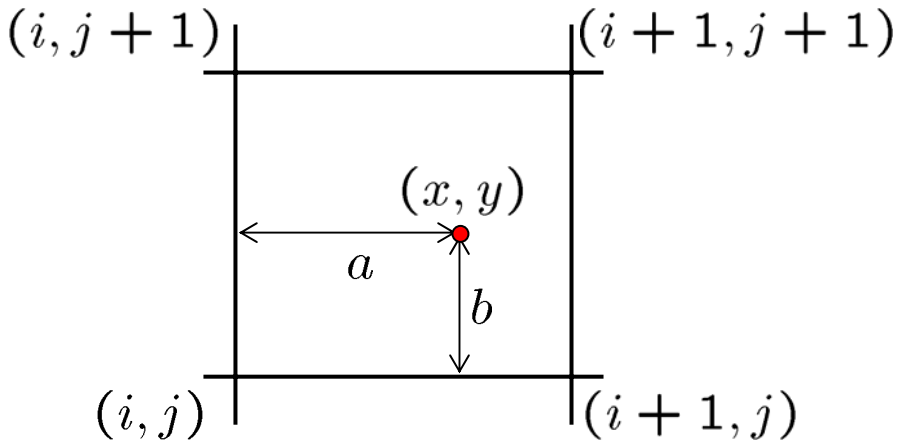
$$g(x,y) = s(f(t_x(x,y), t_y(x,y)))$$

- What kinds of operations can this perform?

Fractal Image coding, Wavelet Image coding……

Bilinear Interpolation(双线性插值)

- A simple method for resampling images



$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) \quad f[i,j] \\
& +a(1-b) \quad\quad f[i+1,j] \\
& +ab \quad\quad\quad f[i+1,j+1] \\
& +(1-a)b \quad\quad f[i,j+1]
\end{aligned}
$$

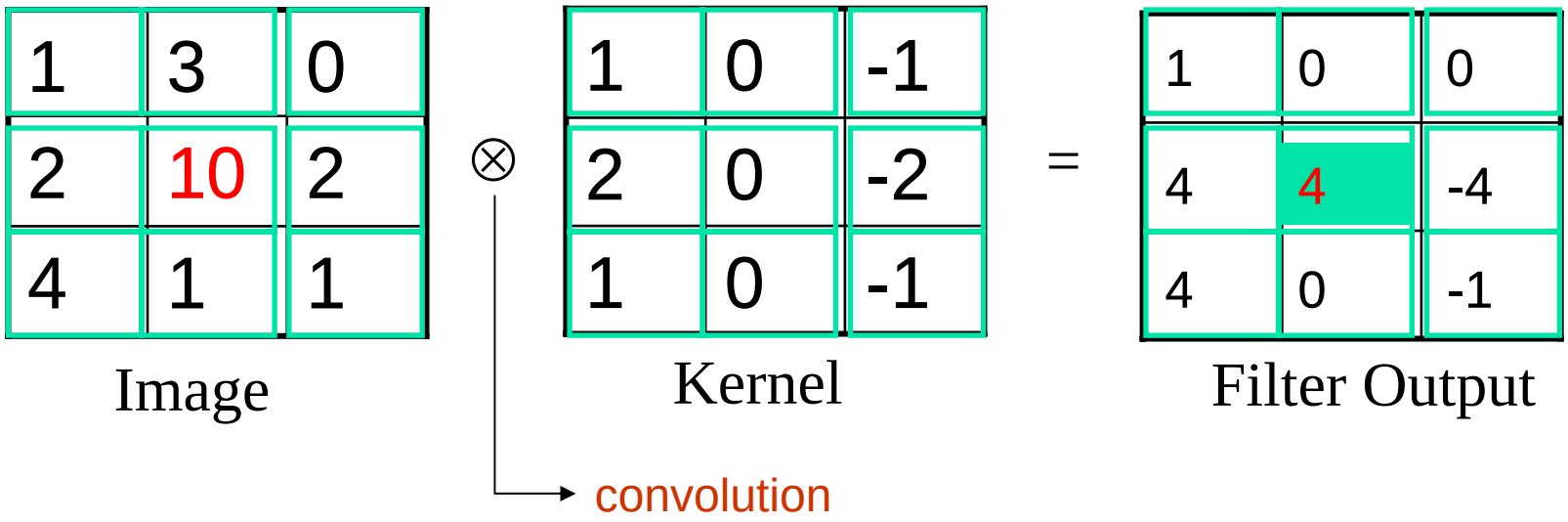
Normalized Correlation

Account for energy differences

$$N_{tf}(i,j) = \frac{\sum_m \sum_n t(m-i, n-j) f(m,n)}{\left[\sum_m \sum_n t^2(m-i, n-i)\right]^{1/2} \left[\sum_m \sum_n f^2(m,n)\right]^{1/2}}$$

Linear Filtering

- The output is the linear combination of the neighborhood pixels
- Weighted Sum(加权和)



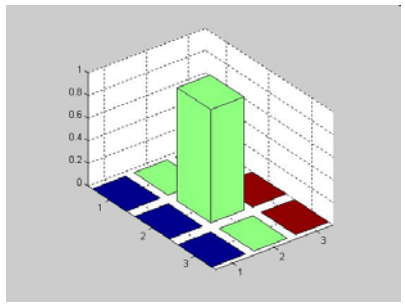
Average Filter(平均滤波器)

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

F

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Filtering Examples



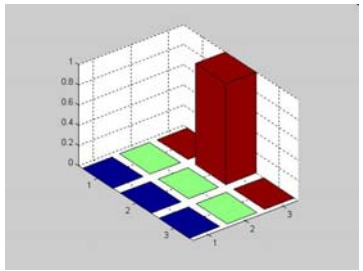
$*$

0	0	0
0	1	0
0	0	0

$=$

Filtering Examples

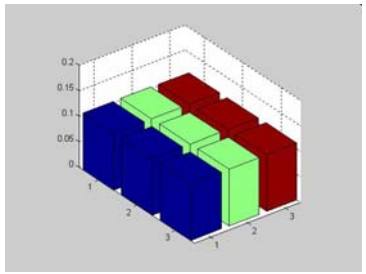
*



0	0	0
0	0	1
0	0	0

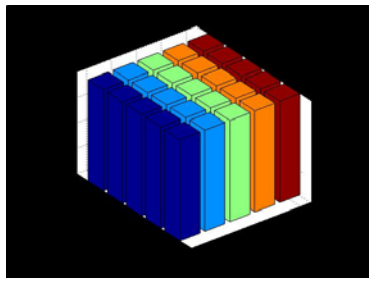
=

Filtering Examples



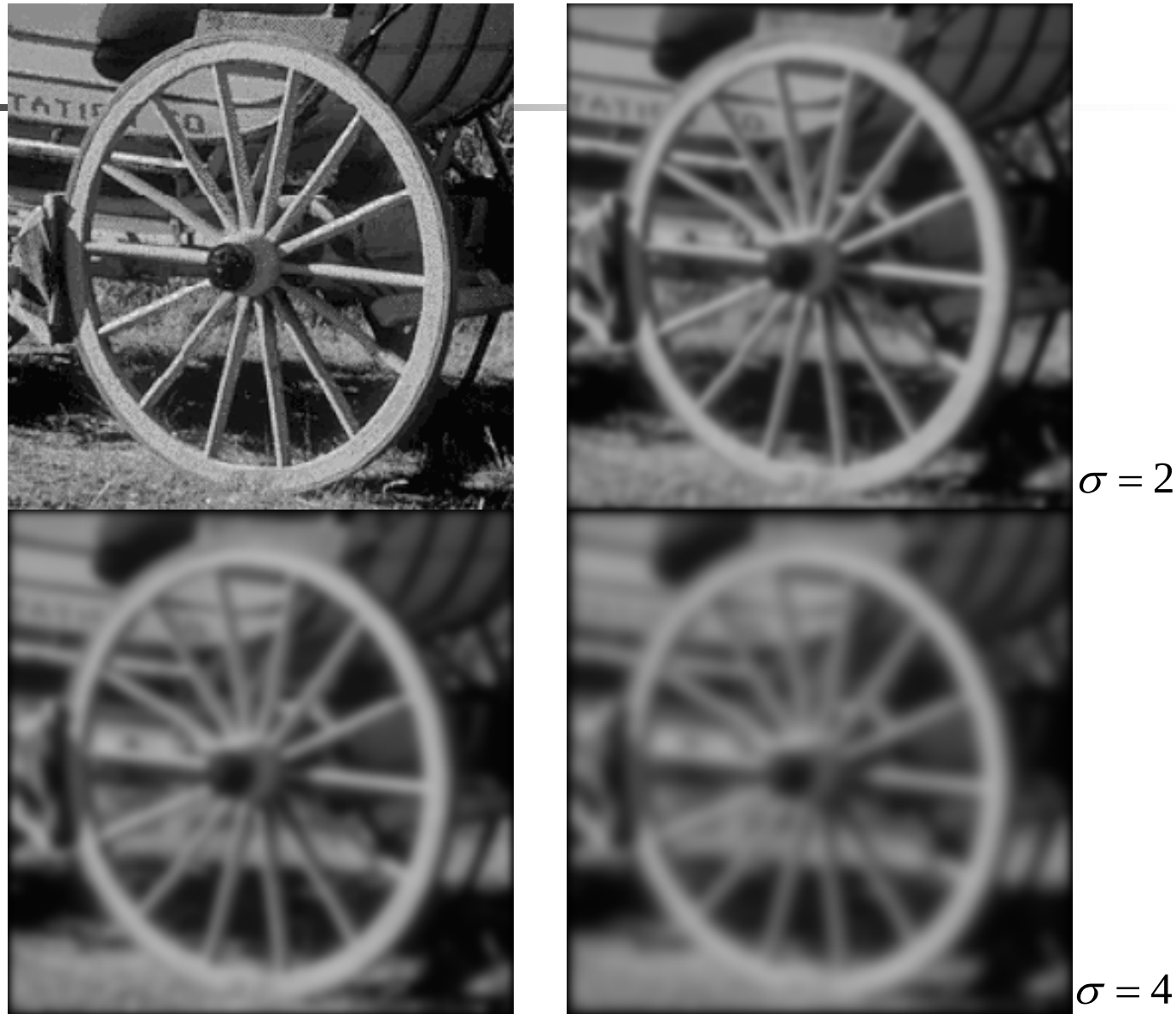
$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Filtering Examples



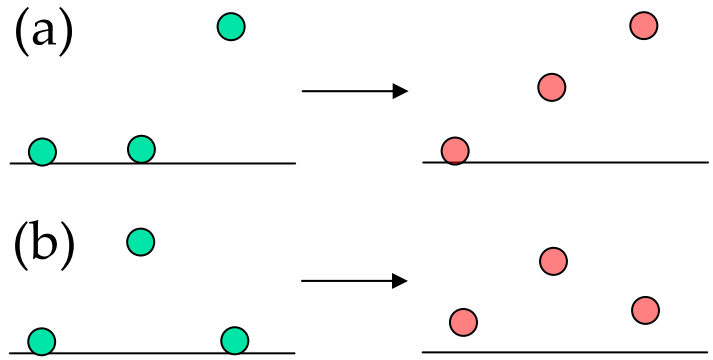
$$* \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

Gaussian Smoothing

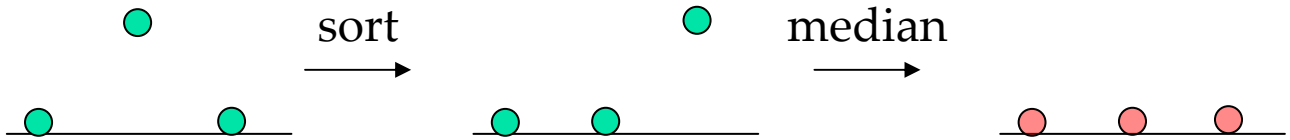


Median Filter(中值滤波)

- Smoothing is averaging
  - (a) Blurs edges
  - (b) Sensitive to outliers



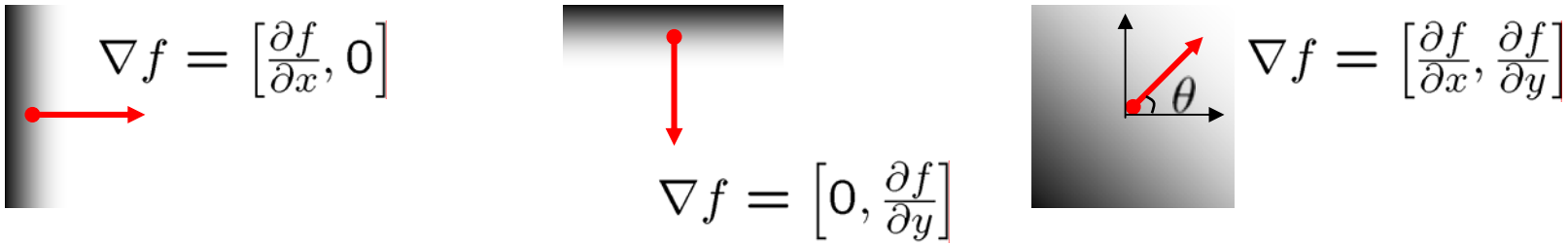
- Median filtering
  - Sort N^2-1 values around the pixel
  - Select middle value (median)



  - Non-linear (Cannot be implemented with convolution)

Image gradient(梯度)

- 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 change in intensity



- The gradient direction is given by:

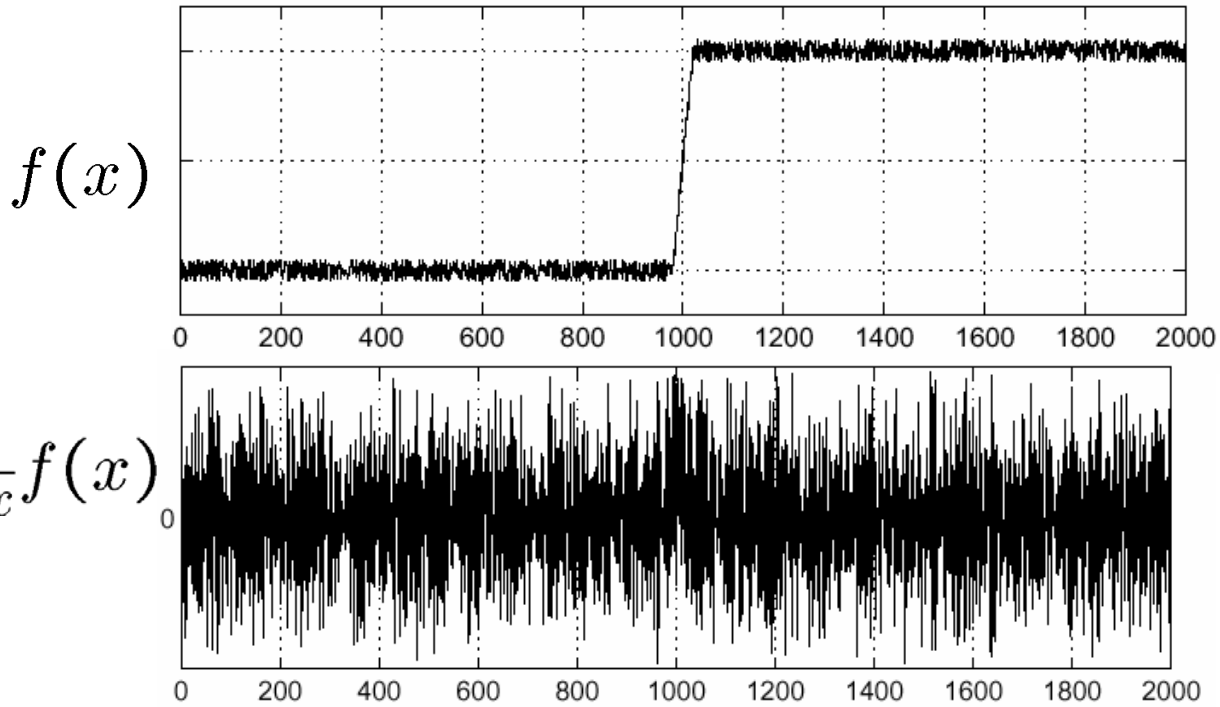
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

  - how does this relate to the direction of the edge?
- The *edge strength* is given by the gradient magnitude

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

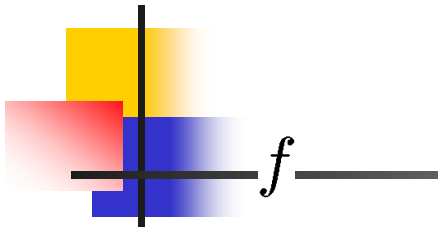
Effects of noise(噪声的影响)

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

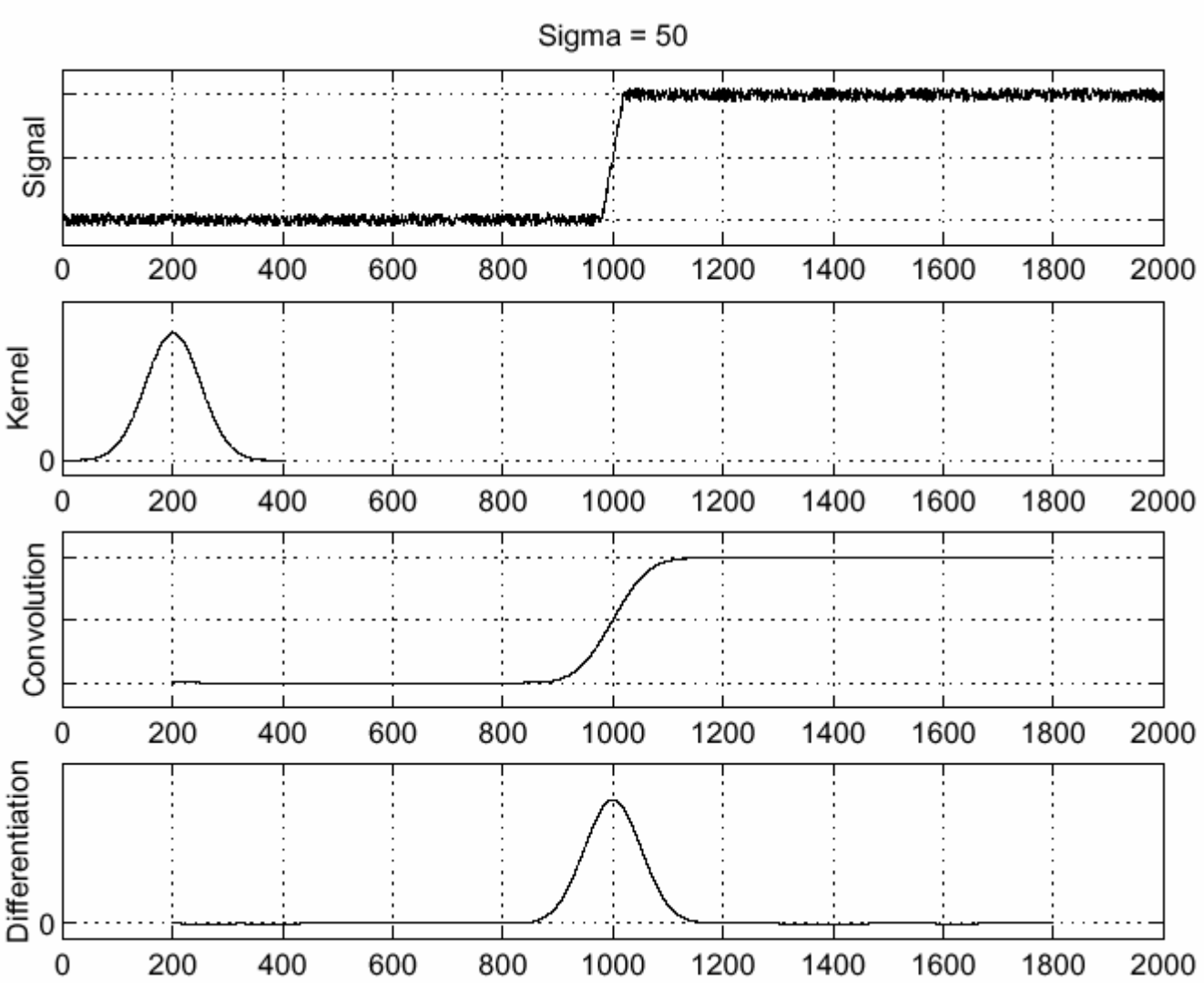


- Where is the edge?

Solution: smooth first



f



h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

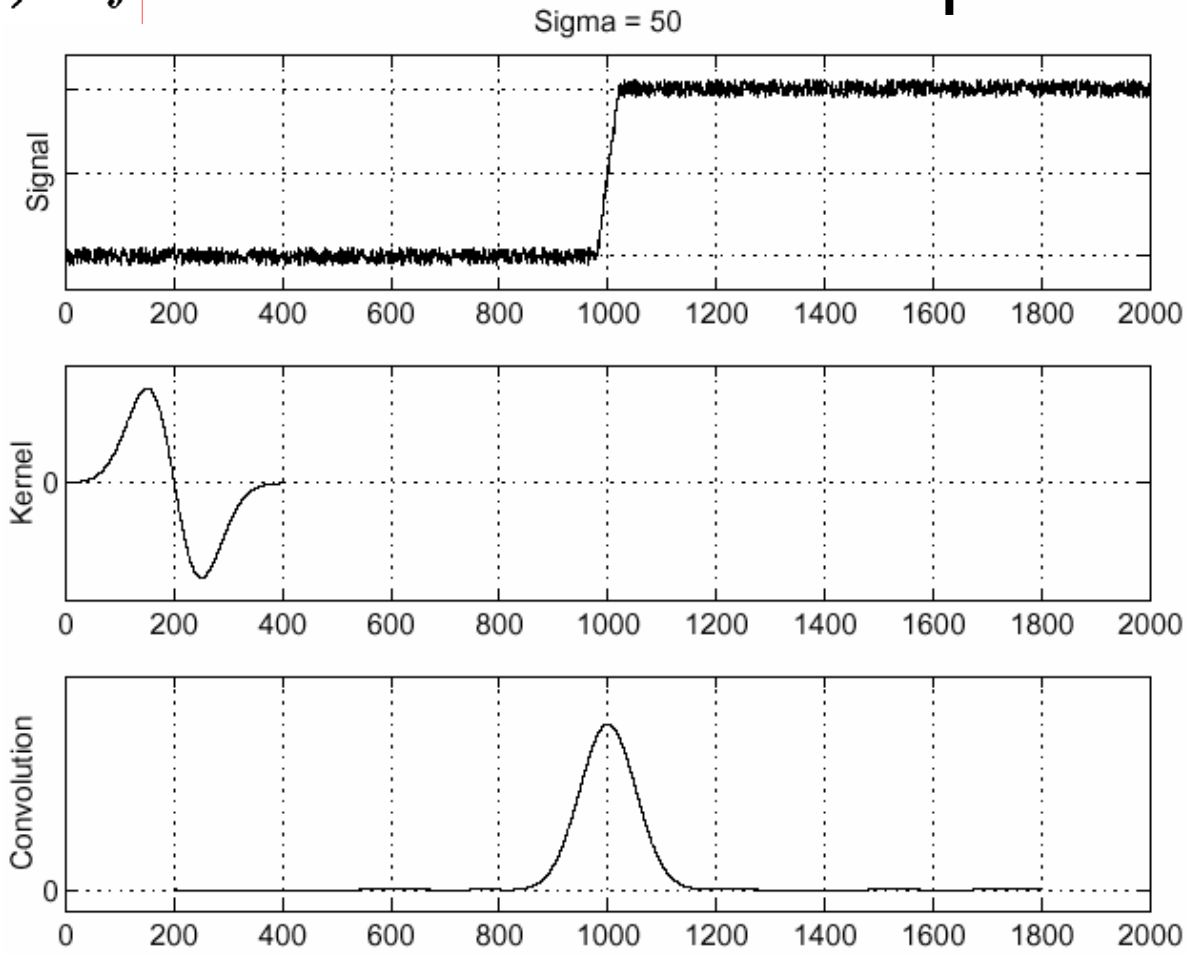
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

f

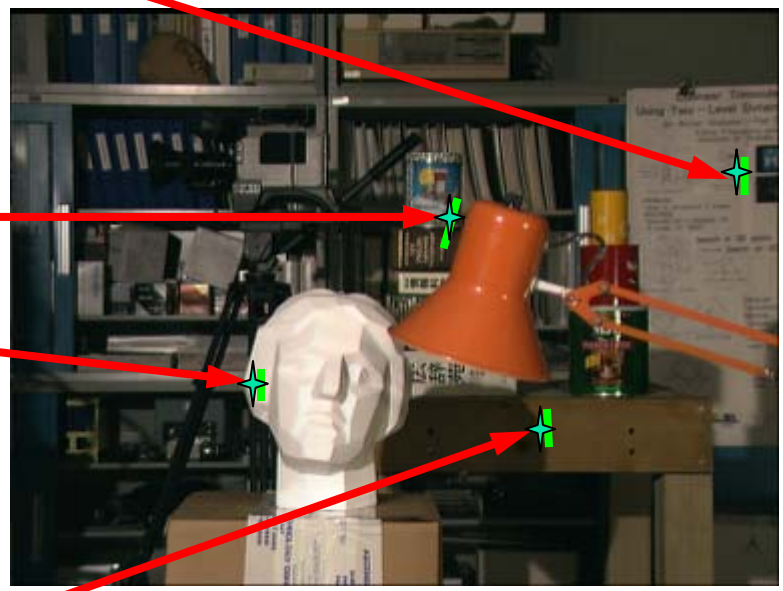
$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



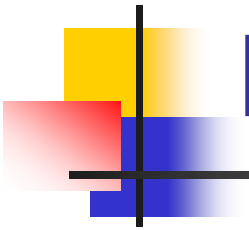
What Causes an Edge?

