מבוא לראייה ממוחשבת – 22928

2016א

מנחה: אמיר אגוזי

egozi5@gmail.com

מפגש מס' 5

בפעם שעברה:

- זיהוי פנים – Viola & Jones
- Adaboost
- Detection by classification
- Sliding window
- Global vs. part-based

היום

- Texture synthesis
- Image quilting
- Image transformations
- Estimation
- RANSAC
- Camera model

Overview

- Texture synthesis [Efros & Leung, ICCV'99]
- Quilting [Efros & Freeman 2001]
- Image Analogies [Hertzmann et al. 2001]
- Super-resolution [Freeman et al. 2002]
- Scene completion [Hays & Efros 2007]

Slides from: Alyosha Efros, Bill Freeman, James Hayes

http://www.cs.nyu.edu/~fergus/teaching/comp_photo/index.html

Texture

- Texture depicts spatially repeating patterns
- Many natural phenomena are textures

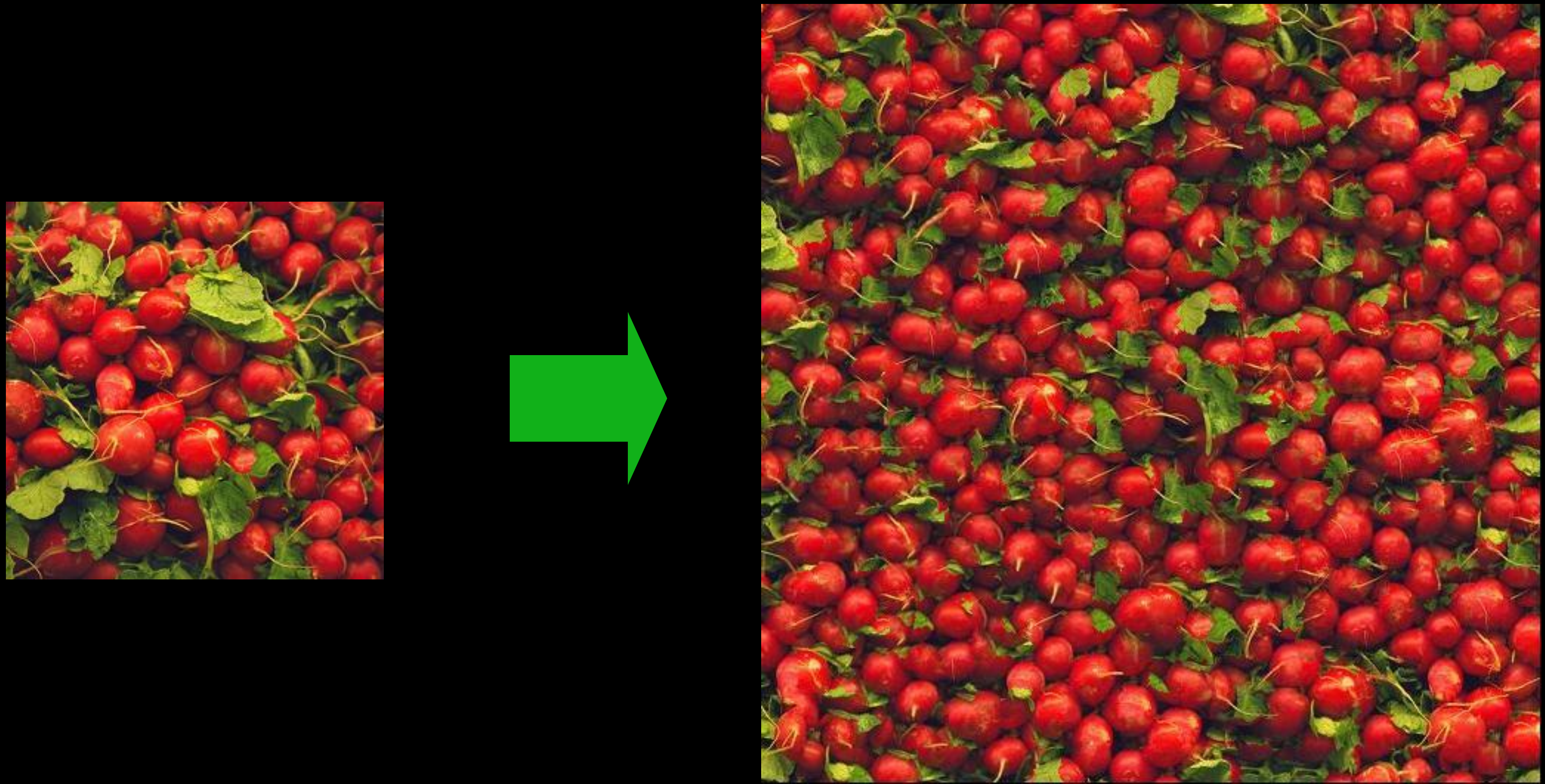
radishes

rocks

yogurt

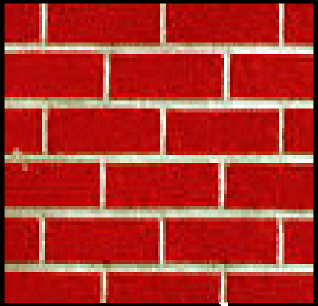
Texture Synthesis

- Goal of Texture Synthesis: create new samples of a given texture
- Many applications: virtual environments, hole-filling, texturing surfaces

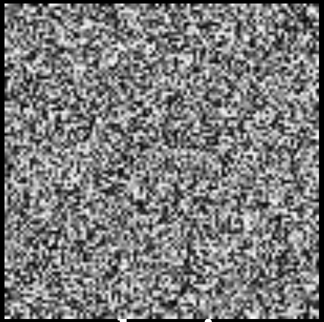


The Challenge

- Need to model the whole spectrum: from repeated to stochastic texture



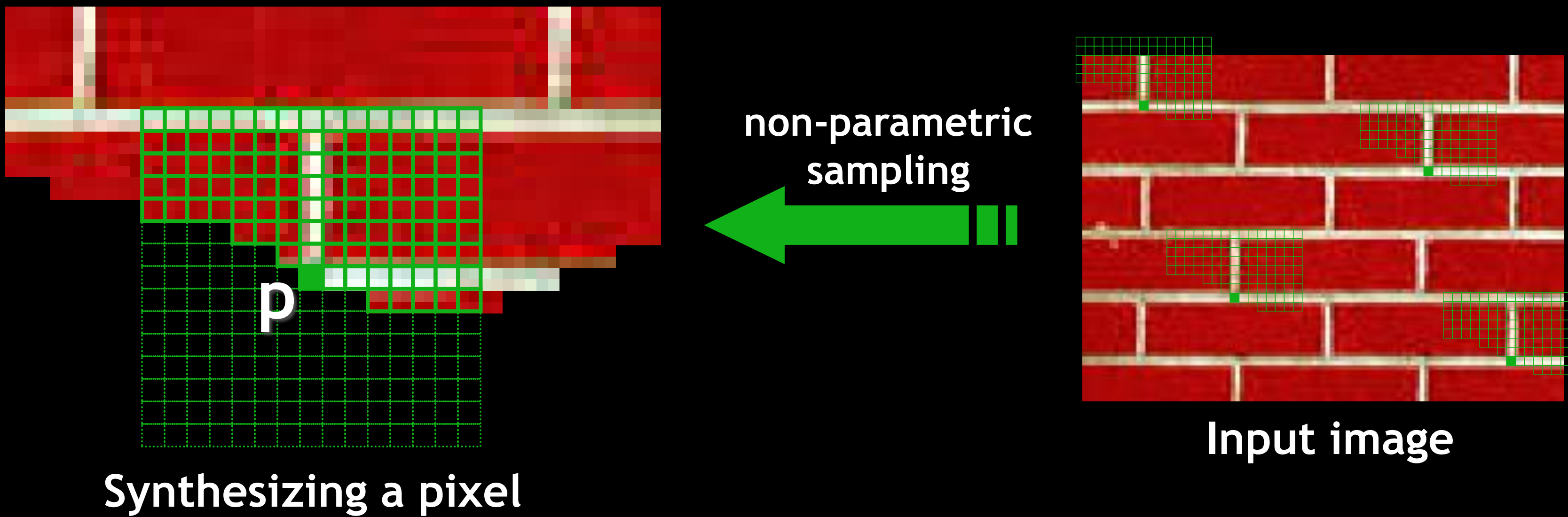
repeated



stochastic

Both?

Efros & Leung Algorithm

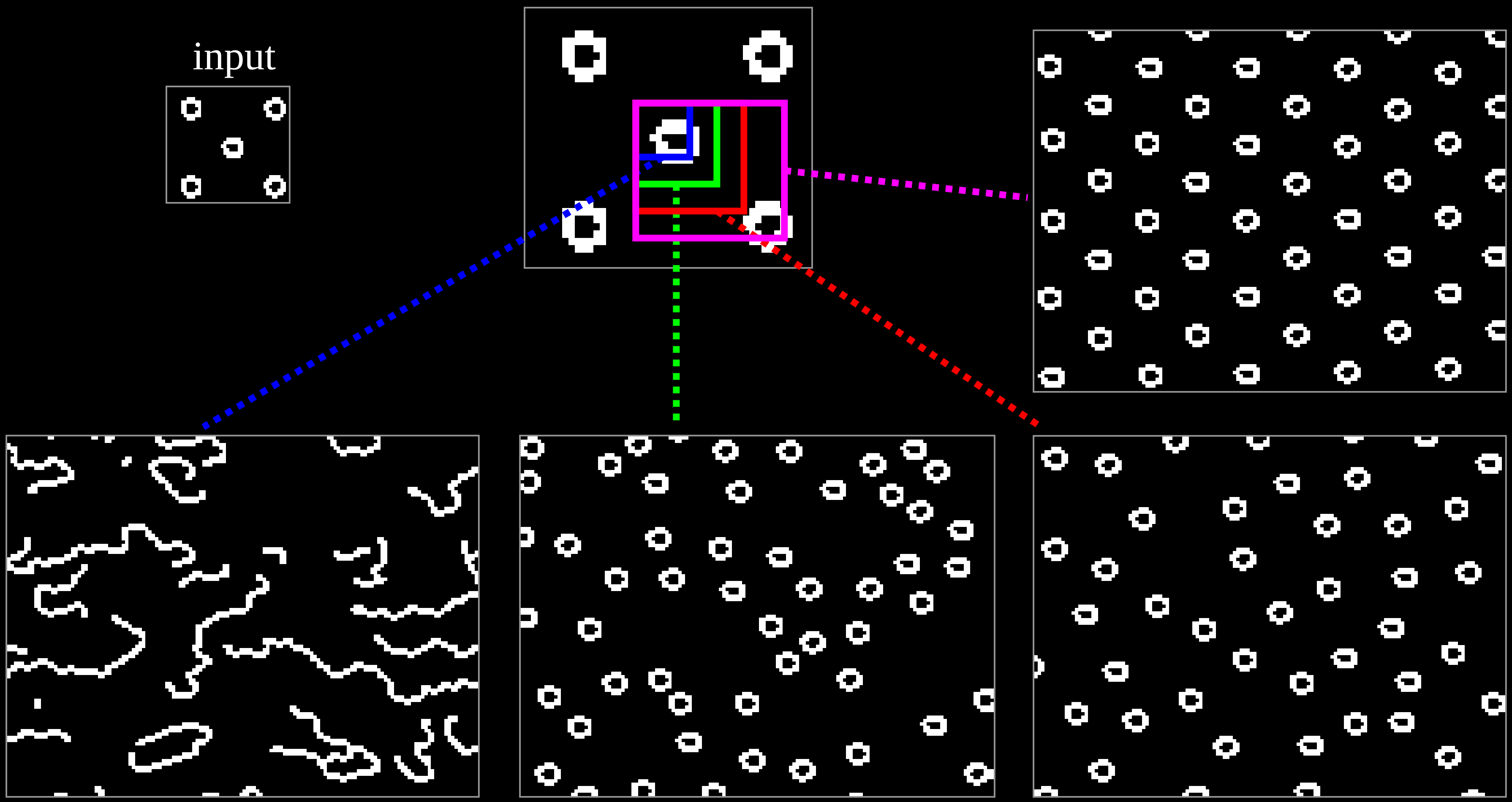


- Assuming Markov property, compute P(**p**|N(**p**))
  - Building explicit probability tables infeasible
  - Instead, we *search the input image* for all similar neighborhoods — that's our pdf for **p**
  - To sample from this pdf, just pick one match at random

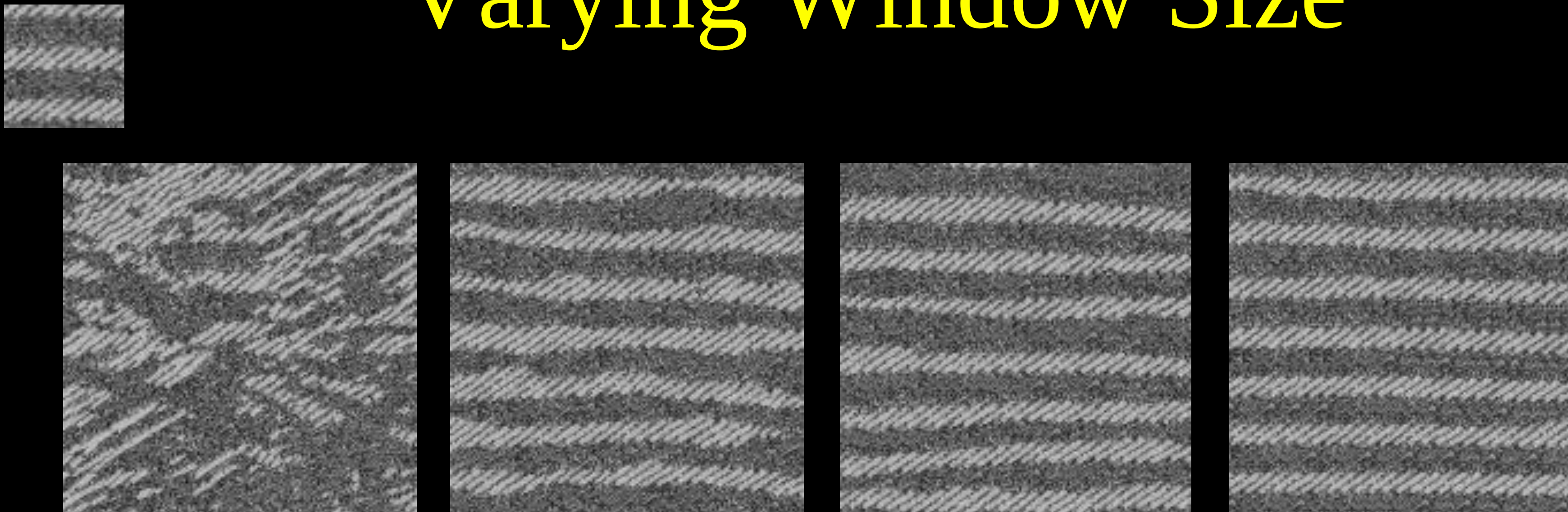
Some Details

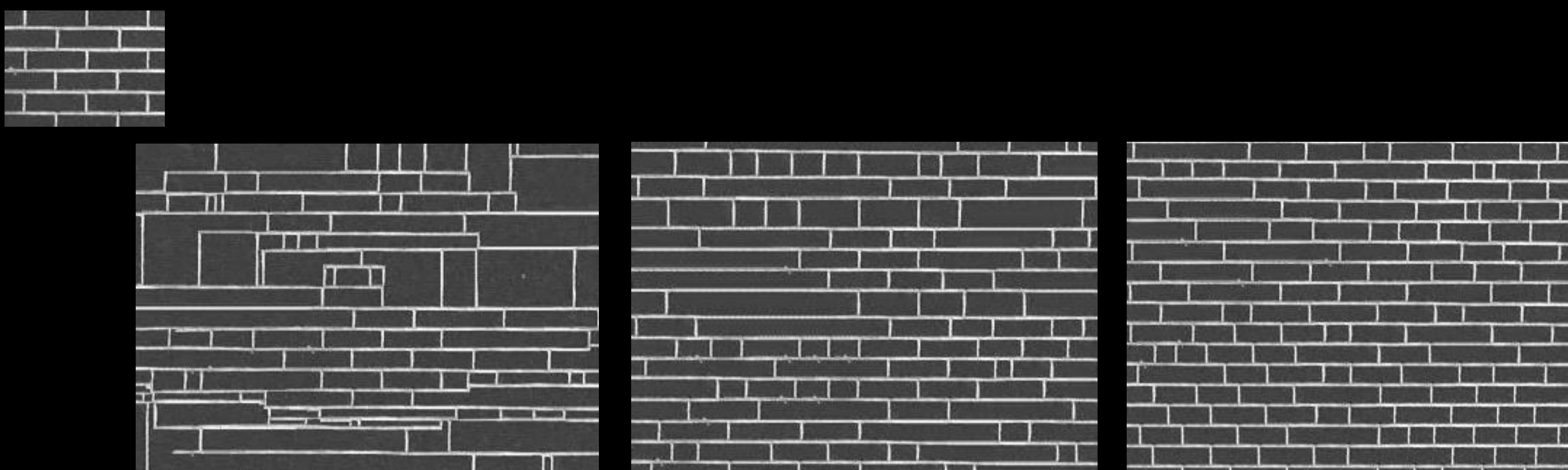
- Growing is in "onion skin" order
  - Within each "layer", pixels with most neighbors are synthesized first
  - If no close match can be found, the pixel is not synthesized until the end
- Using *Gaussian-weighted* SSD is very important
  - to make sure the new pixel agrees with its closest neighbors
  - Approximates reduction to a smaller neighborhood window if data is too sparse

