Perspective, Mosaics and Panoramas

universität bonn

Mosaics and Panoramas

- Basic idea
- Registration
- Resampling
- Blending

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°

Slide from Brown & Lowe

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°

Slide from Brown & Lowe

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°

Slide from Brown & Lowe

Single vs. Multiple Viewpoint

- Single-viewpoint
  - Necessary for creating pure perspective images.
  - Many vision algorithms assume pinhole cameras.
  - Images that aren’t perspective images look distorted.
- Multi-viewpoint
  - Cross-slit panoramas, etc.
  - necessary for scenes which cannot be captured from a single viewpoint

Image Mosaicing

- Register multiple images
- Blend

universitätbonn


323
322
321
320
319
359
360
361
362
363
1021
1020
1019
1018
1017
9186
9185
9184
9183
9182
9219
9220
9221

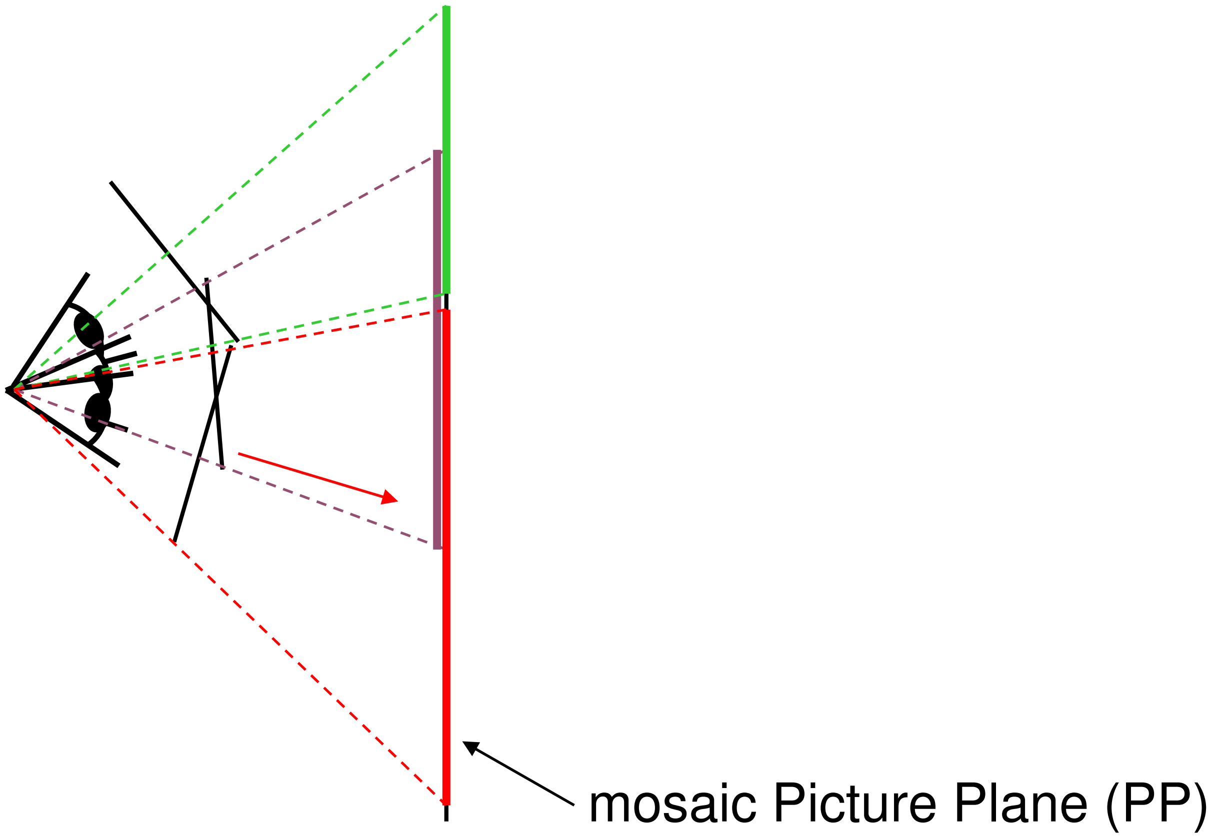
Single Center of Projection

- Take a sequence of images from the same position
  - Rotate the camera about its optical center
  - Where is optical center of thick lens?
- Compute transformation between second image and first
- Transform the second image to overlap with the first
- Blend the two together to create a mosaic
- If there are more images, repeat

- ...why don't we need the 3D geometry?