- Reflectance discontinuity (i.e., change in surface material properties)
- Depth discontinuity
- Surface orientation discontinuity
- Illumination discontinuity (e.g., shadow)



Prewitt and Sobel Edge Detector

- Compute derivatives
  - In x and y directions
- Find gradient magnitude
- Threshold gradient magnitude

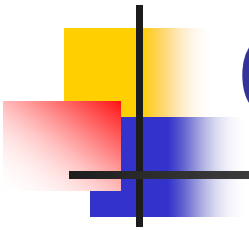


Pyramids

- Many applications
  - small images faster to process
  - good for multiresolution processing
  - compression
  - progressive transmission
- Known as “MIP-maps” in graphics community
- Precursor to wavelets
  - Wavelets also have these advantages

Canny Edge Detector

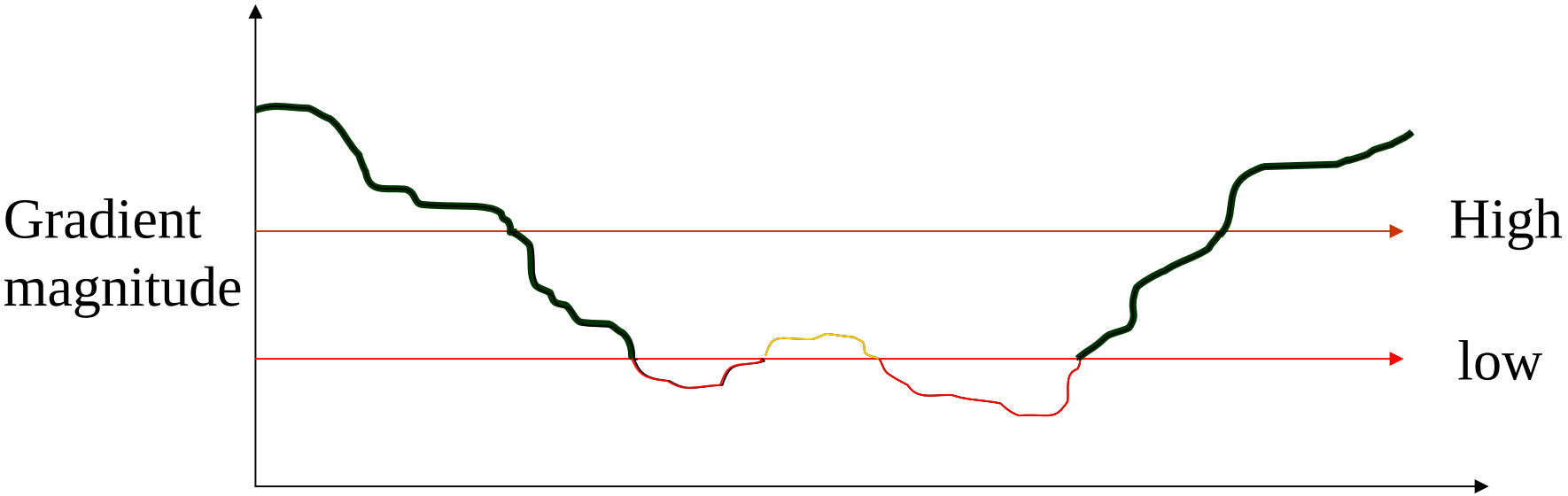
- Criterion 1: Good Detection: The optimal detector must minimize the probability of false positives as well as false negatives.
- Criterion 2: Good Localization: The edges detected must be as close as possible to the true edges.
- Single Response Constraint: The detector must return one point only for each edge point.



Canny Edge Detector Steps

1. Smooth image with Gaussian filter
2. Compute derivative of filtered image
3. Find magnitude and orientation of gradient
4. Apply “Non-maximum Suppression”
5. Apply “Hysteresis Threshold”

Canny Edge Detector
Hysteresis Thresholding



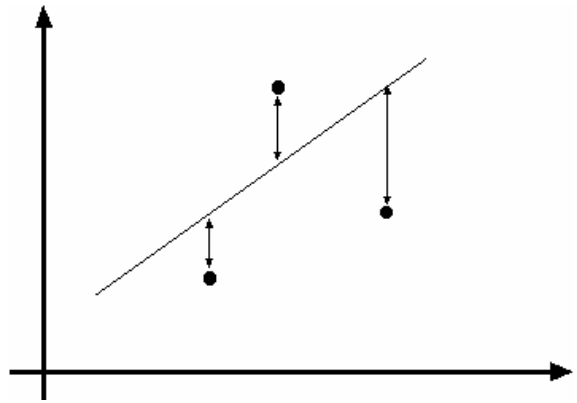
Least Squares Fit(最小二乘拟合)

- Standard linear solution to estimating unknowns.
  - If we know which points belong to which line
  - Or if there is only one line

$$y = ax + b = f(x, a, b)$$

$$\text{Minimize } E = \sum_i [y_i - f(x_i, a, b)]^2$$

Take derivative wrt *a* and *b* set to 0



Advantage of Voting

Example: Two candidates (A & B) run for the president

Assumption: the correct probability of each vote is 0.7

Scheme 1: assigned by the previous president.

Correct rate: 70%=0.7

Scheme 2: general election, the candidate with more votes wins

Correct rate for 3 votes: $C_3^2 \cdot (0.7)^2 \cdot 0.3 + (0.7)^3 = 0.784$

Correct rate for 5 votes: $C_5^3 \cdot (0.7)^3 \cdot (0.3)^2 + C_5^4 \cdot (0.7)^4 \cdot (0.3) + (0.7)^5 = 0.837$

Correct rate for 10,000,000 votes: ……

Recall: Increase transmission reliability by voting

Example: Transmit one “0” or “1” bit in a channel of 80% reliability

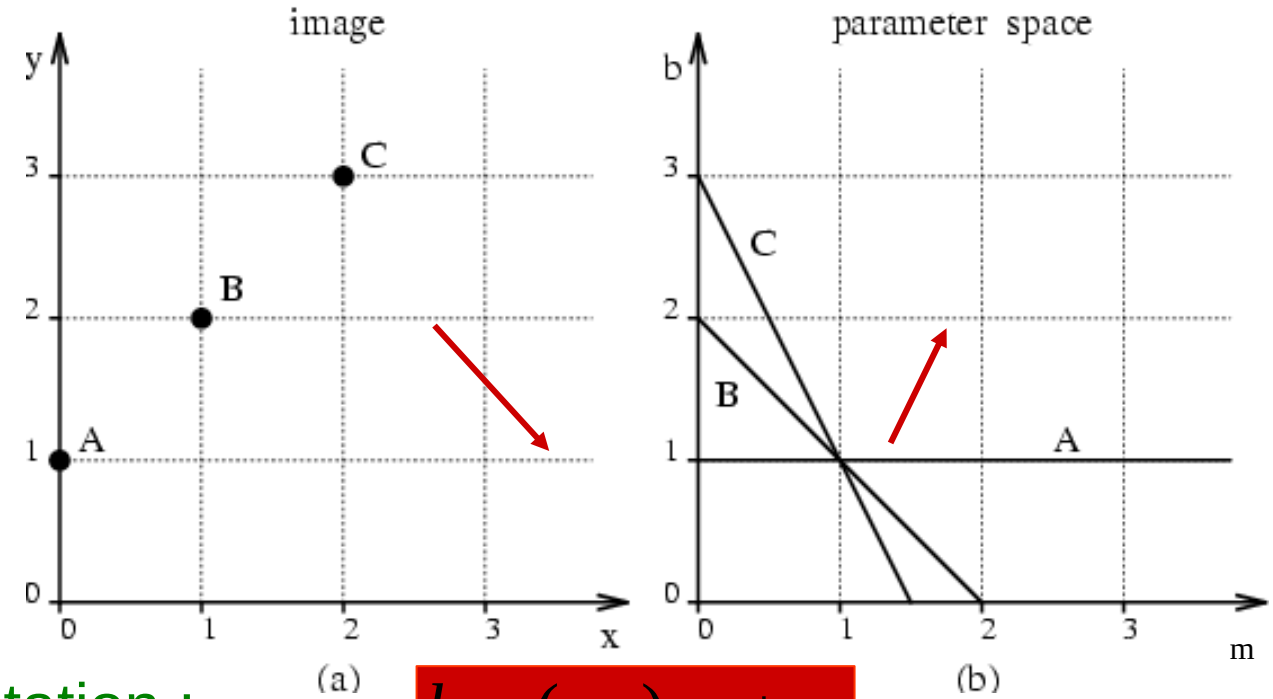
Scheme 1: Directly transmit the bit(0 or 1).

Correct rate: 80%=0.8

Scheme 2: Transmit “000” for the bit 0 and “111” for the bit 1. Take the symbol appears more times in the received sequence as the correct one, eg. “101”->”1” “001”->”0”, “100”->”0”, “111”->”1”

Correct rate: $C_3^2 \cdot (0.8)^2 \cdot 0.2 + (0.8)^3 = 0.896$

Hough transform: Principle



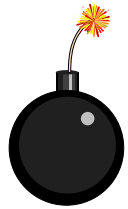
implementation : $b = (-x)\, m + y$

1. the parameter space is discretised
2. a counter is incremented at each cell where the lines pass
3. peaks are detected

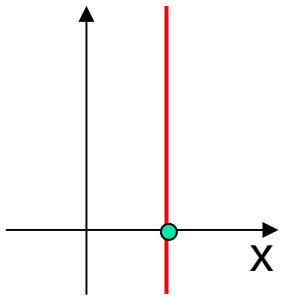
$$b = (-x)\,m + y$$

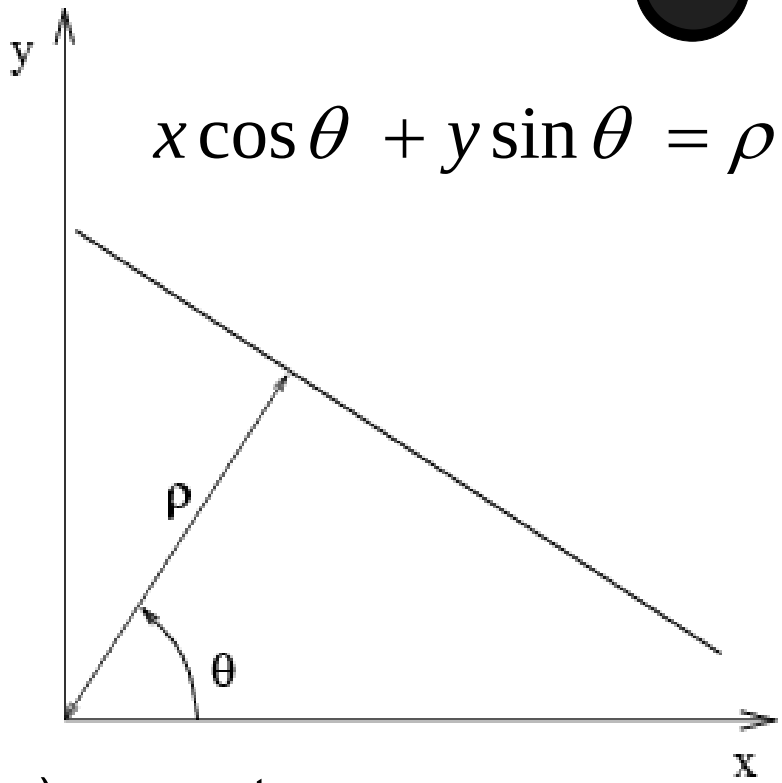
Hough transform: Principle

Problem : unbounded parameter domain, vertical lines require infinite *m*



$m \to \infty$





$$x\cos\theta + y\sin\theta = \rho$$

Solution: alternative representation

θ: determine the slope of the line: *m*

ρ: the distance from the line to the origin

Each point will add a cosine function in the (θ,ρ) parameter space

Hough transform: Principle

$$x\cos\theta + y\sin\theta = \rho$$

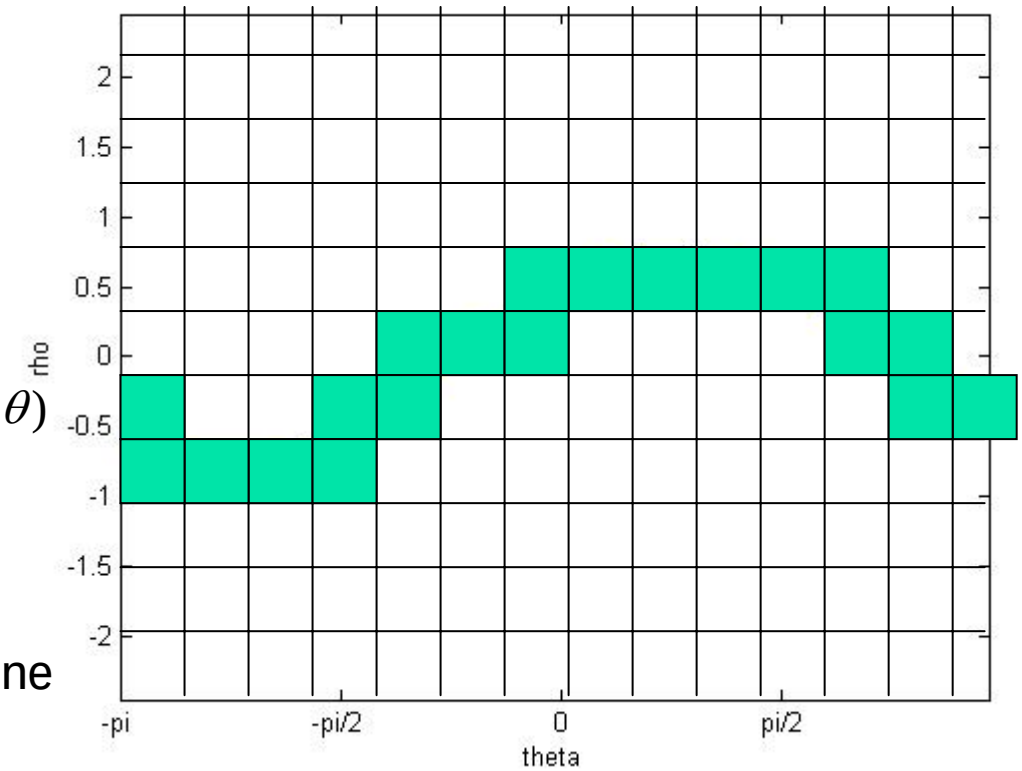
Each green bin get a vote

$$\rho = x\cos\theta + y\sin\theta$$

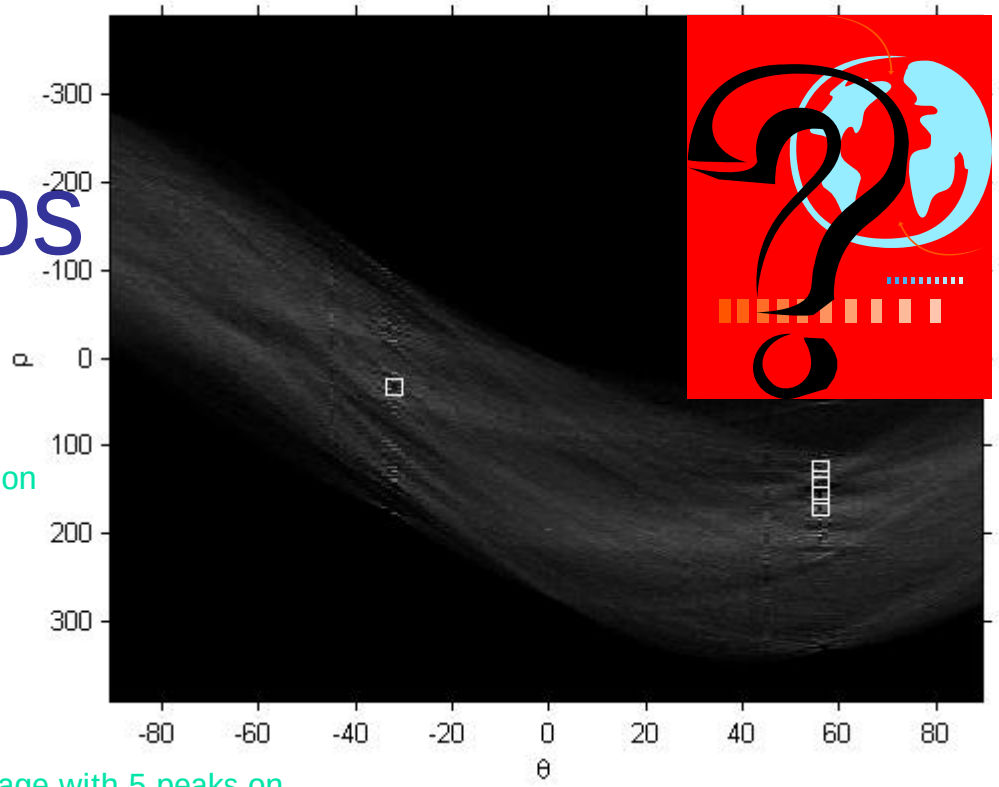
$$= \sqrt{x^2+y^2}\left(\frac{x}{\sqrt{x^2+y^2}}\cos\theta + \frac{y}{\sqrt{x^2+y^2}}\sin\theta\right)$$

$$= \sqrt{x^2+y^2}\,\sin(\theta+\varphi)$$

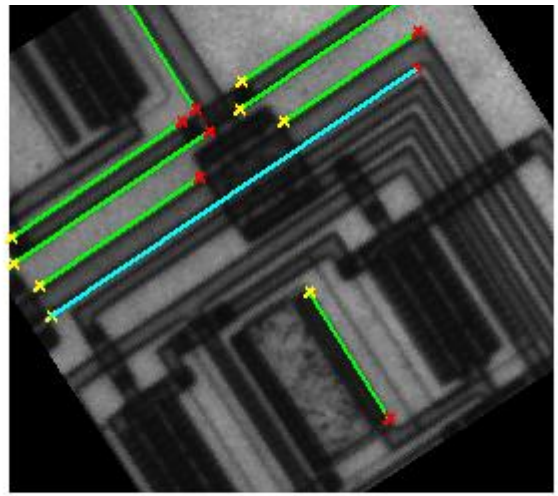
Each feature point corresponds to a sine cuve in Hough space

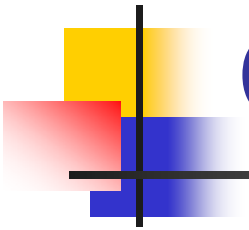


Matlab Demos



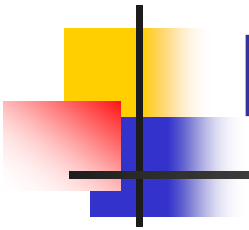

```
I  = imread('circuit.tif');
rotI = imrotate(I,33,'crop'); %rotate to prevent degradation
BW = edge(rotI,'canny');
[H,T,R] = hough(BW);
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
P  = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
x = T(P(:,2)); y = R(P(:,1));
plot(x,y,'s','color','white');      %draw hough transform image with 5 peaks on
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
figure, imshow(rotI), hold on;
max_len = 0;
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');      %green line
   plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');%yellow start point
   plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');    %red end point
   len = norm(lines(k).point1 - lines(k).point2);
   if ( len > max_len)
      max_len = len;
      xy_long = xy;
   end
end
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan');              % highlight the longest line segment
```





Outliers

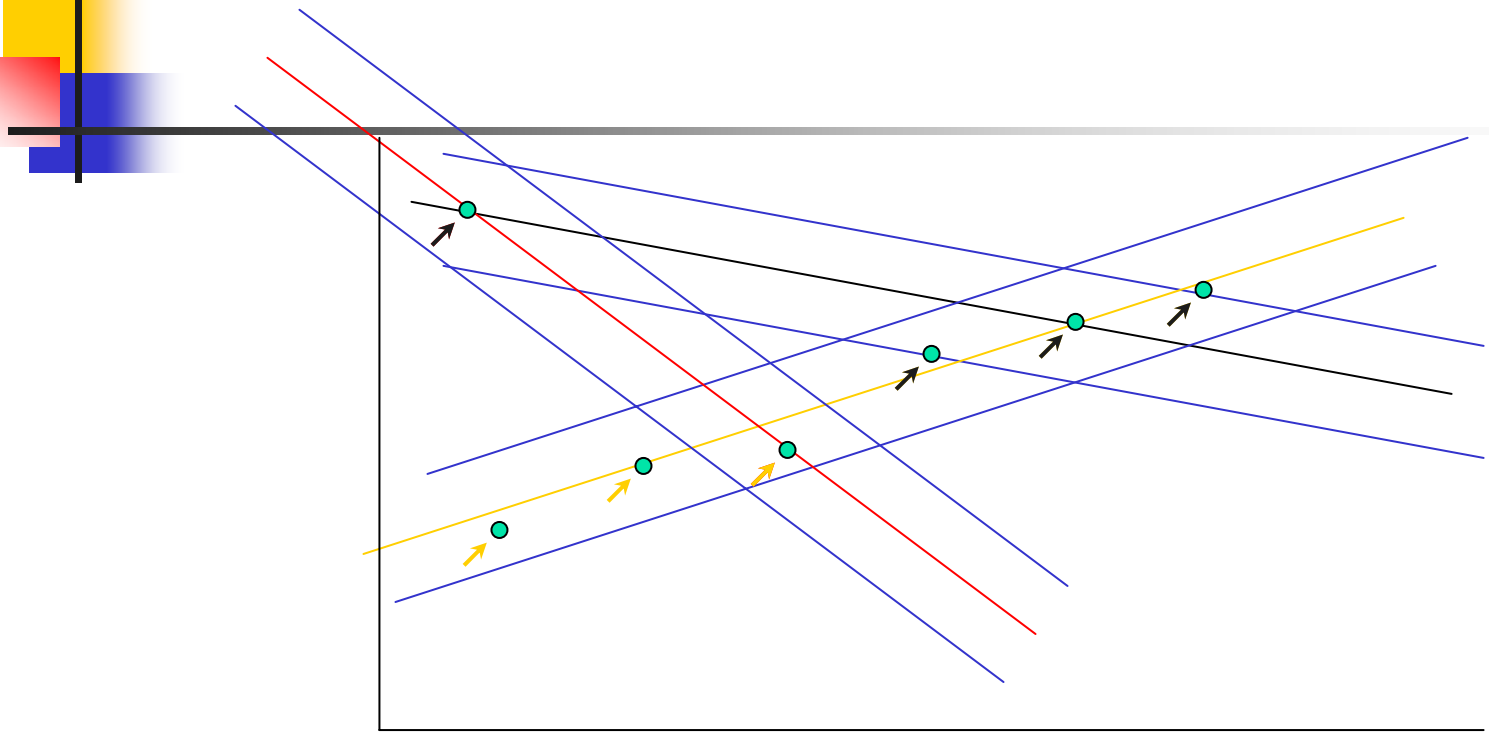
- Least squares assumes Gaussian errors
- Outliers: Points with extremely low probability of occurrence (according to Gaussian statistic)
  - Can result from data association errors
- Strongly influence least squares



RANSAC

- Main idea
  - Select 2 points at random
  - Fit a line
  - “Support” = number of inliers
  - Line with most supports wins
- General algorithm
  - Randomly select s points
  - Instantiate a model
  - Get consensus set Si (supports)
  - Repeat for N trials, return model with max |Si|

A demo for line fitting



Trial 1: Support number=2

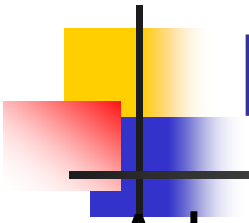
Trial 2: Support number=6

Trial 3: Support number=4

Trial 2 Wins with maximal support number !