Neighborhood Window



Varying Window Size

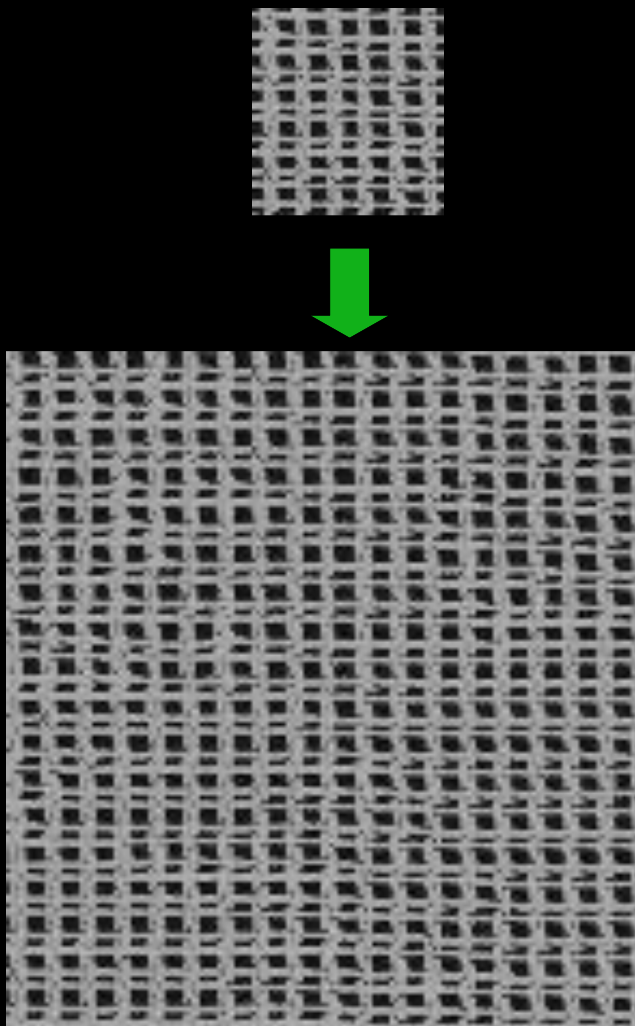




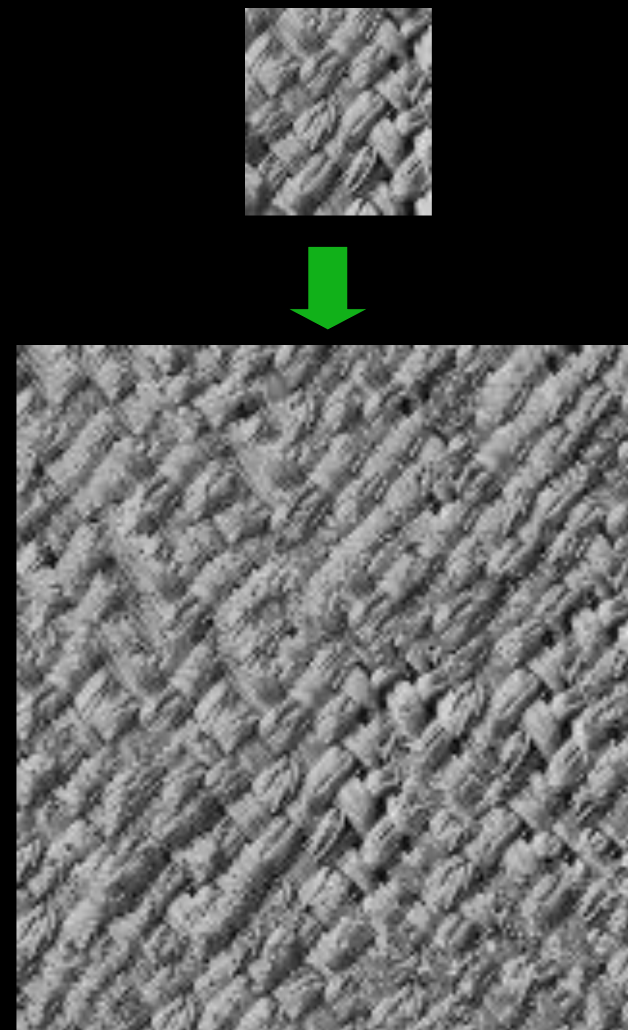
Increasing window size

Synthesis Results

french canvas

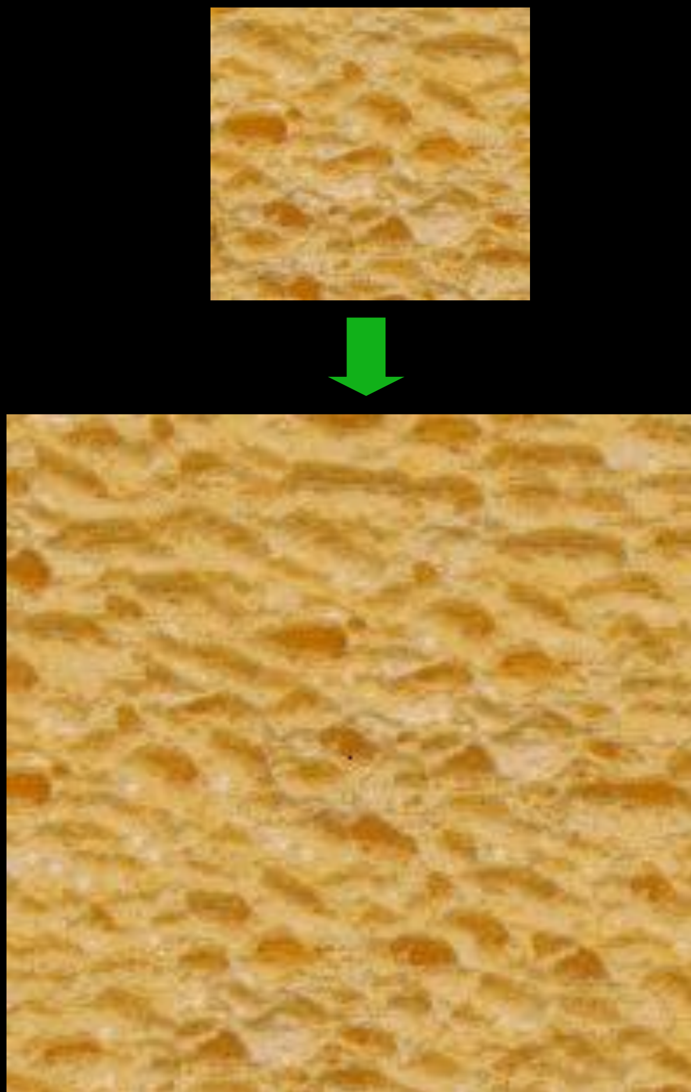


rafia weave

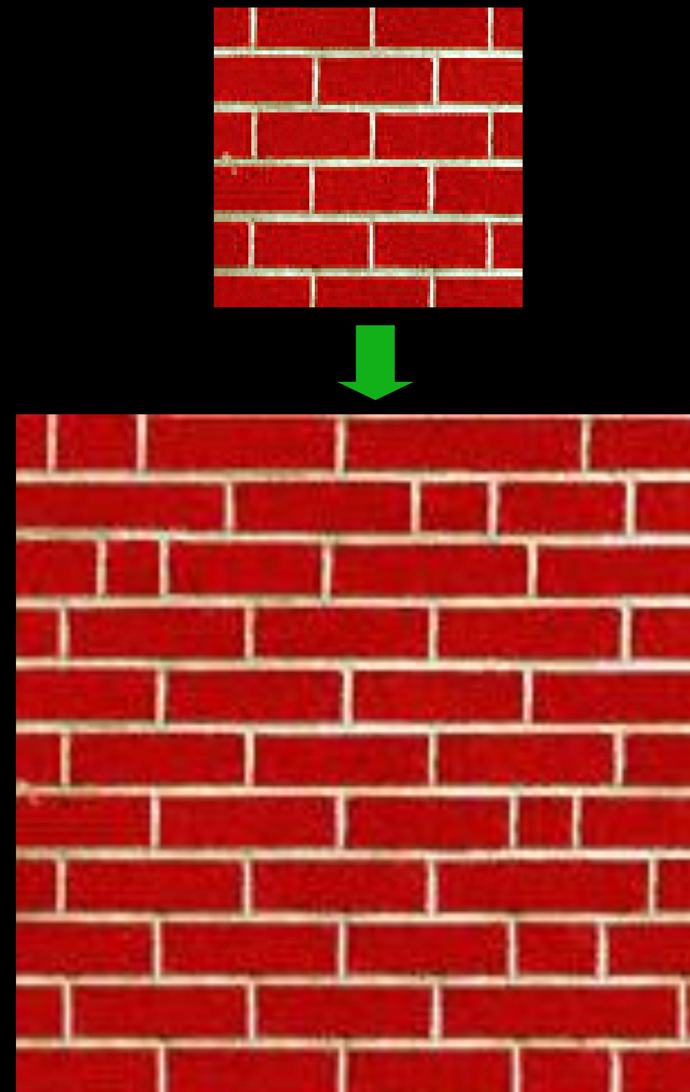


More Results

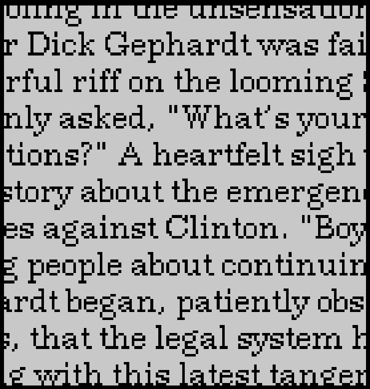
white bread



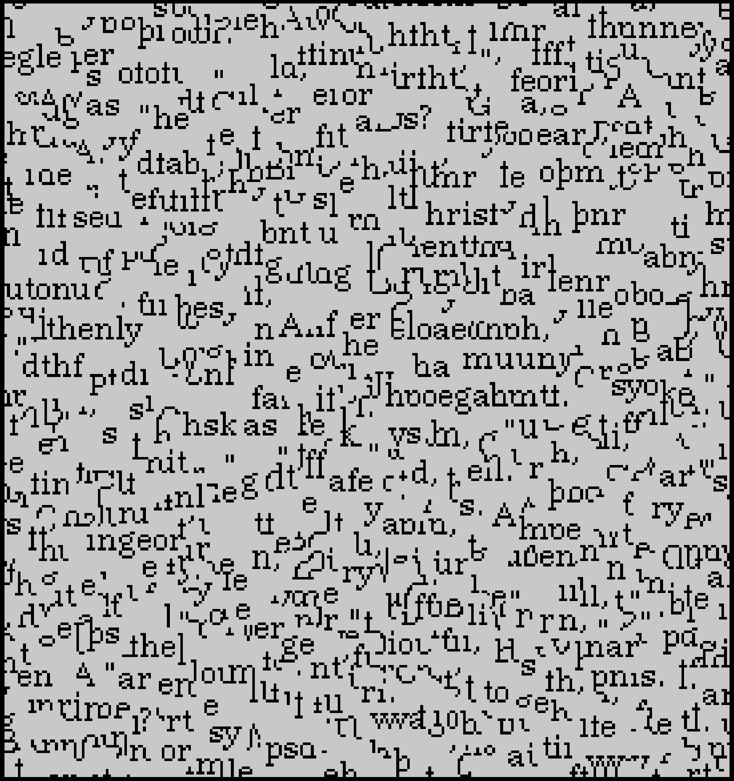
brick wall

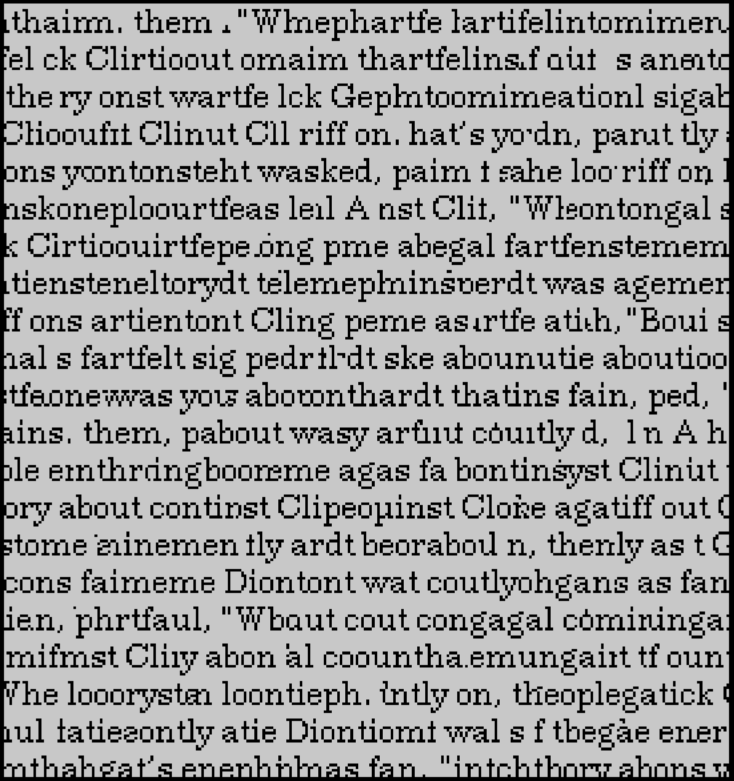


Homage to Shannon



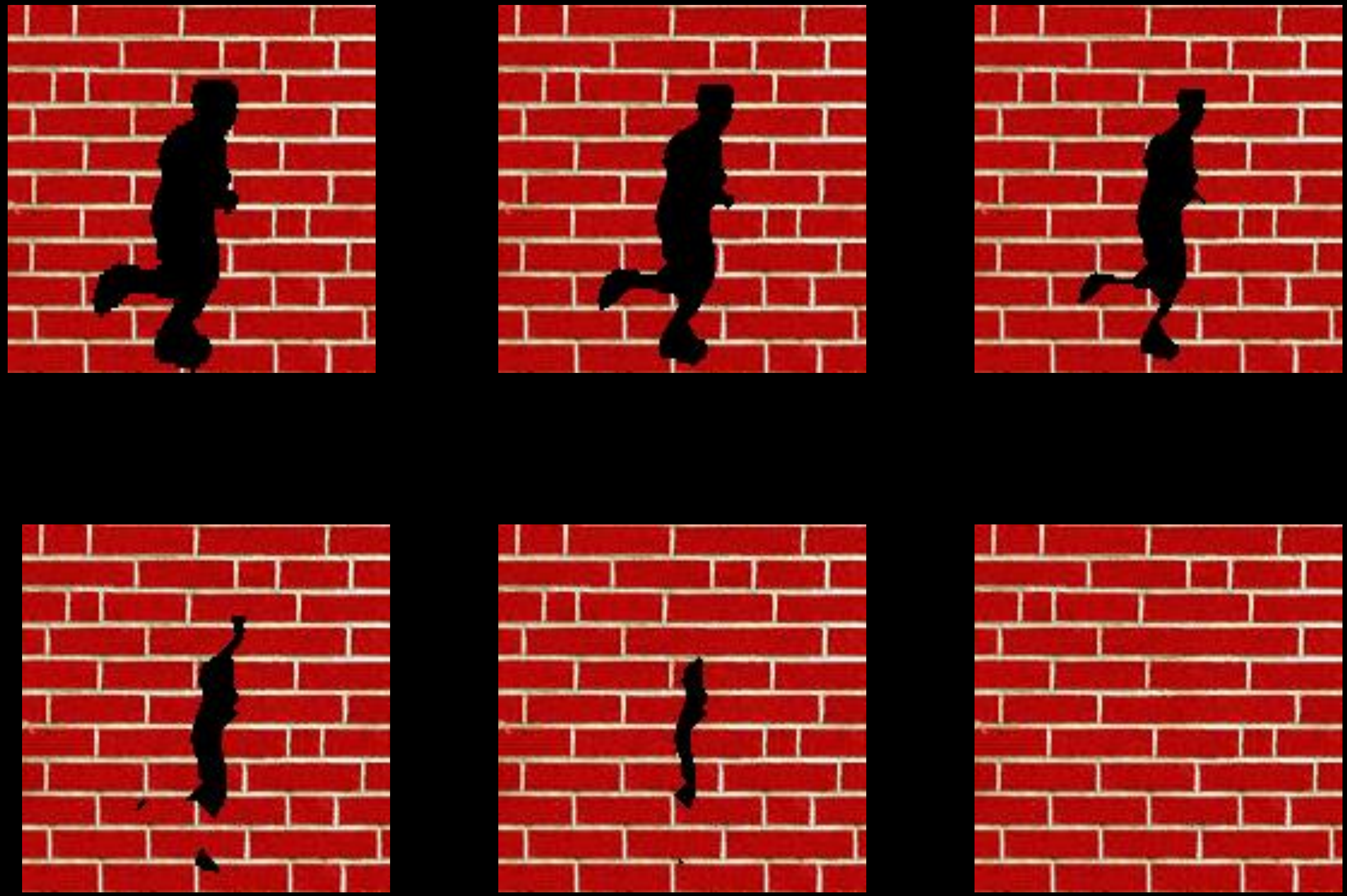
r Dick Gephardt was fai
rful riff on the looming
nly asked, "What's your
tions?" A heartfelt sigh
story about the emergen
es against Clinton. "Boy
g people about continuin
ardt began, patiently obs
, that the legal system h
g with this latest tangen



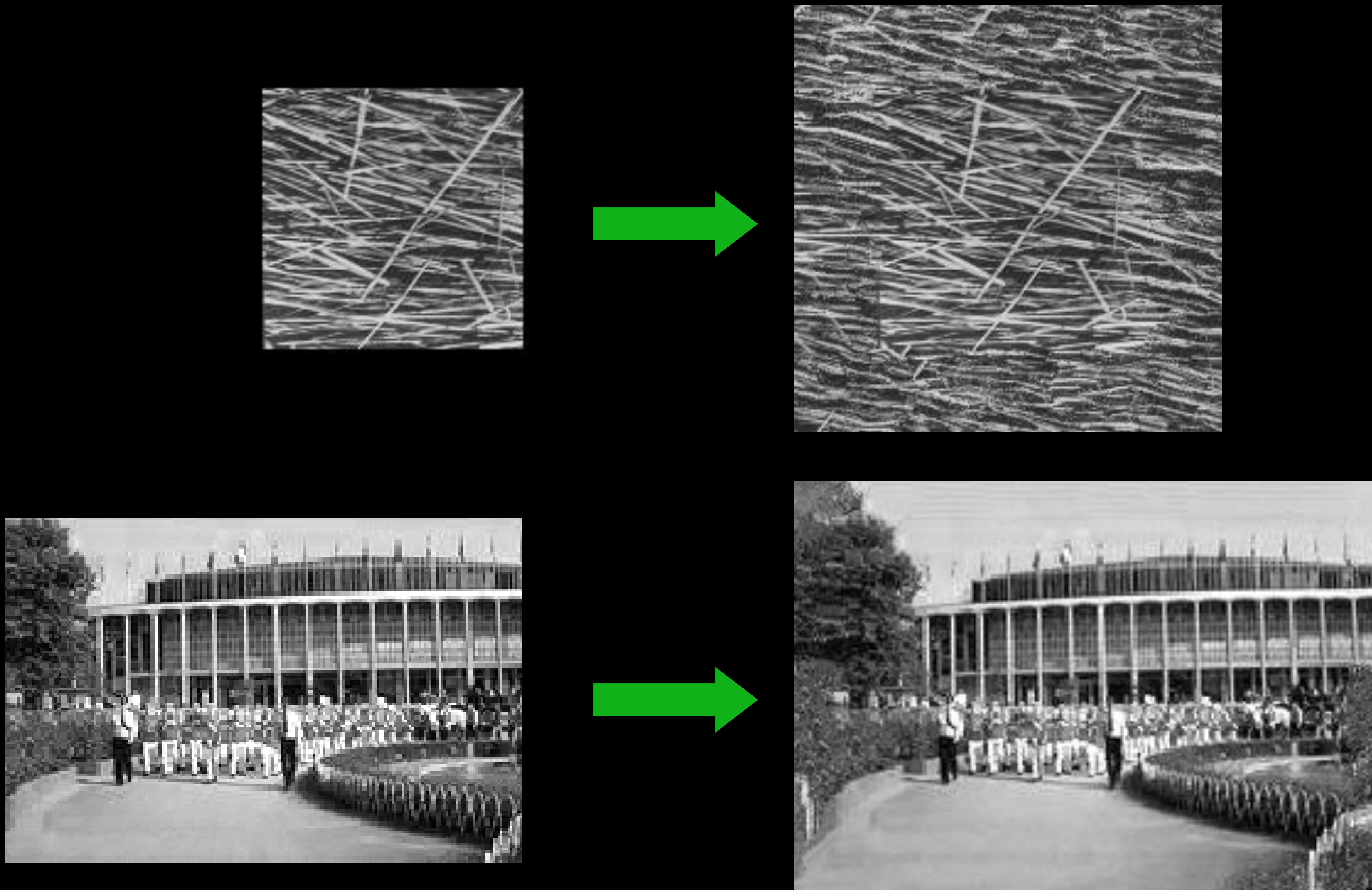


thaim. them ."Whephartfe lartifelintomimen
fel ck Clirtioout omaim thartfelins.f out s anetc
the ry onst wartfe lck Gephtoomimeationl sigab
Clioou fit Clinut Cll riff on. hat's yordn, parut tly
ons yontonsteht wasked, paim t sahe looriff on
nskoneploourtfeas leil A nst Clit, "Wleontongal s
k Cirtioouirtfepe.ong pme abegal fartfenstemem
tiensteneltorydt telemephinsperdt was agemen
ff ons artientont Cling peme as.rtfe atich, "Boui s
hal s fartfelt sig pedrthdt ske abounutie aboutioo
tfeonewwas yous aboronthardt thatins fain, ped, '
ains. them, pabout wasy arfuit coutly d, l n A h
ble emthrdngboomeme agas fa bontinsyst Clinut
ory about continst Clipeopinst Cloke agatiff out C
stome ainemen tly ardt beorabou n, thenly as t G
cons faimeme Diontont wat coutlyohgans as fan
ien, phrtfaul, "Wbout cout congagal comininga
mifmst Cliny abon al coountha.emungairt tf oun
Whe looorystan loontieph. intly on, theoplegatick
hul tatieeontly atie Diontiomt wal s f tbegae ener