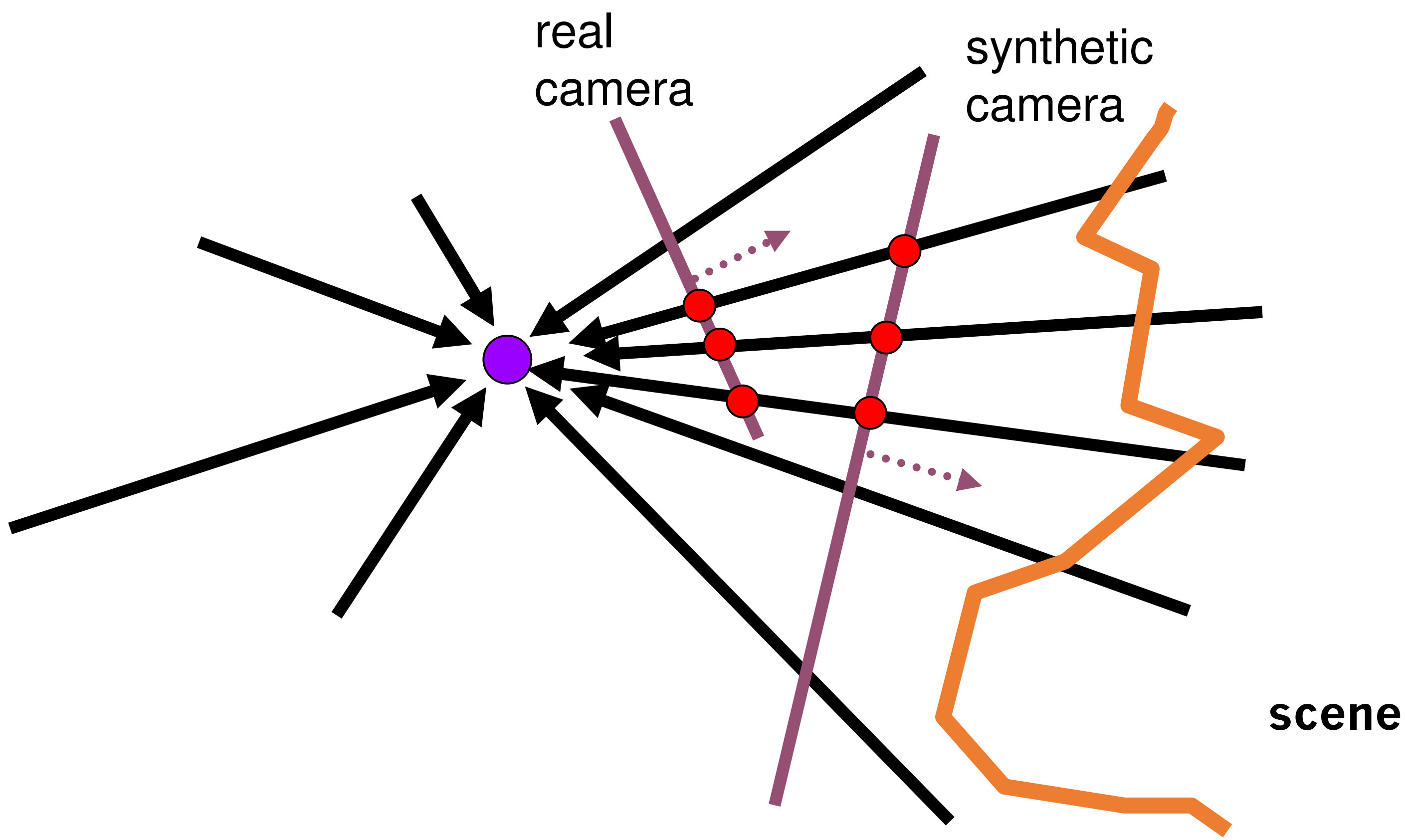
Image Reprojection

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*



universitätbonn

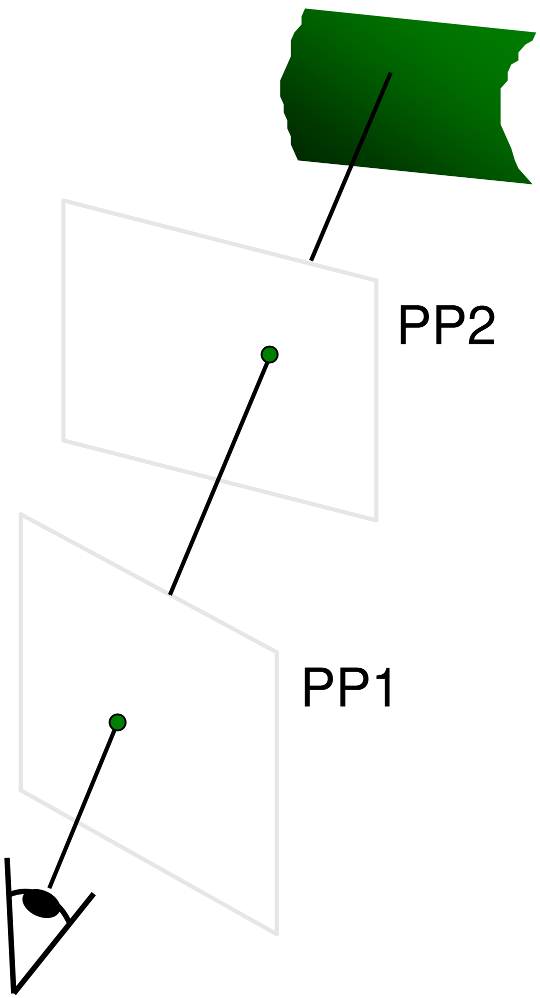
A pencil of rays contains all views



Can generate any synthetic camera view
as long as it has **the same center of projection**!

Image reprojection

- How to relate two images from the same camera center?
- Images contain the same information along the same ray.
- Use 2D image warp instead of ray tracing.



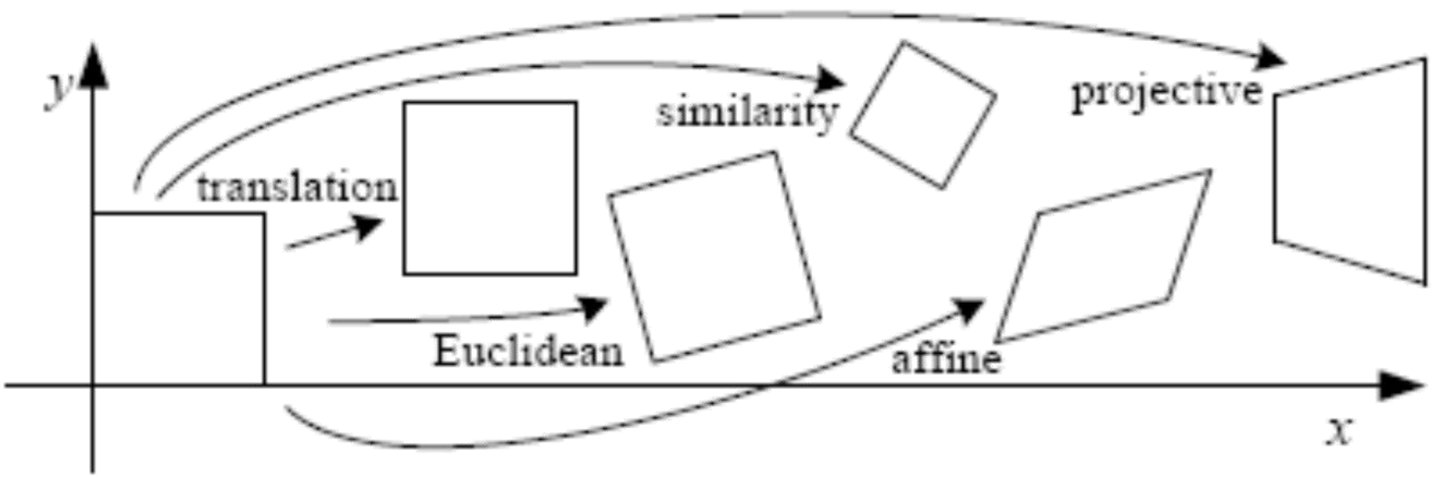
Taxonomy of Projective Transformations

- Homogeneous coordinates: Specify 2D image points using three dimensions.