RANSAC—Summary

- Choose a small subset uniformly at random
- Least squares fit to that subset
- Compute the fitting error
- Determine the consensus set
  - comparing each error with the threshold;
  - Anything close to result is inliers;
  - all others are outliers
- Repeat the above steps for many trials
- Choose the fit that agreed with most points
  - Can perform one final LS with all inliers



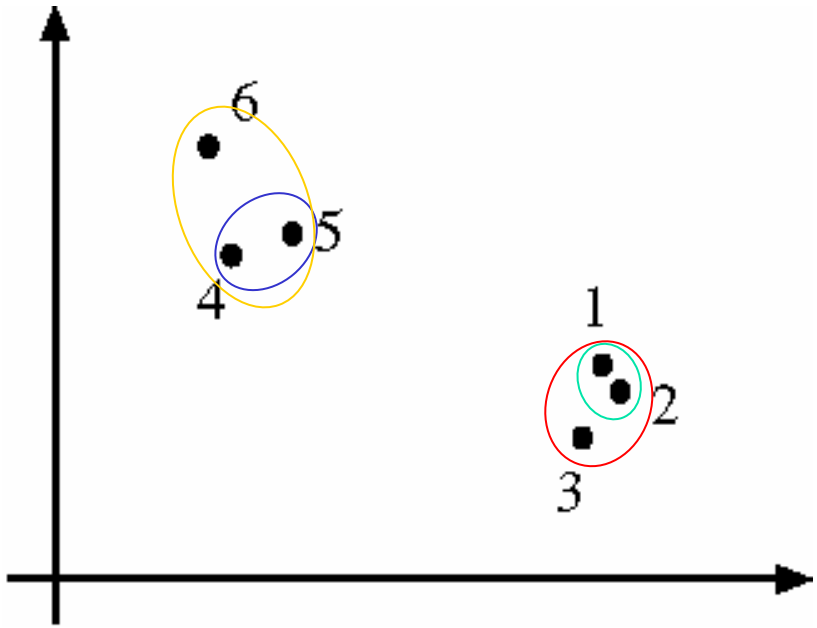
RANSAC—Discussion

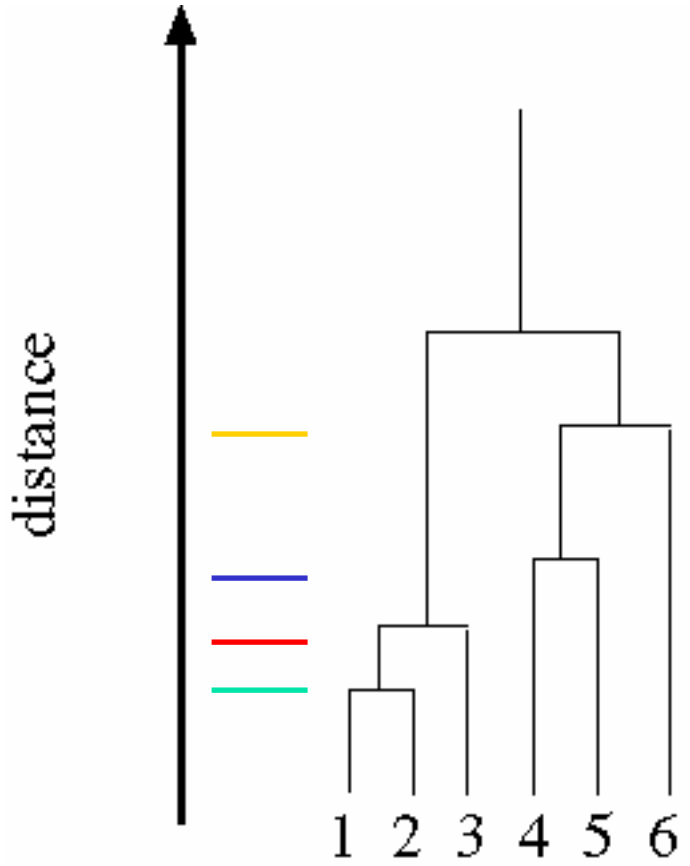
- Advantages:
  - General method suited for a wide range of model fitting problems;
  - Easy to implement and easy to calculate its failure rate;
- Disadvantages:
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)
- Hough transform can handle high percentage of outliers, but false collisions increases with large bins

Segmentation as clustering

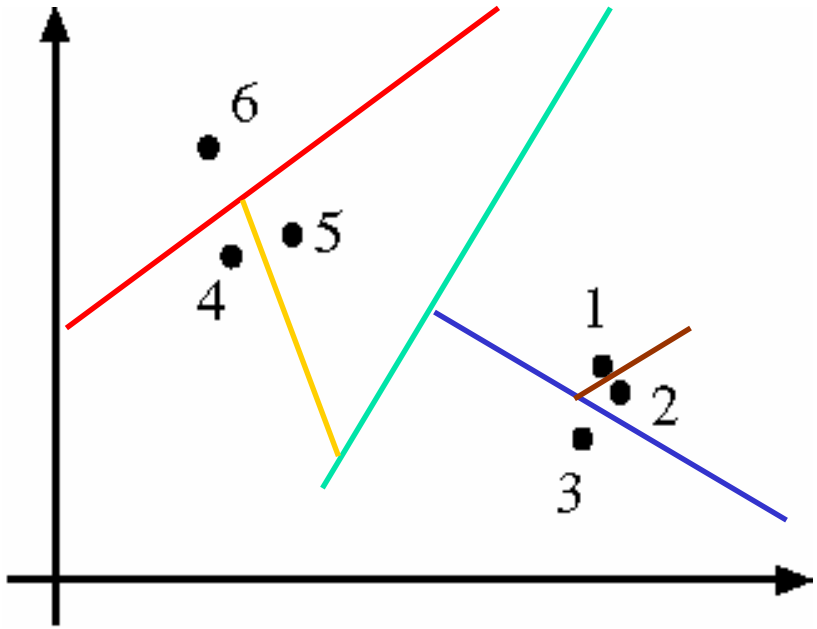
- Cluster together (pixels, tokens, etc.) that belong together
- Agglomerative (凝聚) clustering (bottom-up)
  - attach closest to cluster it is closest to
  - repeat
- Divisive (分裂) clustering (top-down)
  - split cluster along best boundary
  - Repeat
- Point-Cluster distance (merge/split rules)
  - single-link clustering
  - complete-link clustering
  - group-average clustering
- Dendrograms (树形图)
  - yield a picture of output as clustering process continues

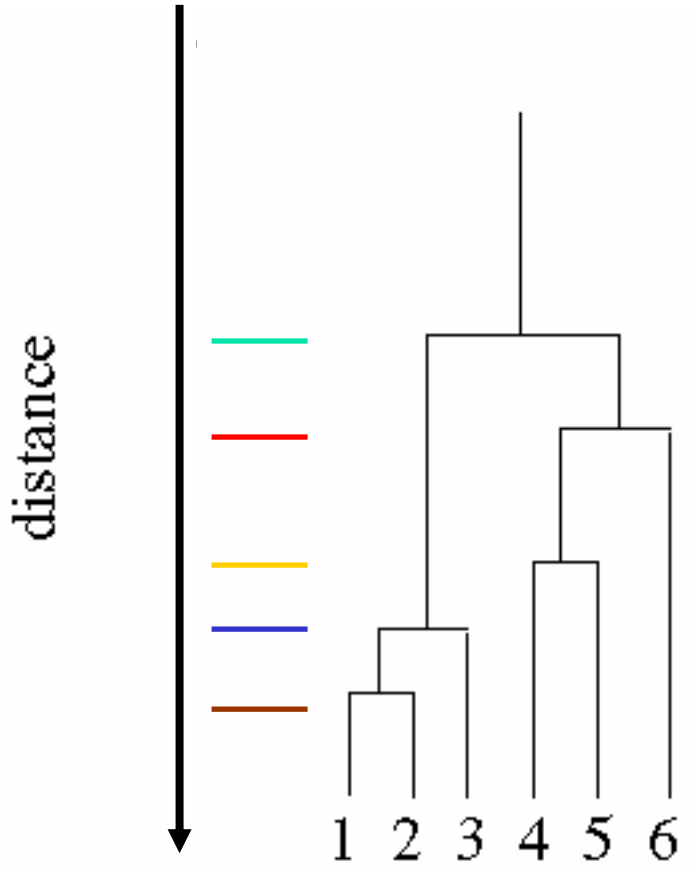
Agglomerative clustering—Clustering by merging





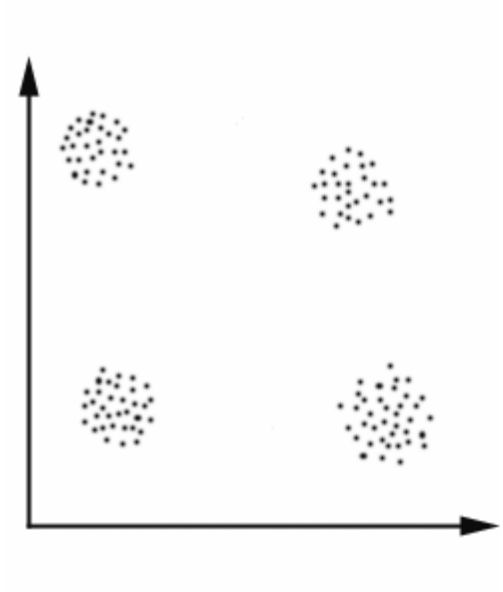
Divisive clustering—Clustering by splitting





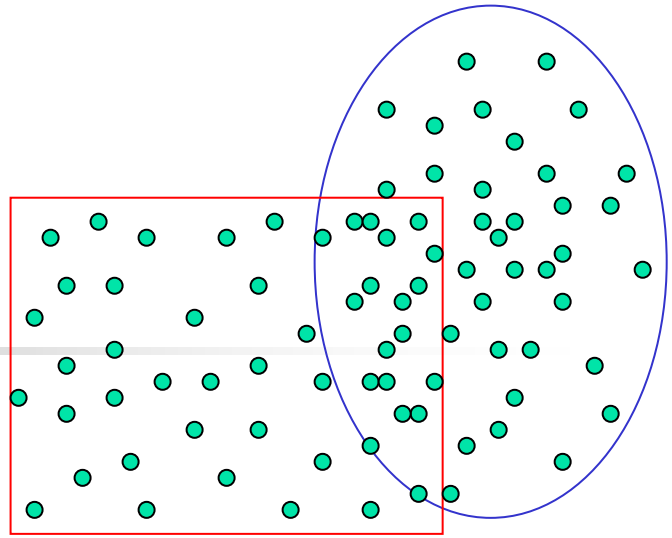
Clustering

- We want to group together some primitives



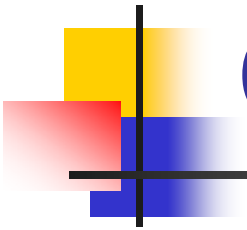
Seems easy, but…

Clustering



$$f_1 = Unit(a,b),\ f_2 = Normal(\mu,\Sigma)$$

- We want to group together some primitives
- If we knew which items belongs to a group…
  - A good description of the groups can be drawn
  - Position, intensity, texture…
- If we knew a good description of the group…
  - We may figure out which primitives belong to which groups
  - Or at least the probability…
- This is a chicken and egg problem…

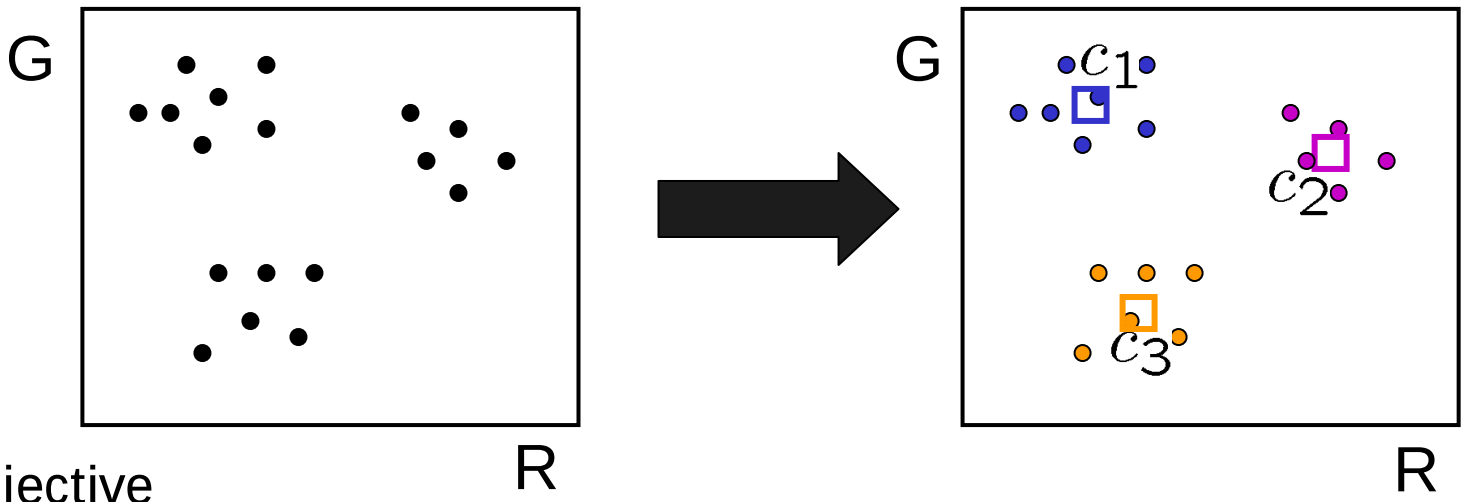


Clustering

- Iterative solution:
  - Guess one side of the answer
  - Figure out the other side
  - Refigure out the first side
  - Keep going till we converge

Clustering

- How to choose the representative colors?
  - This is a clustering problem!

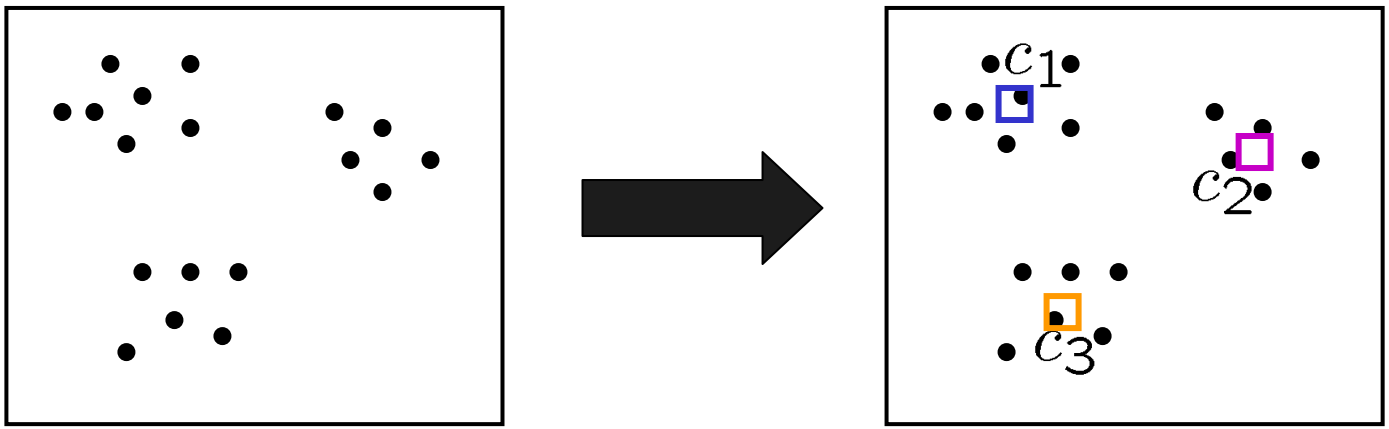


- Objective
  - Each point should be as close as possible to a cluster center
    - Minimize sum squared distance of each point to closest center

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Break it down into subproblems

- Suppose I tell you the cluster centers c_i
  - Q: how to determine which points to associate with each c_i?
  - A: for each point p, choose closest c_i



- Suppose I tell you the points in each cluster
  - Q: how to determine the cluster centers?
  - A: choose c_i to be the mean of all points in the cluster

K-means clustering

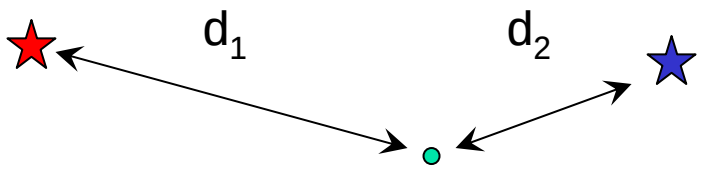
- K-means clustering algorithm
  1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean of points in cluster i
  4. If c_i have changed, repeat Step 2

- Properties
  - Will always converge to *some* solution
  - Can be a "local minimum"
    - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Convergence of the algorithm

- The iteration always reduces the error measure
  - *Reassigning a point to the nearest center reduces error*
  - *The center that minimizes MSE is the average*



$d_2 < d_1$

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Recall – Fitting a constant function

- For constant function *y=a*
  - Minimizing squares gives *a=mean*

$$Min(E = \sum_i (y_i - a)^2)$$

$$\frac{\partial E}{\partial a} = \sum_i -2(y_i - a) = -2(\sum_i y_i - n \cdot a) = 0$$

$$a = \sum_i y_i / n = mean(Y)$$

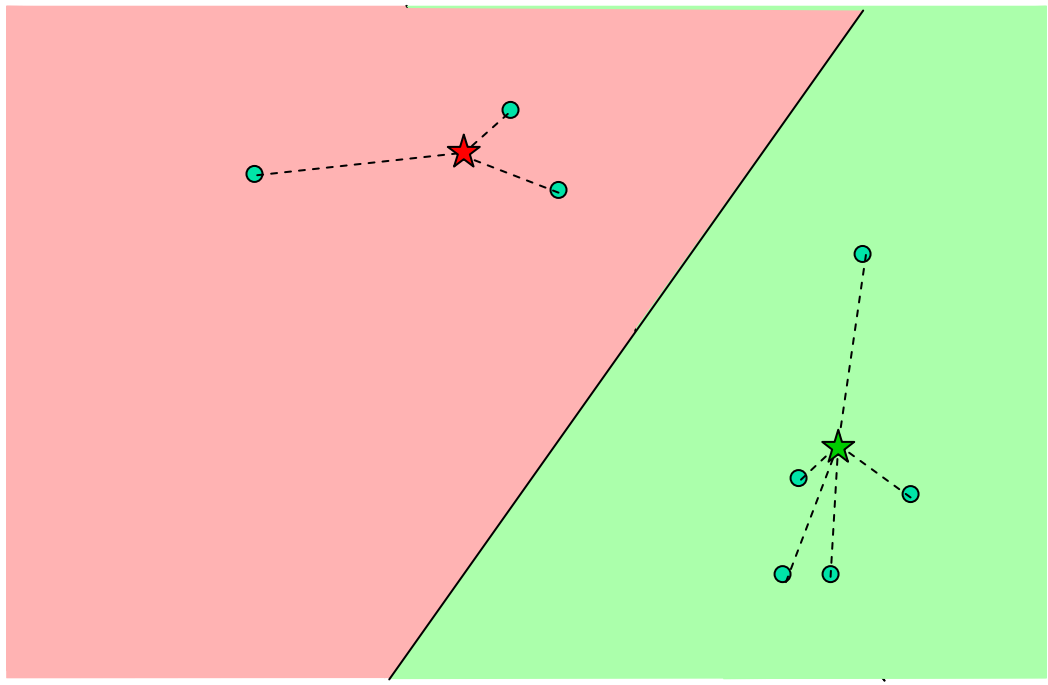
K-Means

- Choose a fixed number of clusters
- Choose cluster centers and point-cluster allocations to minimize error

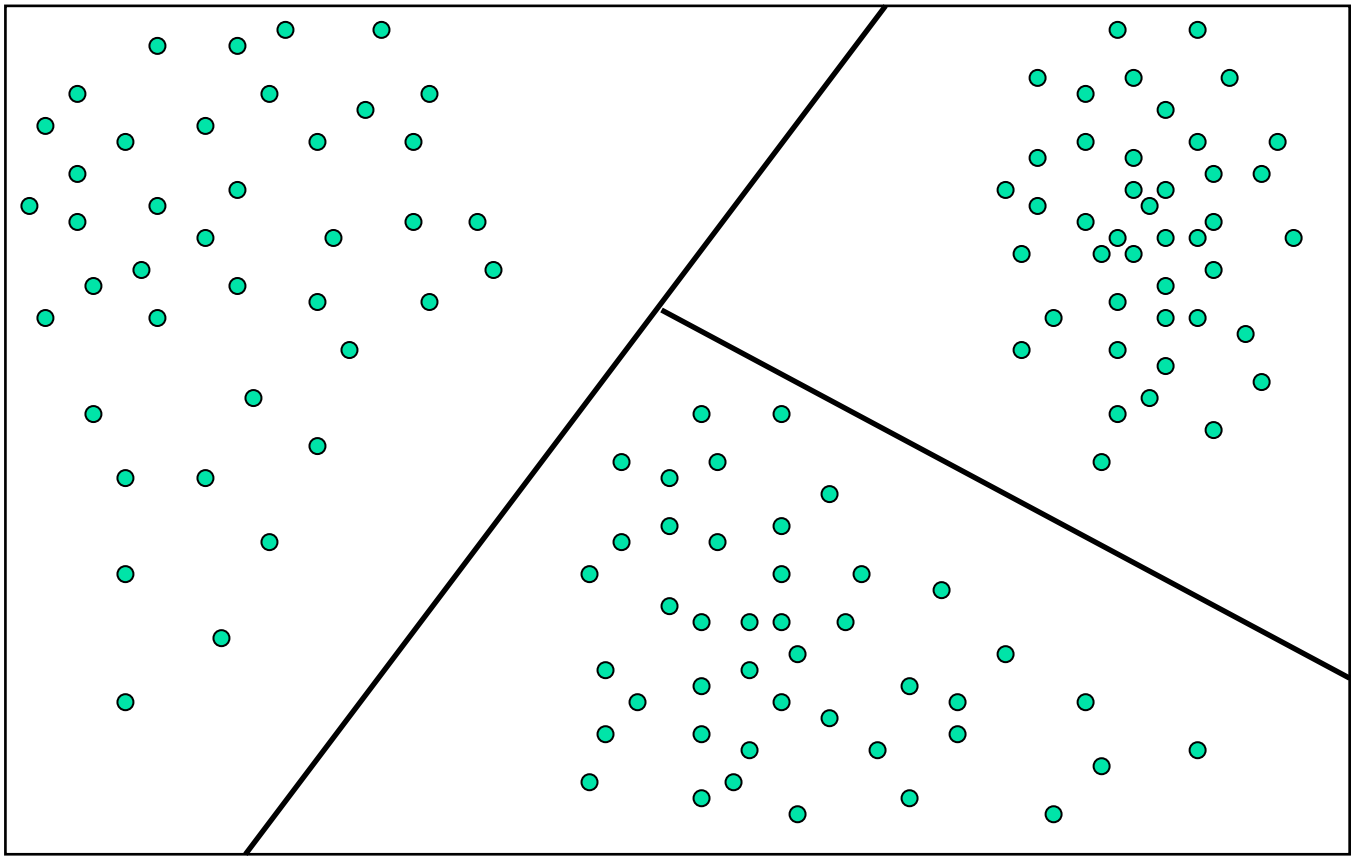
$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

- can't do this by search, because there are too many possible allocations.
- Algorithm
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute best cluster centers
- x could be any set of features for which we can compute a distance (careful about scaling)

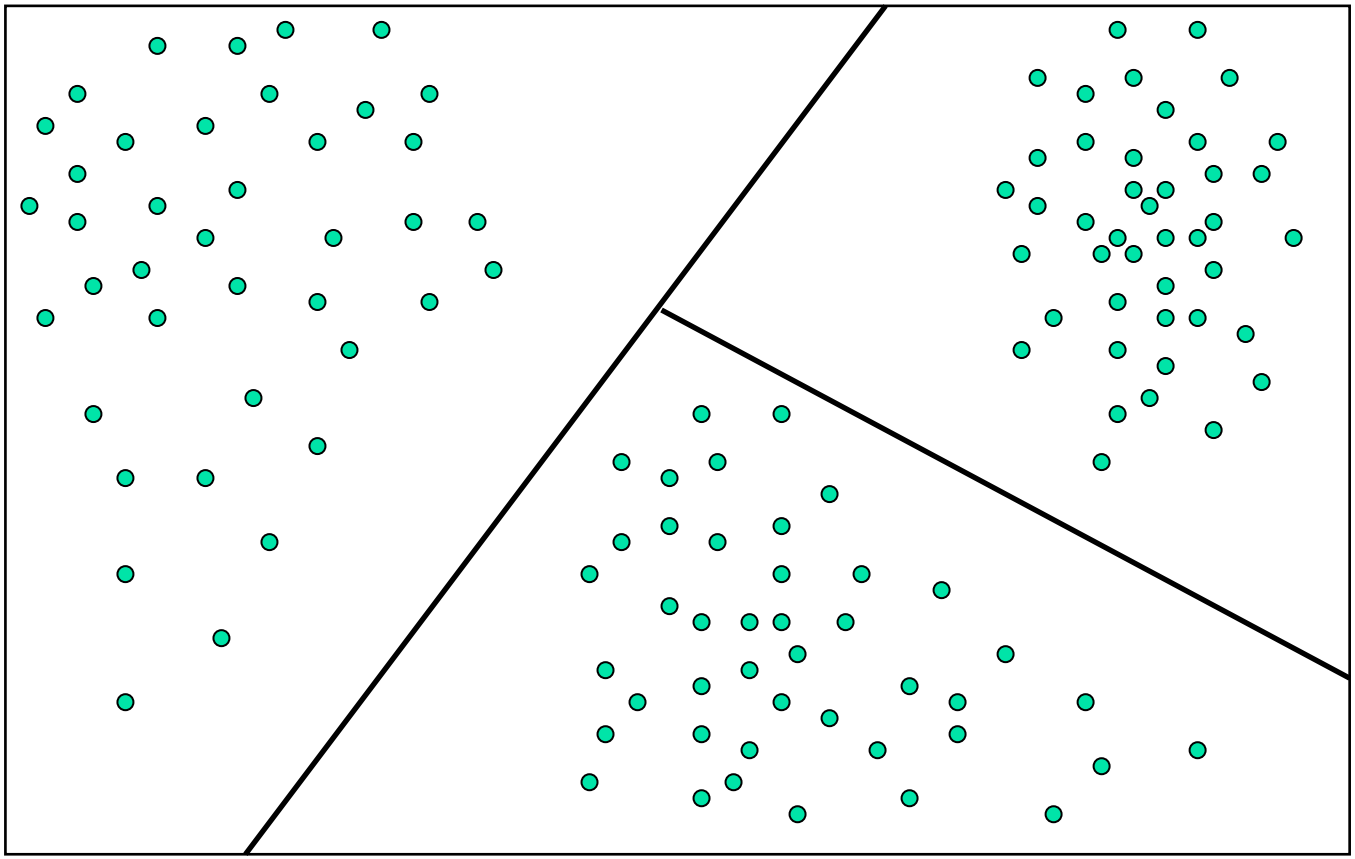
K-Means



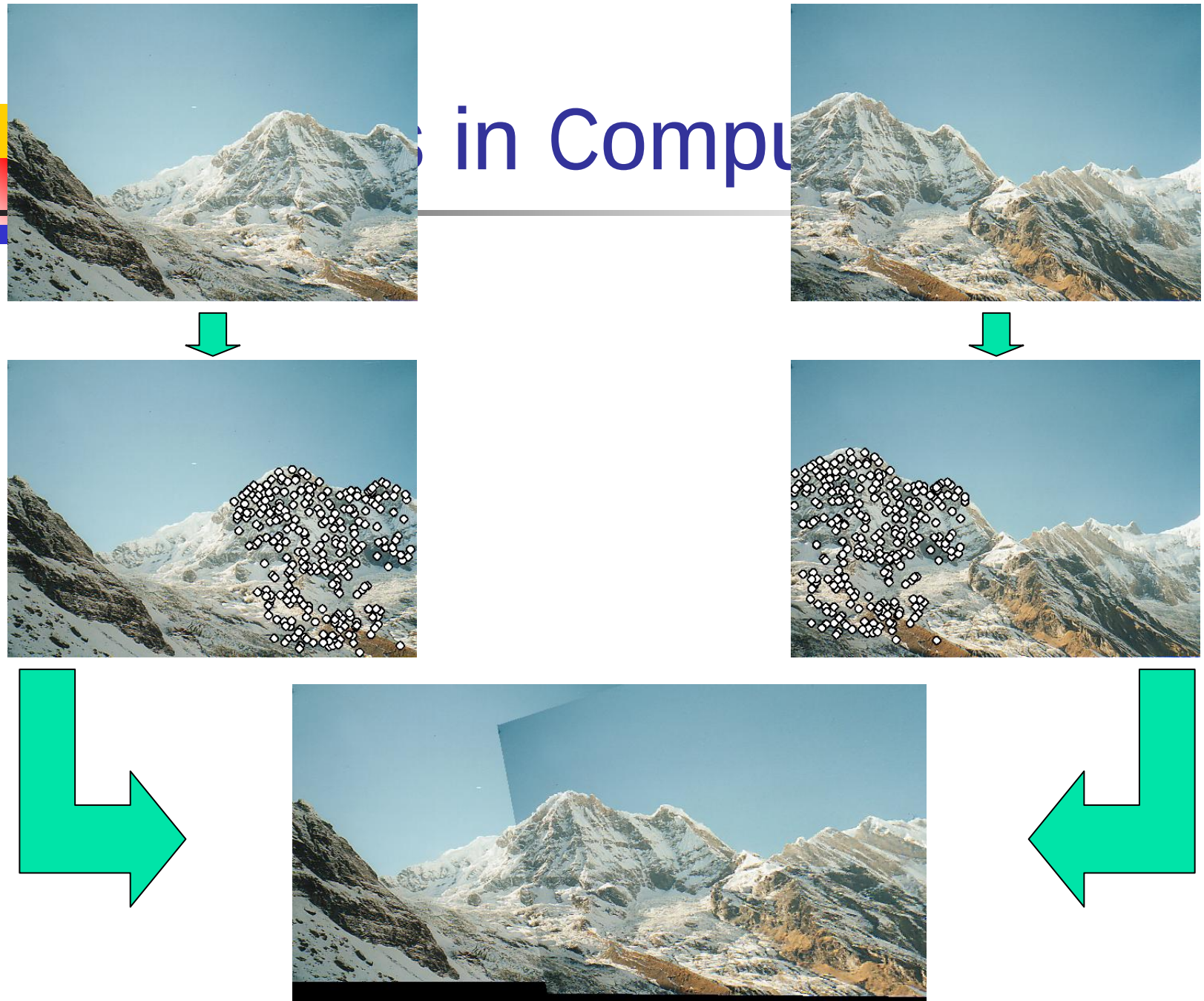
Clustering – Determine Regions



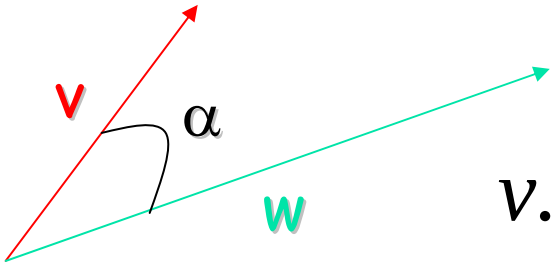
Graph Cut – Determine Boundaries



s in Compu



Inner (dot/scalar) Product



$$v.w = (x_1, x_2).(y_1, y_2) = x_1 y_1 + x_2.y_2$$

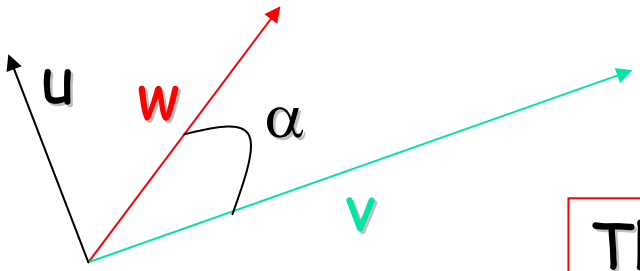
The inner product is a SCALAR!

$$v.w = (x_1, x_2).(y_1, y_2) = v^T w = w^T v = \| v \| \cdot \| w \| \cos\alpha$$

$$v.w = 0 \Leftrightarrow v \perp w$$

The inner product measures the similarity of two vectors

Outer (cross/vector) Product



$$u = v \times w$$

The cross product is a VECTOR!