Hole Filling



Extrapolation



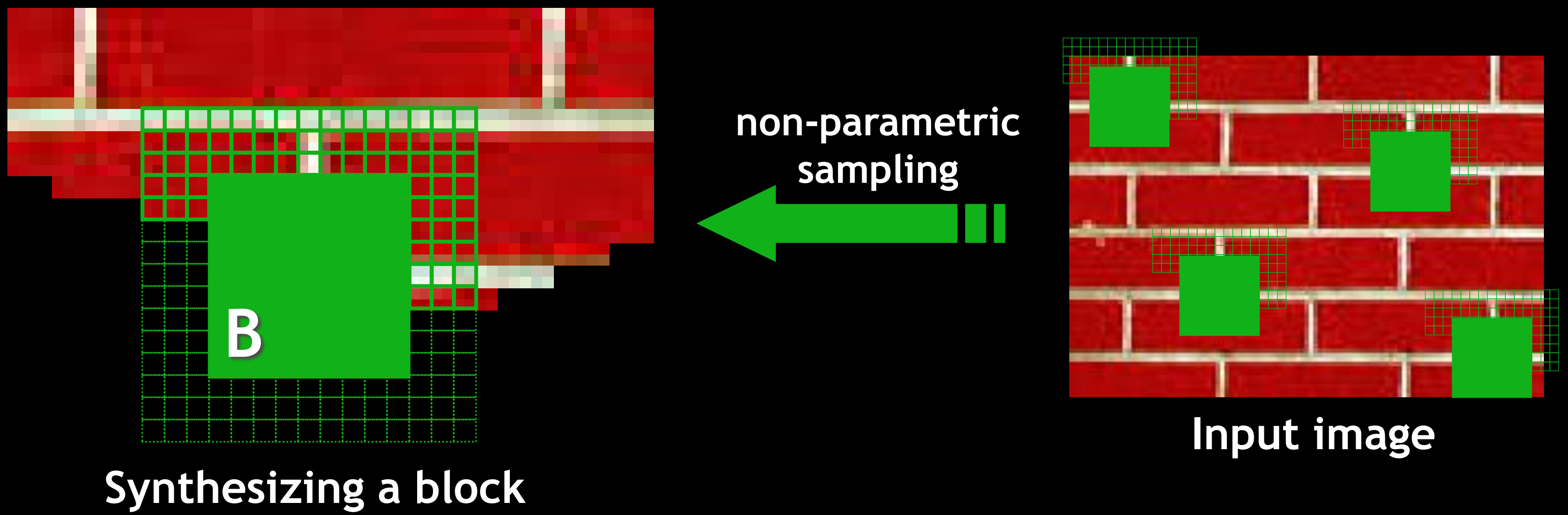
Summary

- The Efros & Leung algorithm
  - Very simple
  - Surprisingly good results
  - Synthesis is easier than analysis!
  - …but very slow

Overview

- Texture synthesis
- Quilting
- Image Analogies
- Super-resolution
- Scene completion

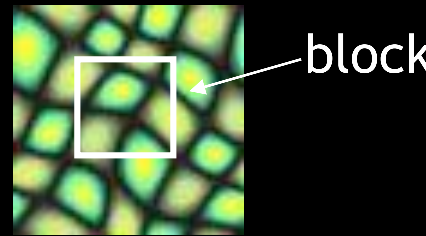
Image Quilting [Efros & Freeman]



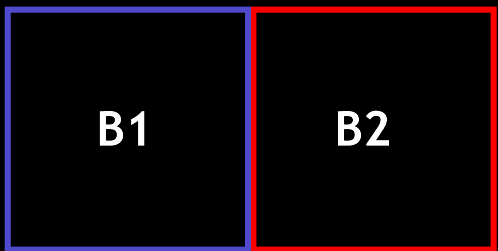
- <u>Observation:</u> neighbor pixels are highly correlated

<u>Idea:</u> unit of synthesis = block

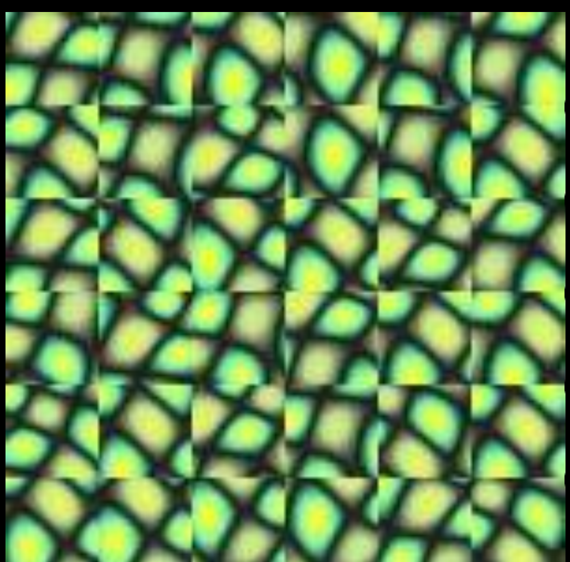
- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once
- Not the same as multi-scale!

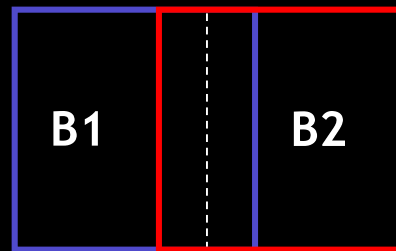


Input texture

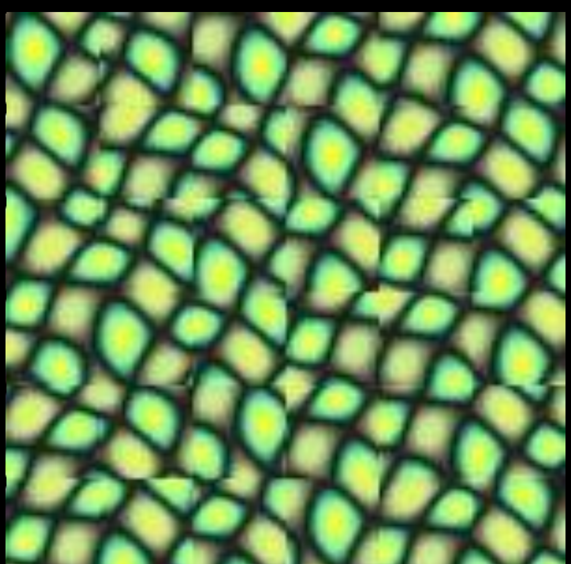


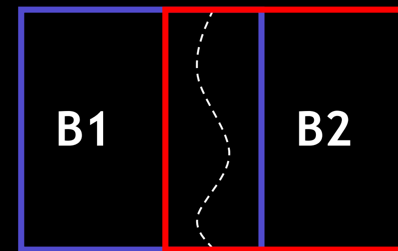
Random placement of blocks



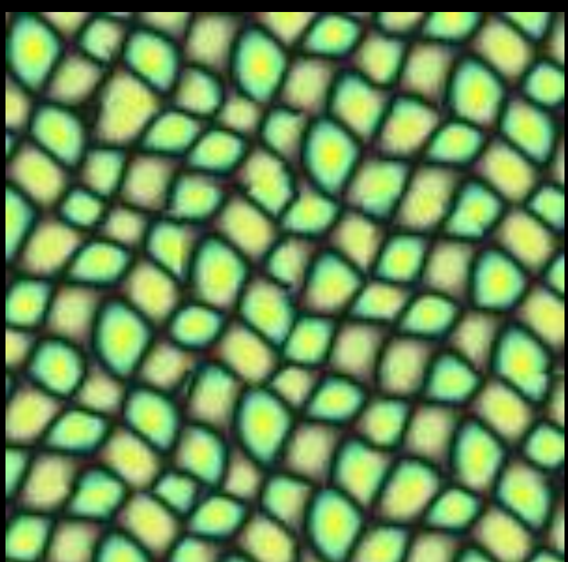


Neighboring blocks constrained by overlap

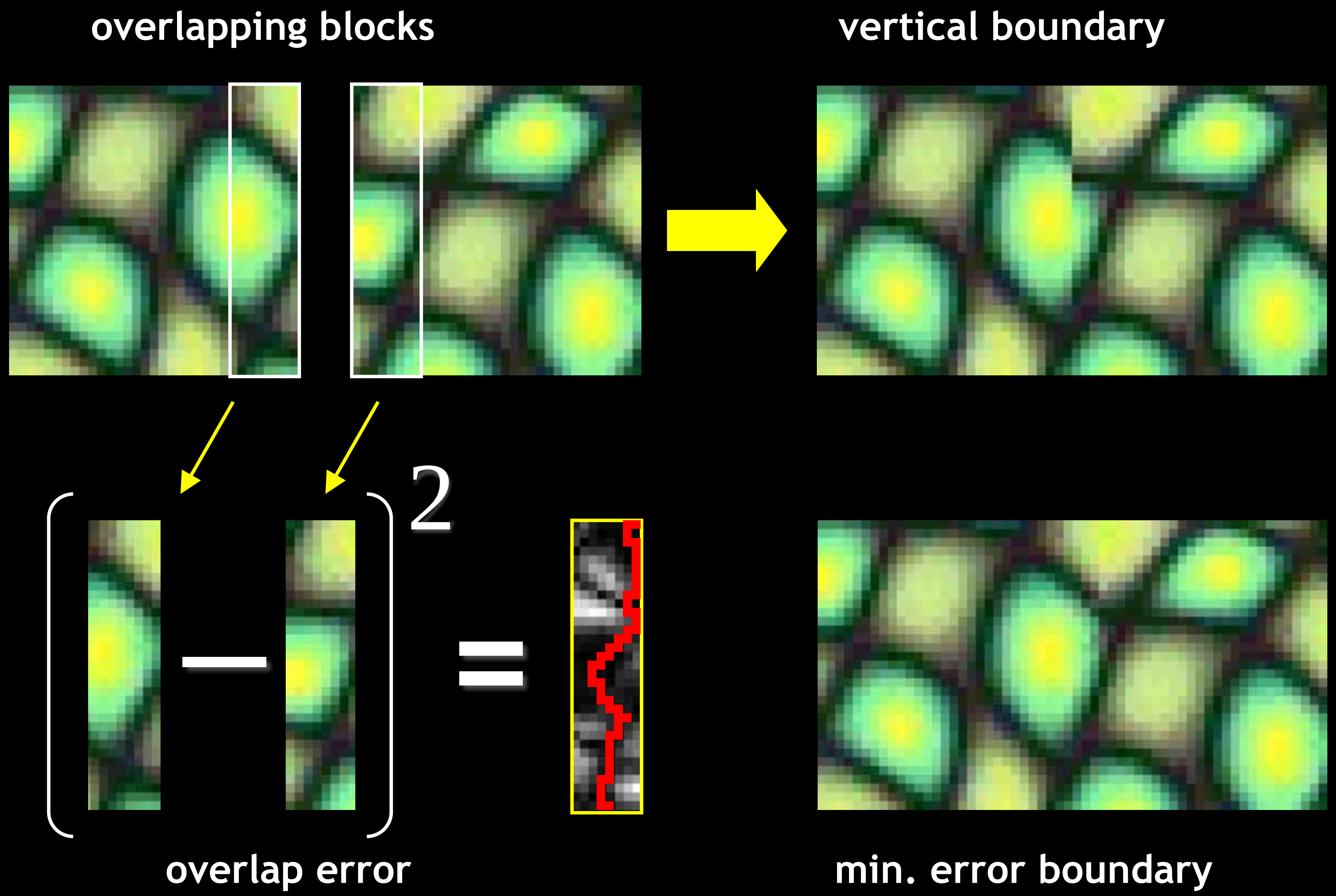




Minimal error boundary cut

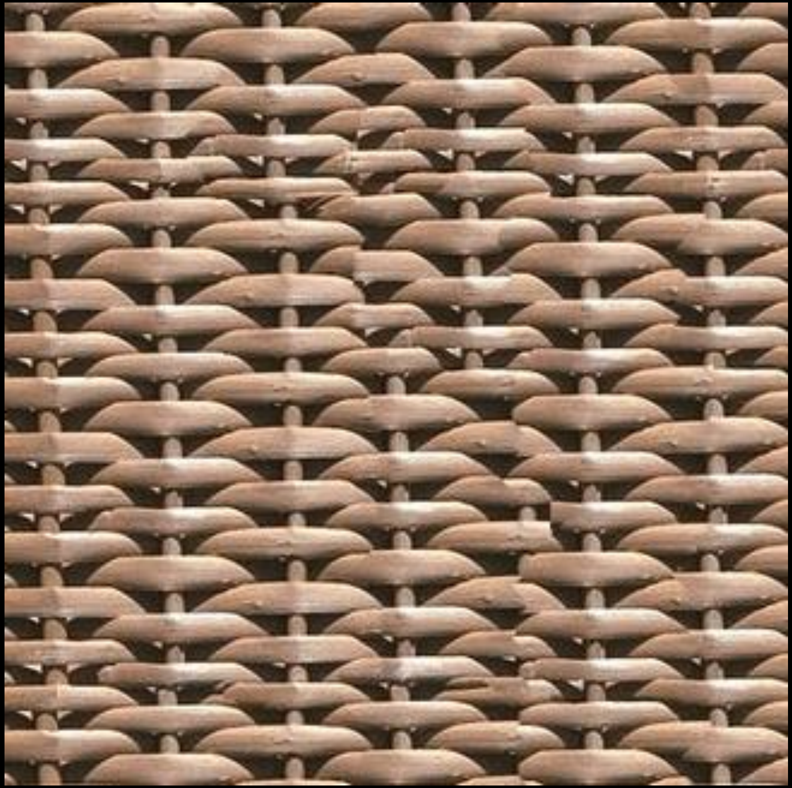


Minimal error boundary

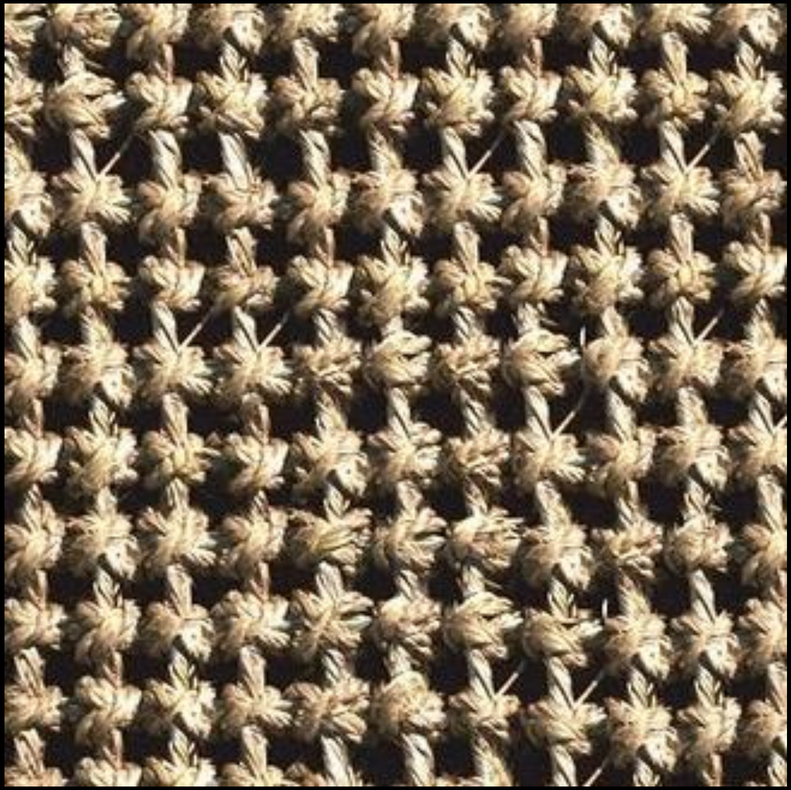


Our Philosophy

- The “Corrupt Professor’s Algorithm”:
  - Plagiarize as much of the source image as you can
  - Then try to cover up the evidence
- Rationale:
  - Texture blocks are by definition correct samples of texture so problem only connecting them together