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

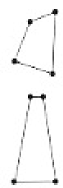
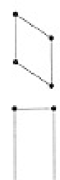
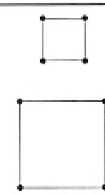
Points that differ only by scalar factor are equivalent.

Projective transformation: $\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}$

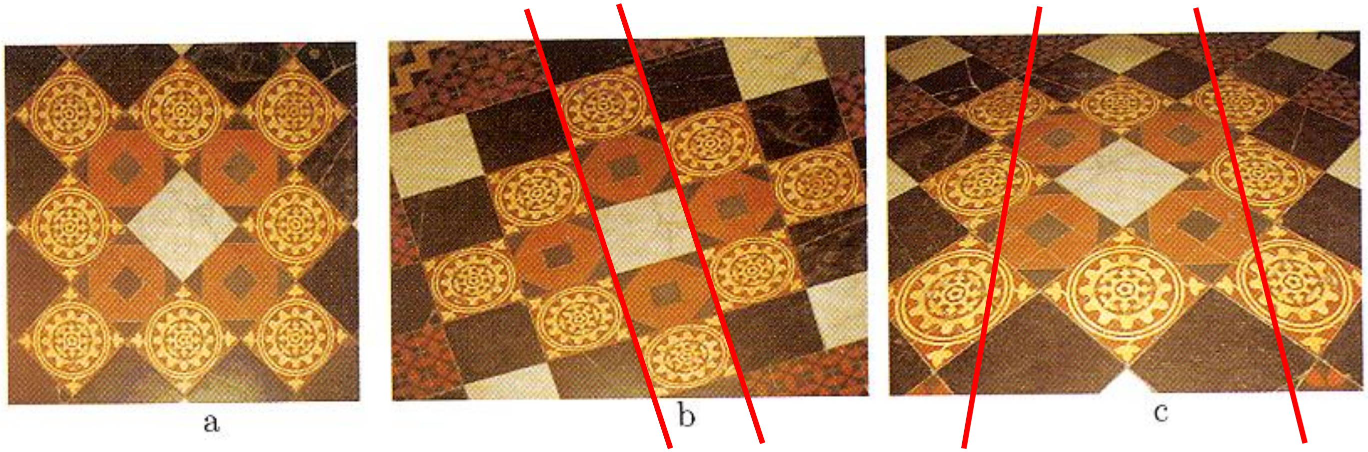


Taxonomy of Projective Transformations

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle.
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

Distortions under Central Projection



- **Similarity**: circle remains circle, square remains square

⇒ line orientation is preserved

- **Affine**: circle becomes ellipse, square becomes rhombus

⇒ parallel lines remain parallel

- **Projective**: imaged object size depends on distance from camera

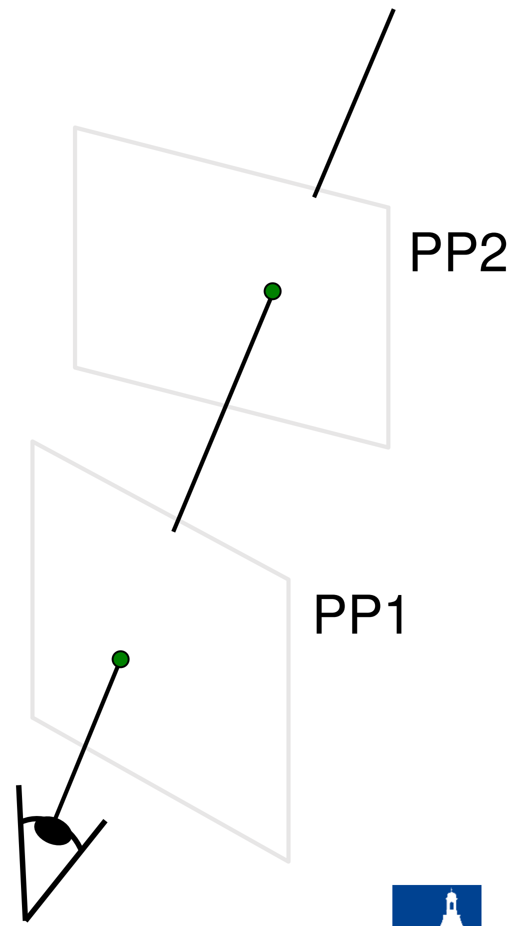
⇒ parallel lines converge

Homography

- Projective mapping between any two PPs with the same center of projection
  - rectangle should map to arbitrary quadrilateral
  - parallel lines no longer parallel
  - but straight lines preserved
  - same as: project, rotate, reproject
- called Homography

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

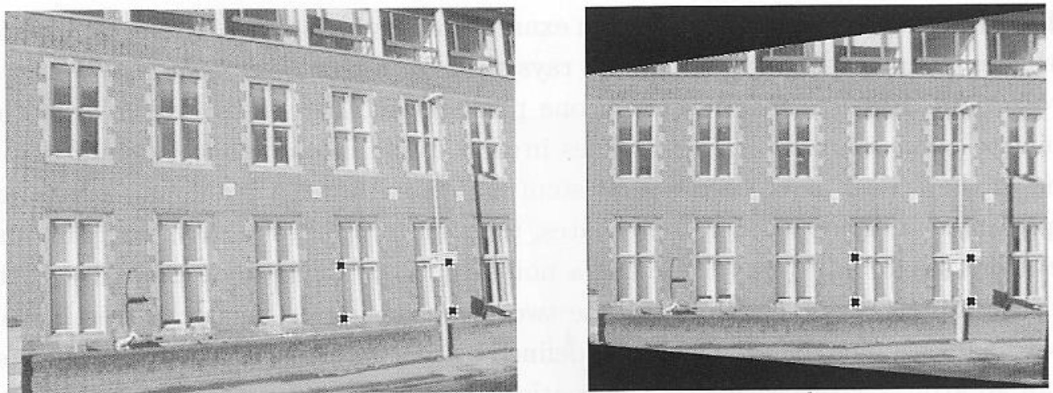
$\mathbf{p}'$ $\quad\quad\quad$ H $\quad\quad\quad$ $\mathbf{p}$