Magnitude: $\| u \| = \| v.w \| = \| v \| \| w \| \sin\alpha$

Orientation:

$$u \perp v \Rightarrow u \cdot v = (v \times w) \cdot v = 0$$

$$u \perp w \Rightarrow u \cdot w = (v \times w) \cdot w = 0$$

Homogeneous Coordinates (齐次坐标)

- Multiply the coordinates by a non-zero scalar and add an extra coordinate equal to that scalar. For example,

$$(x, y) \rightarrow (x \cdot z, y \cdot z, z) \quad z \neq 0$$

$$(x, y, z) \rightarrow (x \cdot w, y \cdot w, z \cdot w, w) \quad w \neq 0$$

- NOTE: If the scalar is 1, there is no need for the multiplication!

Example:

$$(2,3) \rightarrow (2,3,1) \sim (4,6,2) \sim (-4,-6,-2)\ldots$$

$$(3,-1,2) \rightarrow (3,-1,2,1) \sim (6,-2,4,2) \sim (-6,2,-4,-2)\ldots$$

Back to Cartesian Coordinates:

- Divide by the last coordinate and eliminate it. For example,

$$(x, y, z) \quad z \neq 0 \rightarrow (x/z, y/z)$$

$$(x, y, z, w) \quad w \neq 0 \rightarrow (x/w, y/w, z/w)$$

Question: What if *z=0*?

Scaling, Translating & Rotating

Order matters!

$P' = S.P$

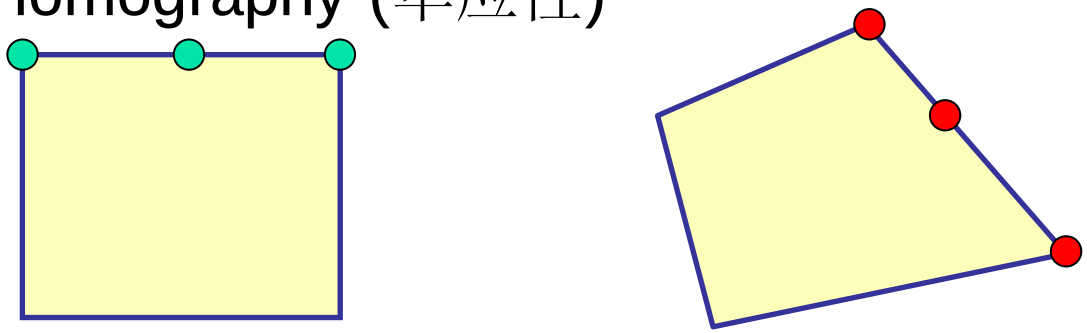
$P'' = T.P' = (T.S).P$

$P''' = R.P'' = R.(T.S).P = (R.T.S).P$

$R.T.S \neq R.S.T \neq T.S.R \ldots$

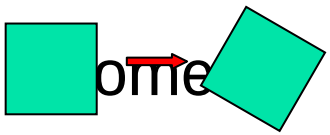
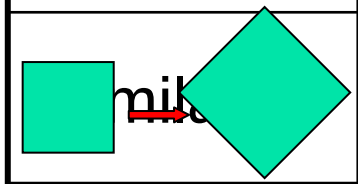
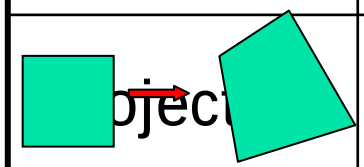
Projective Transformations in a Plane (射影变换/透视变换)

- Projectivity (直射)
  - Mapping from points in plane to points in plane
  - 3 aligned points are mapped to 3 aligned points
- Also called
  - Collineation (共线，直射变换)
  - Homography (单应性)



Same shapes are related by a projective transformation

Invariants(不变量)

	Length Area	Angle Shape	Parallelism Area ratio	Collinearity Cross-ratio
				
				
				
				

Special Projectivities

Invariants

Projectivity
8 dof

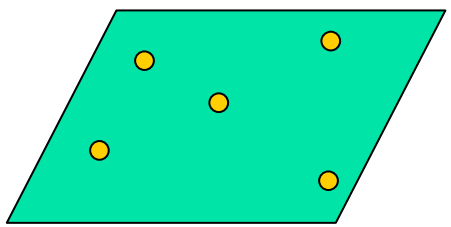
Affine transform
6 dof

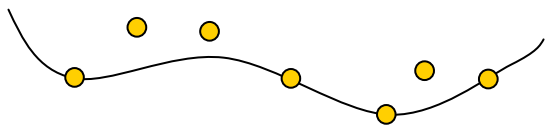
Similarity
4 dof

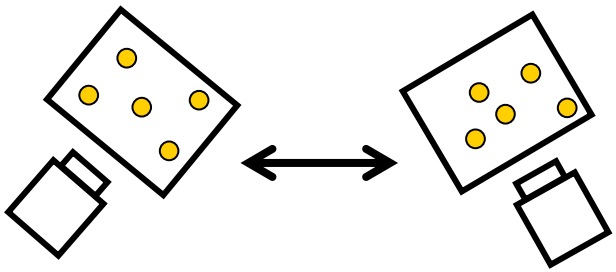
Euclidean transform
3 dof

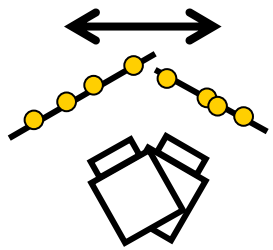
Homography

- Homography is a singular case of the Fundamental Matrix(基本矩阵)
  - Two views of **coplanar points**
  - Two views that **share the same center of projection**









Homographies

- Perspective projection of a plane
  - Lots of names for this:
    - **homography**, collineation, planar projective map
  - Modeled as a 2D warp using homogeneous coordinates

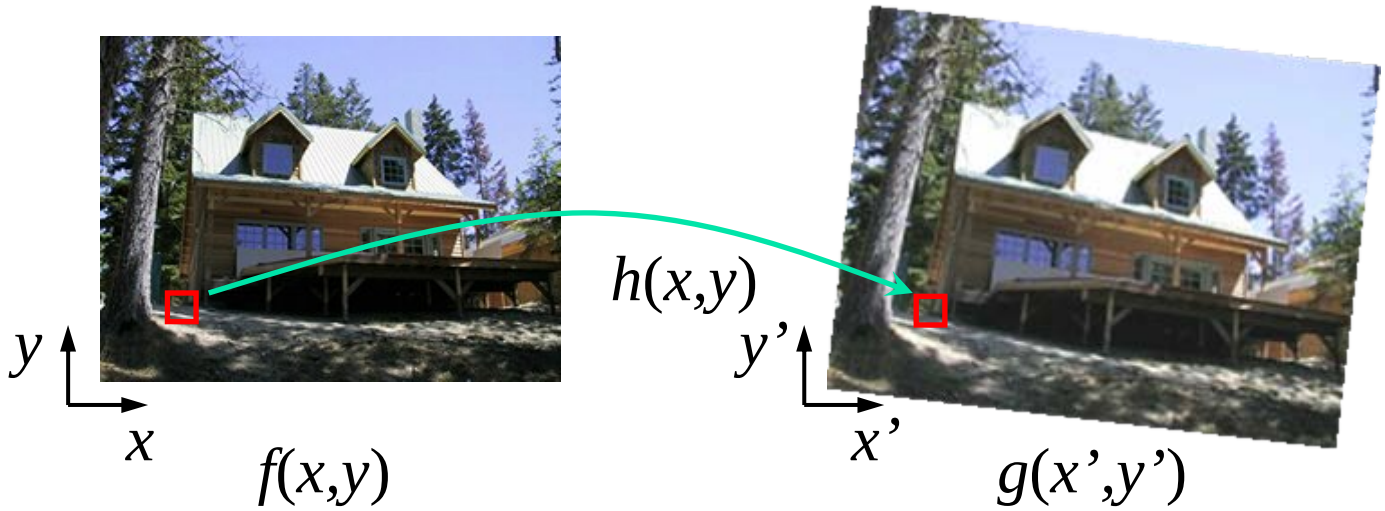
$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{p'} \qquad \mathbf{H} \qquad \mathbf{p}$$

To apply a homography $\mathbf{H}$

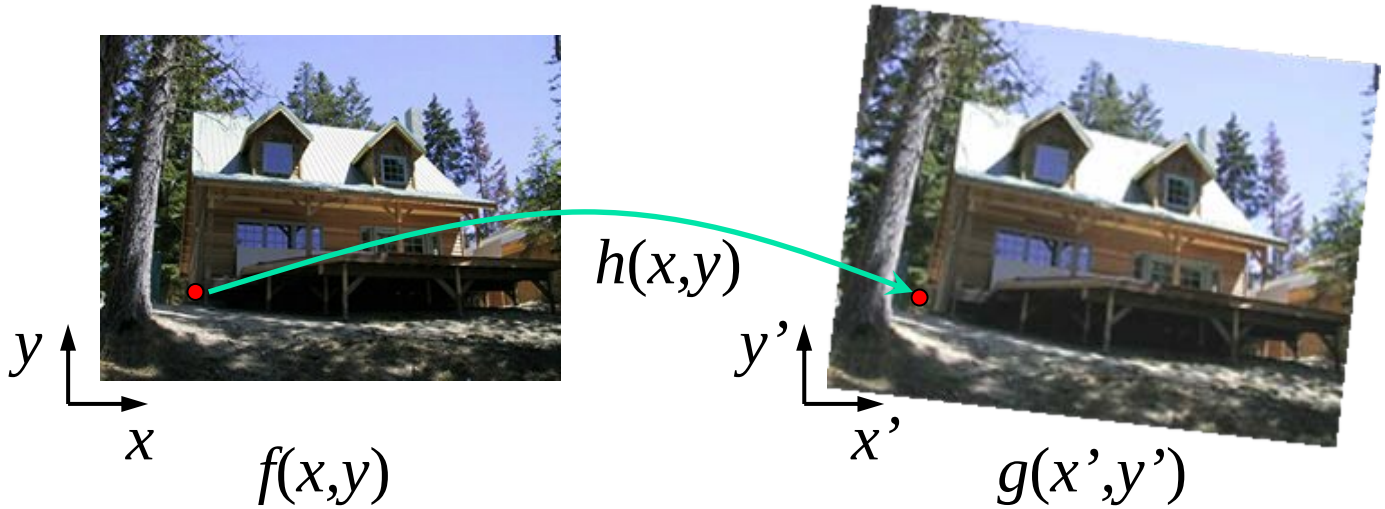
- Compute $\mathbf{p'} = \mathbf{Hp}$ (regular matrix multiplication)
- Convert $\mathbf{p'}$ from homogeneous to image coordinates
  - divide by w (third) coordinate

Image Warping



- Given a coordinate transform *(x',y')* = *h*(*x*,*y*) and a source image *f*(*x*,*y*), how do we compute a transformed image *g(x',y')* = *f*(*h*(*x*,*y*))?

Forward Warping

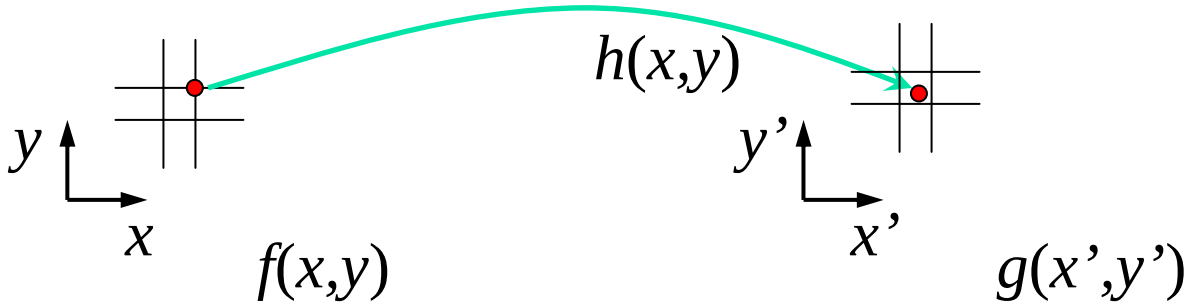


- Send each pixel $f(x,y)$ to its corresponding location

  $(x',y') = h(x,y)$ in the second image

Q: what if pixel lands "between" two pixels?

Forward Warping



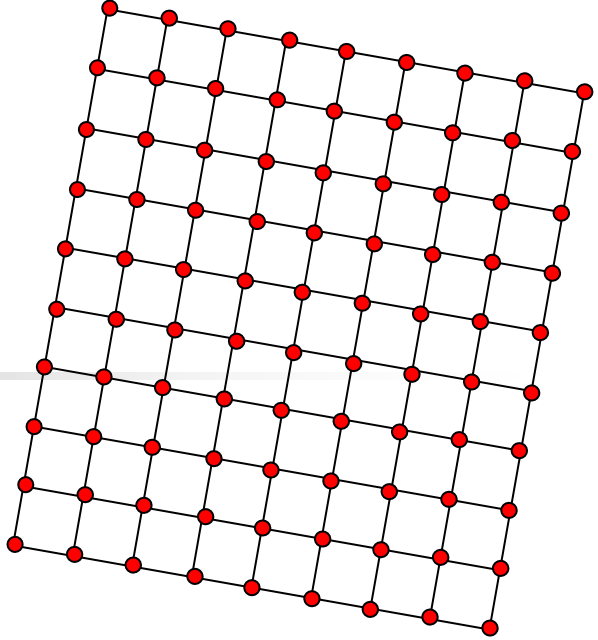
- Send each pixel $f(x,y)$ to its corresponding location $(x',y') = h(x,y)$ in the second image

Q: what if pixel lands "between" two pixels?

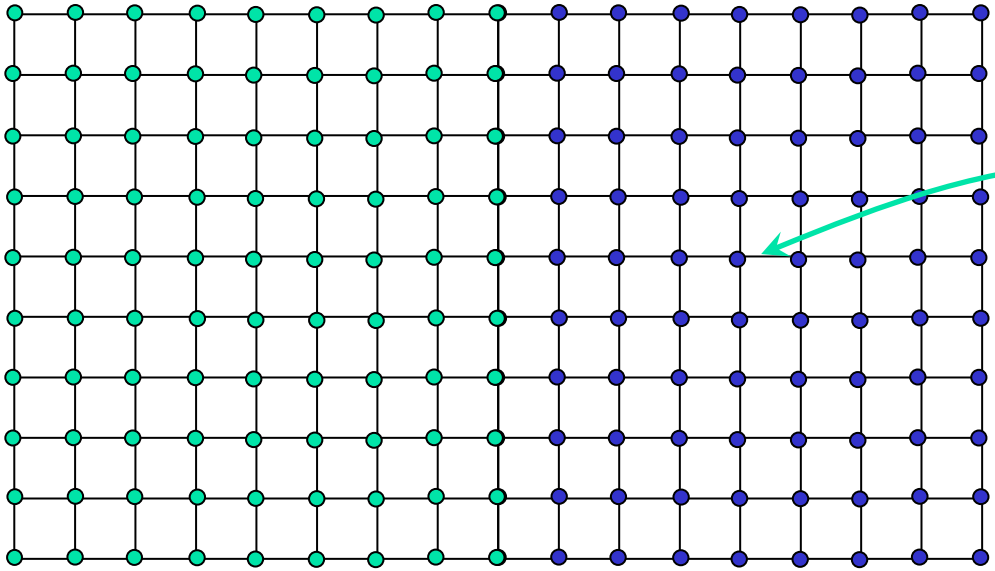
A: distribute color among neighboring pixels (x',y')

– Known as "splatting"

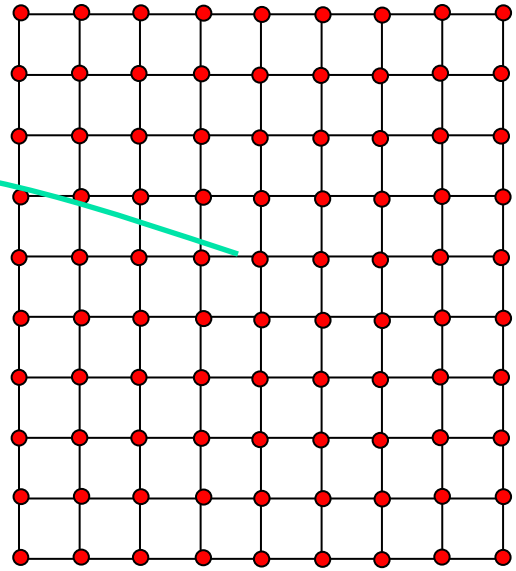
Forward Warping



Forward warped image



$h(x,y)$

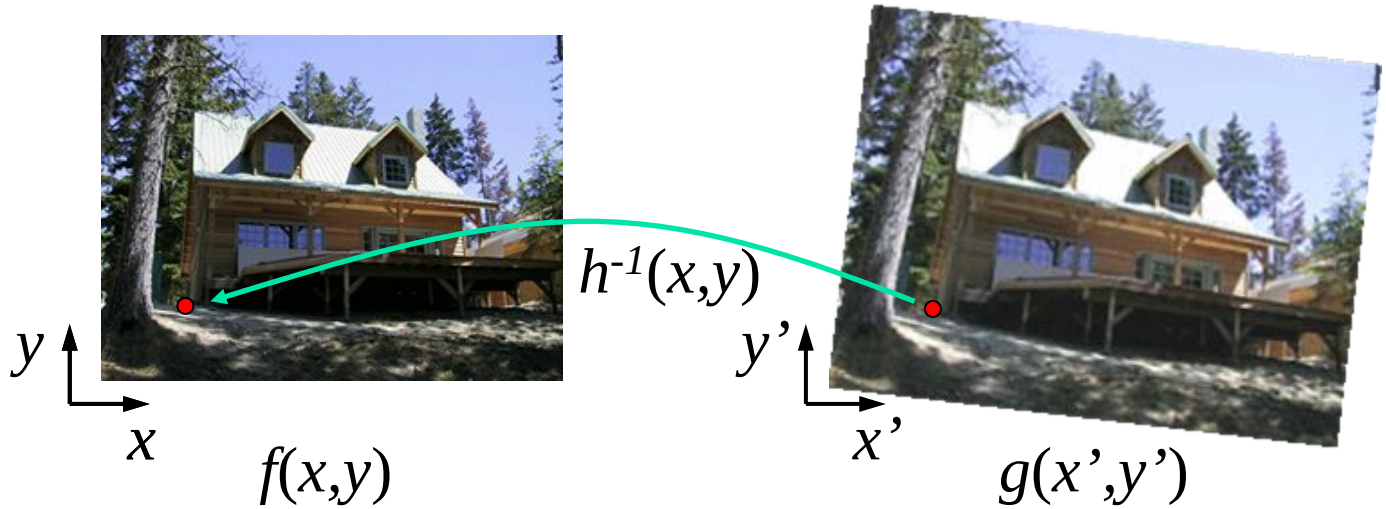


Reference image

Extended region

Target image

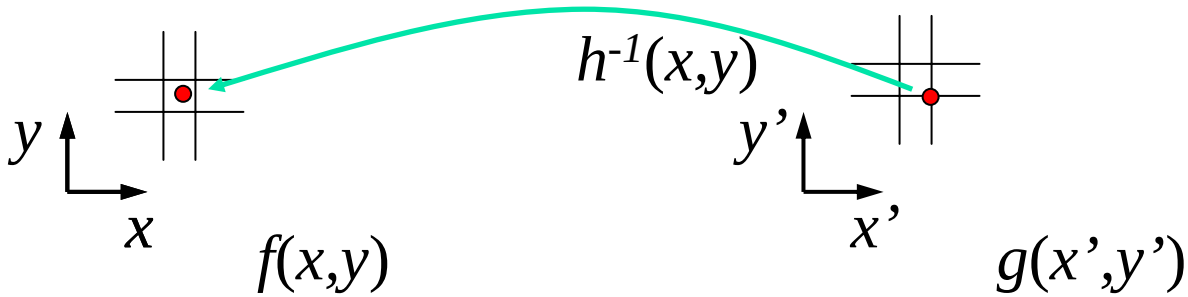
Inverse Warping



- Get each pixel $g(x',y')$ from its corresponding location
- $(x,y) = h^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

Inverse Warping

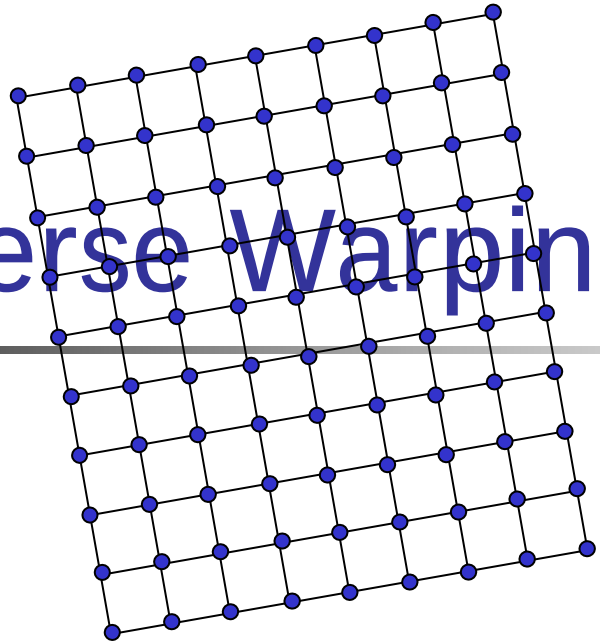


- Get each pixel $g(x',y')$ from its corresponding location $(x,y) = h^{-1}(x',y')$ in the first image

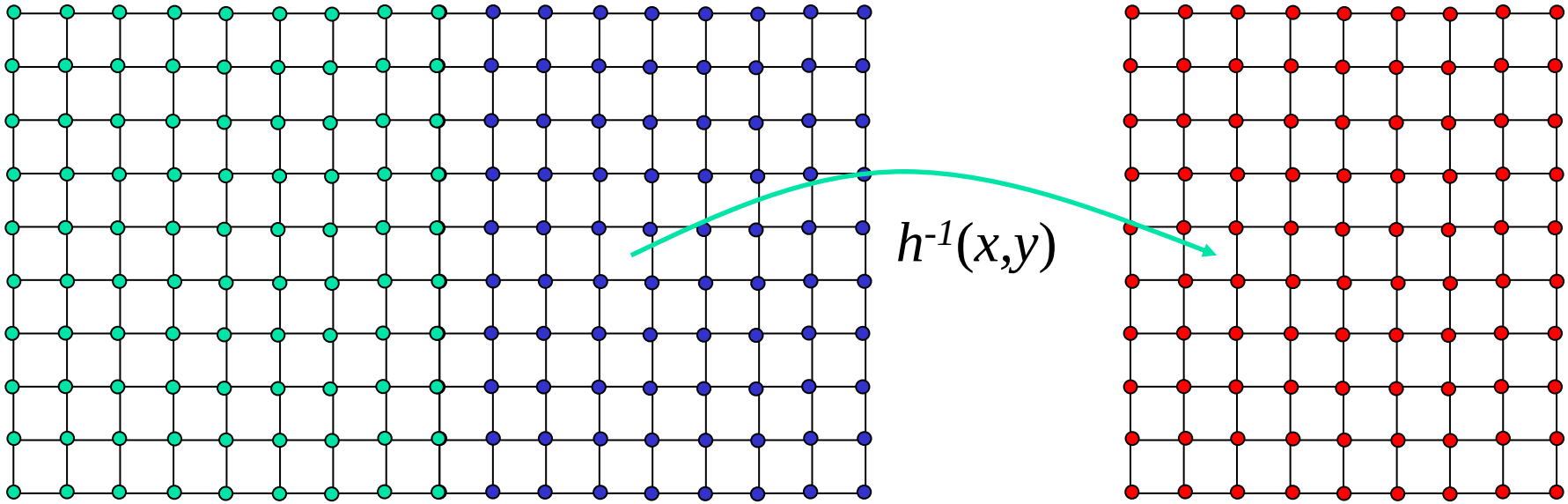
Q: what if pixel comes from "between" two pixels?