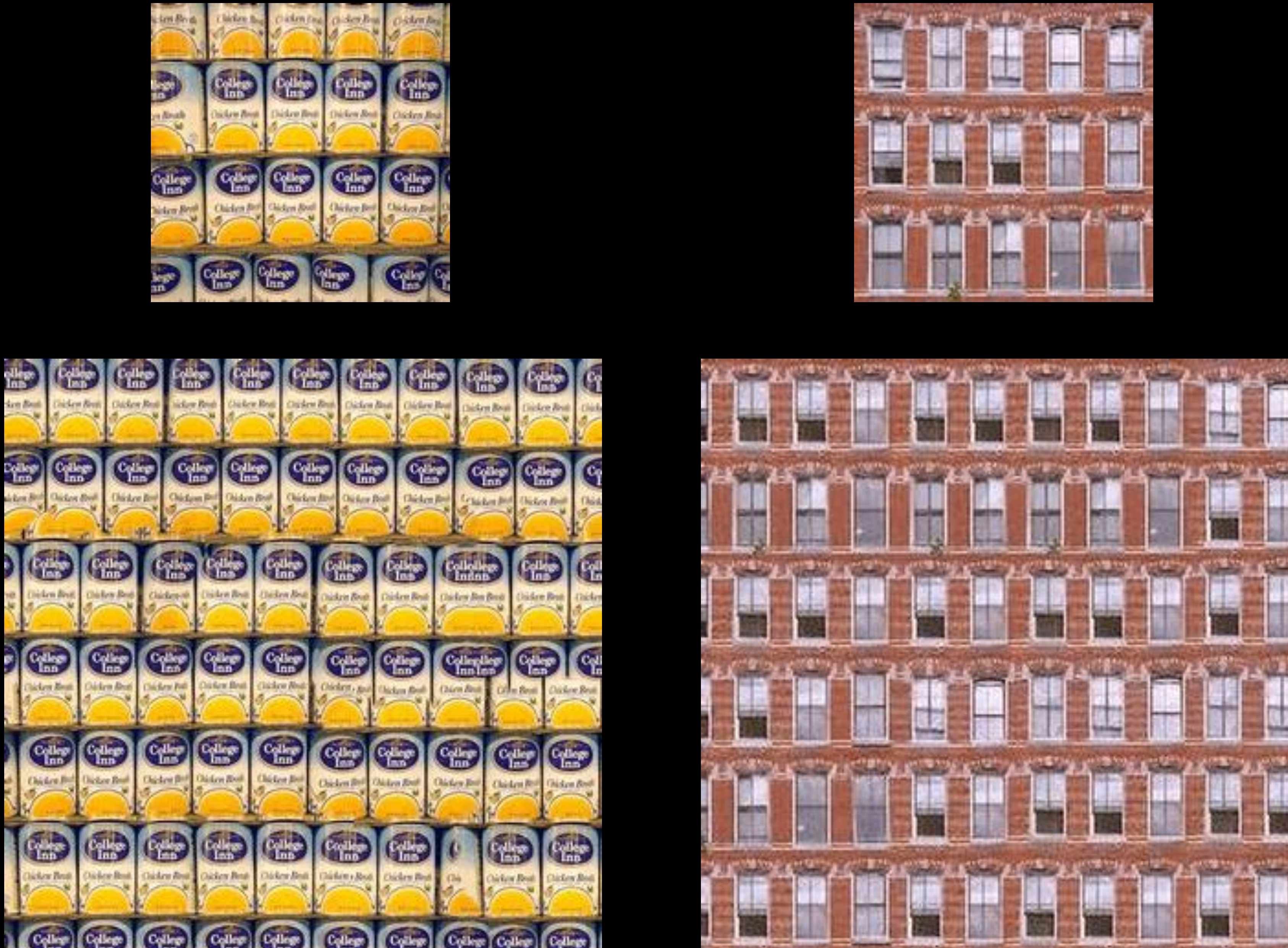




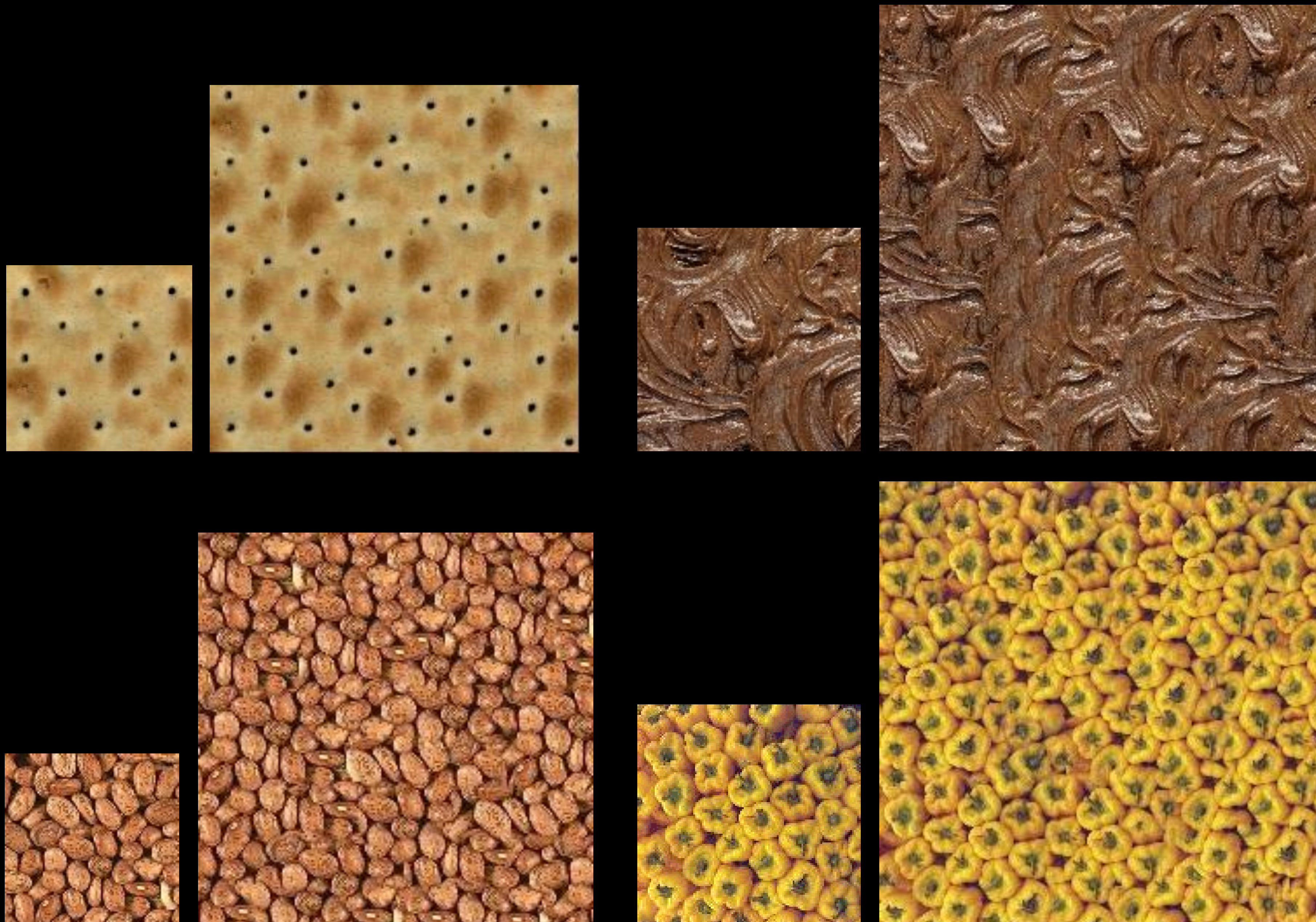




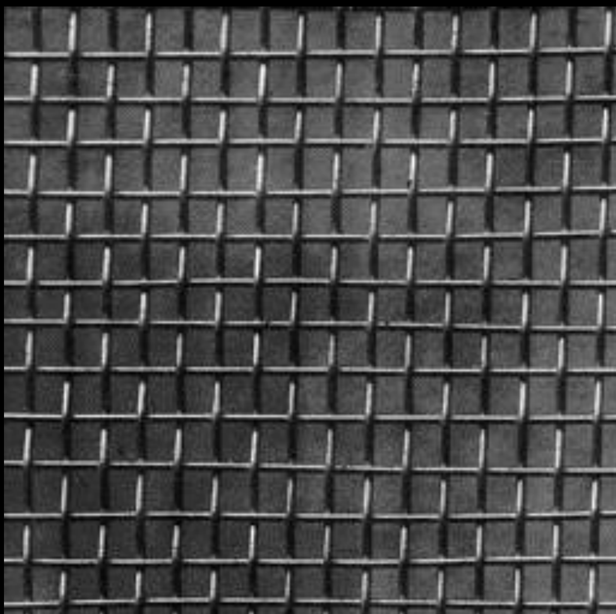



College Inn

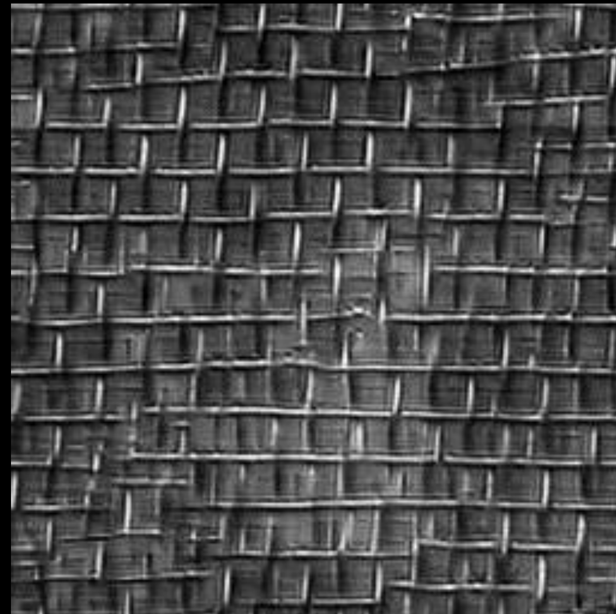


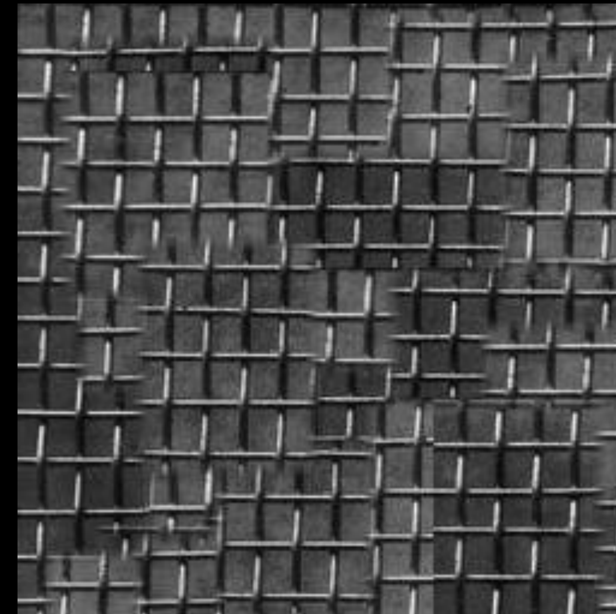
Failures
(Chernobyl Harvest)