To apply a homography H

- Compute $\mathbf{p}' = \mathrm{H}\mathbf{p}$ (regular matrix multiply)
- Convert $\mathbf{p}'$ from homogeneous to image coordinates

Removing Projective Distortion



a b

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ \mathbf{1} \end{pmatrix}$$

Projective transformation in inhomogeneous form

$$x'^{(2D)} = \frac{x'}{w'} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad y'^{(2D)} = \frac{y'}{w'} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

4 general point correspondences ($x, y \rightarrow x'^{(2D)}, y'^{(2D)}$) on the planar facade lead to eight linear equations of the type

$$x'^{(2D)}(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}$$
$$y'^{(2D)}(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}$$

Sufficient to solve for H up to multiplicative factor

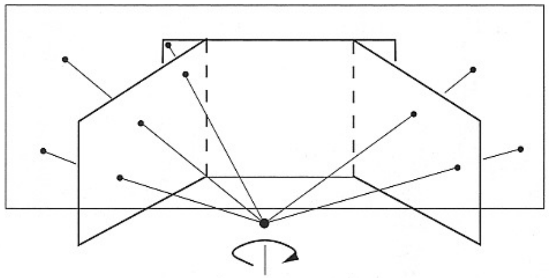
Getting it numerically stable is not trivial! **Best use proven libraries like OpenCV**

Panoramic Mosaicing

Rotation about camera center: homography

- choose one image as reference
- compute homography to map neighboring image to reference image plane
- projectively warp image, add to reference plane
- repeat for all images

$\Rightarrow$ bow tie shape



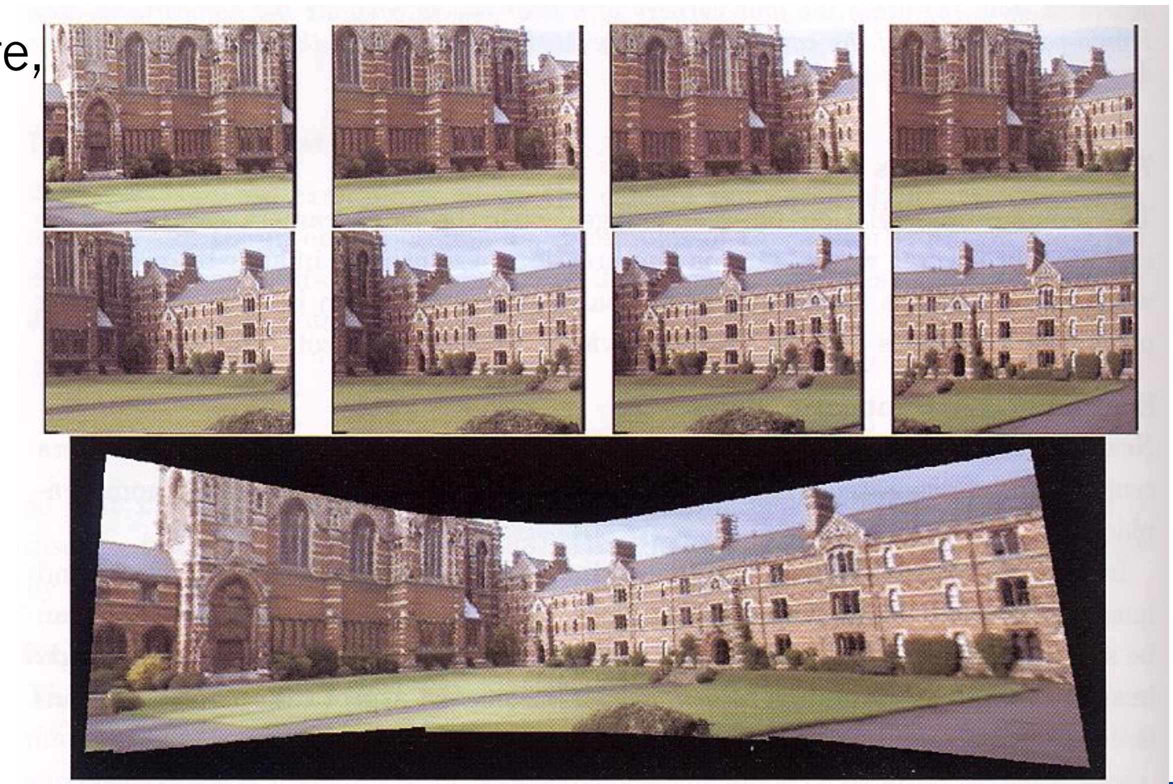
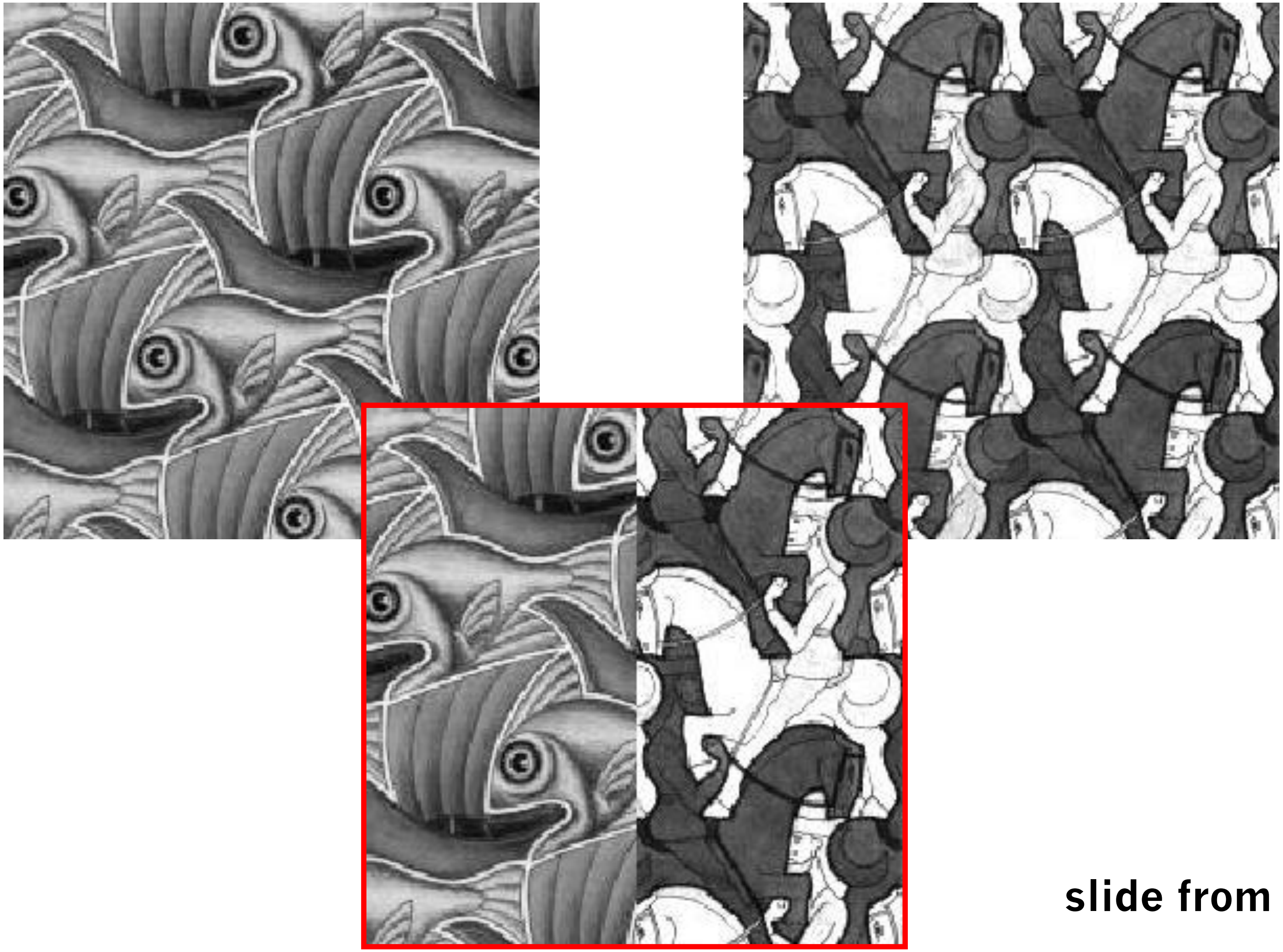
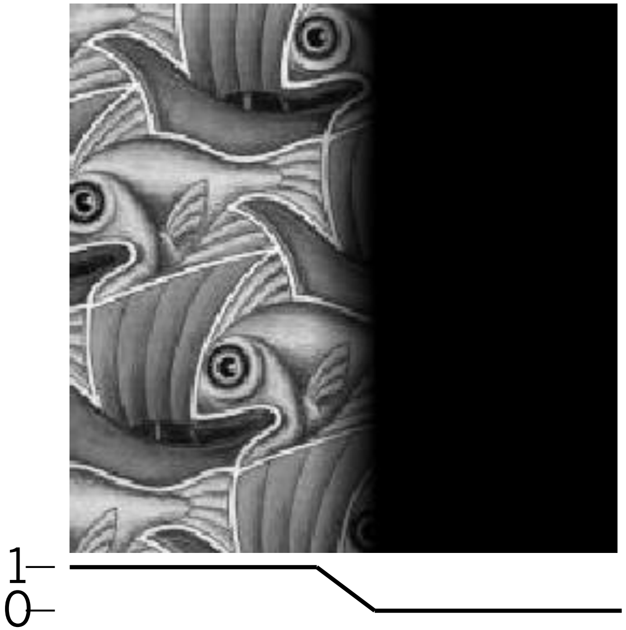