A: *resample* color value

Inverse Warping

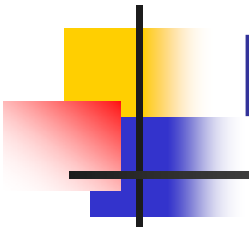


Inverse warped image



Reference image Extended region Target image

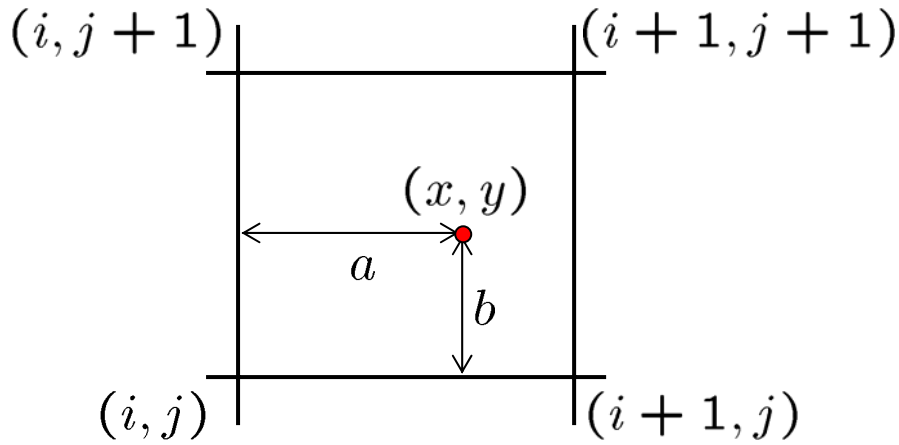
Forward vs. Inverse Warping



- Q: which is better?
- A: usually inverse—eliminates holes
  - however, it requires an invertible warp function—not always possible...

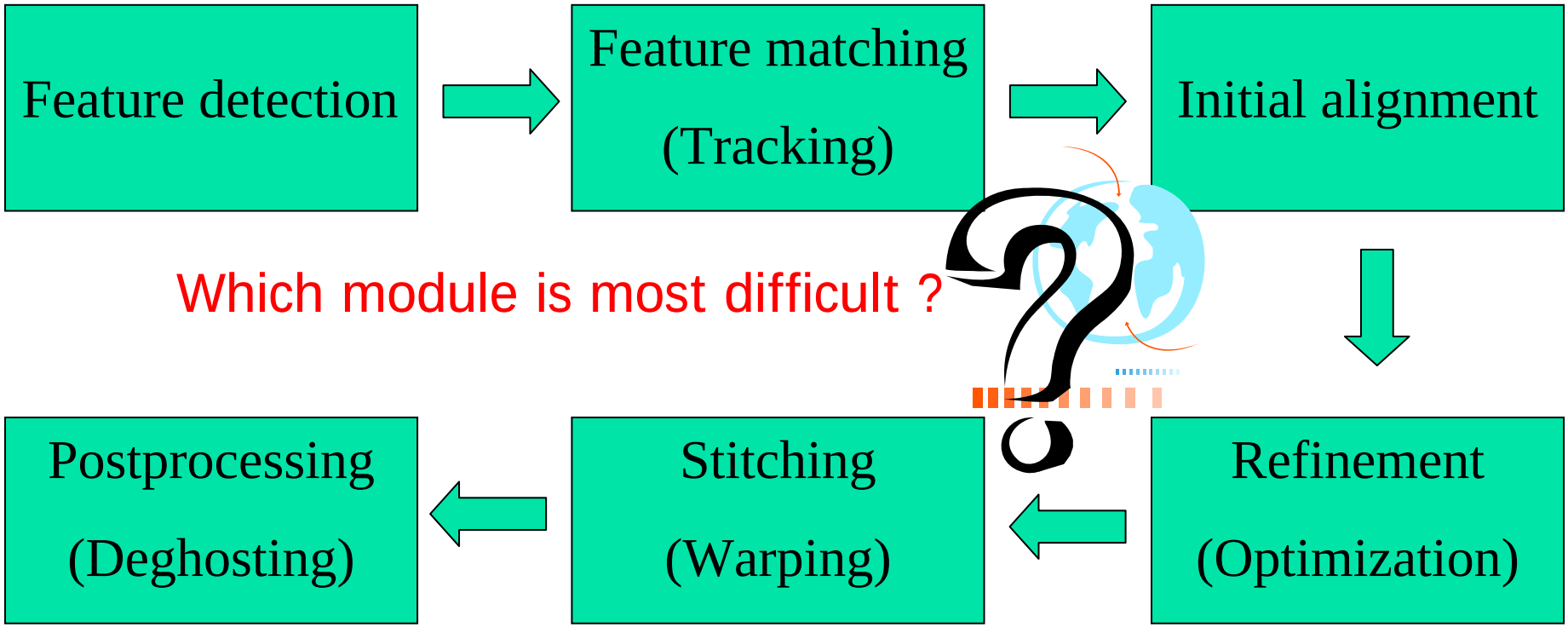
Bilinear Interpolation

- A simple method for resampling images



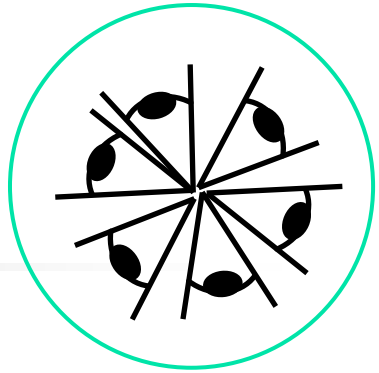
$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) \quad f[i,j] \\
& +a(1-b) \quad f[i+1,j] \\
& +ab \quad f[i+1,j+1] \\
& +(1-a)b \quad f[i,j+1]
\end{aligned}
$$

A general pipeline



MultiView/Extraction/matching/tracking/morphing/optimization /blending/inpainting/editing/…

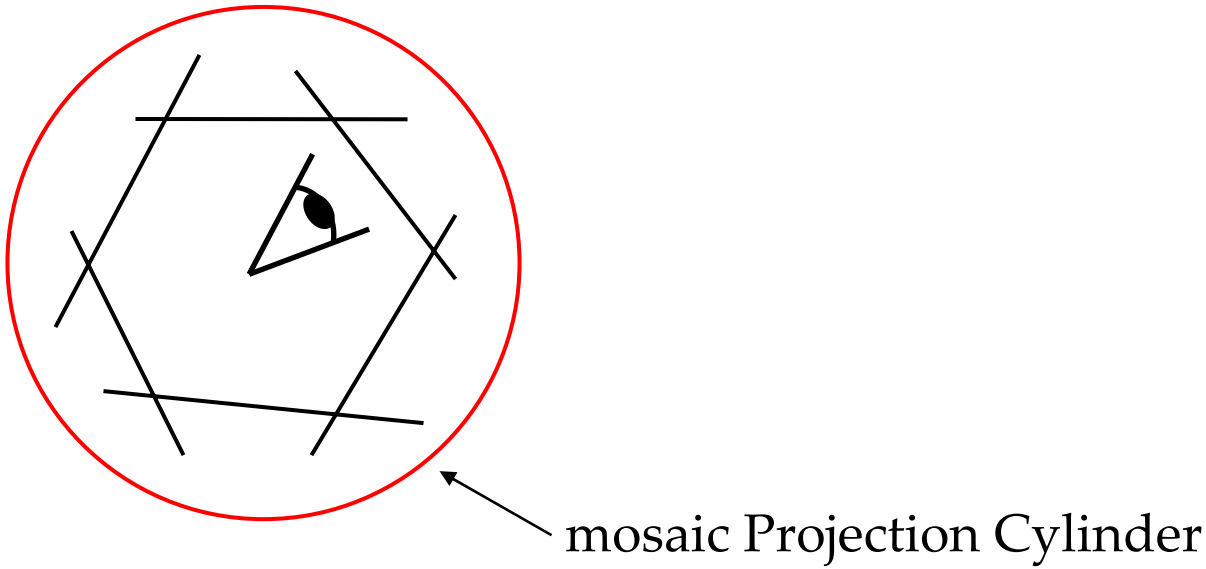
Wide-angle Imaging



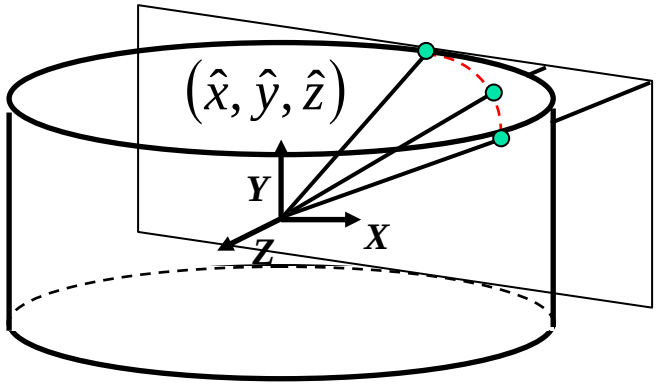
- Goal
  - Stitch together several images into a seamless composite

Panoramas

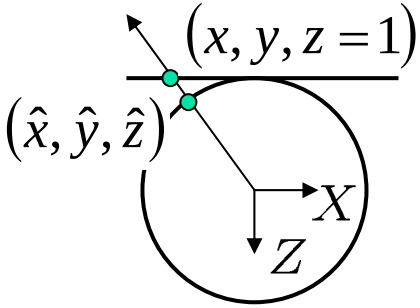
- What if you want a 360° field of view?



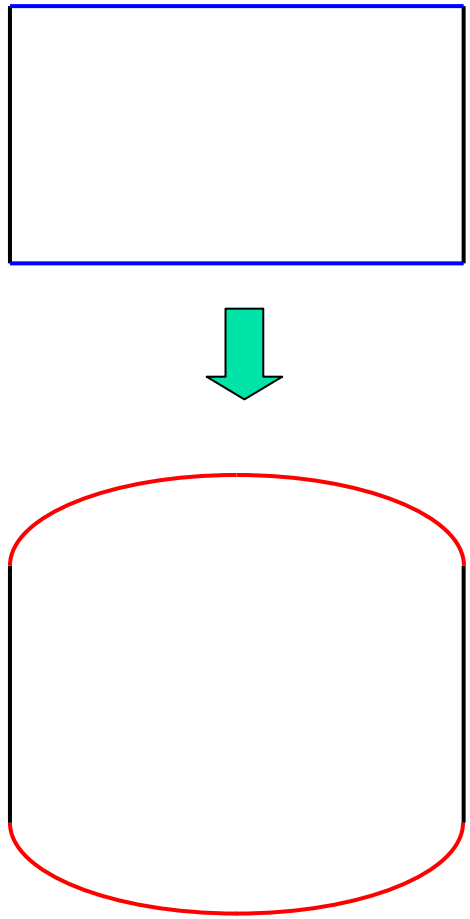
Cylindrical Reprojection



side view



top-down view



Homogeneous Coordinates

- Multiply the coordinates by a non-zero scalar and add an extra coordinate equal to that scalar. For example,

$$(x, y) \rightarrow (x \cdot z, y \cdot z, z) \quad z \neq 0$$

$$(x, y, z) \rightarrow (x \cdot w, y \cdot w, z \cdot w, w) \quad w \neq 0$$

- NOTE: If the scalar is 1, there is no need for the multiplication!

Example:

$$(2,3) \rightarrow (2,3,1) \sim (4,6,2) \sim (-4,-6,-2)\ldots$$

$$(3,-1,2) \rightarrow (3,-1,2,1) \sim (6,-2,4,2) \sim (-6,2,-4,-2)\ldots$$

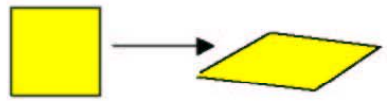
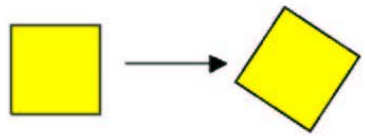
Back to Cartesian Coordinates:

- Divide by the last coordinate and eliminate it. For example,

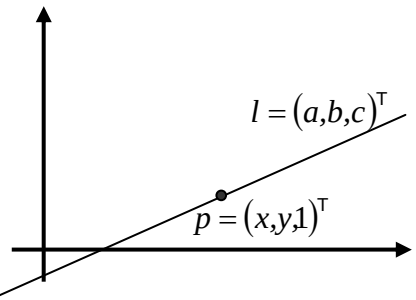
$$(x, y, z) \quad z \neq 0 \rightarrow (x/z, y/z)$$

$$(x, y, z, w) \quad w \neq 0 \rightarrow (x/w, y/w, z/w)$$

Special Projectivities

		Invariants	
Projectivity 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$	Collinearity, Cross-ratios	
Affine transform 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_x \\ 0 & 0 & 1 \end{bmatrix}$	Parallelism, Ratios of areas, Length ratios	
Similarity 4 dof	$\begin{bmatrix} s\,r_{11} & s\,r_{12} & t_x \\ s\,r_{21} & s\,r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	Angles, Length ratios	
Euclidean transform 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	Angles, Lengths, Areas	

Homogeneous coordinates



Homogeneous representation of lines

$ax+by+c=0 \qquad (a,b,c)^T$

$(ka)x+(kb)y+kc=0, \forall k \neq 0 \qquad (a,b,c)^T \sim k(a,b,c)^T$

equivalence class of vectors, any vector is representative

Set of all equivalence classes in $\mathbf{R}^3-(0,0,0)^T$ forms $\mathbf{P}^2$

Homogeneous representation of points

$\mathrm{x}=(x,y)^T$ on $\mathrm{l}=(a,b,c)^T$ if and only if $ax+by+c=0$

$(x,y,1)(a,b,c)^T=(x,y,1)\mathrm{l}=0 \qquad (x,y,1)^T \sim k(x,y,1)^T, \forall k \neq 0$

The point x lies on the line l if and only if $\mathrm{x}^T\mathrm{l}=\mathrm{l}^T\mathrm{x}=0$

Homogeneous coordinates $(x_1,x_2,x_3)^T$ but only 2DOF

Inhomogeneous coordinates $(x,y)^T$

Points and lines

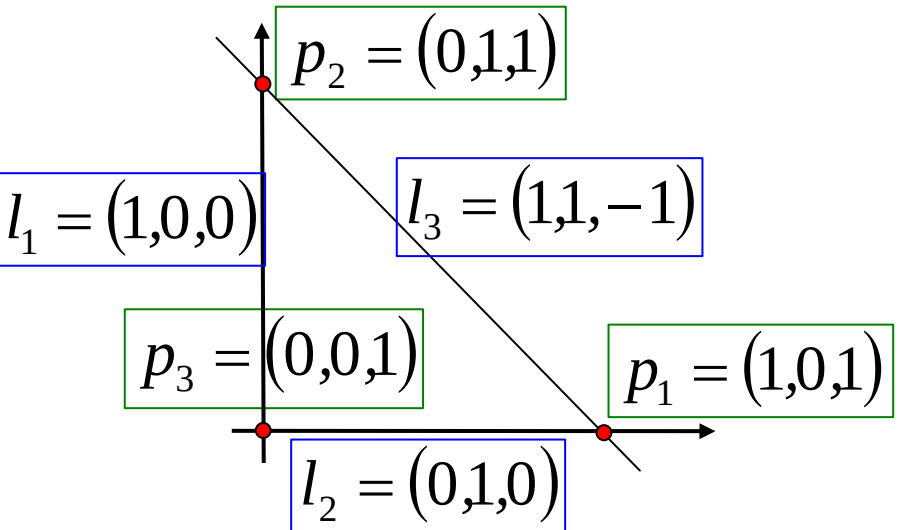
The point $p(x,y,1)^T$ lies on the line $l(a,b,c)^T$ if and only if $p^T l = l^T p = 0$

i.e. ax+by+c=0

The line l pass through two points $p_1(x_1,y_1,1)$ and $p_2(x_2,y_2,1)$ is homogeneously defined by l=p1xp2

Note that $(p_1 x p_2)^T p_1 = 0$, $(p_1 x p_2)^T p_2 = 0$

The intersection point p of two lines $l_1(a_1,b_1,c_1)$ and $l_2(a_2,b_2,c_2)$ is homogeneously defined by $p = l_1 x l_2$



We verify:

$$p_1 \cdot l_3 = 1 \cdot 1 + 0 \cdot 1 + 1 \cdot -1 = 0$$

$$p_1 \times p_2 = (0 \cdot 1 - 1 \cdot 1, 1 \cdot 0 - 1 \cdot 1, 1 \cdot 1 - 0 \cdot 0)$$

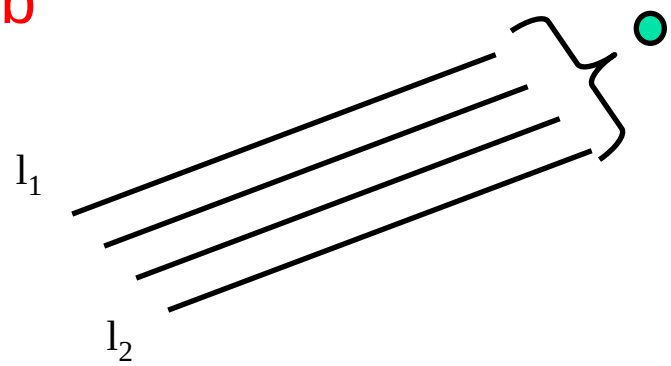
$$= (-1,-1,1) \propto (1,1,-1) = l_3$$

$$l_1 \times l_2 = (0 \cdot 0 - 0 \cdot 1, 0 \cdot 0 - 1 \cdot 0, 1 \cdot 1 - 0 \cdot 0)$$

$$= (0,0,1) = p_3$$

Ideal points

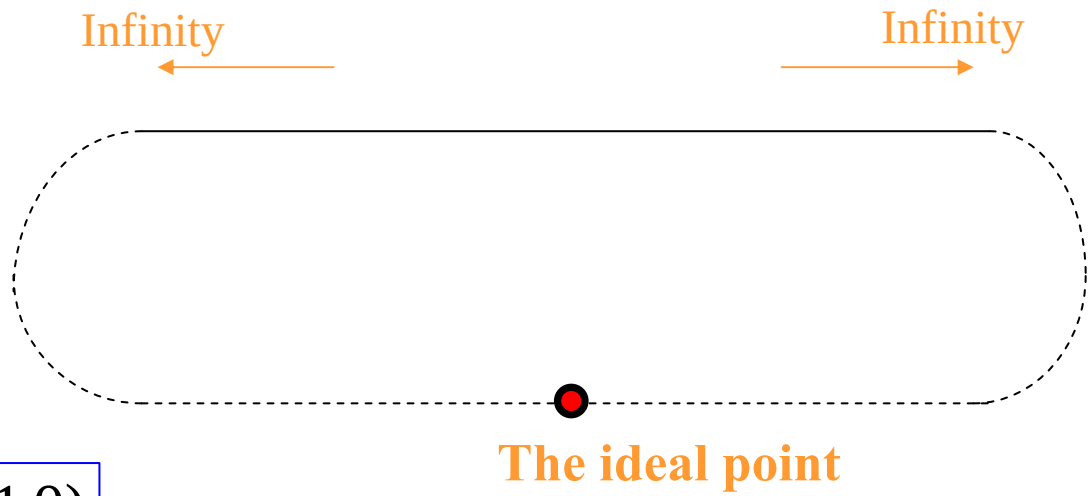
It is independent of the third coordinate c
It is solely dependent on the ratio a/b

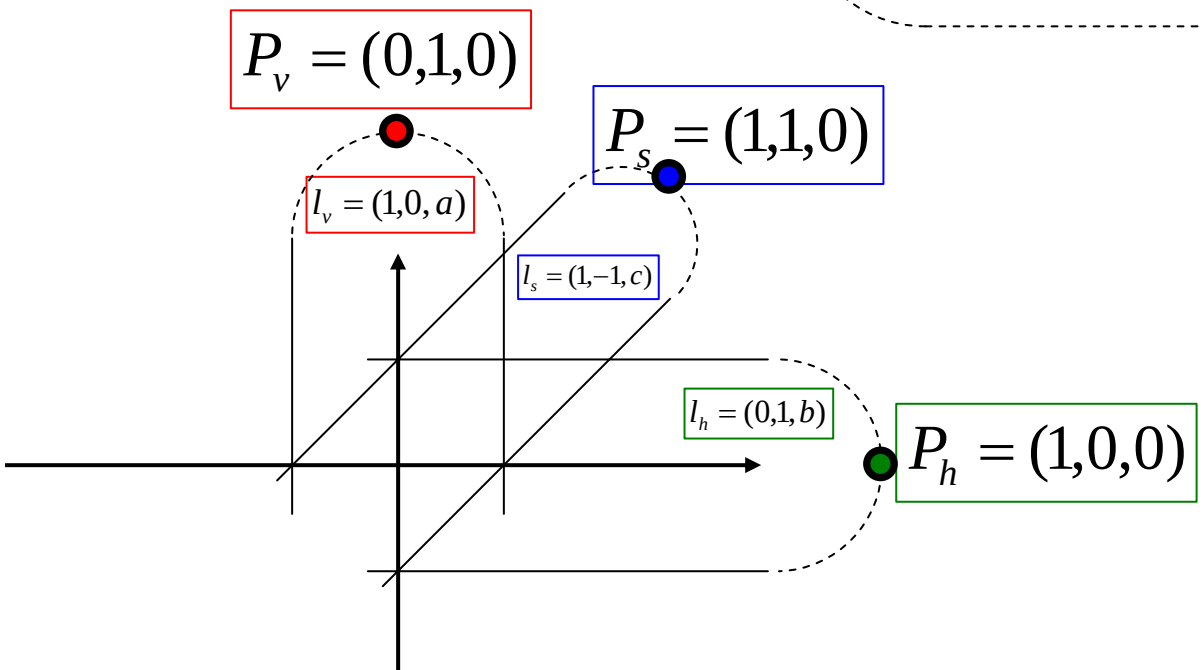


Q: How many ideal points are there in P^2?
A: 1 degree of freedom family – the line at infinity

Ideal Points

All parallel lines uniquely determine one ideal point





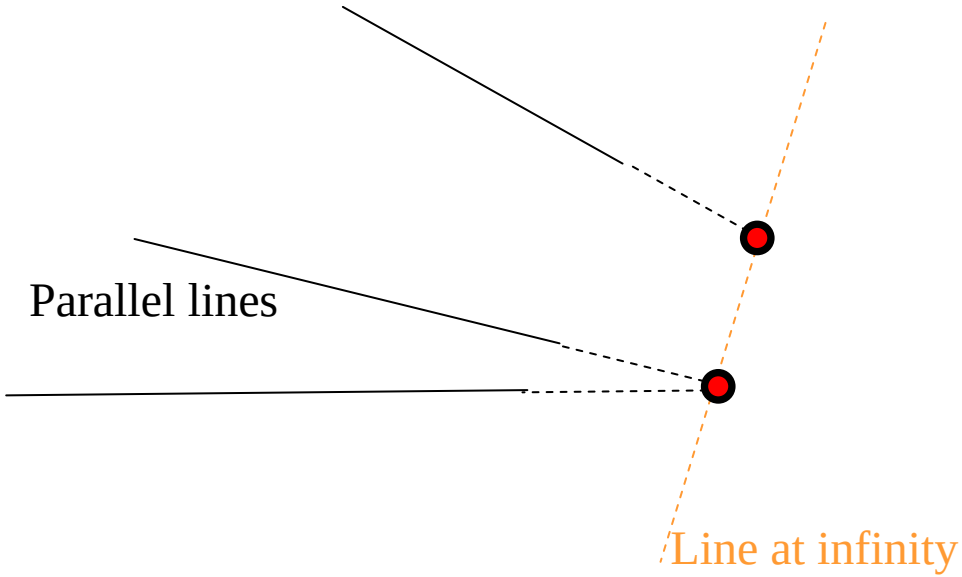
We validate:

$$l_v^T P_v = 0$$

$$l_h^T P_h = 0$$

$$l_s^T P_s = 0$$

Line at infinity



- All ideal points of a 2D plane form an ideal line, which is called the line at infinity of this 2D plane.