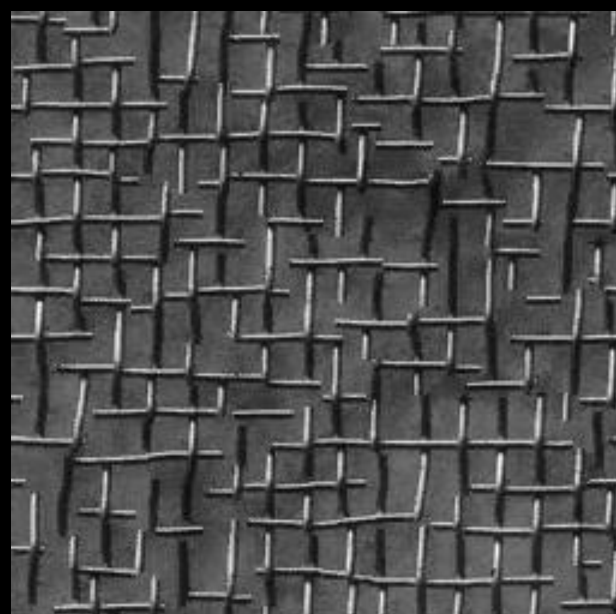
input image



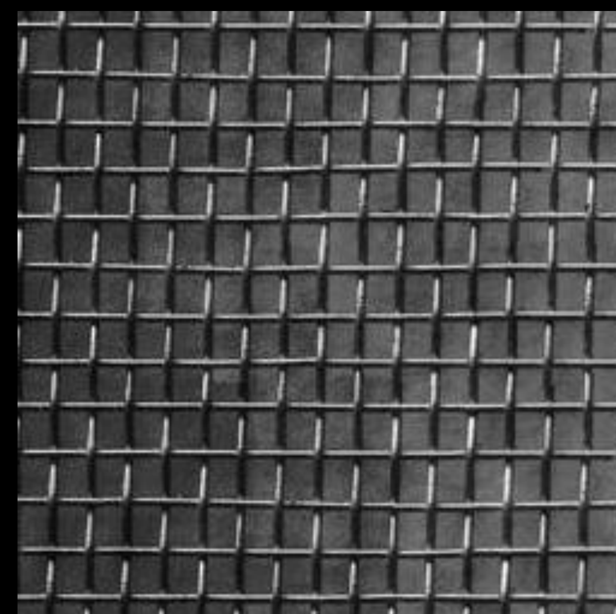
Portilla & Simoncelli



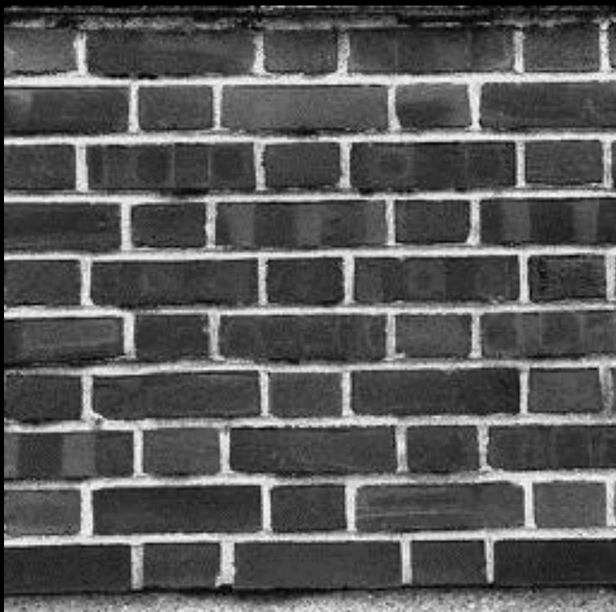
Xu, Guo & Shum



Wei & Levoy

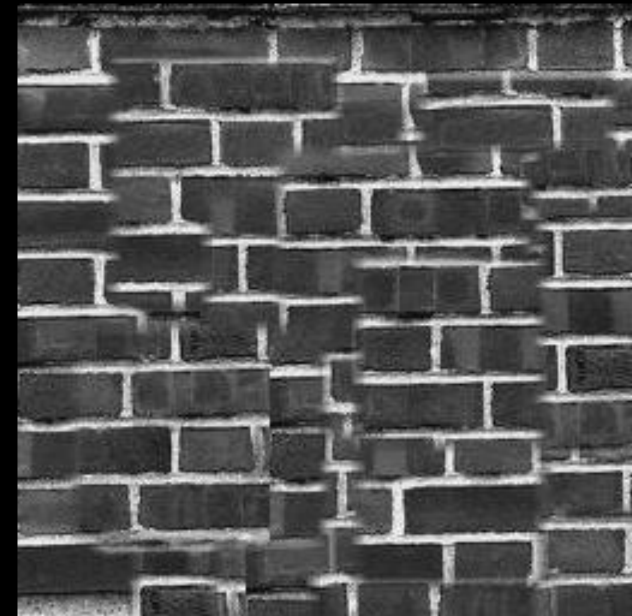


Our algorithm

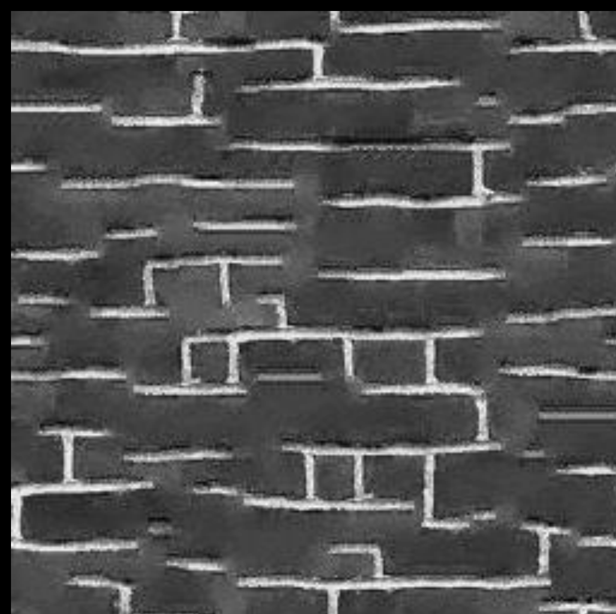


input image

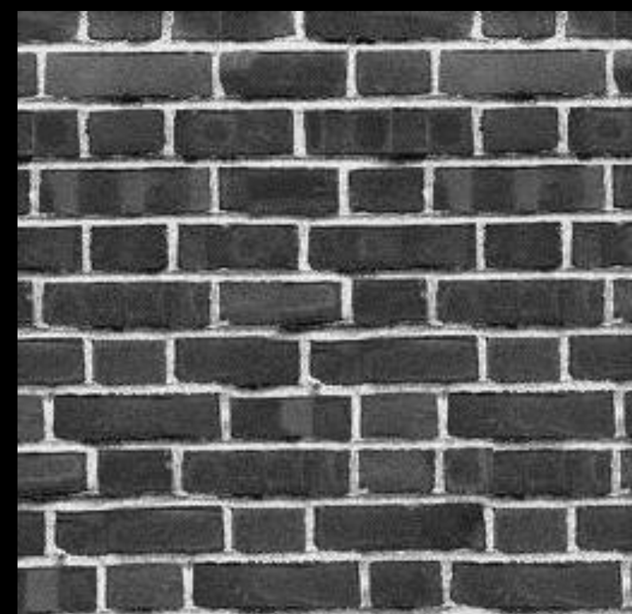
Portilla & Simoncelli



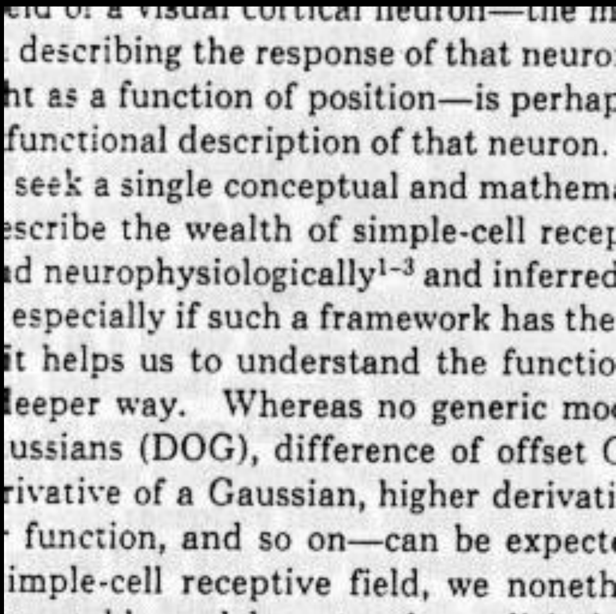
Xu, Guo & Shum



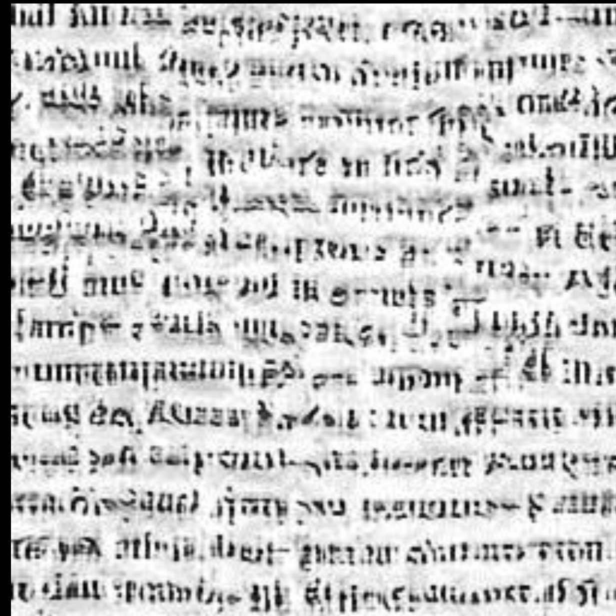
Wei & Levoy



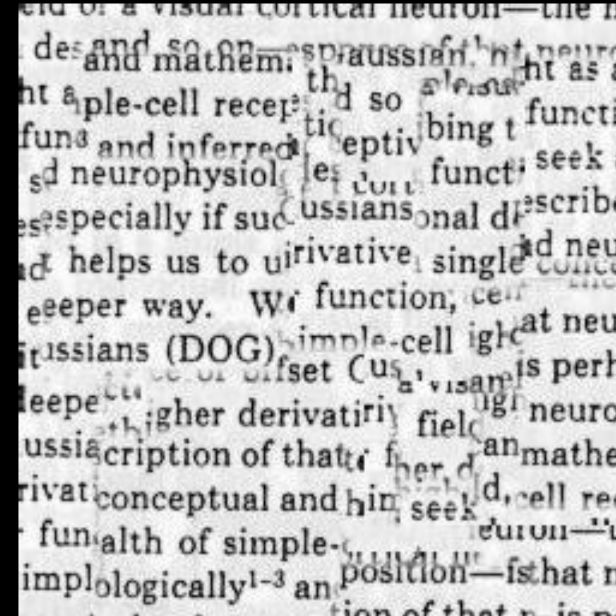
Our algorithm



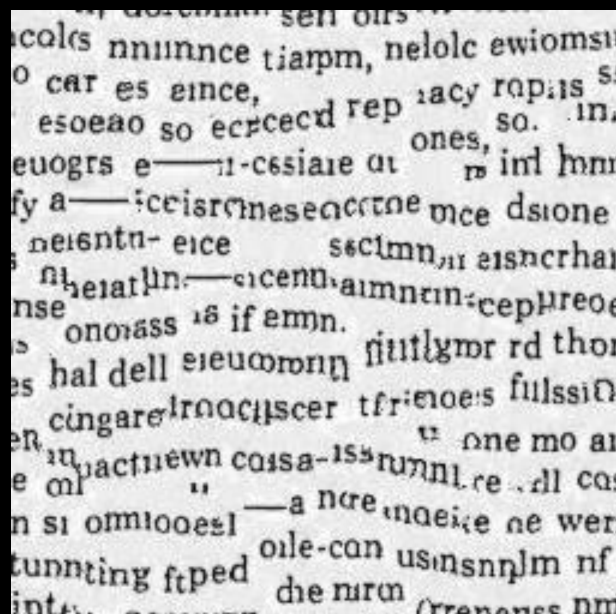
input image



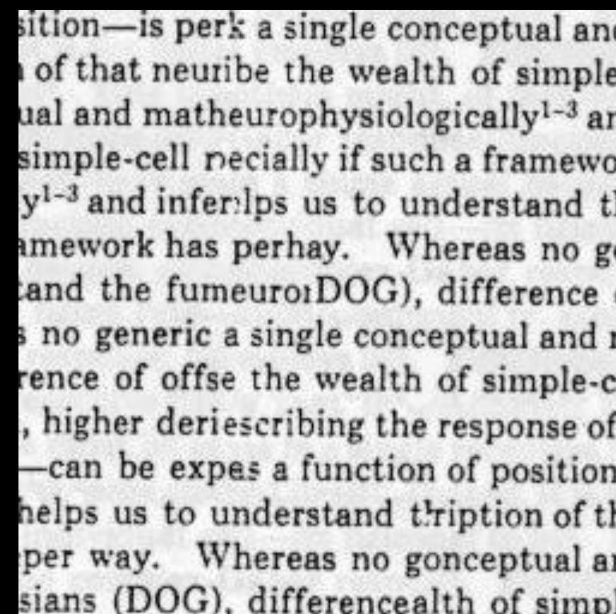
Portilla & Simoncelli



Xu, Guo & Shum



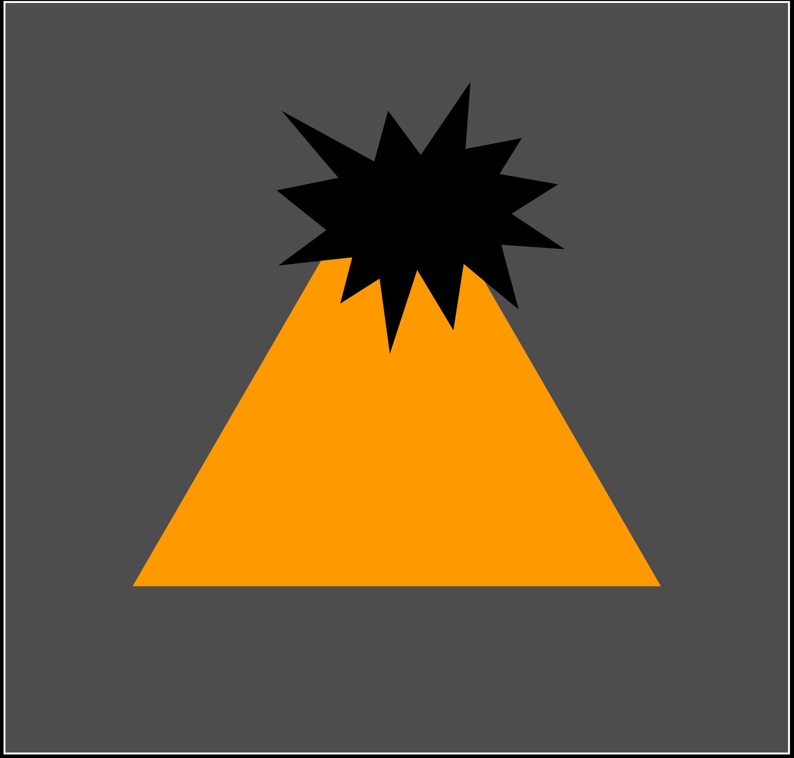
Wei & Levoy



Our algorithm

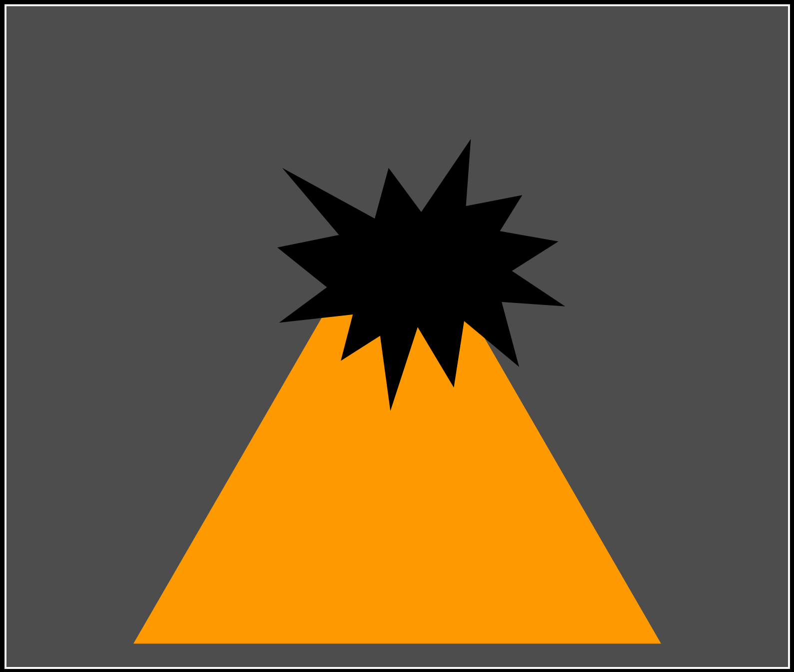
Political Texture Synthesis!

Fill Order



- In what order should we fill the pixels?

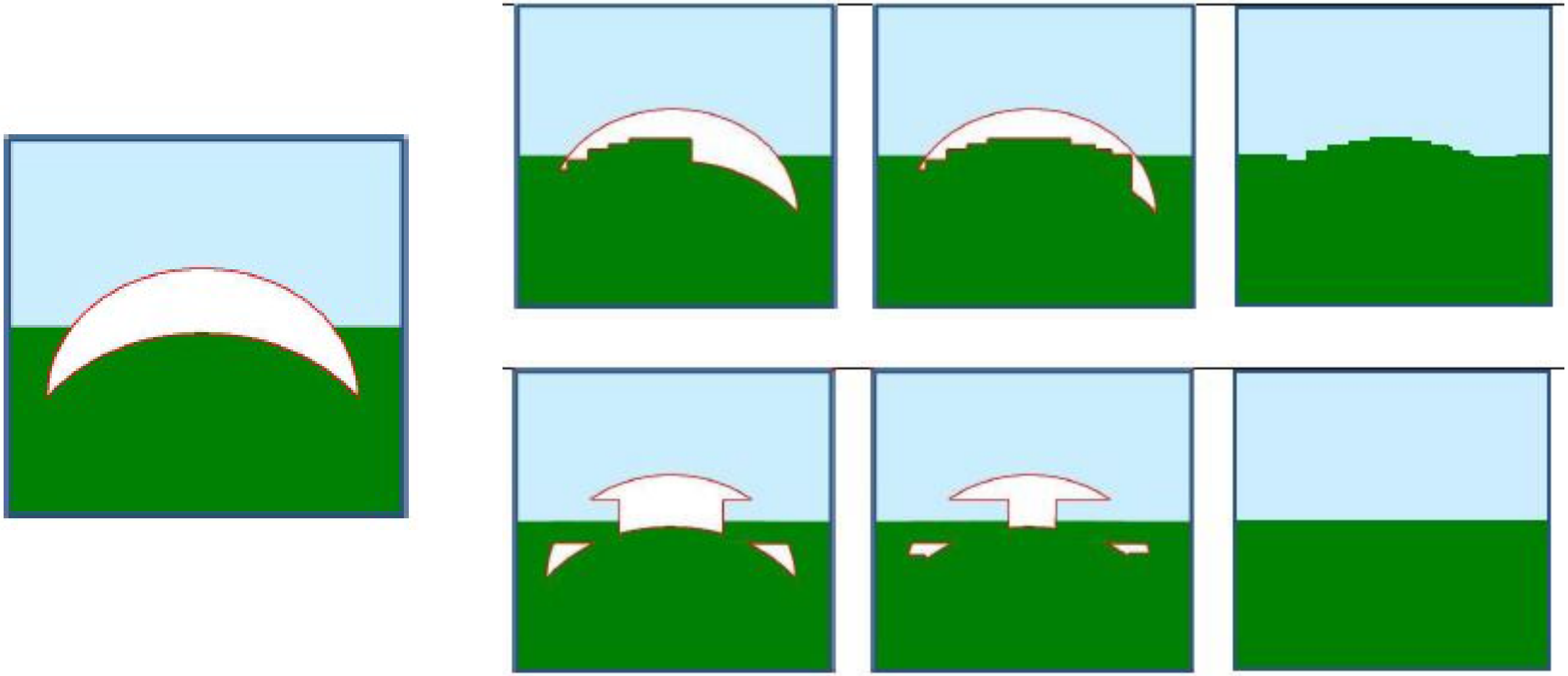
Fill Order



- In what order should we fill the pixels?
  - choose pixels that have more neighbors filled
  - choose pixels that are continuations of lines/curves/edges

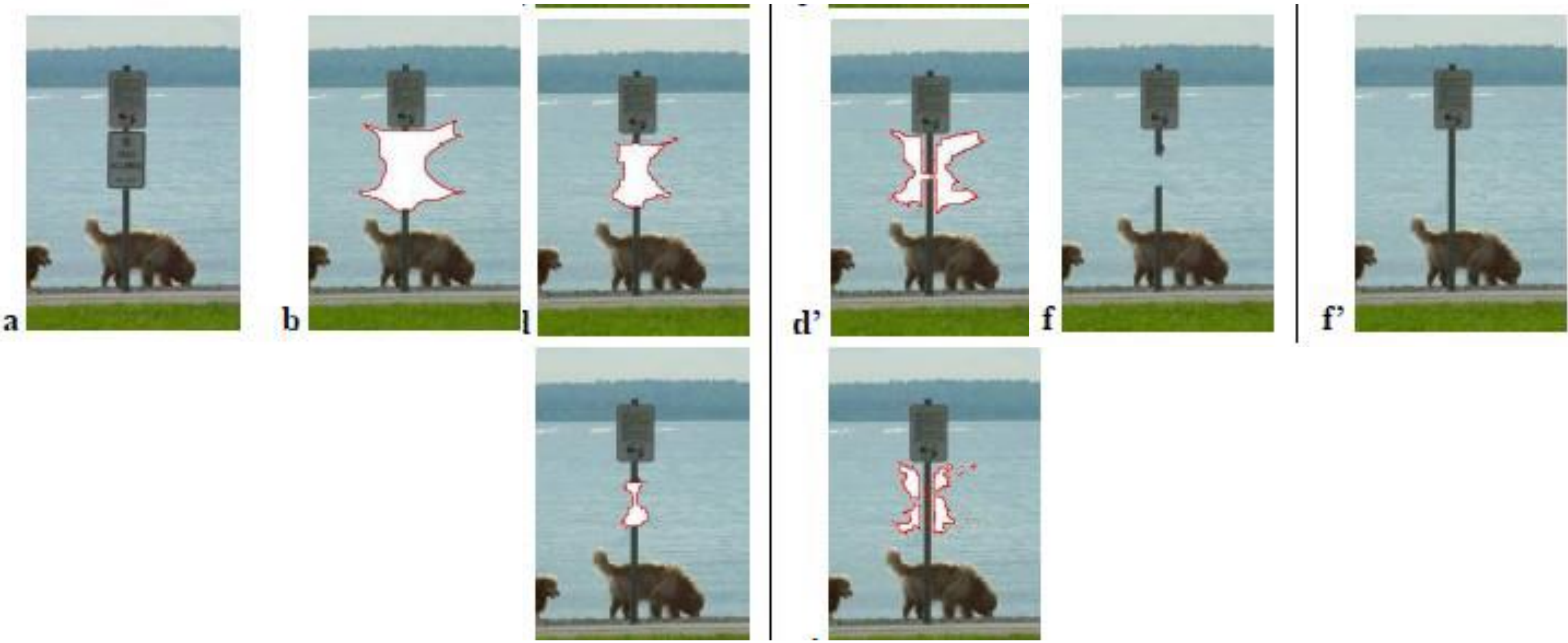
Criminisi, Perez, and Toyama. "Object Removal by Exemplar-based Inpainting," Proc. CVPR, 2003.

Image quilting – order problem



Region Filling and Object Removal by
Exemplar-Based Image Inpainting, Criminisi et al. '04

Image quilting – order problem



Region Filling and Object Removal by
Exemplar-Based Image Inpainting, Criminisi et al. '04

Patch-based image analysis

Super resolution from single image
http://www.wisdom.weizmann.ac.il/~vision/SingleImageSR.html

The patch transform, Cho et a. '08
http://people.csail.mit.edu/taegsang/patchTransform.html

Space-time video completion, Wexler et al. '04
http://www.wisdom.weizmann.ac.il/~vision/VideoCompletion.html

Seam carving, Avidan & Shamir '07
http://www.faculty.idc.ac.il/arik/site/seam-carve.asp

Image transformations

Image transformations

image filtering: change ***range*** of image

$$g(x) = T(f(x))$$

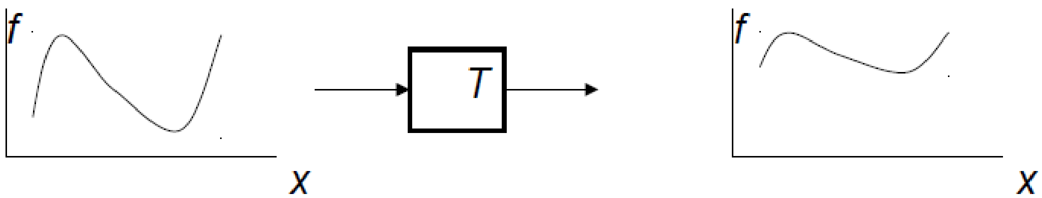


image warping: change ***domain*** of image

$$g(x) = f(T(x))$$

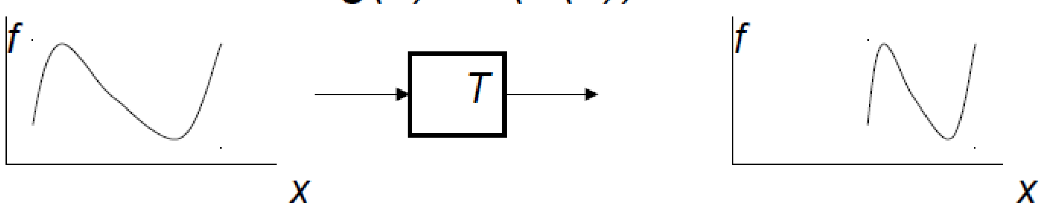


Image transformations

image filtering: change ***range*** of image

$$g(x) = T(f(x))$$

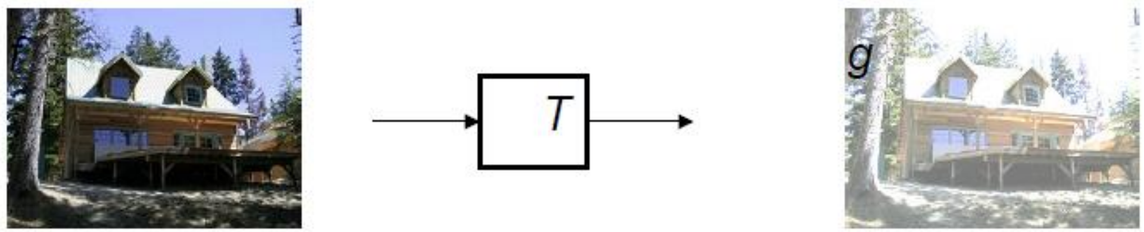
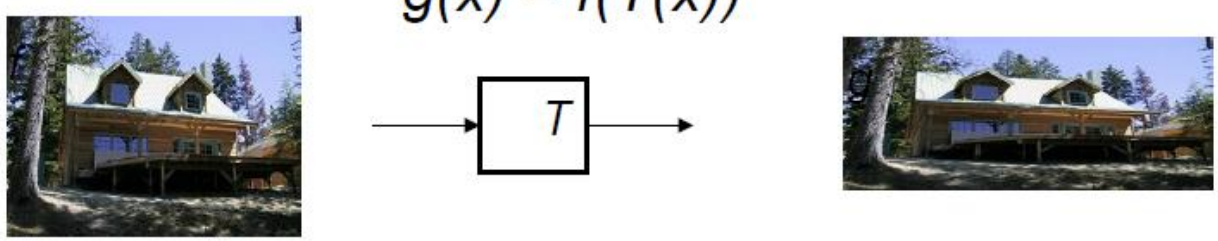
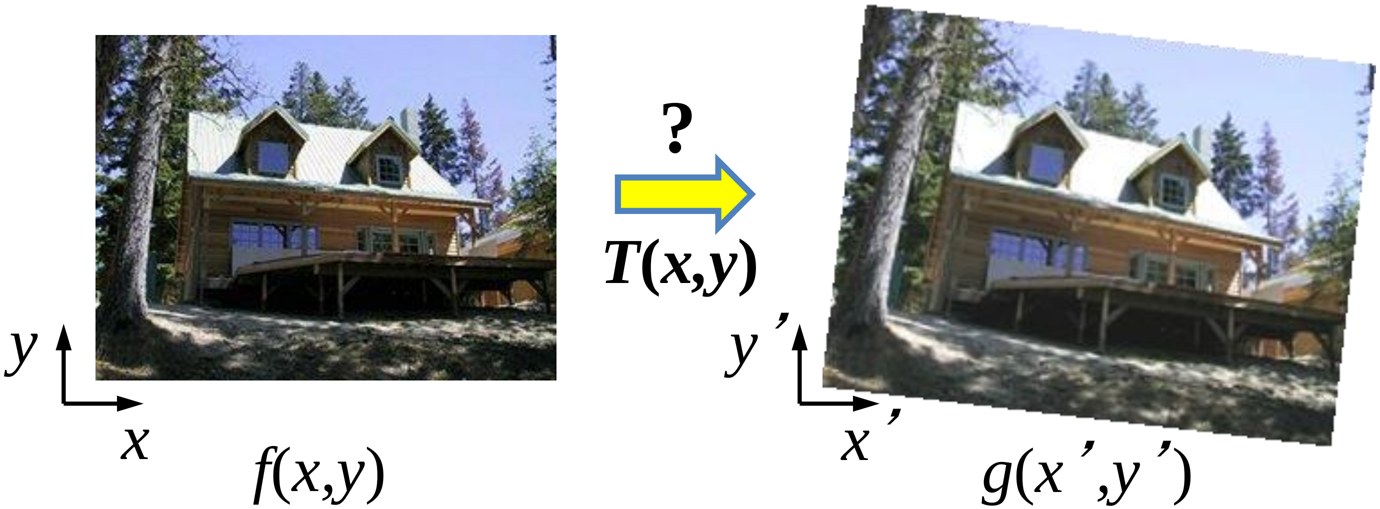


image warping: change ***domain*** of image

$$g(x) = f(T(x))$$

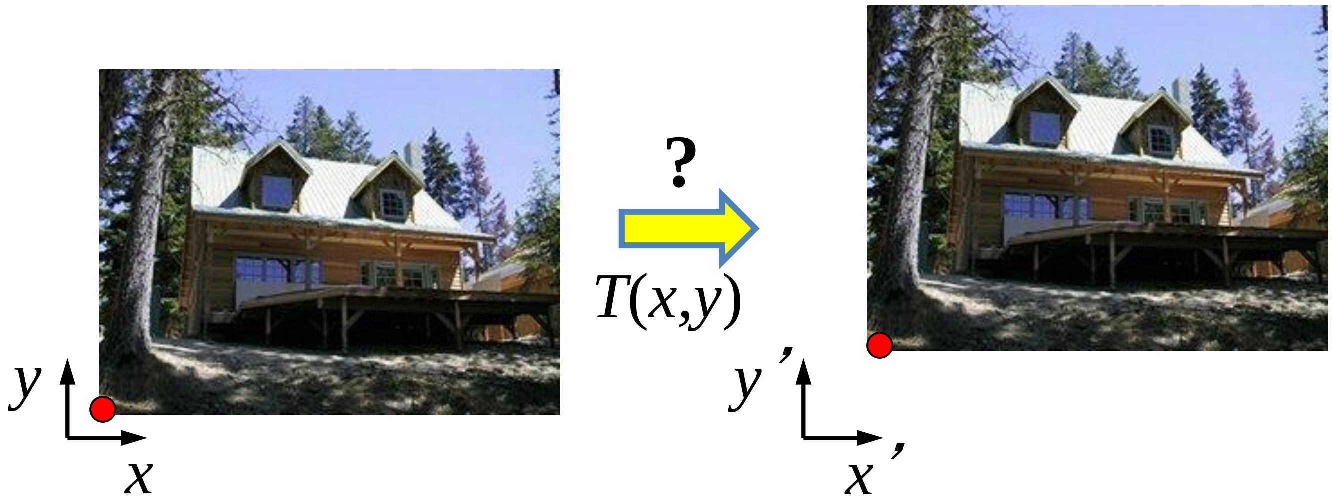


Recovering Transformations



- What if we know *f* and *g* and want to recover the transform T?
  - e.g. better align photographs you've taken
  - willing to let user provide correspondences
    - How many do we need?