Image Blending

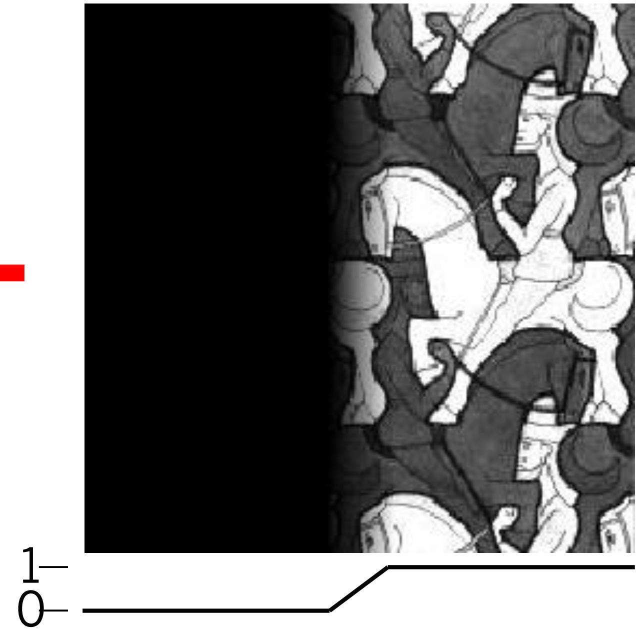


slide from Efros

Feathering



+



=

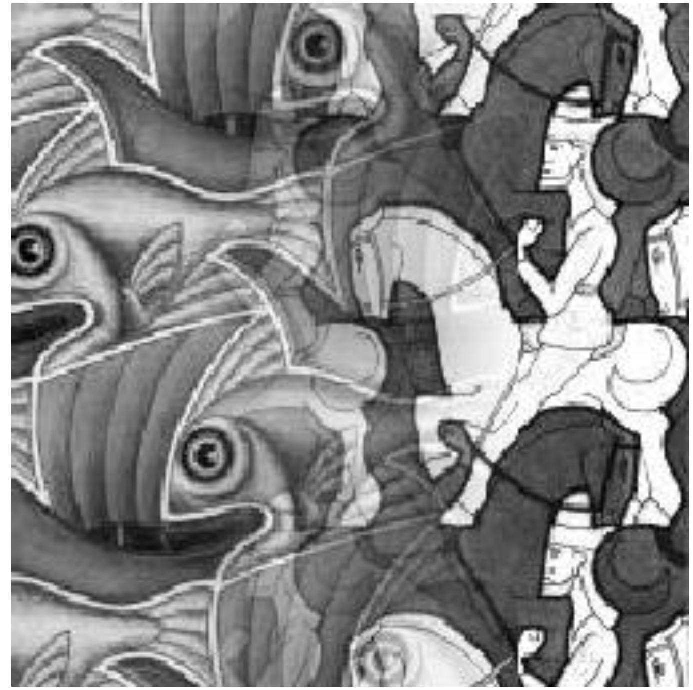
Encoding transparency

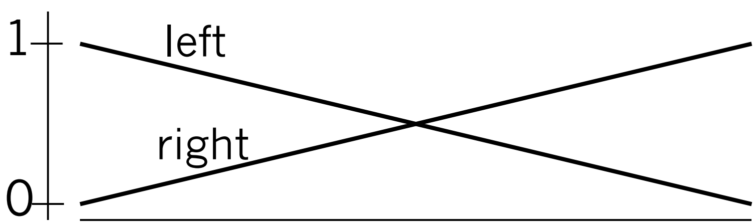
$$I(x,y) = (\alpha R, \alpha G, \alpha B, \alpha)$$