Points from lines and vice-versa

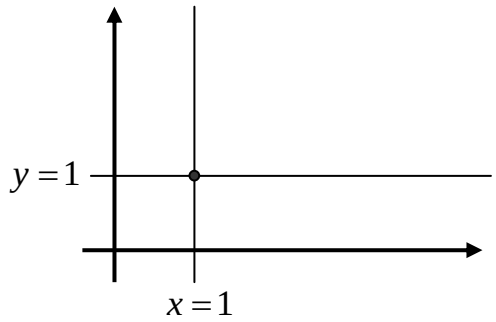
Intersections of lines

The intersection of two lines l and l' is $\mathrm{x} = \mathrm{l} \times \mathrm{l}'$

Line joining two points

The line through two points x and x' is $\mathrm{l} = \mathrm{x} \times \mathrm{x}'$

Example

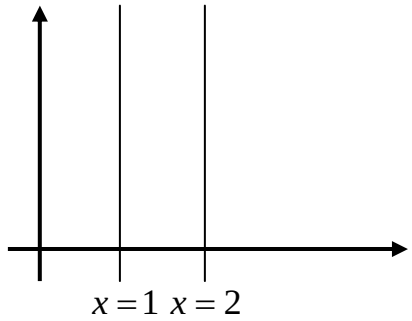


Ideal points and the line at infinity

Intersections of parallel lines

$$\mathrm{l} = (a, b, c)^{\mathsf{T}} \text{ and } \mathrm{l}' = (a, b, c')^{\mathsf{T}} \qquad \mathrm{l} \times \mathrm{l}' = (b, -a, 0)^{\mathsf{T}}$$

Example



$(b, -a)$ tangent vector

(a, b) normal direction

Ideal points $(x_1, x_2, 0)^{\mathsf{T}}$

Line at infinity $\mathrm{l}_\infty = (0, 0, 1)^{\mathsf{T}}$

$$\mathbf{P}^2 = \mathbf{R}^2 \cup \mathrm{l}_\infty$$

Note that in $\mathbf{P}^2$ there is no distinction between ideal points and others

Practice

- All ideal points are on l_∞:
  - Proof: $(0,0,1)\bullet(x_1,x_2,0)^T = 0$
- Any line l intersects with l_∞ line at an ideal point
  - Proof: $(a,b,c)x(0,0,1) = (b,-a,0)$
- Two parallel lines l and l' always meet at an ideal point
  - Proof: Let $l = (a,b,c)^T$ and $l' = (a,b,c')^T$
  - ……

2D Projective Geometry: Basics

- A point: $(x, y, z)^T \in P^2 \quad \Leftrightarrow \quad (\frac{x}{z}, \frac{y}{z})^T \in \square^2$
- A line: $ax + by + cz = 0 \quad \Leftrightarrow \quad a(\frac{x}{z}) + b(\frac{y}{z}) + c = 0$

  we denote a line with a 3-vector $(a, b, c)^T$
- Points and lines are dual: p is on l if $l^T p = 0$
- Intersection of two lines: $l_1 \times l_2,$
- A line through two points: $p_1 \times p_2$

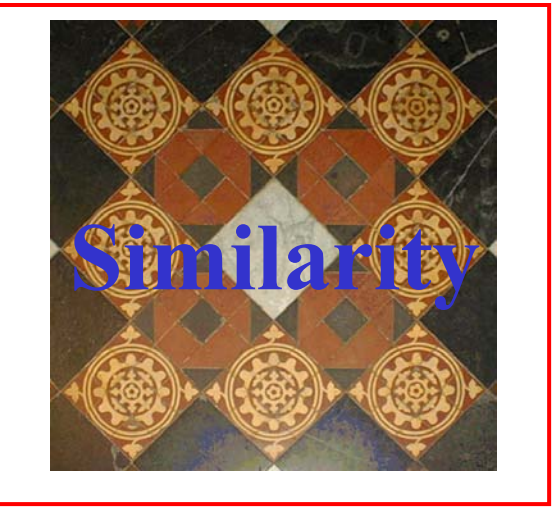
A hierarchy of transformations

Euclidean group (upper left 2x2 orthogonal)

Similarity groun (scaled Euclidean)

Affine group (last row (0,0,1))

Projective linear group (general)

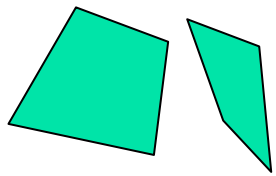


- *Can be described algebraically*
  - *characterized by invertible 3x3 matrices*
  - *or in terms of invariants*

Overview transformations

Projective
8dof

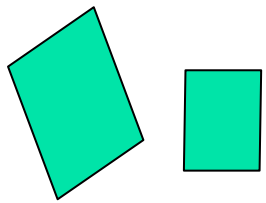
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$



Concurrency, collinearity, order of contact (intersection, tangency, inflection, etc.), cross ratio

Affine
6dof

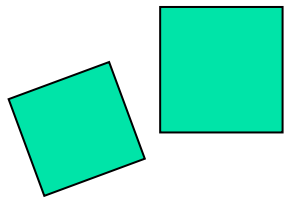
$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Parallellism, ratio of areas, ratio of lengths on parallel lines (e.g midpoints), linear combinations of vectors (centroids).
The line at infinity l_∞

Similarity
4dof

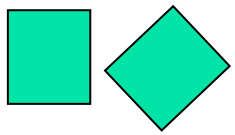
$$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Ratios of lengths, angles.
The circular points I,J

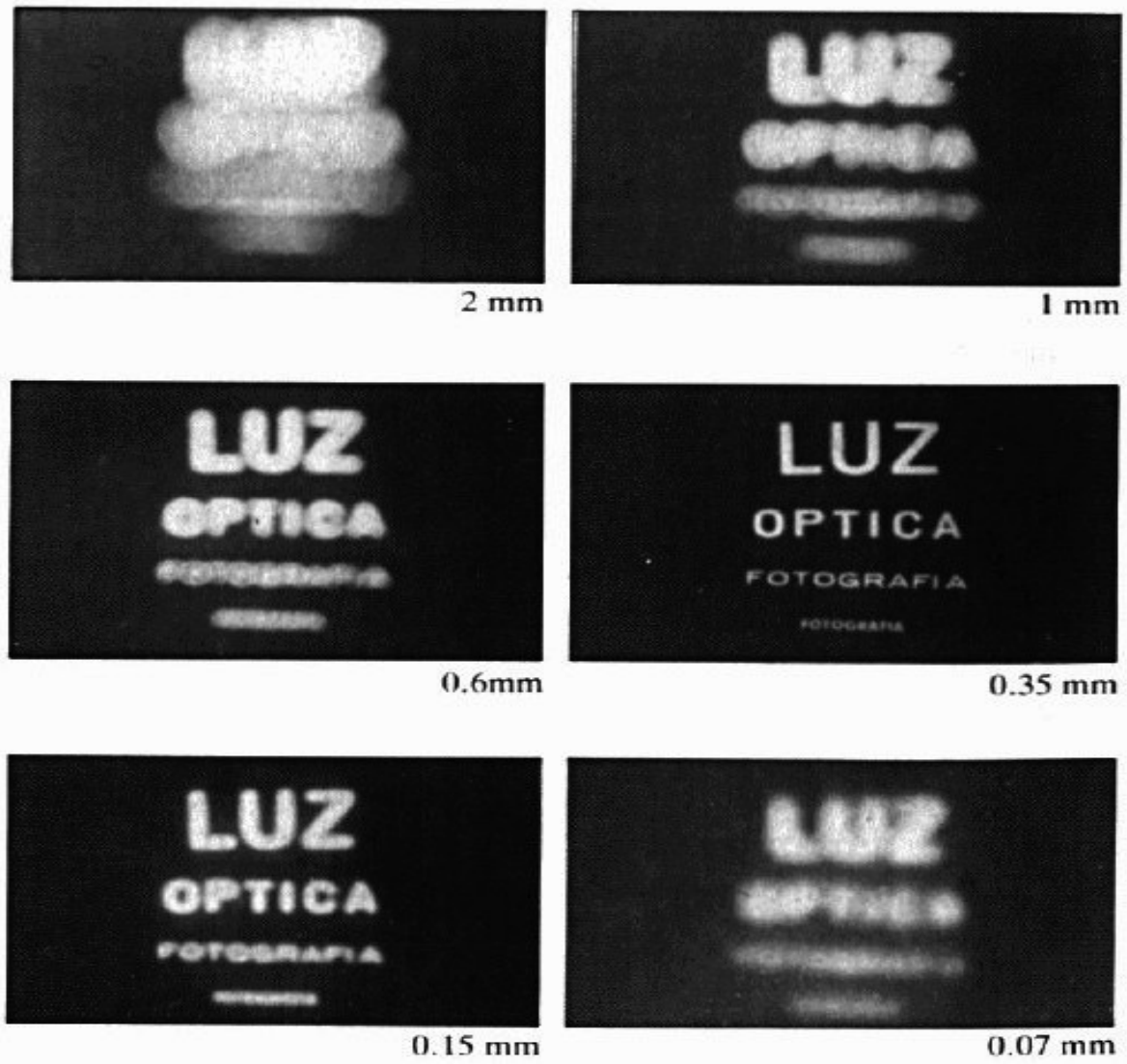
Euclidean
3dof

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



lengths, areas.

Shrinking the aperture

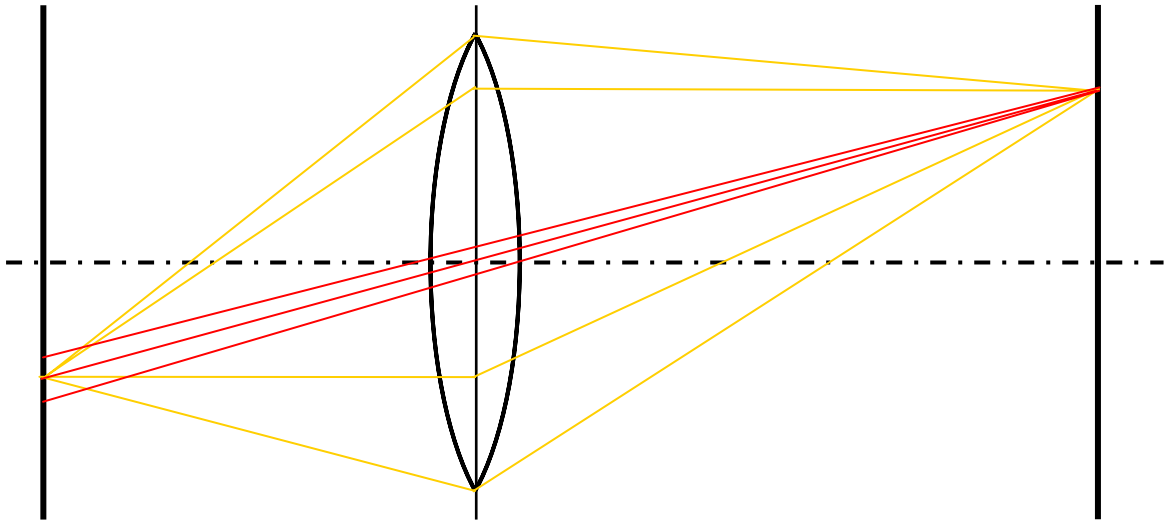


Why not make the aperture as small as possible?

- Less light gets through
- *Diffraction* effects...

Lens

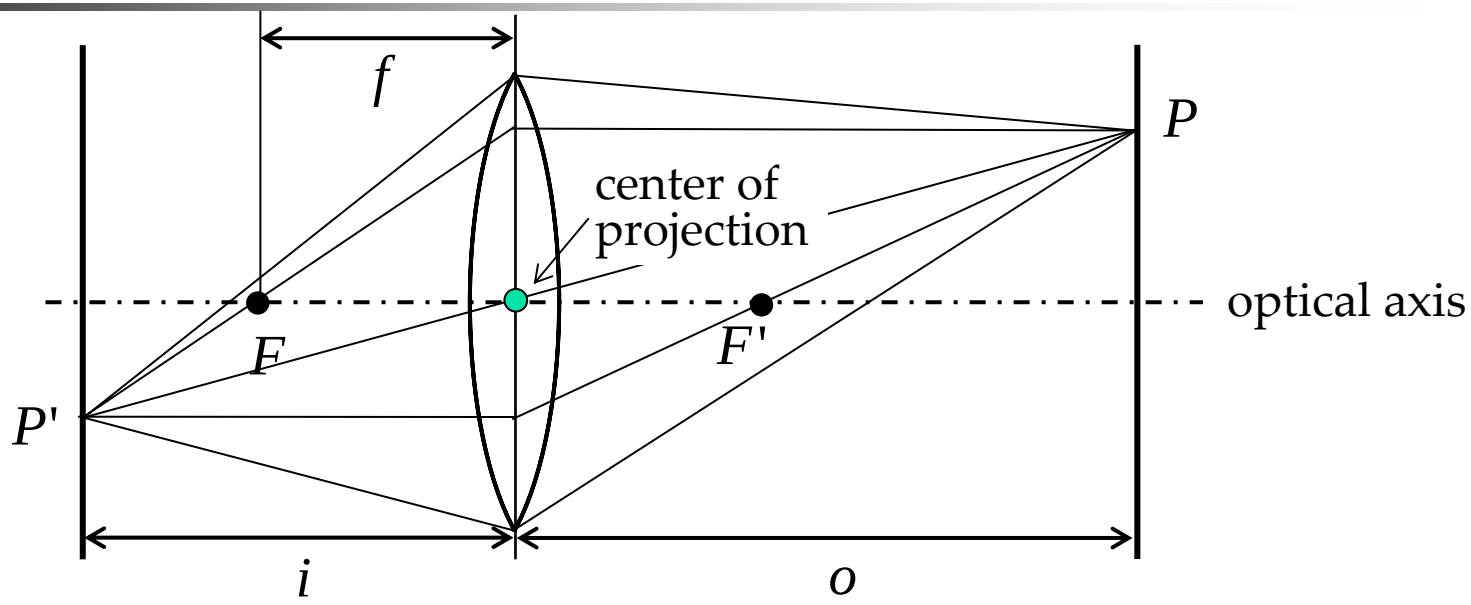
- Ideal lens realizes the same projection as a pinhole but **gathers much more light**!



Thin Lens: Properties

1. Any ray entering a thin lens parallel to the optical axis must go through the focus on other side
2. Any ray entering through the focus on one side will be parallel to the optical axis on the other side
3. Any ray passing through the optical center does not change its direction

Thin-lens Law (Gaussian Lens Formula)



f : focal length

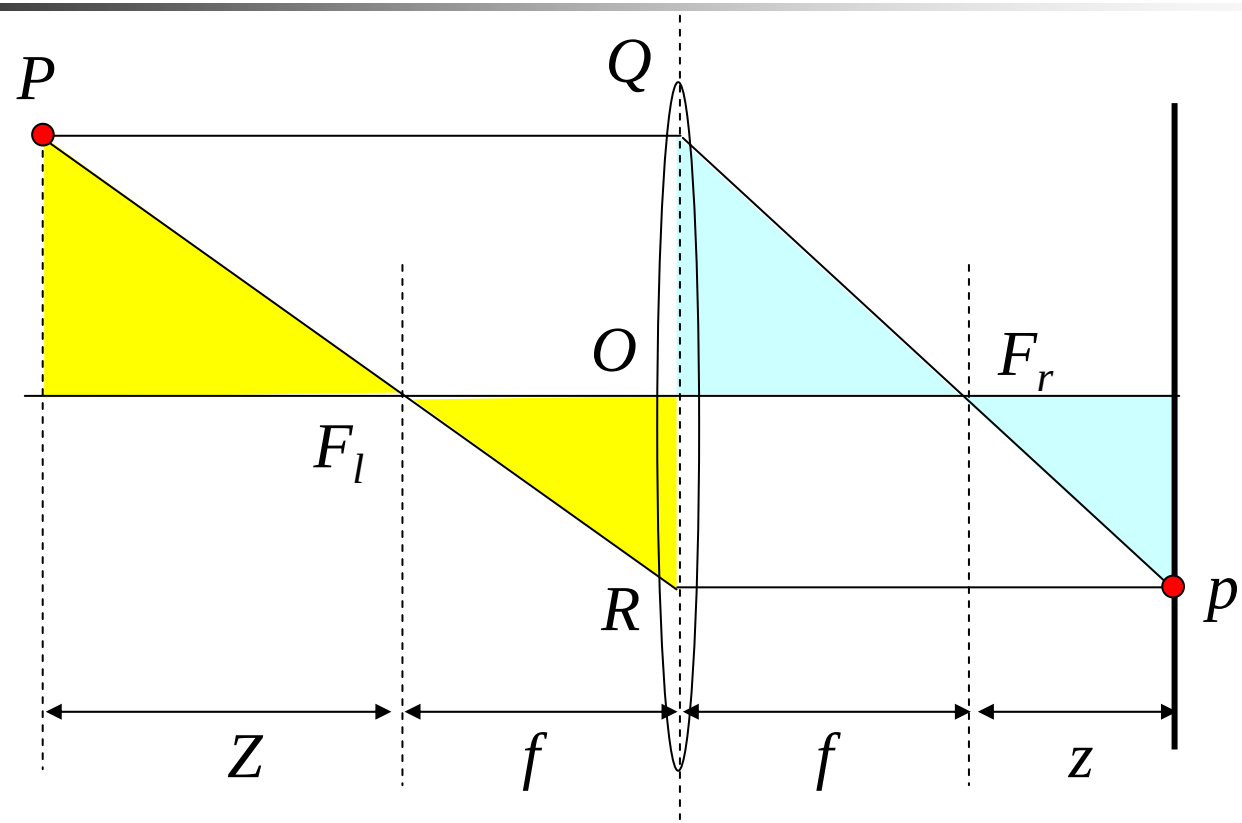
$$\frac{1}{i}+\frac{1}{o}=\frac{1}{f}$$

For example, $f=50\text{mm}$ $o=300\text{mm}$ $\longrightarrow$ $i=60\text{mm}$

Thin Lens: Model

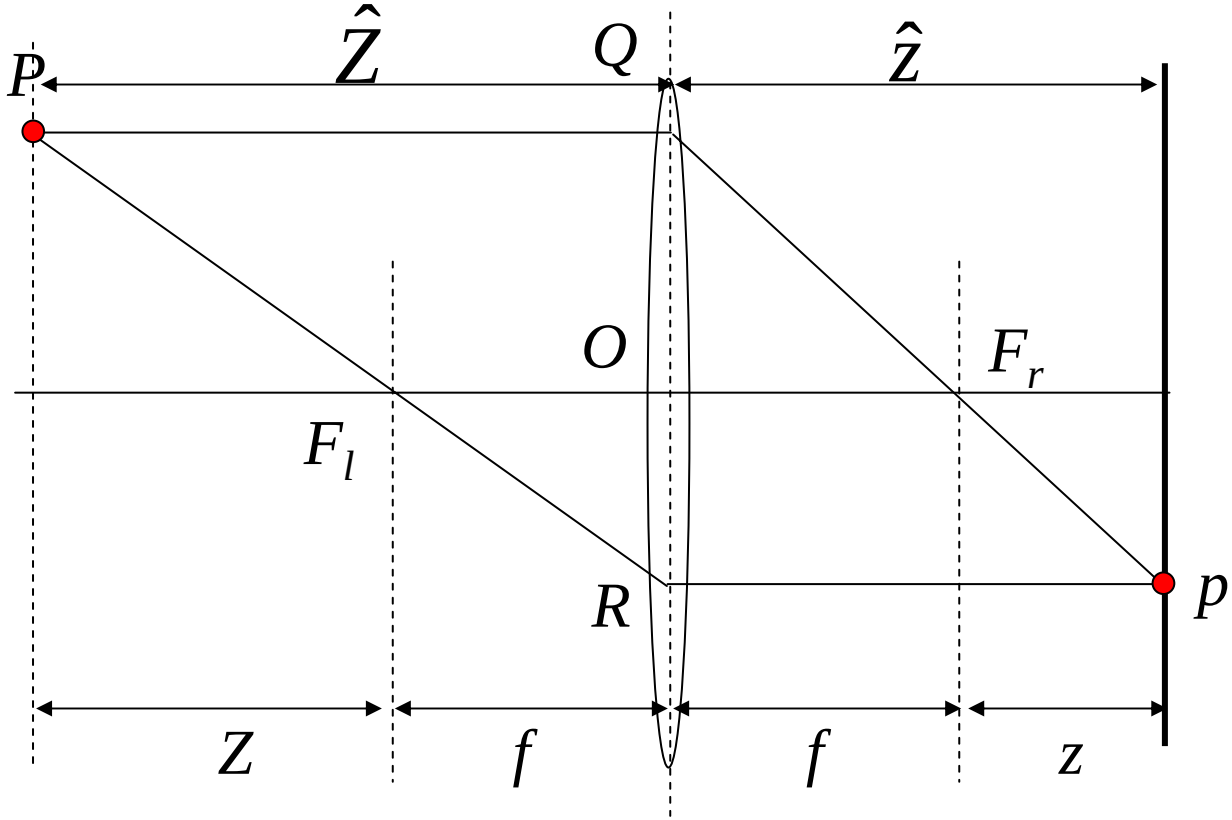
$$\frac{QO}{Z} = \frac{OR}{f}$$

$$\frac{OR}{z} = \frac{QO}{f}$$



$$OR = \frac{f}{Z} QO = \frac{z}{f} QO \qquad \frac{f}{Z} = \frac{z}{f} \Rightarrow f^2 = zZ$$

The Thin Lens Law



$$f^2 = zZ$$

$$\hat{z} = z + f$$

$$\hat{Z} = Z + f$$

$$\frac{1}{\hat{z}} + \frac{1}{\hat{Z}} = \frac{1}{f}$$

Limits of the Thin Lens Model

3 assumptions :

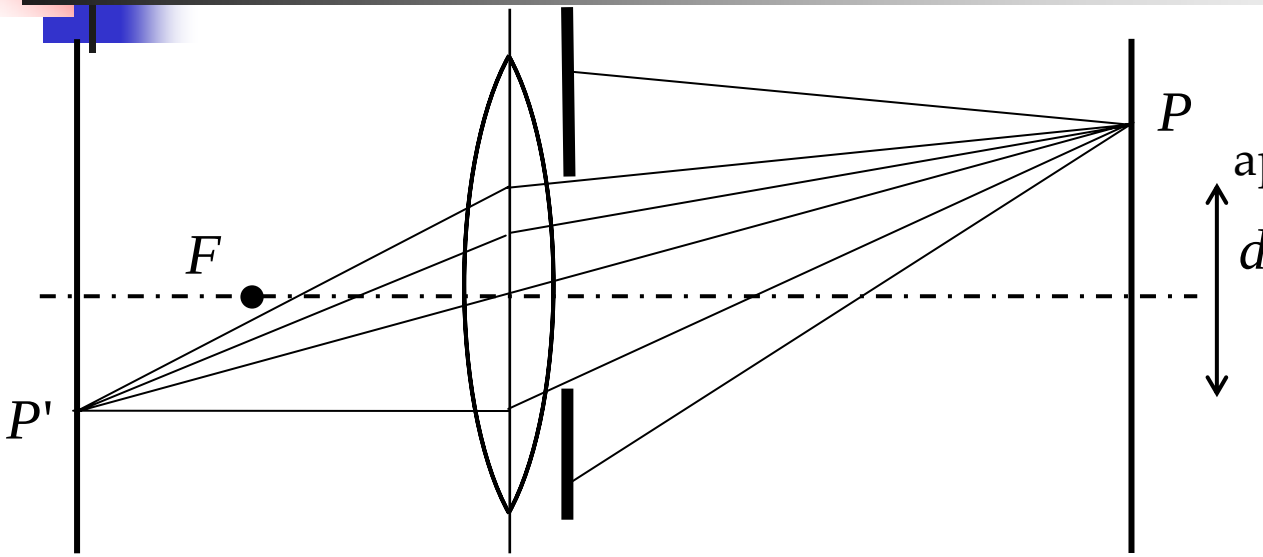
1. all rays from a point are focused onto 1 image point
   - Remember thin lens small angle assumption

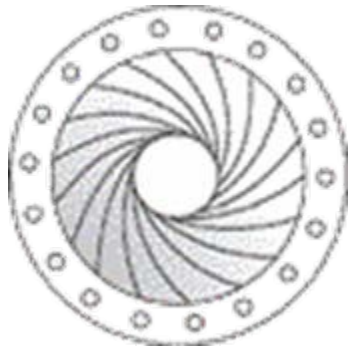
2. all image points in a single plane

3. magnification $m = \frac{f'}{z_0}$ is constant

Deviations from this ideal are *aberrations*

Aperture and F/stop





iris diaphragm

F/stop: for instance, f/1.0 f/1.4 f/2.0 f/2.8 f/4 f/5.6 f/8 f/11

less light

aperture areas is halved at each stop

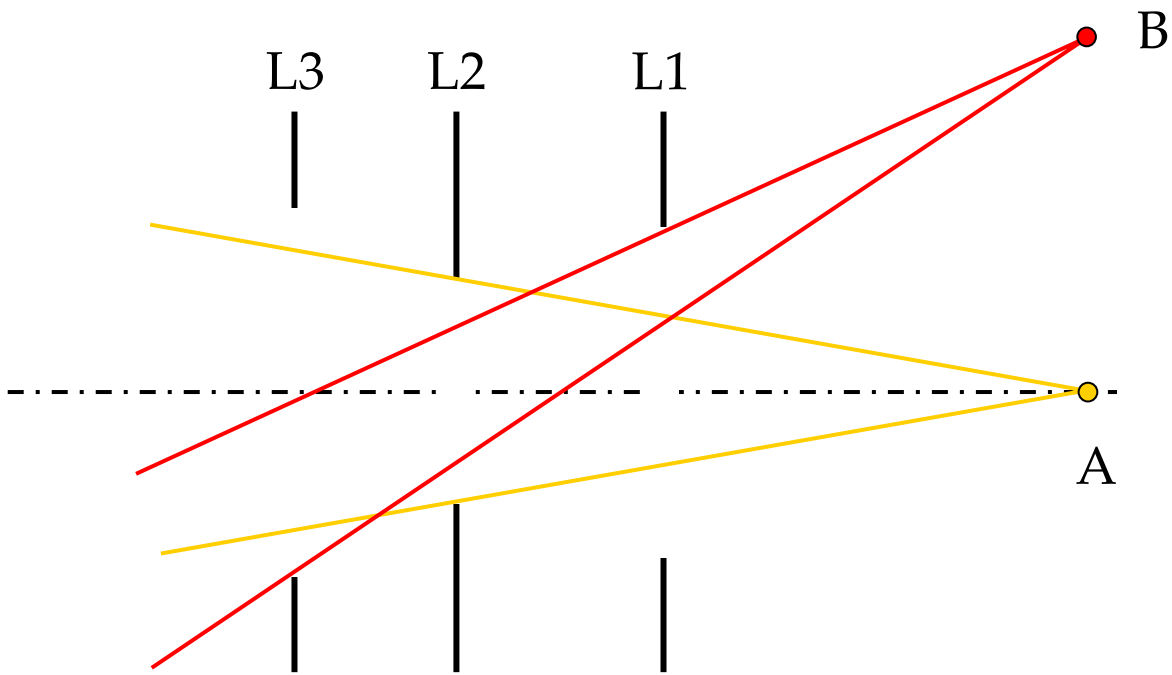
f/stop =	f/1.4	f/2.0	f/2.8	f/4	f/5.6	f/8	f/11
shutter speed =	1/1000	1/500	1/250	1/125	1/60	1/30	1/15

Vignetting

Vignetting

Optical Vignetting - Aperture dependency

At wider aperture, on the edge of the field, the entrance pupil can be partially shielded by the lens body. This is why optical vignetting increases with aperture.