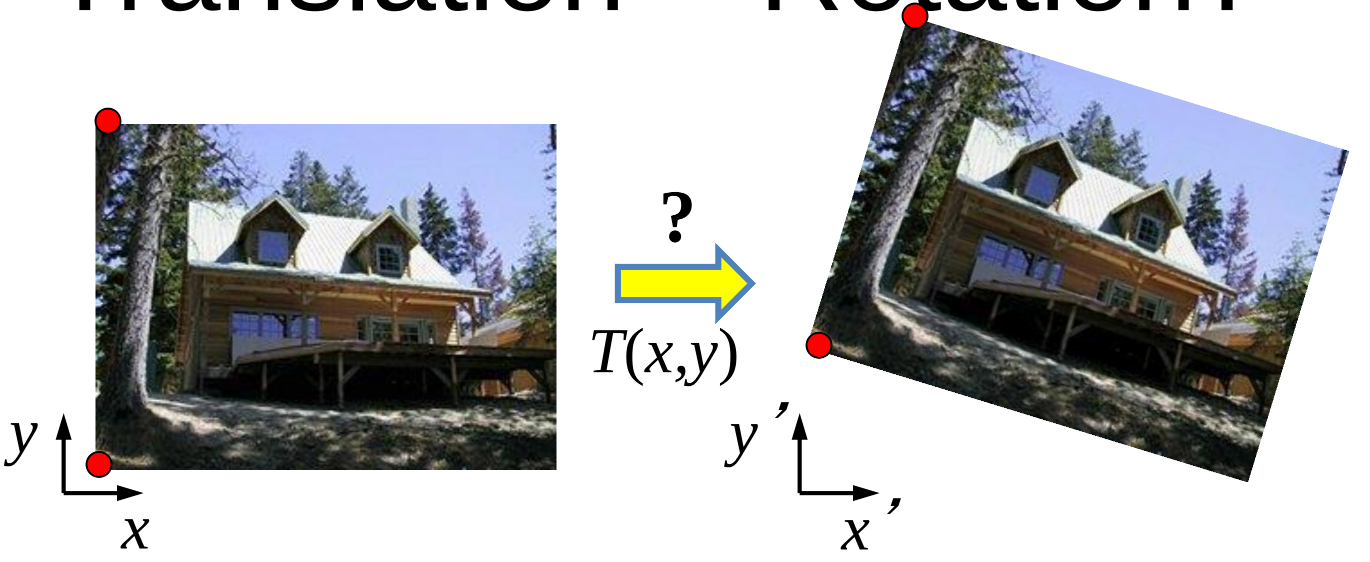
Translation: # correspondences?



- How many correspondences needed for translation?
- How many Degrees of Freedom?
- What is the transformation matrix?

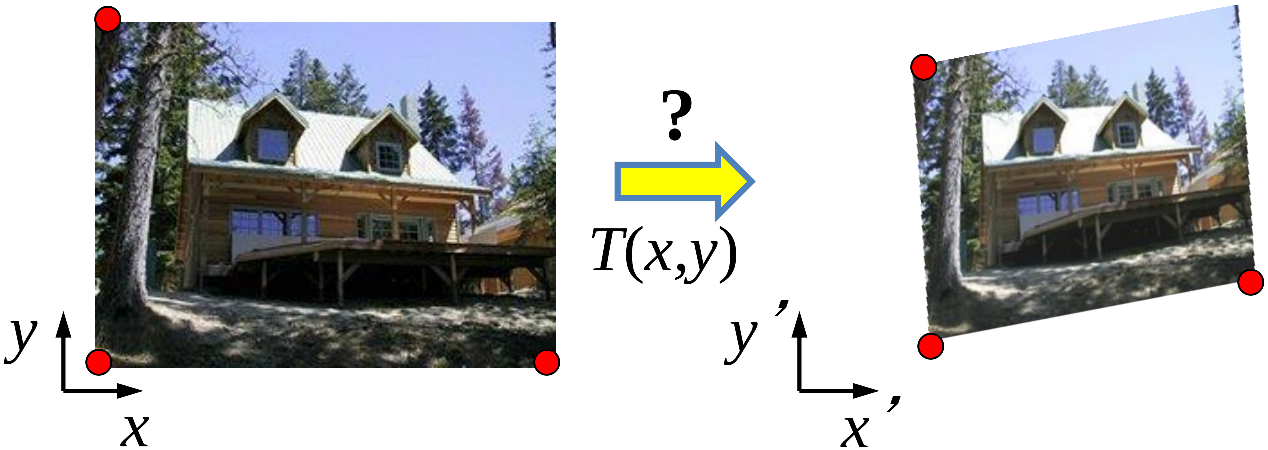
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Translation + Rotation?



- How many correspondences needed for translation+rotation?
- How many DOF?

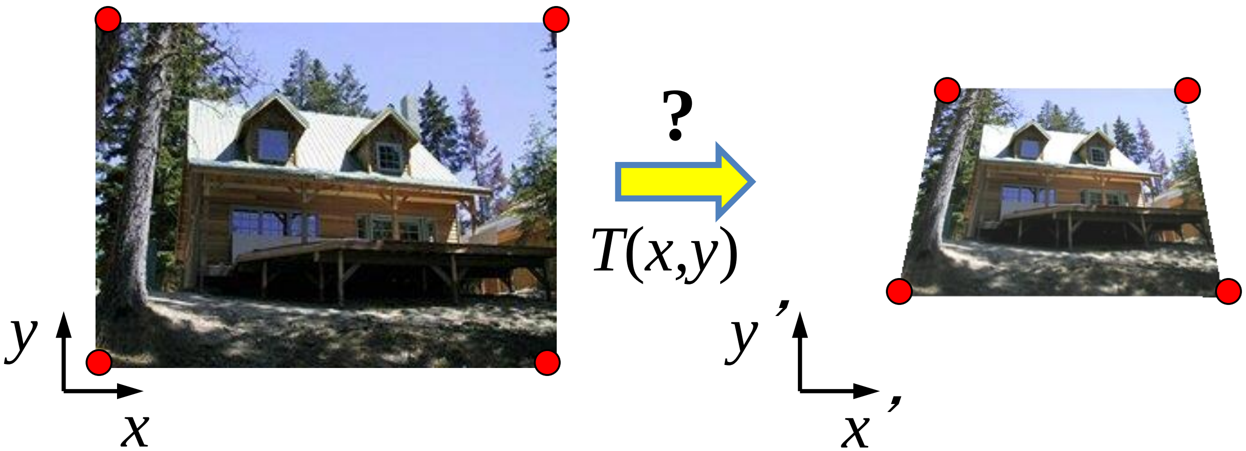
Affine: # correspondences?



- How many correspondences needed for affine transform?
- How many DOF?

$$T(x,y) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

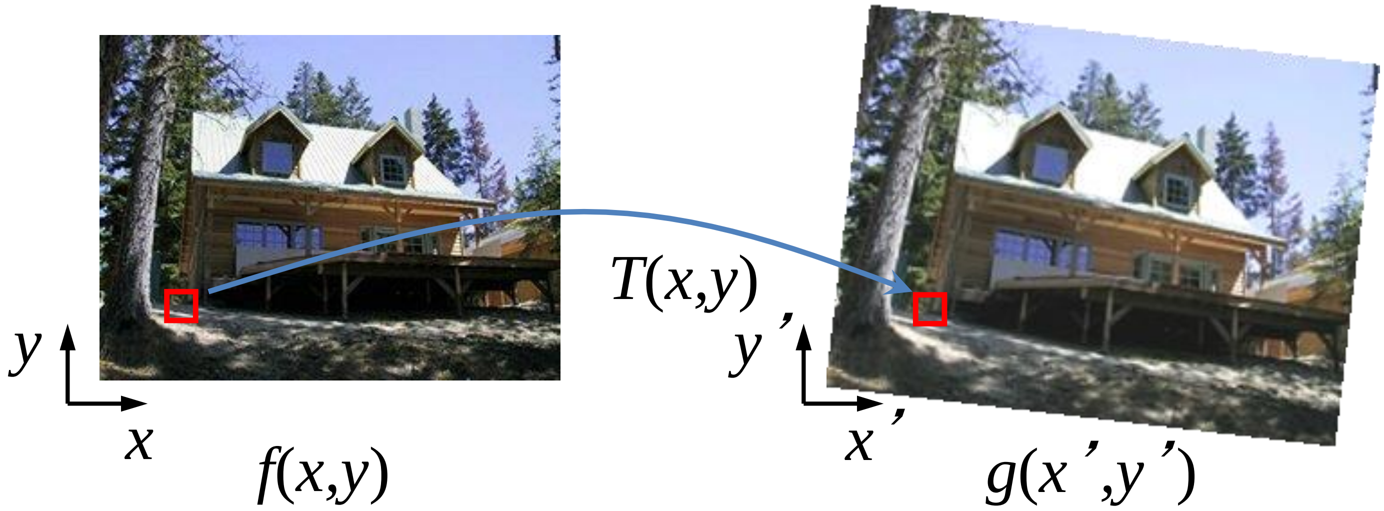
Projective / Homography



- How many correspondences needed for projective? How many DOF?

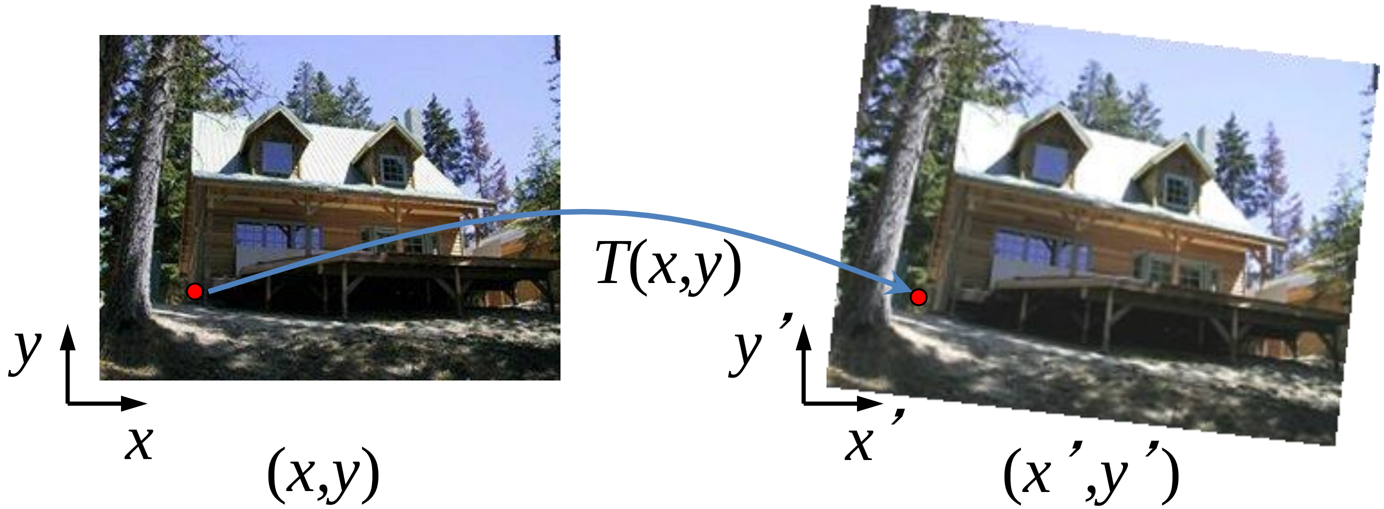
$$T(x,y) = h\left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\right) \qquad h(x,y,z) = (x/z, y/z)$$

Image Warping



- Given a coordinate transform $(x',y') = T(x,y)$ and a source image $f(x,y)$, how do we compute a transformed image $g(x',y') = f(T(x,y))$?

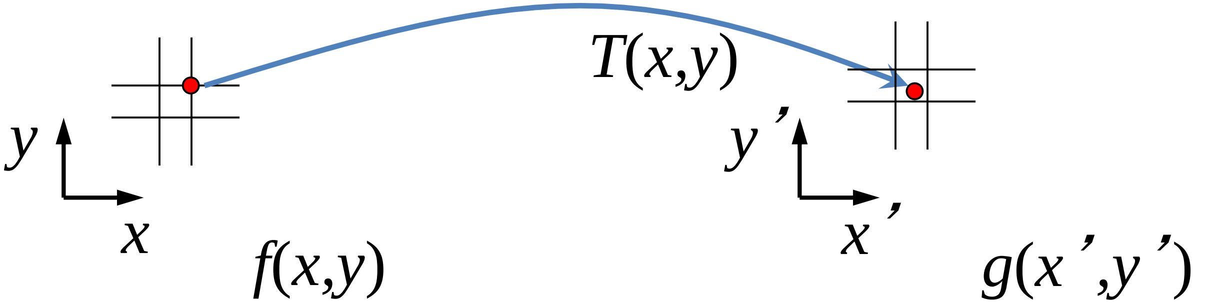
Forward warping



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

$(x',y') = T(x,y)$ in the second image

Forward warping

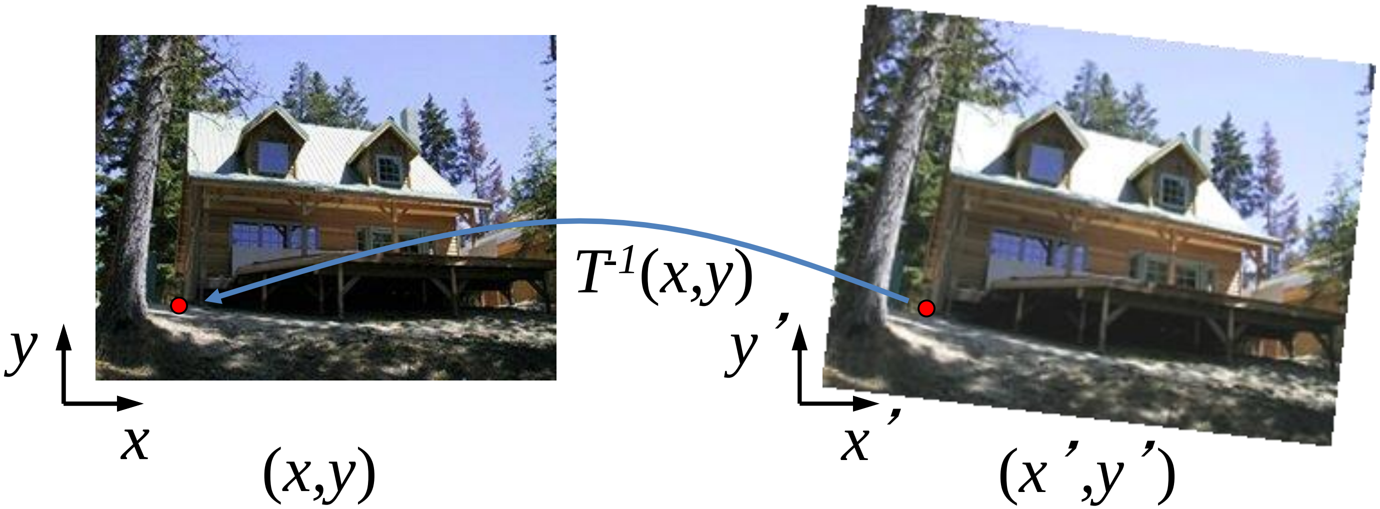


Q: what if pixel lands "between" two pixels?

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

– Known as "splatting"
– Can also interpolate points in target image: [`griddata`]() (Matlab), [`scipy.interpolate.griddata`]() (Python)

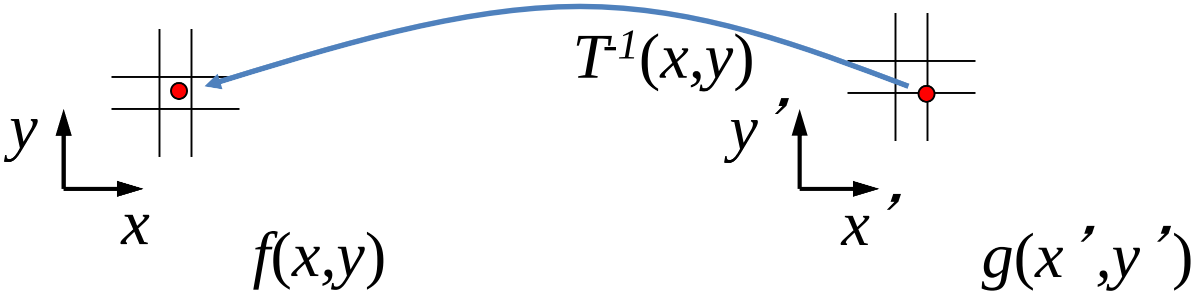
Inverse warping



- Get each pixel color $g(x',y')$ from its corresponding location

  $(x,y) = T^{-1}(x',y')$ in the first image

Inverse warping



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

A: *Interpolate* color value from neighbors

- nearest neighbor, bilinear, Gaussian, bicubic
- See `interp2` (Matlab), `scipy.interpolate.interp2d` (Python)

Forward vs. inverse warping

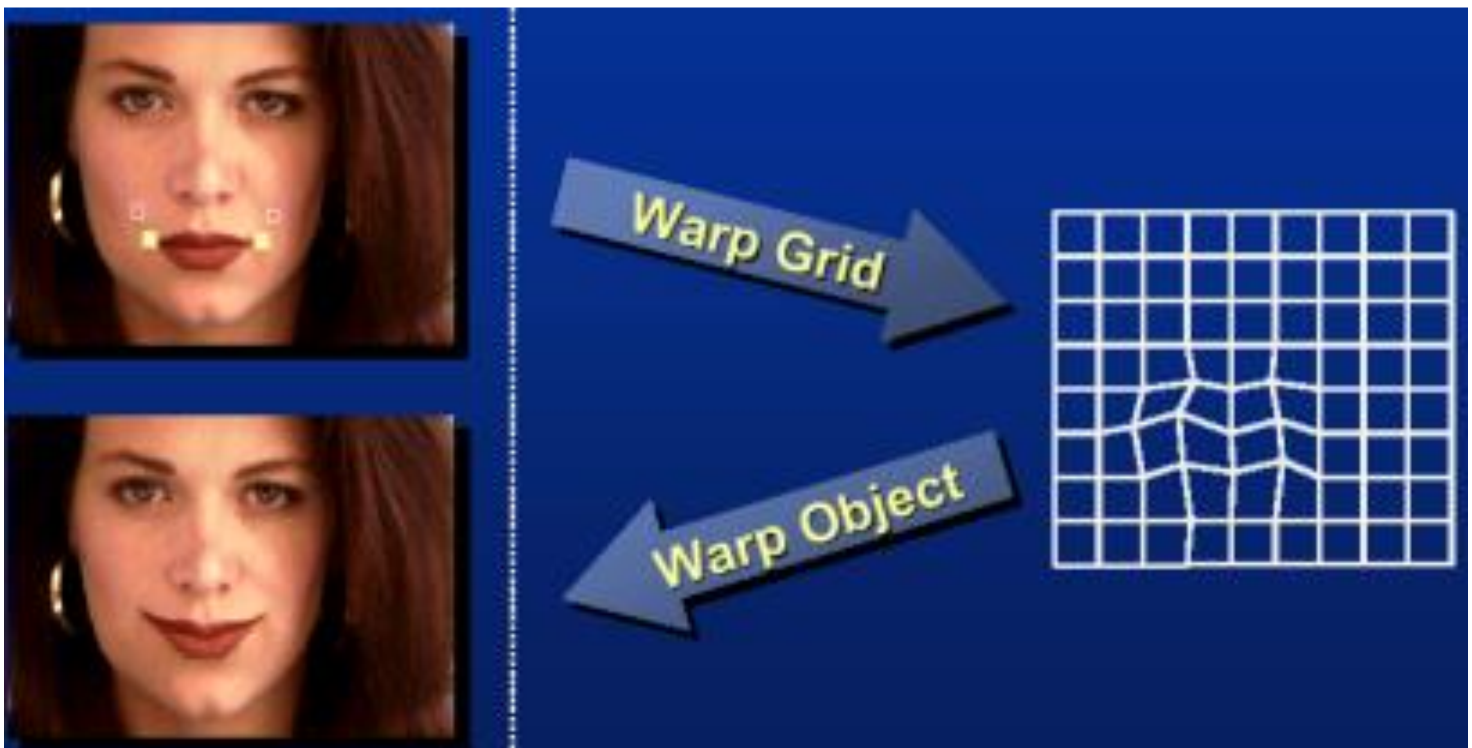
- Q: Which is better?