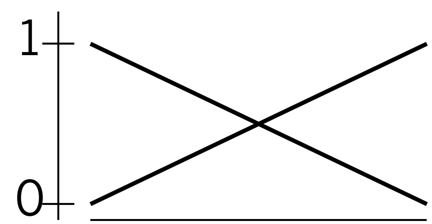
$$I_{blend} = I_{left} + I_{right}$$

slide from Efros

Effect of Window Size

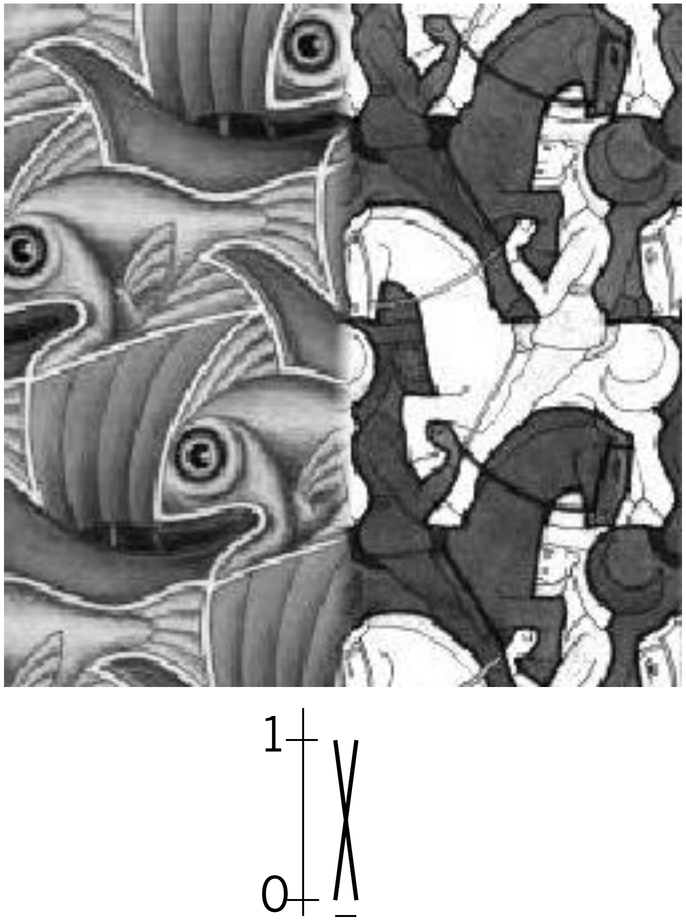


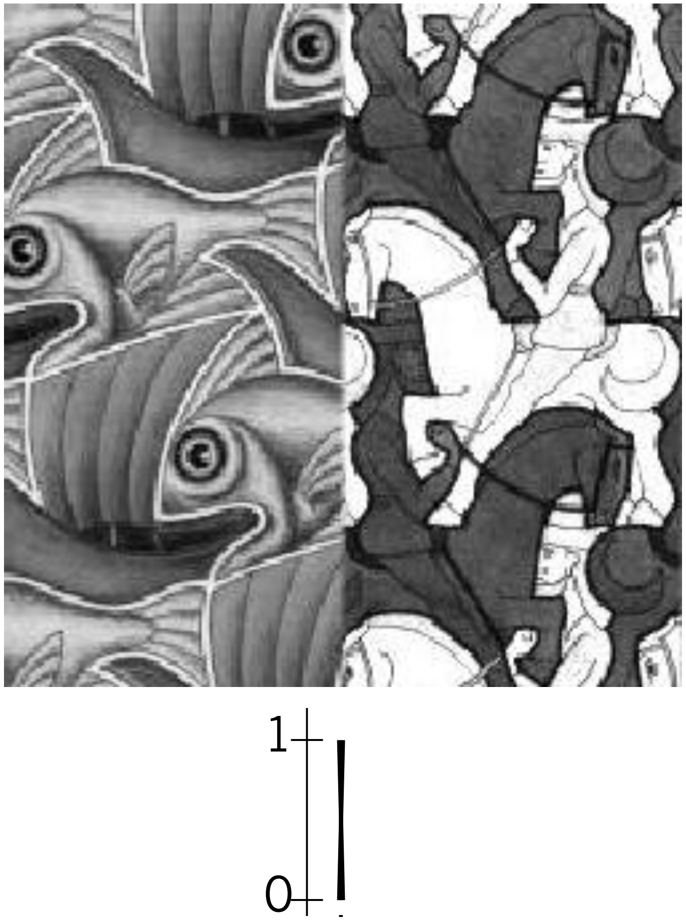




slide from Efros

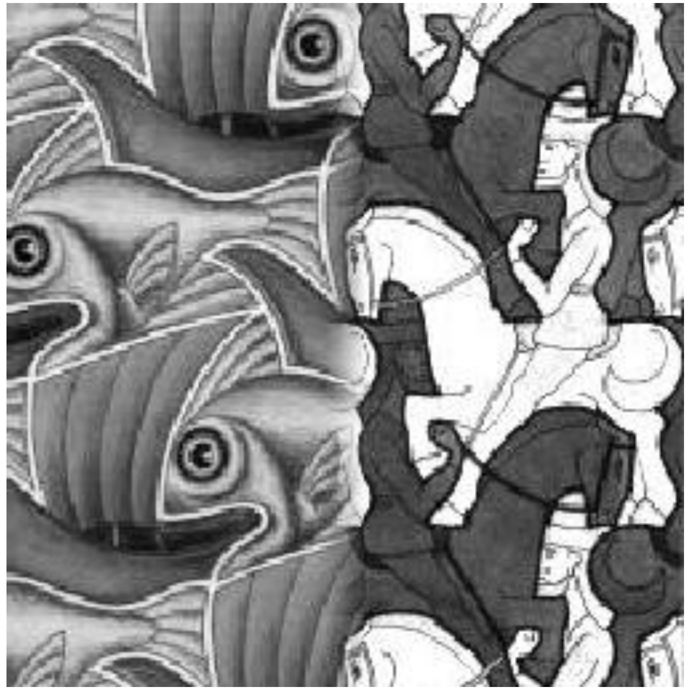
Effect of Window Size

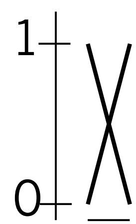




slide from Efros

Good Window Size





"Optimal" Window: smooth but not ghosted

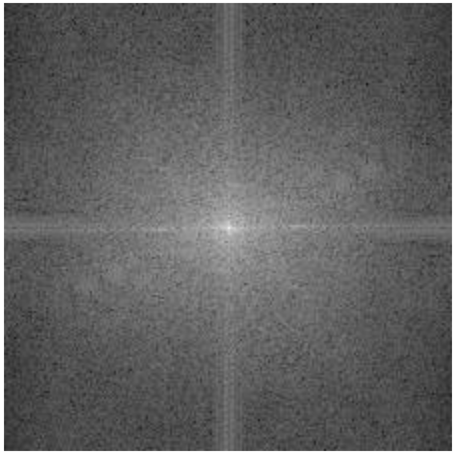
[Slide: Efros]

What is the Optimal Window?

- To avoid seams
  - window >= size of largest prominent feature
- To avoid ghosting
  - window <= 2*size of smallest prominent feature
- Natural to cast this in the *Fourier domain*
  - largest frequency <= 2*size of smallest frequency
  - do blending in different frequency bands