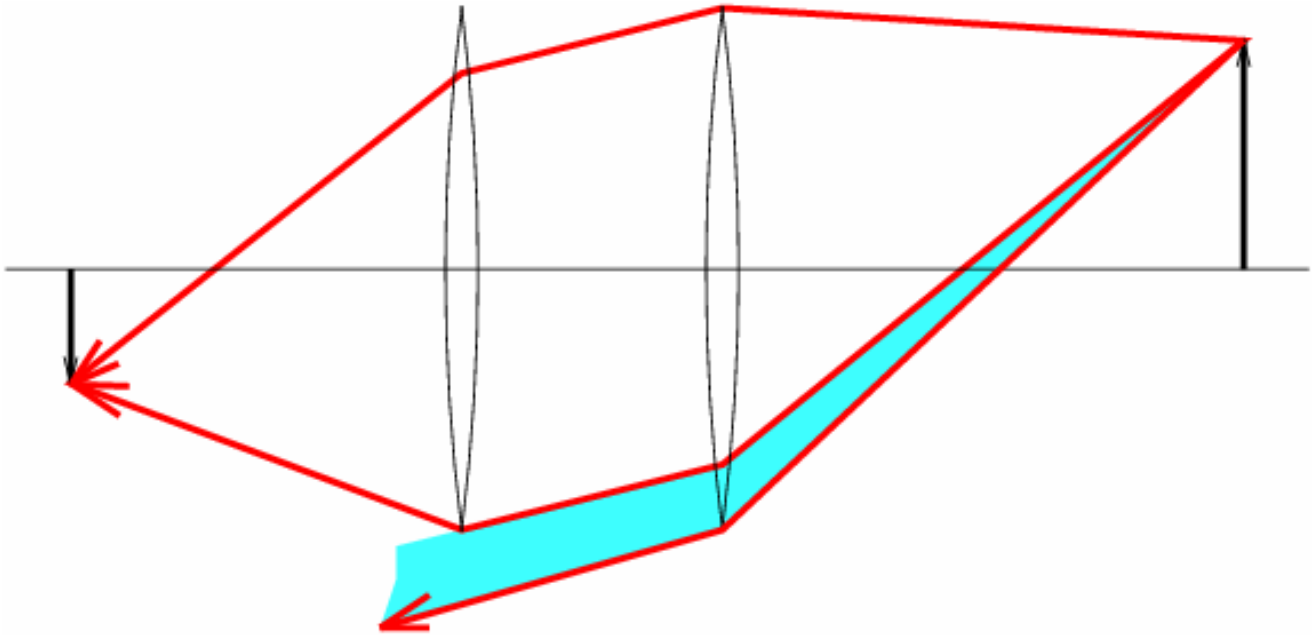


More light passes through lens L3 for scene point A than scene point B.
Results in spatially non-uniform brightness (in the periphery of the image)

Vignetting

Natural Vignetting - Lens Dependency

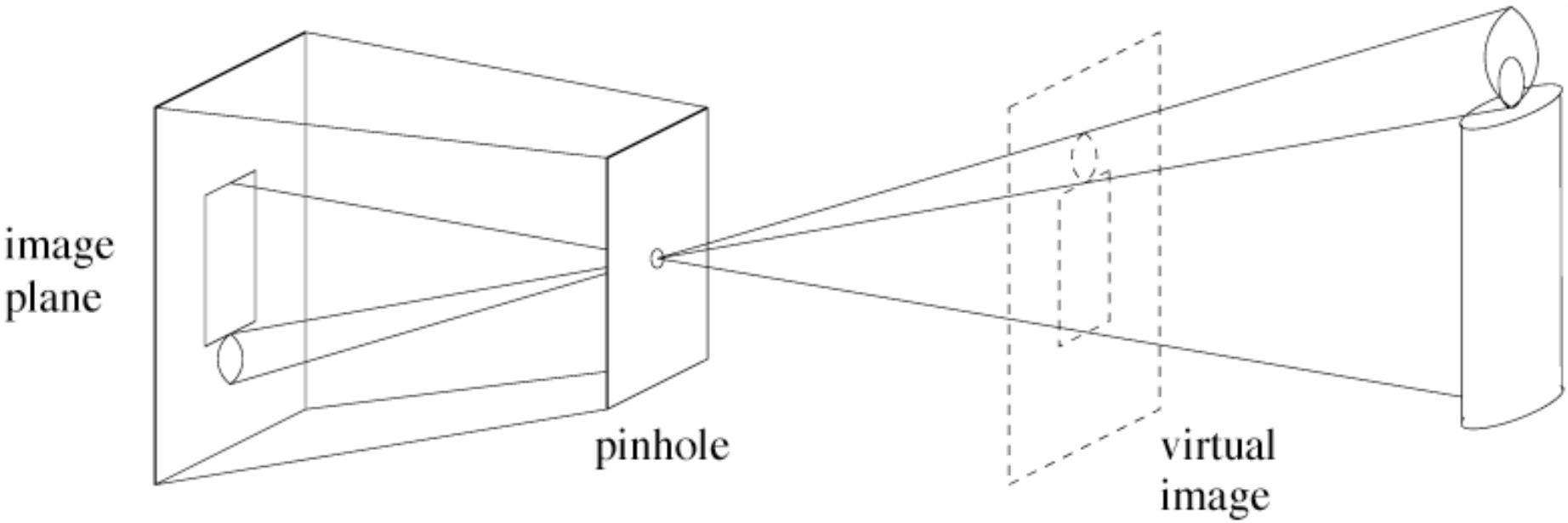
- Natural vignetting is inherent to lens design, regardless of aperture.
- With a zoom lens, it generally increases as the focal length decreases.



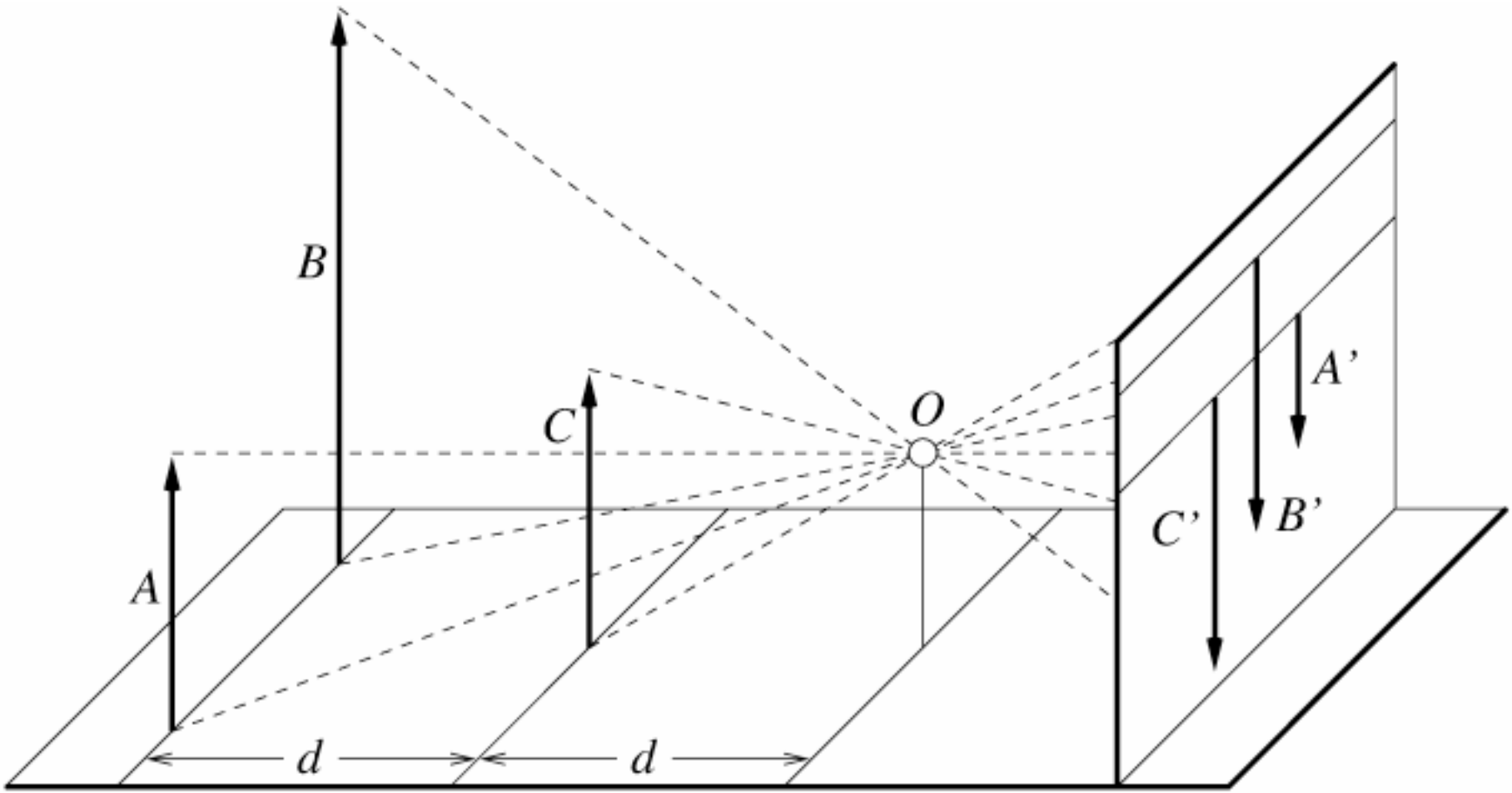
Effect: Darkens pixels near the image boundary

Pinhole cameras

- Abstract camera model - box with a small hole in it
- Pinhole cameras work in practice

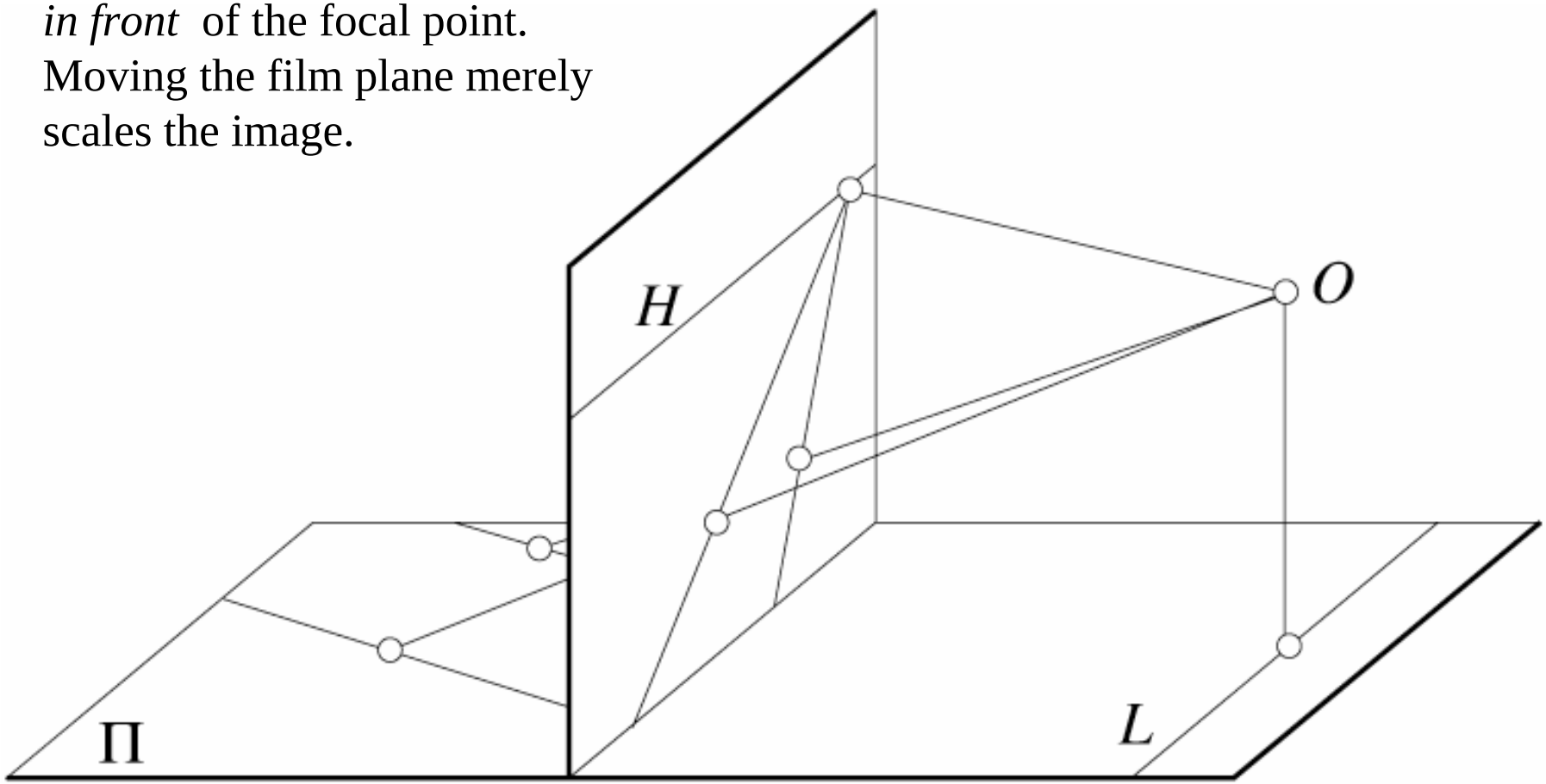


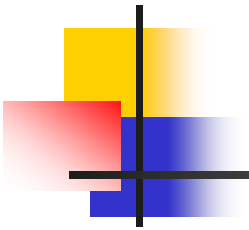
Distant objects are smaller



Parallel lines meet

Common to draw film plane *in front* of the focal point. Moving the film plane merely scales the image.

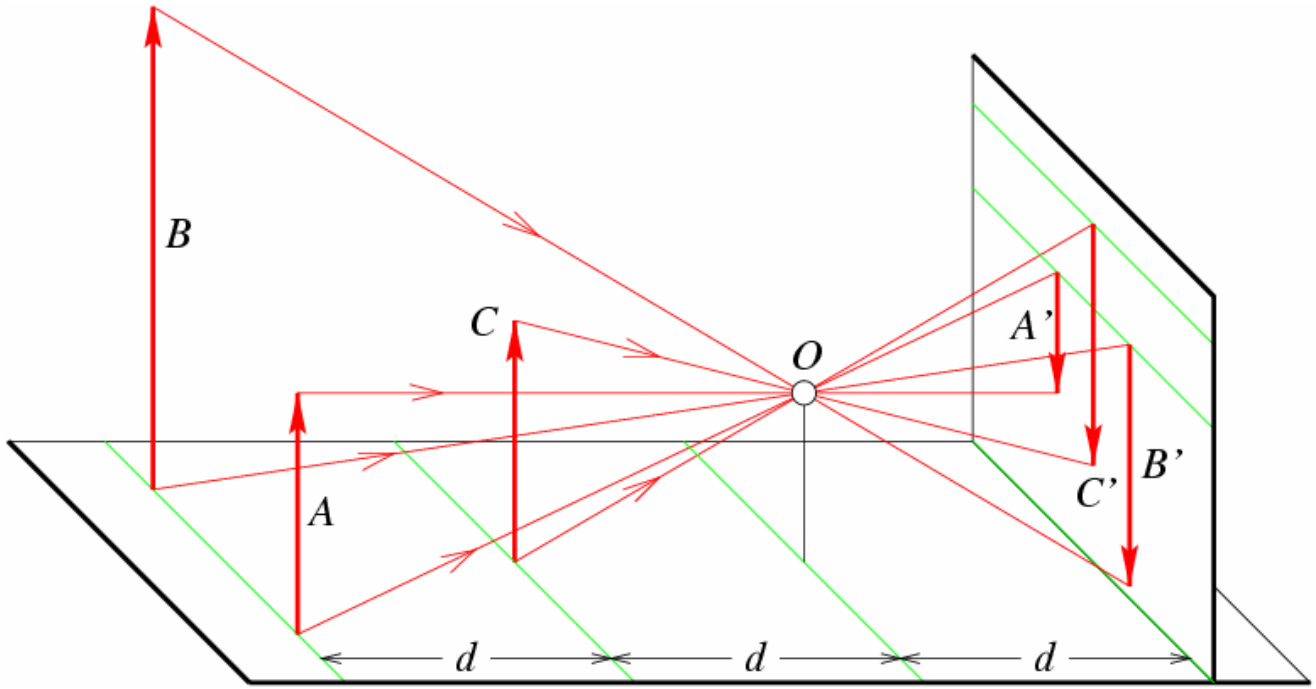




Vanishing points

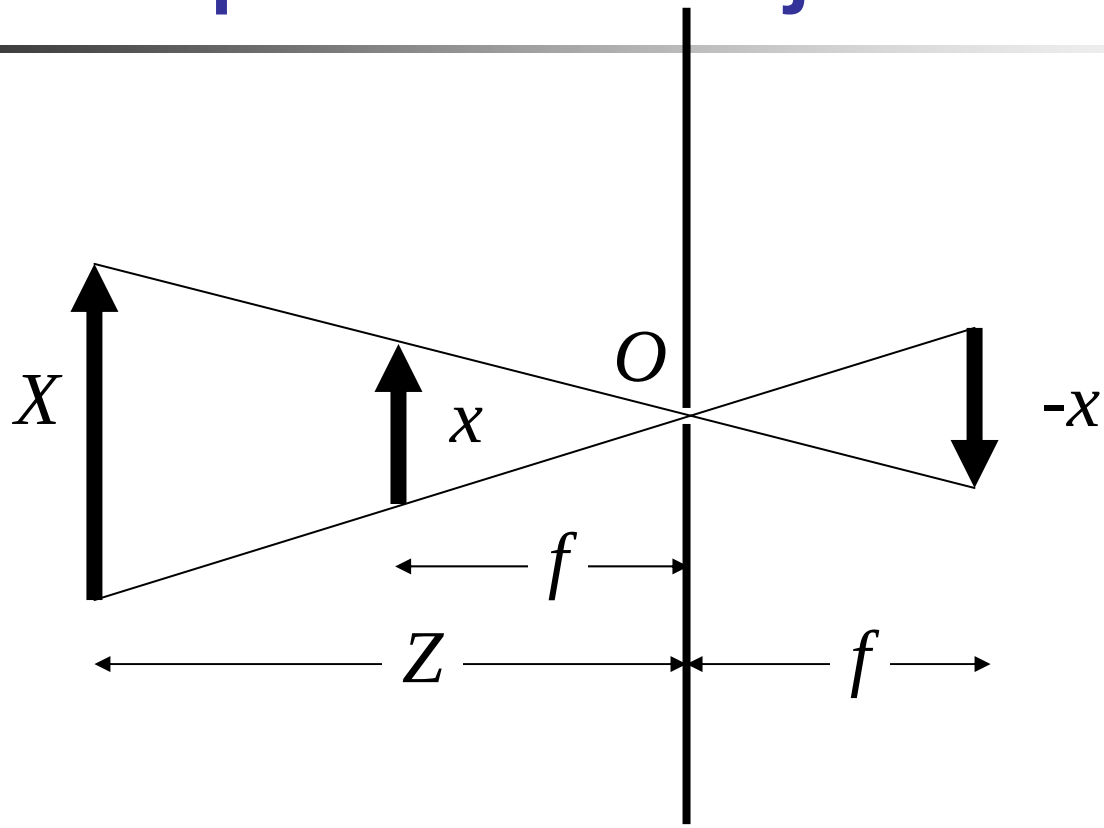
- Each set of parallel lines meets at a different point
  - The *vanishing point* for this direction
- Sets of parallel lines on the same plane lead to *collinear* vanishing points.
  - The line is called the *horizon (or vanishing line)* for that plane

Perspective Projection

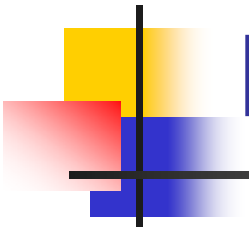


A “similar triangle’s” approach to vision.

Perspective Projection



$$x = -X \frac{f}{Z}$$

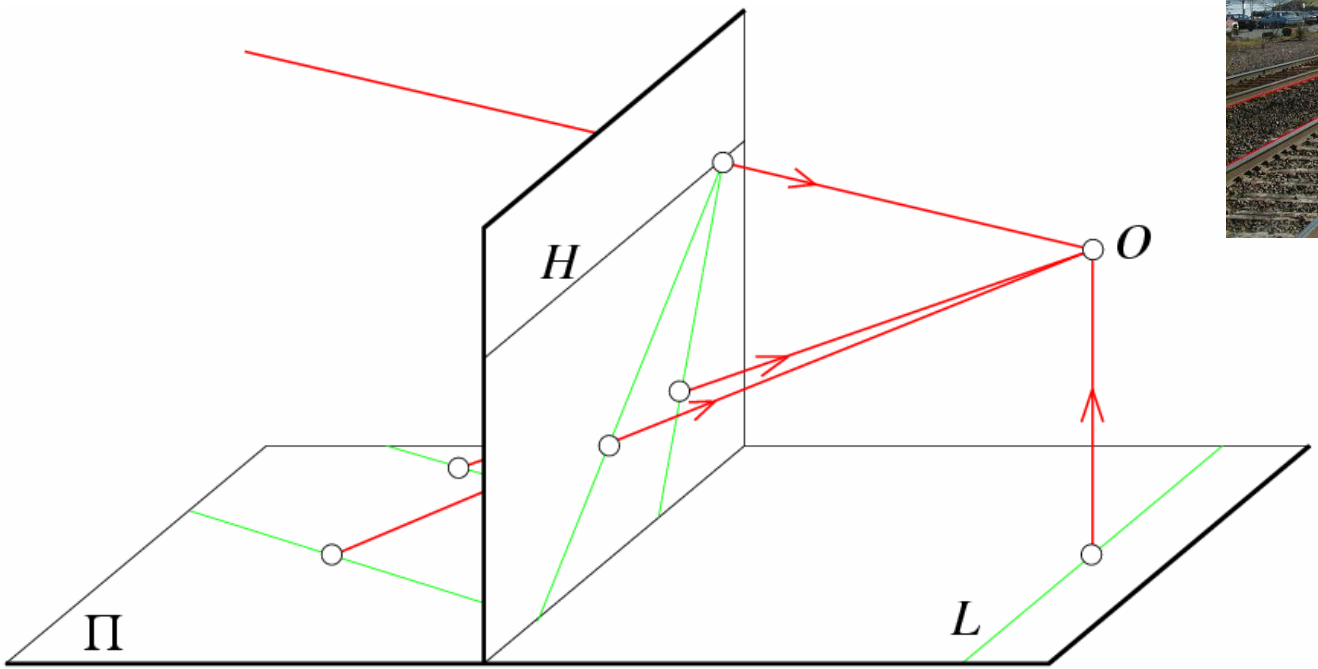


Properties of Projection

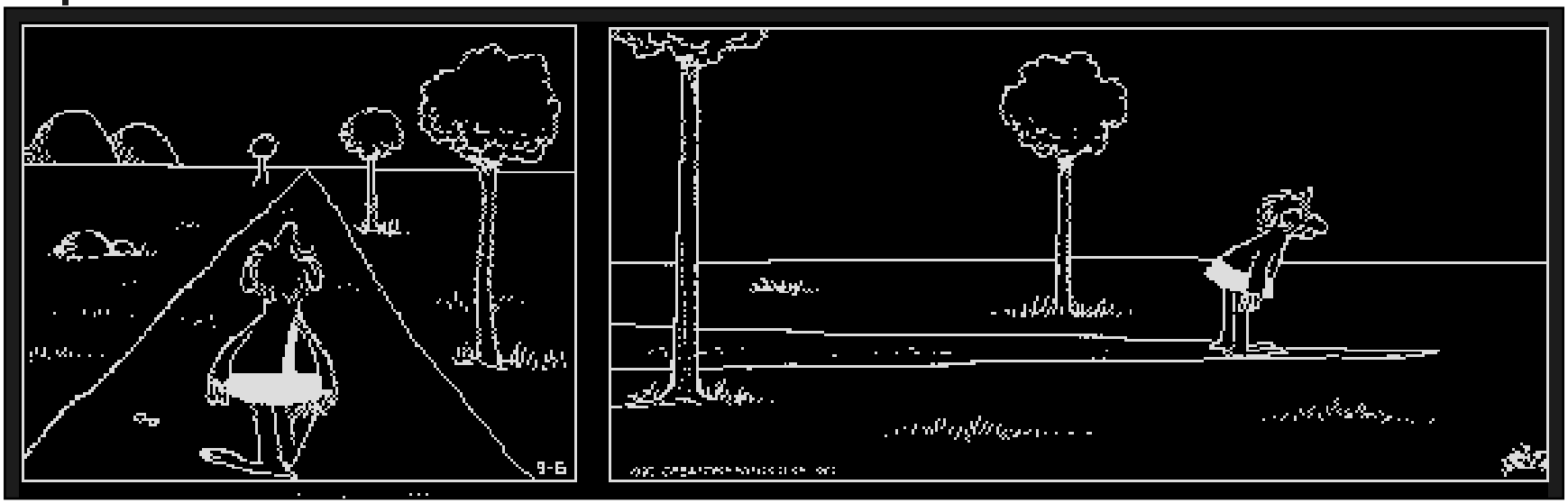
- Points project to points
- Lines project to lines
- Planes project to the whole image or a half image
- Angles are not preserved
- Degenerate cases
  - Line through focal point projects to a point.
  - Plane through focal point projects to line