Forward vs. inverse warping

- Q: Which is better?
- A: Usually inverse – eliminates holes
  - However, it requires an invertible warp function
  - Not always possible

How to Obtain Warp Field?

- Move control points to specify a spline warp
- Spline produces a smooth vector field *T*(*x*, *y*)



Slide Alyosha Efros

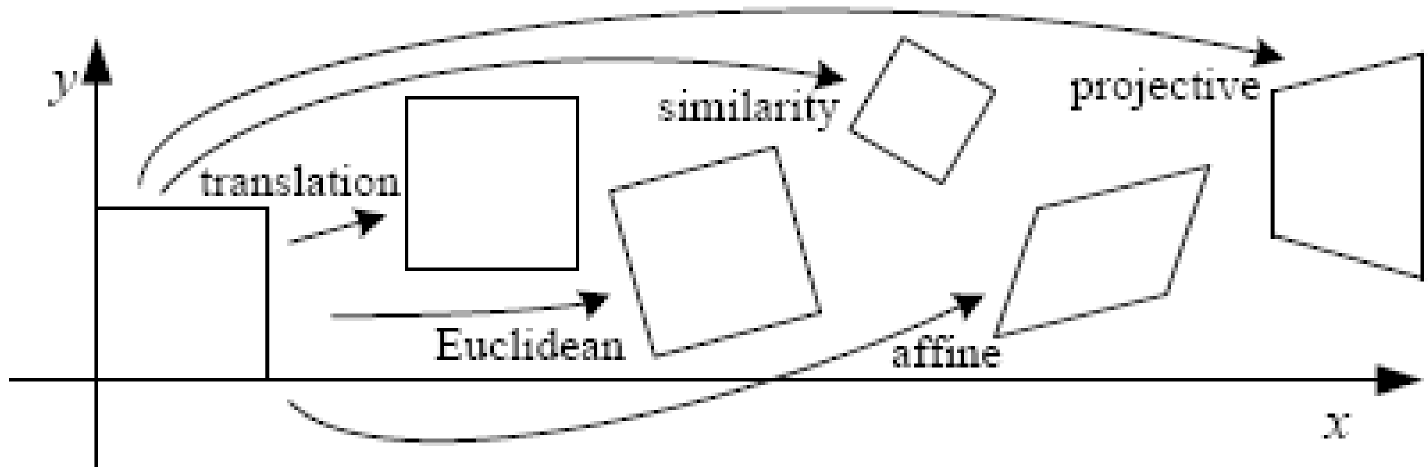
Warp as Interpolation

- **We are looking for a warping field**
  - A function that given a 2D point, returns a warped 2D point
- **We have a sparse number of correspondences**
  - These specify values of the warping field
- **This is an interpolation problem**
  - Given sparse data, find smooth function

Geometry transformations

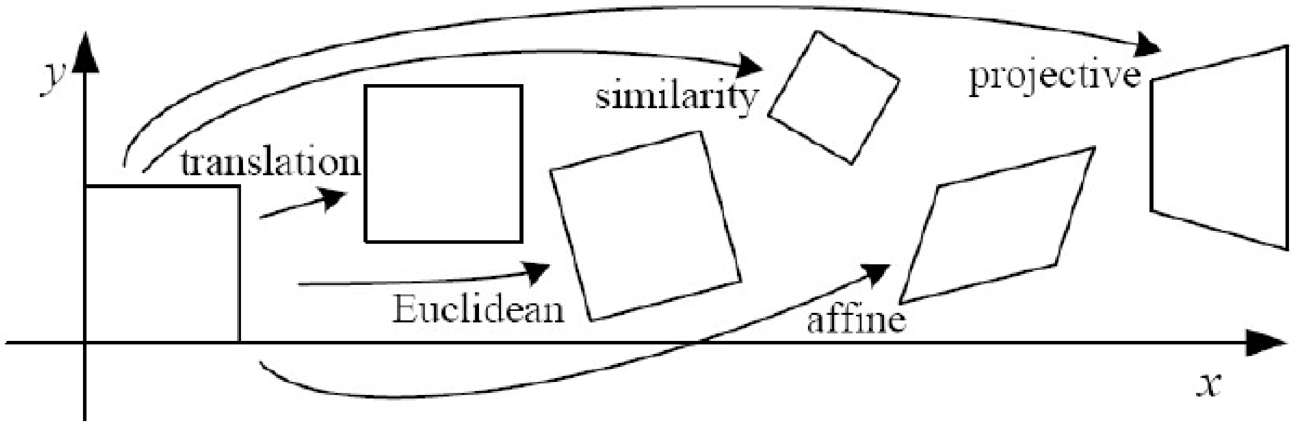
Hartley, R., and Zisserman, A. Multiple View Geometry in Computer Vision, Cambridge University Press, 2004, Chapters 1–3

2D Geometry transformations



$$\text{Translation} \subset \text{Euclidean} \subset \text{Similarity} \subset \text{Affine} \subset \text{Projective}$$

2D Geometry transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+ \cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+ \cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles $+ \cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism $+ \cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

$$\text{Translation} \subset \text{Euclidean} \subset \text{Similarity} \subset \text{Affine} \subset \text{Projective}$$

Basic 2D transformations as 3x3 matrices

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Transformations can be combined by matrix multiplication

Affine Transformations

- Affine transformations are combinations of …
  - –Linear transformations, and
  - –Translations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - – Lines map to lines
  - – Parallel lines remain parallel
  - – Ratios are preserved
  - – Closed under composition
  - – Models change of basis

- Will the last coordinate *w* always be 1?

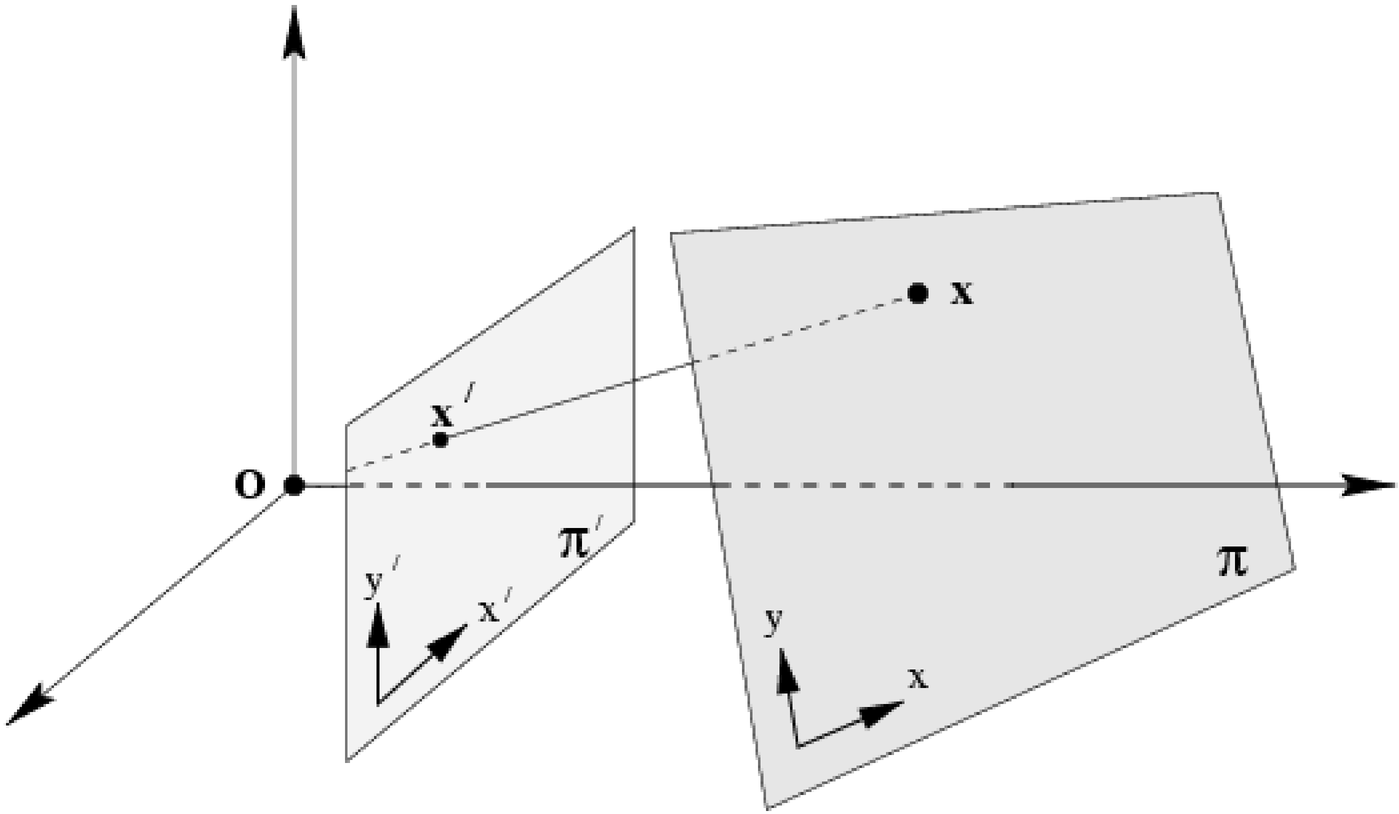
Projective Transformations

- Projective transformations …
  - Affine transformations, and
  - Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

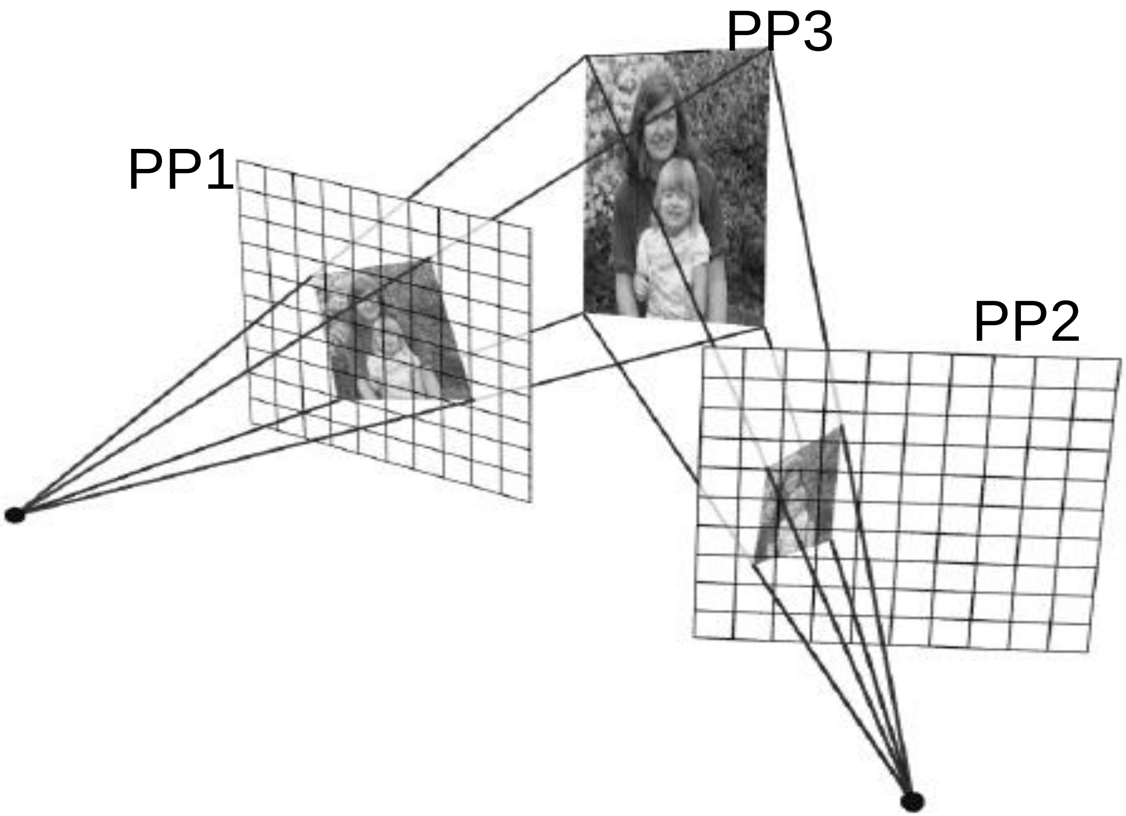
- Properties of projective transformations:
  - •Origin does not necessarily map to origin
  - Lines map to lines
  - •Parallel lines do not necessarily remain parallel
  - •Ratios are not preserved
  - Closed under composition
  - Models change of basis

Mapping between planes



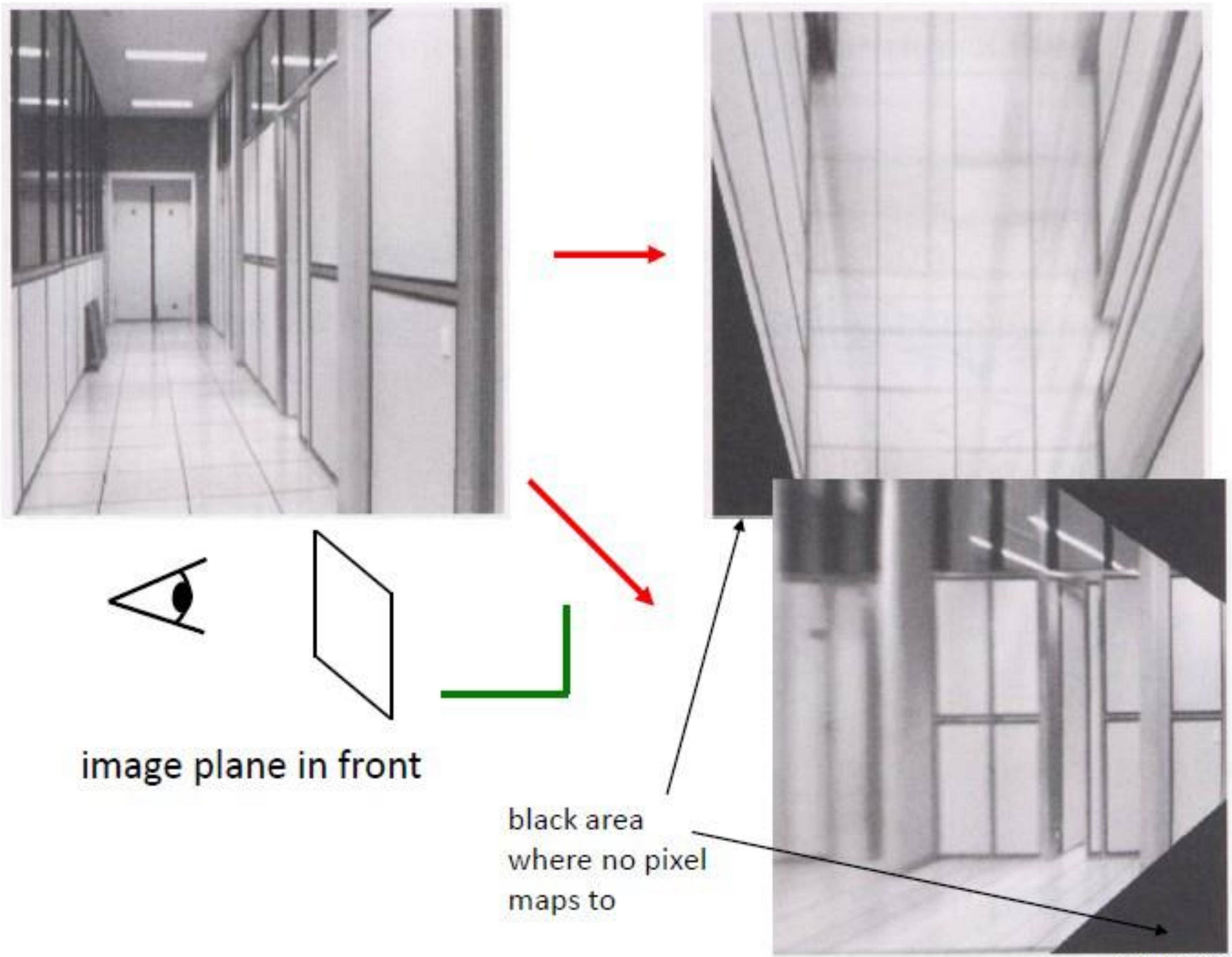
central projection may be expressed by x’=Hx
(application of theorem)

Planar scene (or far away)



- PP3 is a projection plane of both centers of projection, so we are OK!
- This is how big aerial photographs are made

Image warping with homographies



Source: Steve Seitz

Removing projective distortion

select four points in a plane with know coordinates

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

$$x'\left(h_{31}x + h_{32}y + h_{33}\right) = h_{11}x + h_{12}y + h_{13}$$
$$y'\left(h_{31}x + h_{32}y + h_{33}\right) = h_{21}x + h_{22}y + h_{23}$$

(linear in h_{ij})

(2 constraints/point, 8DOF $\Rightarrow$ 4 points needed)

Remark: no calibration at all necessary,
better ways to compute (see later)

Solving for homographies

p' = Hp

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Can set scale factor *i*=1. So, there are 8 unknowns.
- Set up a system of linear equations:
  - **Ah = b**
- where vector of unknowns h = [a,b,c,d,e,f,g,h]^T
- Need at least 8 eqs, but the more the better…
- Solve for h. If overconstrained, solve using least-squares
- Use RANSAC to throw outliers

Parameter estimation

- 2D homography
  Given a set of (x_i, x_i'), compute H
  (optimize $x_i' = Hx_i$)
- 3D to 2D camera projection
  Given a set of (X_i, x_i), compute P ($x_i = PX_i$)
- Fundamental matrix
  Given a set of (x_i, x_i'), compute F
  ($x_i'^T F x_i = 0$)
- Trifocal tensor
  Given a set of (x_i, x_i', x_i''), compute T

Projective geometry

Homogeneous representation of lines and points

- Equation of line in the plane $ax + by + c = 0$
- As an inner product of vectors

$$(x, y, 1)^T (a, b, c) = 0$$

- This is true for any $(kx, ky, k)^T,\ k \neq 0$
- Therefore, the set of vectors $(kx, ky, k)^T,\ k \neq 0$ can represent the point $(x, y) \in \mathbb{R}^2$
- The set of equivalent vectors are called *homogeneous vectors.*

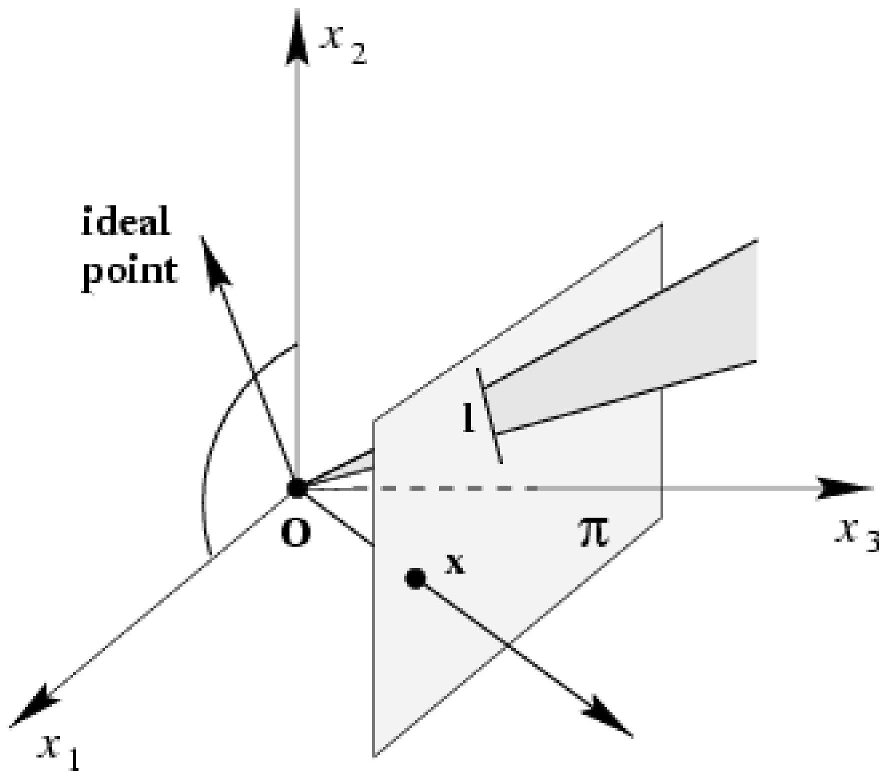
The projective plane

The set of equivalence classes of vectors in

$$\mathbb{R}^3 - (0,0,0)^T$$

forms the projective space $\mathbb{P}^2$.