What if the Frequency Spread is Wide

FFT



- Idea (Burt and Adelson)
  - Compute $F_{left} = FFT(I_{left})$, $F_{right} = FFT(I_{right})$
  - Decompose Fourier image into octaves (bands)
    - $F_{left} = F_{left}^{1} + F_{left}^{2} + \ldots$
  - Feather corresponding octaves F_{left}^{i} with F_{right}^{i}
    - Can compute inverse FFT and feather in spatial domain
  - Sum feathered octave images in frequency domain
- Better implemented in *spatial domain*

What does blurring take away?

original

What does blurring take away?

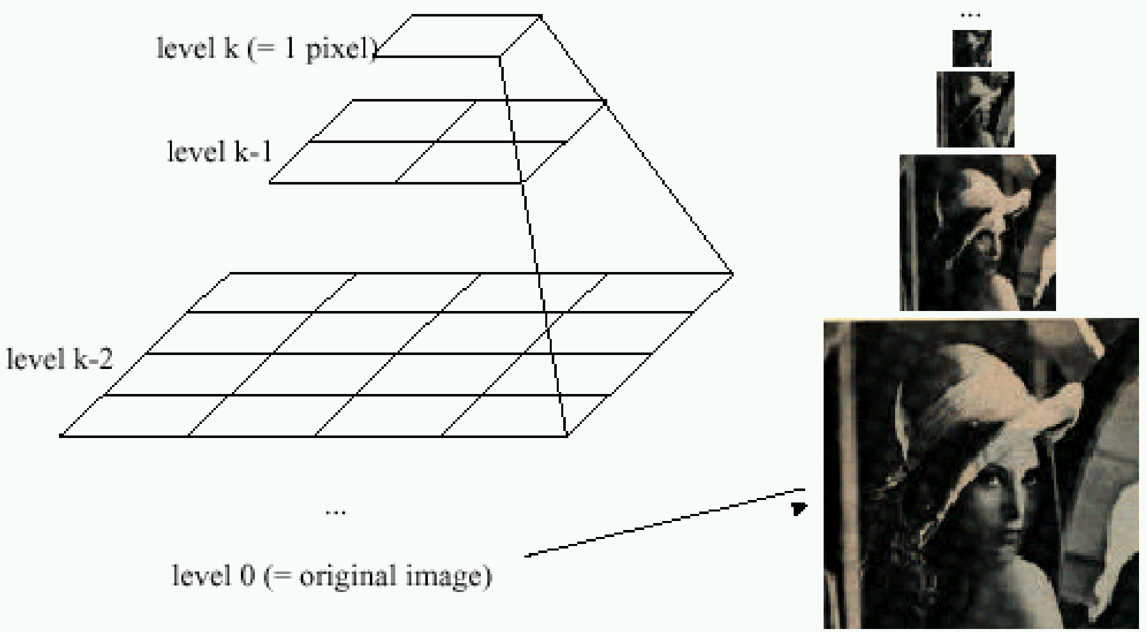
smoothed (5x5 Gaussian)

High-Pass Filter

smoothed – original

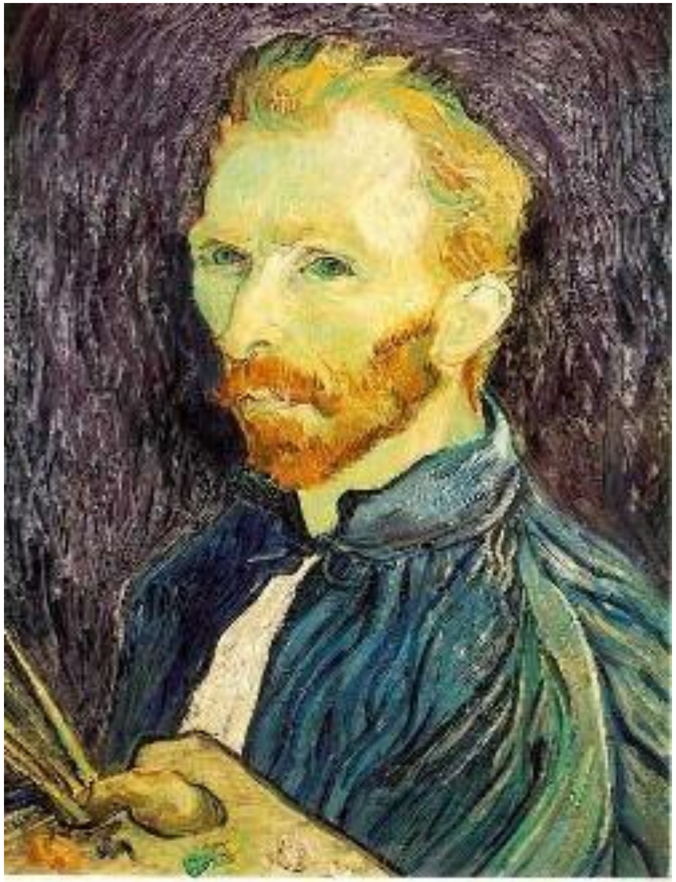
Image Pyramids

Idea: Represent NxN image as a "pyramid" of 1×1, 2×2, 4×4,…, $2^k \times 2^k$ images (assuming $N=2^k$)