Consequences: Parallel lines meet

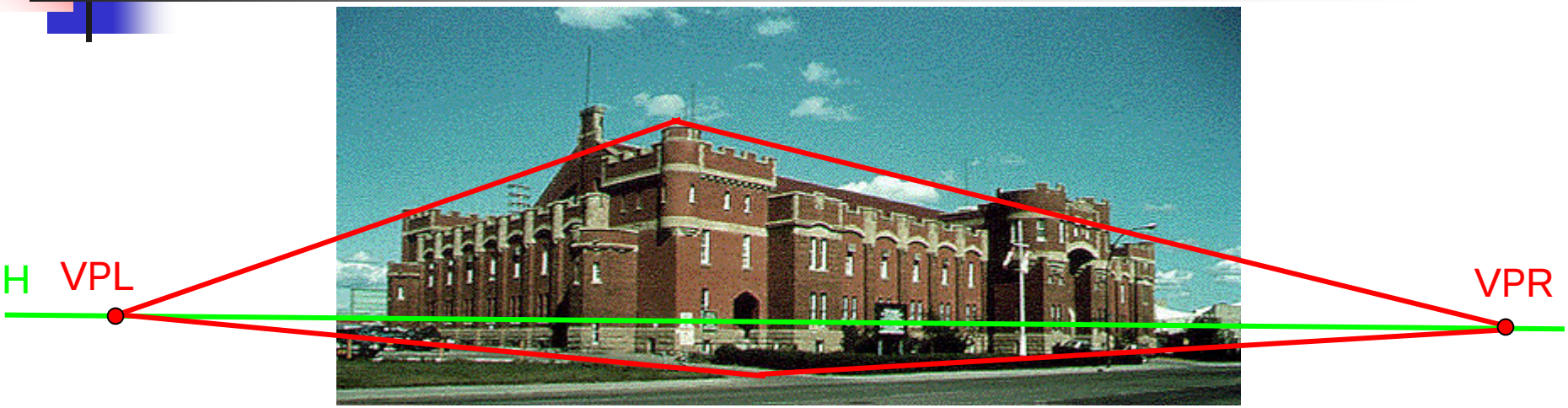
- There exist vanishing points

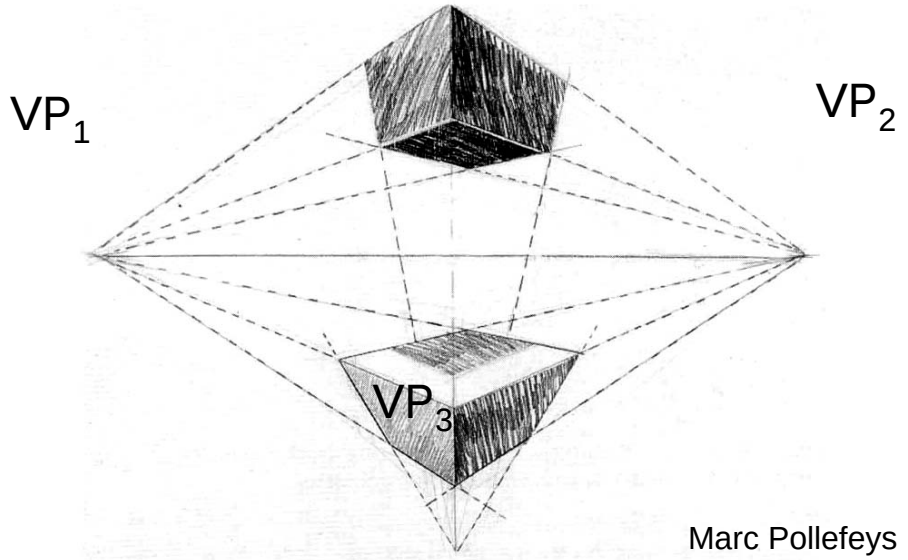


The Effect of Perspective



Vanishing points

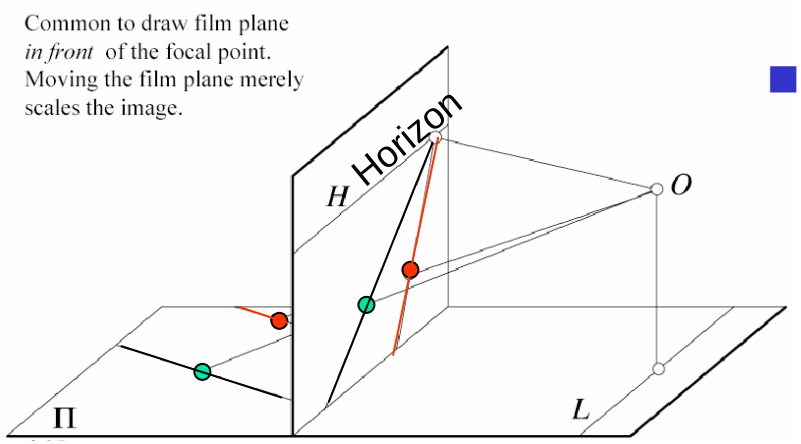




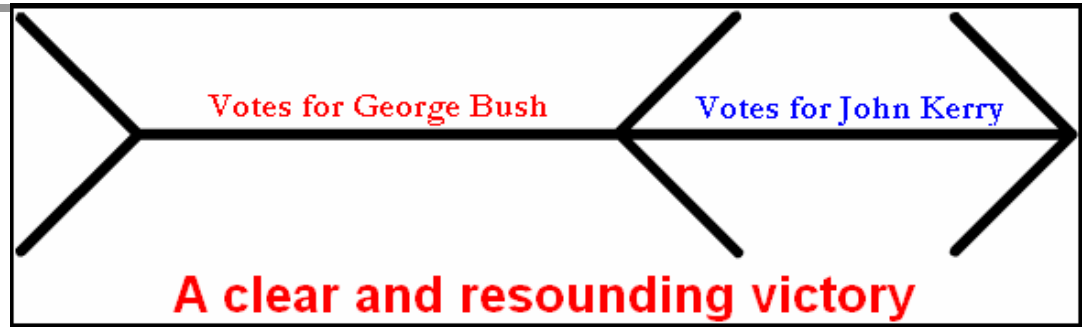
Different directions correspond to different vanishing points

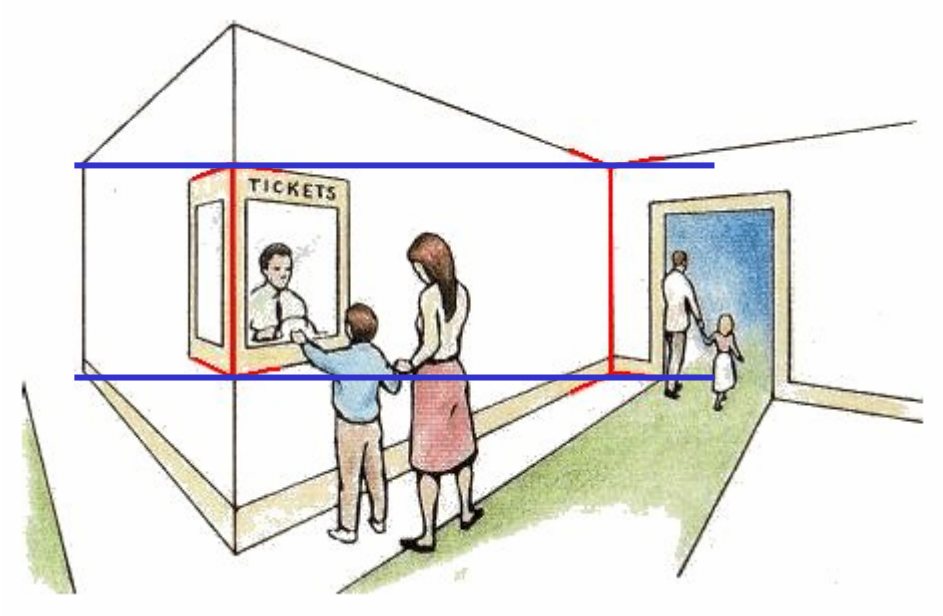
Perspective Projection

- Objects farther appear smaller
- Points go to Points
- Lines go to Lines
- Polygons go to Polygons
- Parallel lines meet



Müller-Lyer Illusion

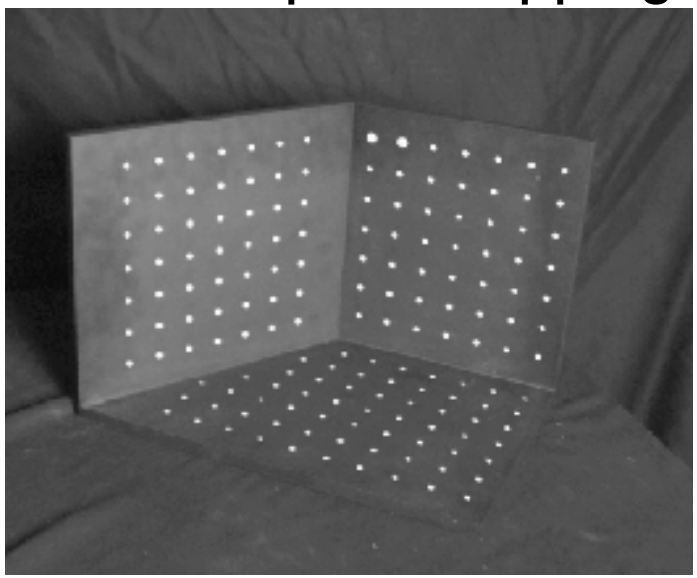




http://www.michaelbach.de/ot/sze_muelue/index.html

Estimating the Projection Matrix

- Place a known object in the scene
  - identify correspondence between image and scene
  - compute mapping from scene to image

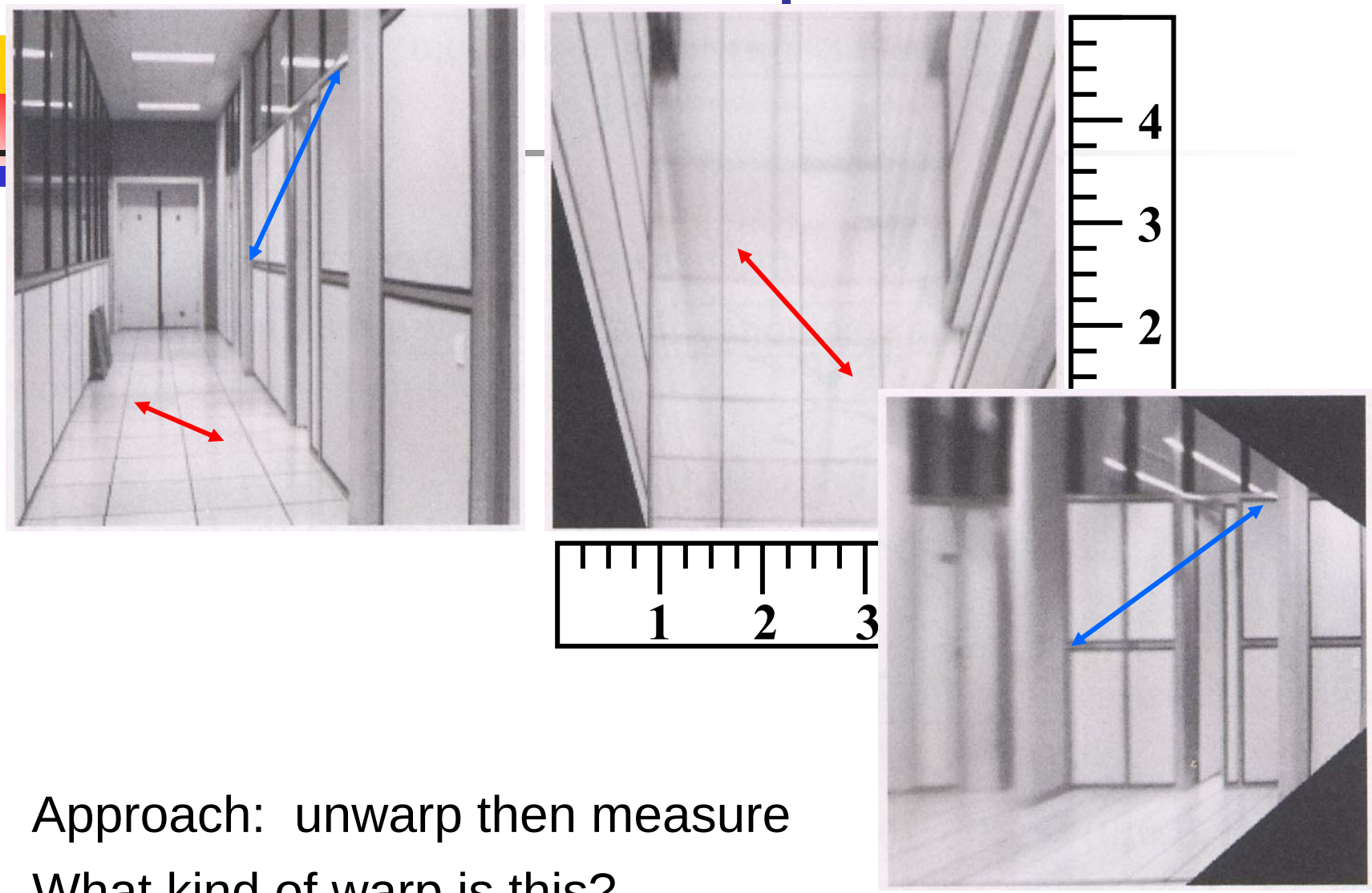


$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Issues

- must know geometry very accurately
- must know 3D->2D correspondence

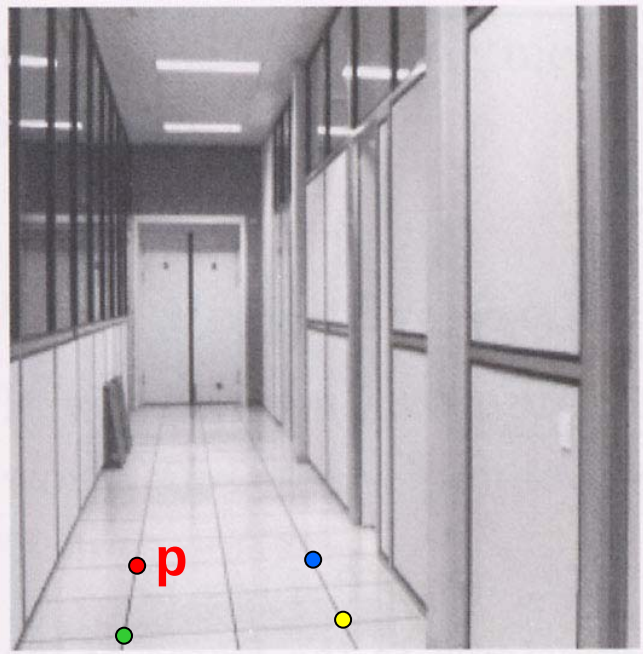
Measurements on planes

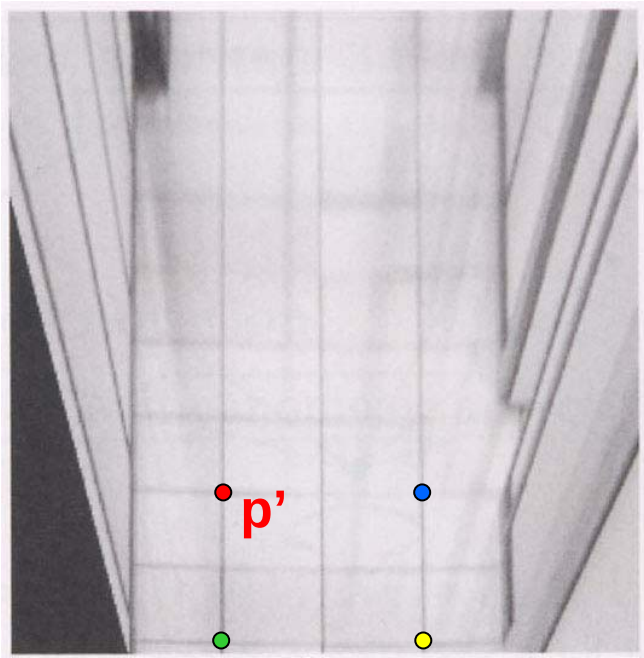


Approach: unwarp then measure

What kind of warp is this?

Image rectification

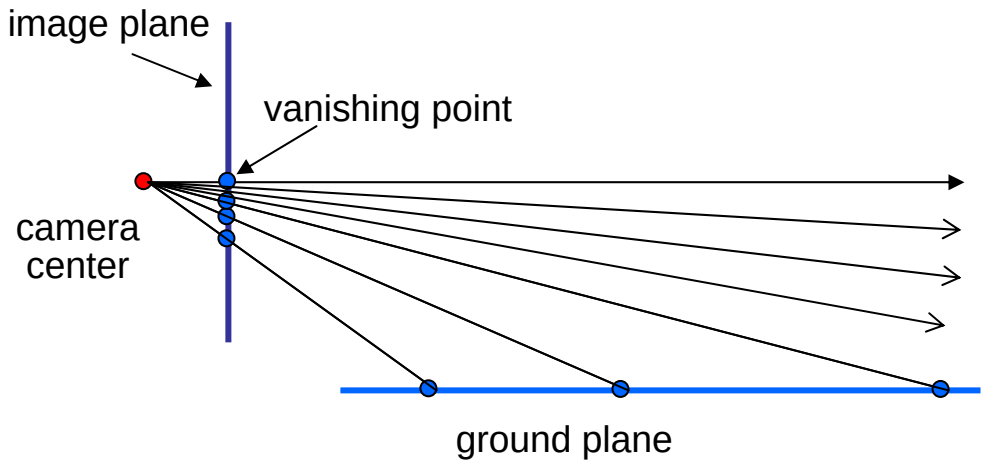




To unwarp (rectify) an image

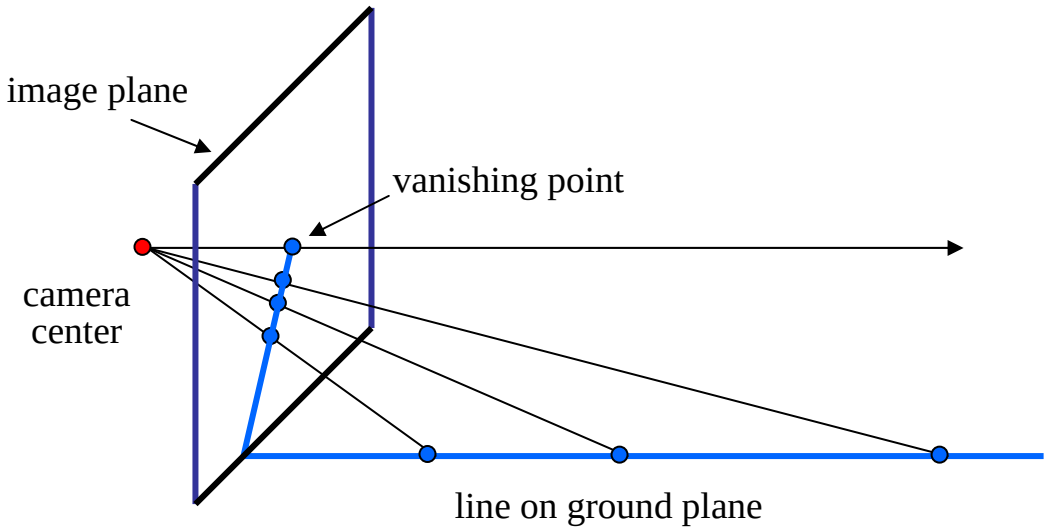
- solve for homography **H** given **p** and **p’**
- solve equations of the form: w**p’** = **Hp**
  - linear in unknowns: w and coefficients of **H**
  - H is defined up to an arbitrary scale factor
  - how many points are necessary to solve for **H**?

Vanishing points

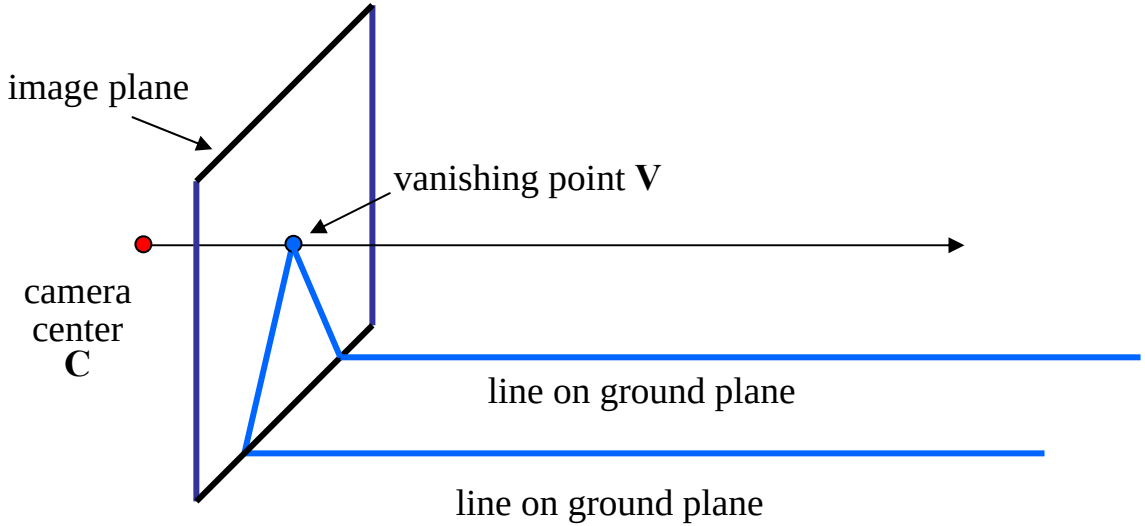


- Vanishing point
  - projection of a point at infinity

Vanishing points (2D)



Vanishing points

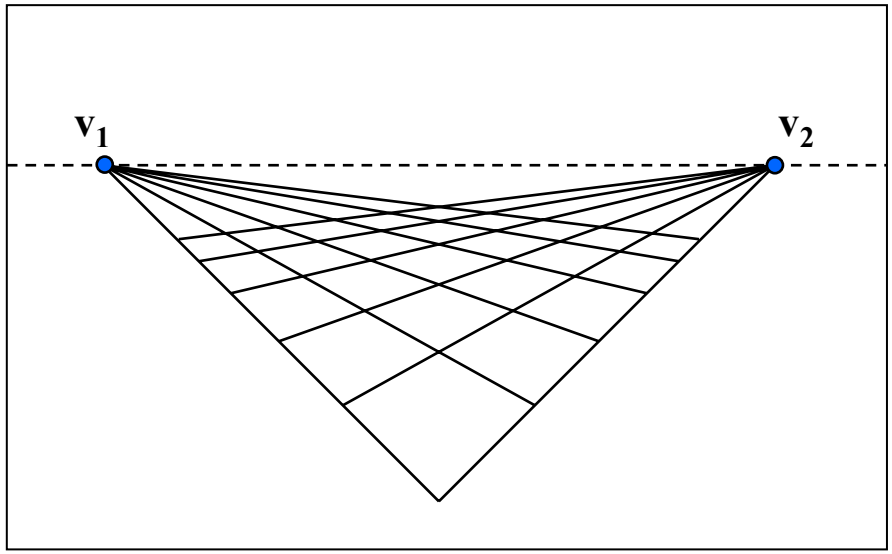


- Properties
  - Any two parallel lines have the same vanishing point **v**
  - The ray from **C** through **v** is parallel to the lines
  - An image may have more than one vanishing point
    - in fact every pixel is a potential vanishing point

Vanishing points

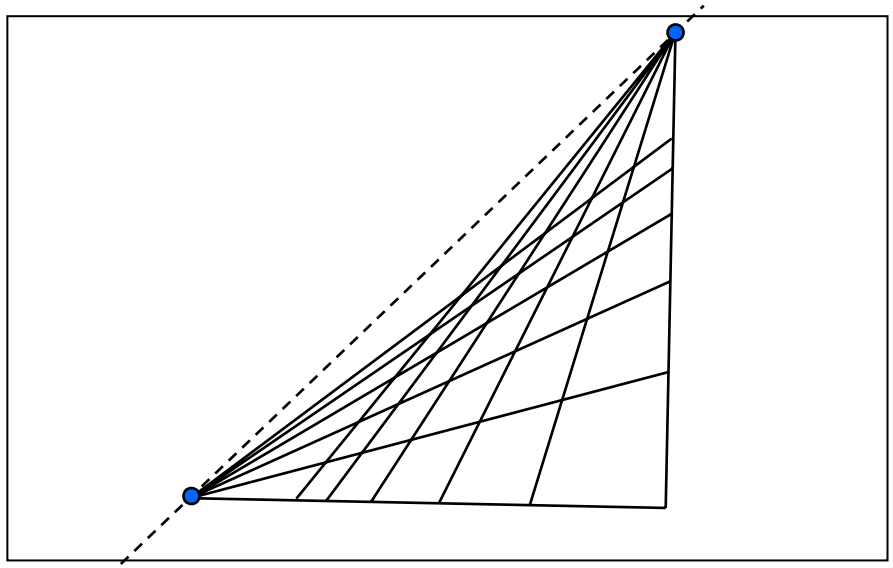
Image by Q-T. Luong (a vision researcher & photographer)

Vanishing lines



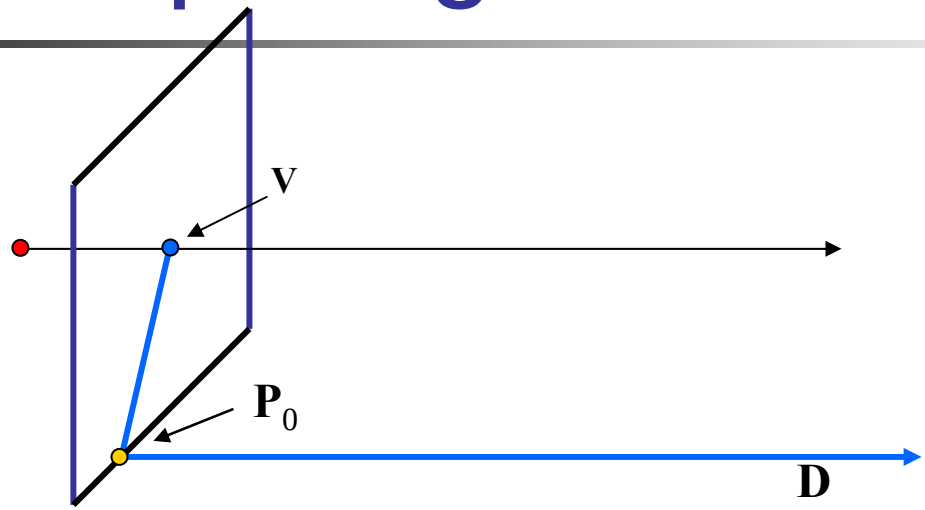
- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the *horizon line*
    - also called *vanishing line*
  - Note that different planes define different vanishing lines

Vanishing lines



- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the *horizon line*
    - also called *vanishing line*
  - Note that different planes define different vanishing lines

Computing vanishing points

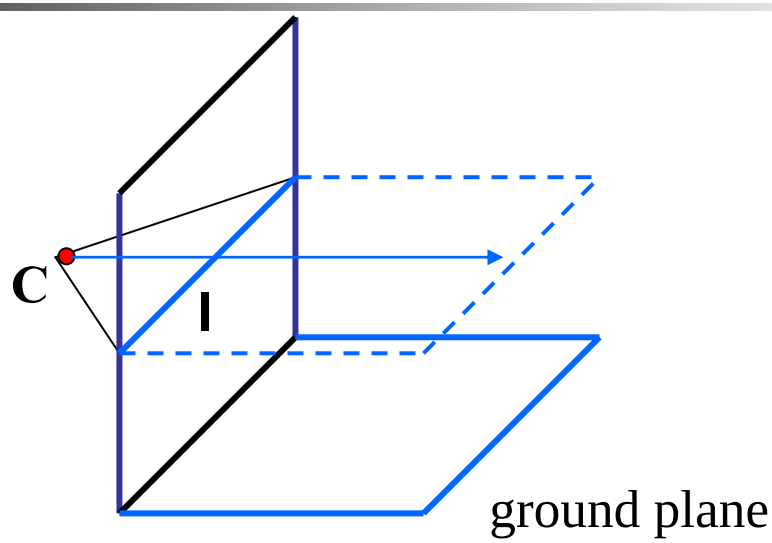


$$\mathbf{P} = \mathbf{P}_0 + t\mathbf{D}$$

$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X/t + D_X \\ P_Y/t + D_Y \\ P_Z/t + D_Z \\ 1/t \end{bmatrix} \qquad t \to \infty \qquad \mathbf{P}_\infty \cong \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix}$$

- Properties $\mathbf{v} = \mathbf{\Pi P}_\infty$
  - $\mathbf{P}_\infty$ is a point at *infinity*, **v** is its projection
  - They depend only on line *direction*
  - Parallel lines $\mathbf{P}_0 + t\mathbf{D}$, $\mathbf{P}_1 + t\mathbf{D}$ intersect at $\mathbf{P}_\infty$

Computing vanishing lines



- Properties
  - **l** is intersection of horizontal plane through **C** with image plane
  - Compute **l** from two sets of parallel lines on ground plane
  - All points at same height as **C** project to **l**
    - points higher than C project above l
  - Provides way of comparing height of objects in the scene

Measuring height