A model for the projective plane



exactly one line through two points
exactly one point at intersection of two lines

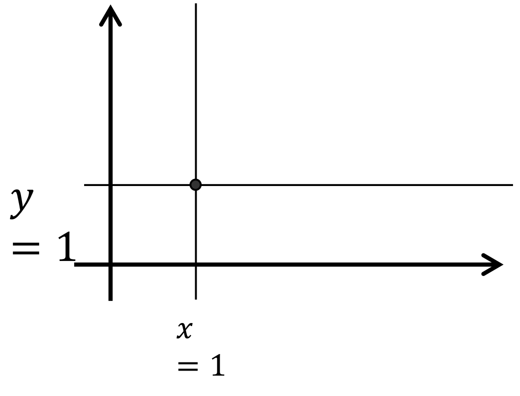
Points and lines

- The point x lies on the line l if and only if

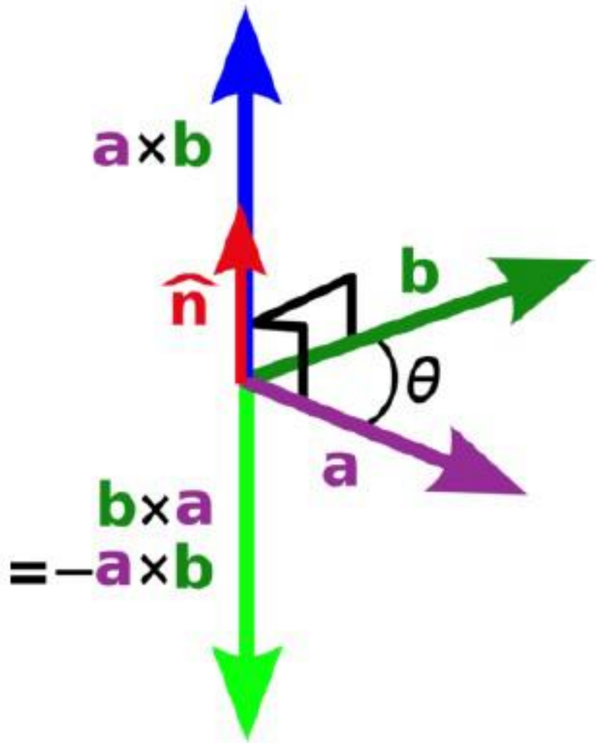
$$\mathbf{x}^T\mathbf{l} = \mathbf{l}^T\mathbf{x} = 0$$

- The intersection of two lines $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$
- The line through two points $\mathbf{l} = \mathbf{x} \times \mathbf{x}'$

Example



מכפלה ווקטורית – cross product

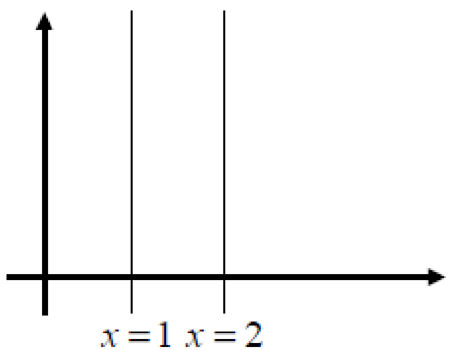


Ideal points and the line at infinity

Intersections of parallel lines

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

Example



$(b, -a)$ tangent vector

(a, b) normal direction

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

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

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

Note that in **P**² there is no distinction between ideal points and others

נקודות אידיאליות

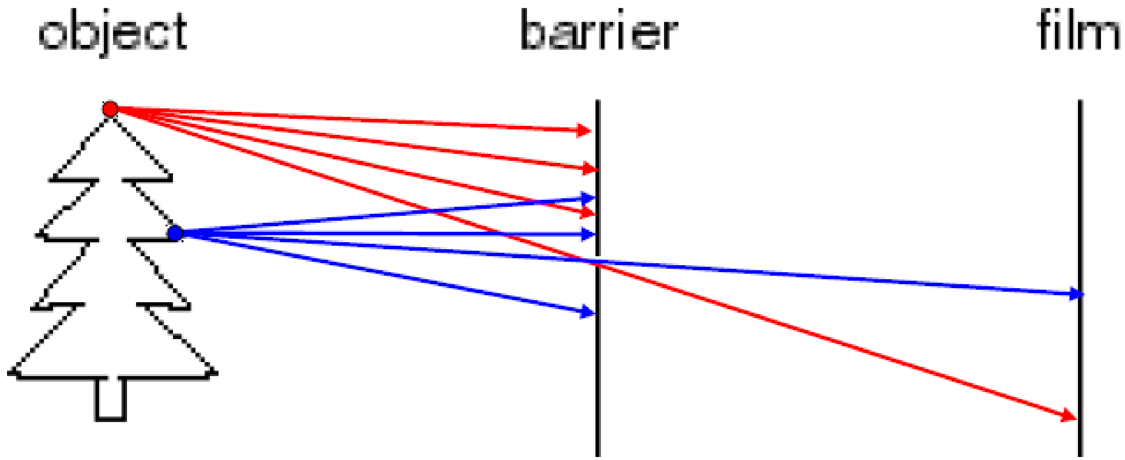
- הישרים: $\mathbf{l}_1 = [a, b, c_1]^T$ ו- $\mathbf{l}_2 = [a, b, c_2]^T$ הם ישרים מקבילים. איפה הם נחתכים?
- $\mathbf{l}_1 \times \mathbf{l}_2 = (c_2 - c_1)[b, -a, 0]^T \approx [b, -a, 0]^T$
- נקודות עם קואורדינטה שלישית 0 נקראות
  - נקודות אידיאליות
  - כיוונים
  - נקודות באינסוף
- אין להן ייצוג בגיאומטריה האוקלידית

הישר באינסוף

- כל הנקודות באינסוף נמצאות על אותו הישר, "הישר באינסוף": $\mathbf{l}_\infty = [0,0,1]^T$
- קל לודא על ידי $\mathbf{l}_\infty{}^T\mathbf{p} = [0,0,1][x,y,0]^T = 0$
- נקרא גם: קו האופק
- לא בהכרח נמצא בתוך התמונה

Camera model

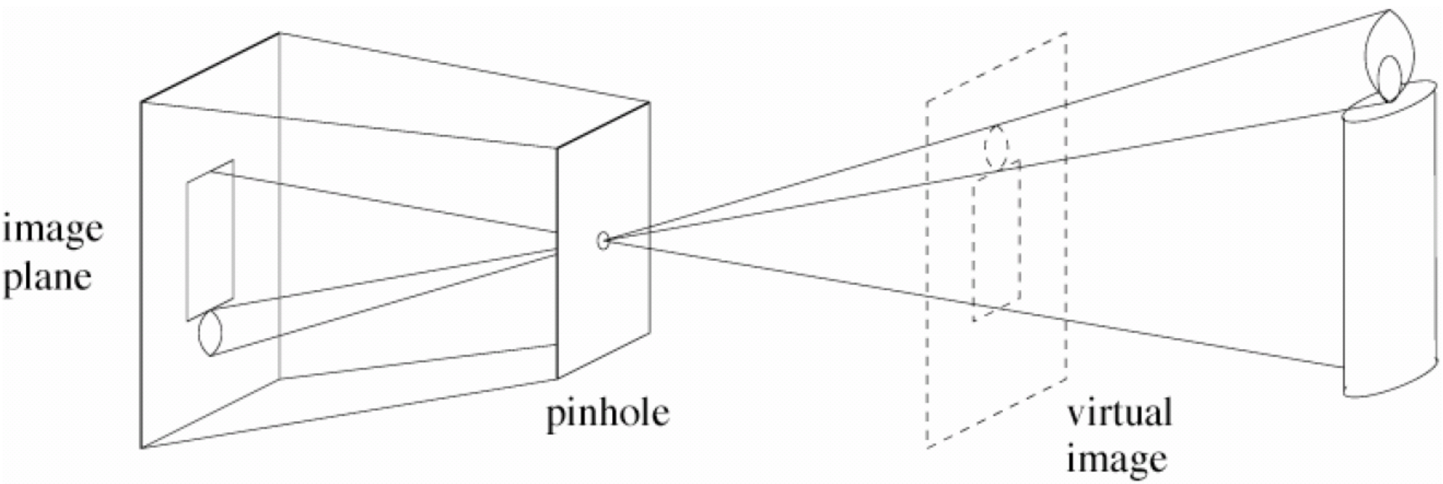
Pinhole Camera



- Add a barrier to block off most of the rays
  - This reduces blurring
  - The opening is known as the **aperture**
  - How does this transform the image?

Pinhole Camera

- Pinhole camera is a simple model to approximate imaging process, perspective **projection**.



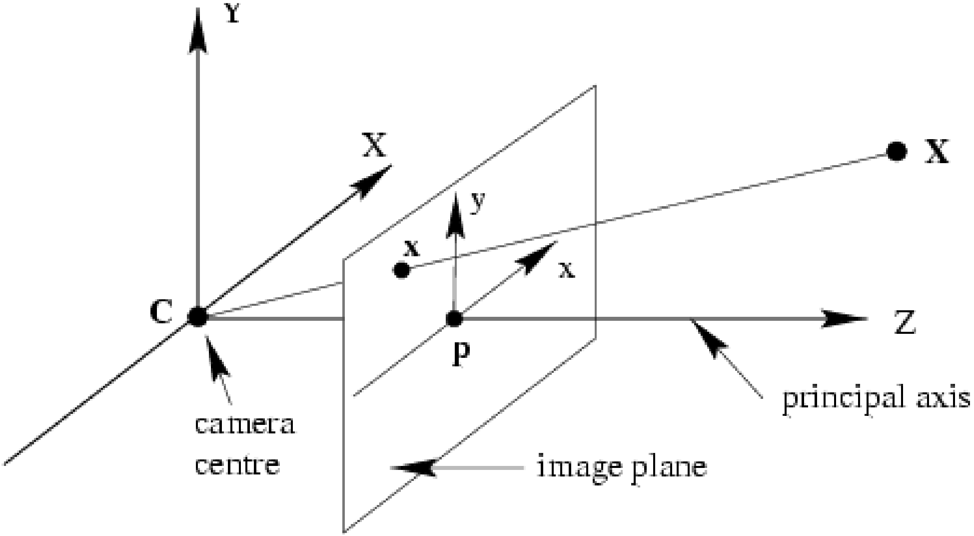
- If we treat pinhole as a point, only one ray from any given point can enter the camera.

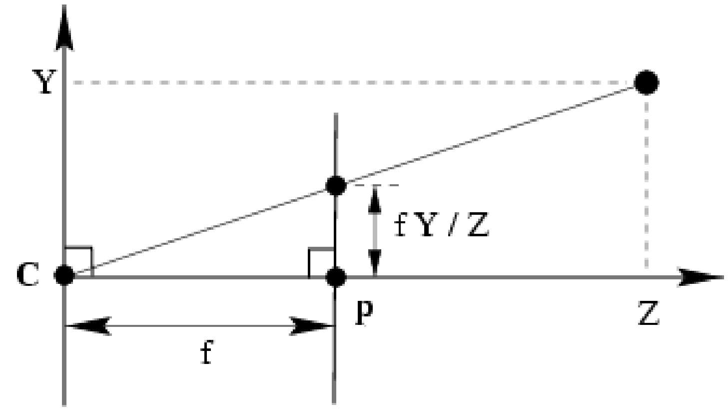
Fig from Forsyth and Ponce

Perspective effects

Perspective effects

Pinhole camera model

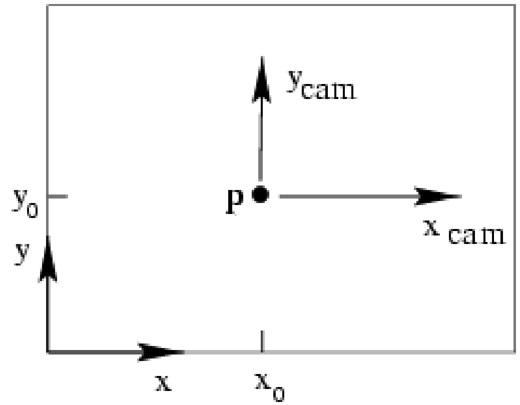




$$\begin{bmatrix} X & Y & Z \end{bmatrix}^T \mapsto \begin{bmatrix} fX/Z & fY/Z \end{bmatrix}^T$$

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

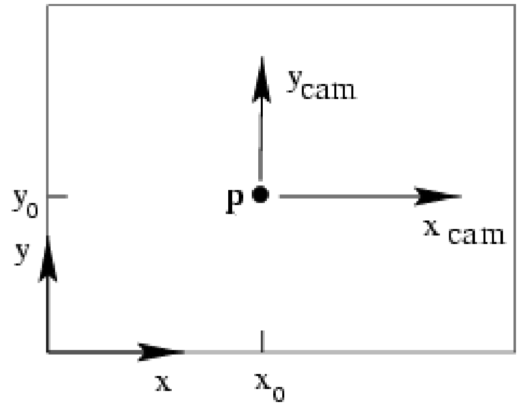
Principle point offset



$$\begin{bmatrix} X & Y & Z \end{bmatrix}^T \mapsto \begin{bmatrix} fX/Z + p_x & fY/Z + p_y \end{bmatrix}^T$$

$$\begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Principle point offset



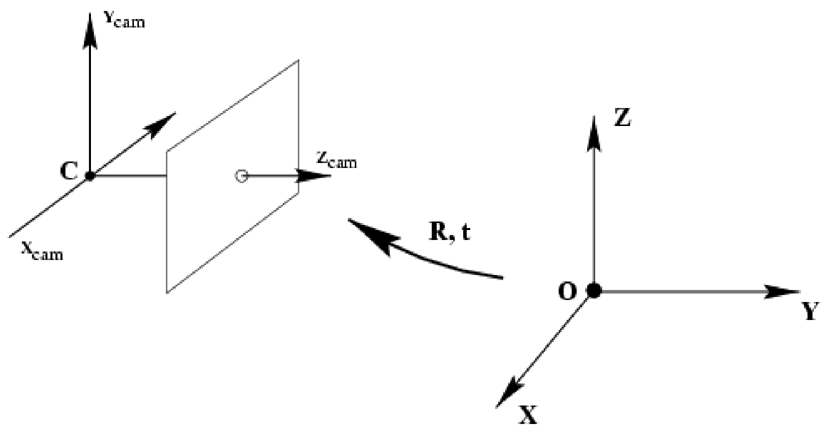
$$\mathbf{x} = K[\, I \mid \mathbf{0}]\mathbf{X}_{cam}$$

$$\begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

Calibration matrix

Camera rotation and translation



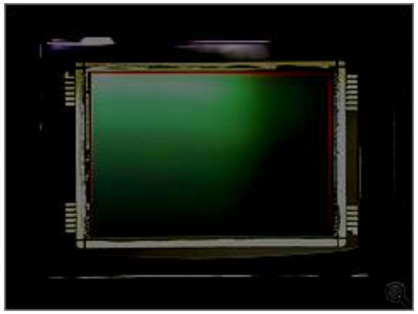
$$\mathbf{X}_{cam} = R(\mathbf{X} - C)$$

$$\mathbf{X}_{cam} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \mathbf{X}$$

$$\mathbf{x} = K[\, I \mid \mathbf{0}]\mathbf{X}_{cam} = K[\, R \mid -RC]\mathbf{X} = P\mathbf{X}$$

$$P = K[\, R \mid \mathbf{t}], \quad \mathbf{t} = -RC$$

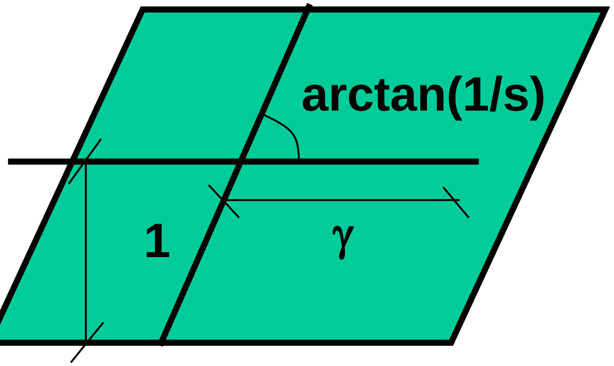
CCD camera



$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

When skew angle is not zero:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$



Finite projective camera

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

$$P = K[\, R \mid \mathbf{t}], \quad \mathbf{t} = -RC$$

11 DOF (5 + 3 + 3)

תרגיל

- מצלמת חריר אידיאלית (ideal pinhole camera) בעלת מרחק מוקד של 7mm.
- גודל כל פיקסל הוא 0.02mm X 0.03mm.
- ומרכז הצילום נמצא ב 550×650 - כאשר הקורדינטות מתחילות מפינה שמאלית עליונה ב - (0,0).
- מהי מטריצת הקליברציה של המצלמה?

תרגיל -תשובה

- מצלמת חריר אידיאלית (ideal pinhole camera)בעלת מרחק מוקד של 7mm.
- גודל כל פיקסל הוא 0.02mm X 0.03mm.
- ומרכז הצילום נמצא ב 550×650 -כאשר הקורדינטות מתחילות מפינה שמאלית עליונה ב-(0,0).
- מהי מטריצת הקליברציה של המצלמה?

<u>תשובה:</u>

$$K = \begin{bmatrix} f \cdot k_u & 0 & x_0 \\ 0 & f \cdot k_v & y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 \cdot \frac{1}{0.02} & 0 & 550 \\ 0 & 7 \cdot \frac{1}{0.03} & 650 \\ 0 & 0 & 1 \end{bmatrix}$$

סיכום

- טקסטורה
- Image transformations
- Geometry
- Projective geometry
- Camera model

בפעם הבאה (והאחרונה):

- Camera model (cont.)
- Stereo vision

- מצגות פרויקט