Mipmap or precursor of wavelets

Image Sub-sampling

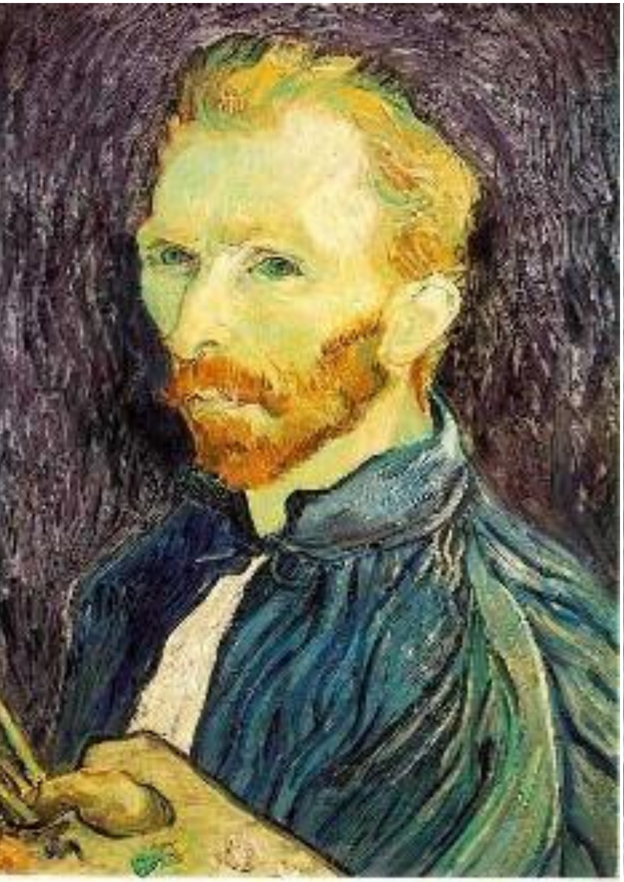


1/4

1/8

Throw away every other row and column to create a 1/2 size image

Image Sub-sampling



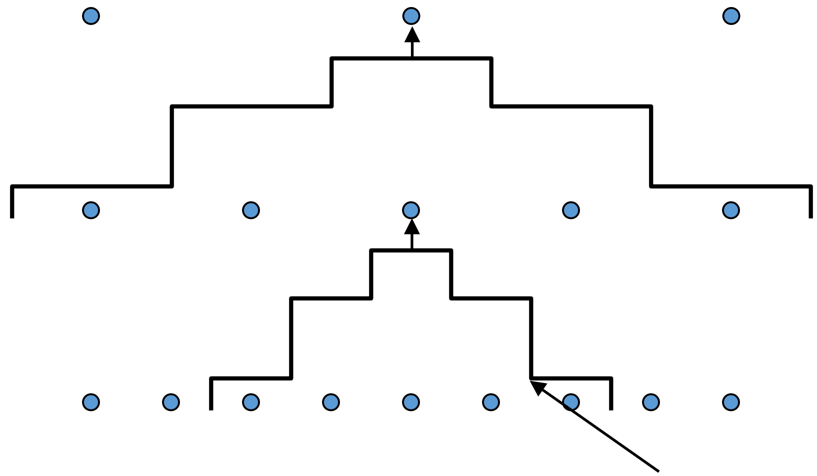
1/2

1/4 (2x zoom)

1/8 (4x zoom)

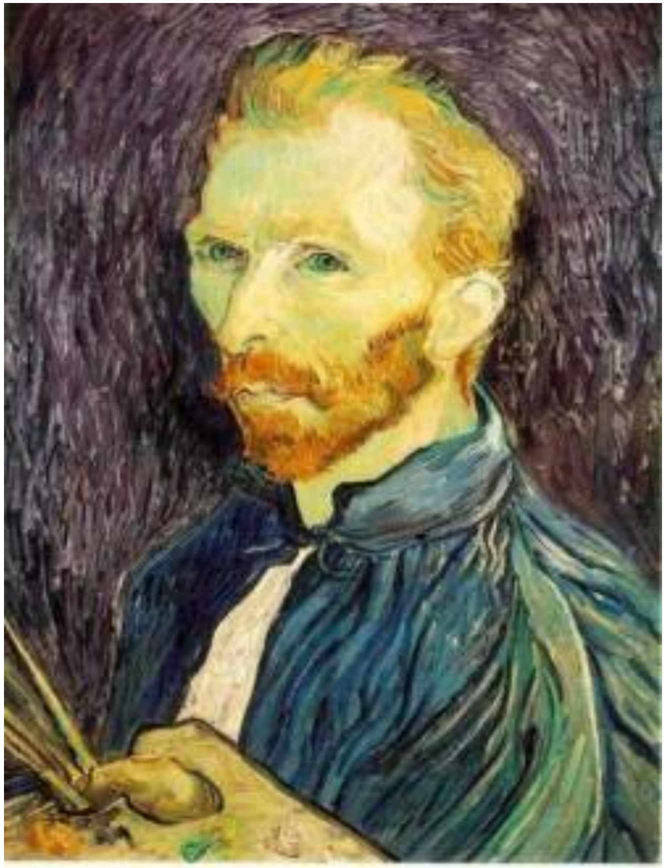
Why does this look so bad?

Gaussian Pyramid Construction



- Repeat
  - Filter
  - Subsample
- Until minimum resolution reached
- Good filter kernel: $^1/_{256}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix}[1\ 4\ 6\ 4\ 1] = \frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\4 & 16 & 24 & 16 & 4\\6 & 24 & 36 & 24 & 6\\4 & 16 & 24 & 16 & 4\\1 & 4 & 6 & 4 & 1\end{bmatrix}$
- Whole pyramid is only 4/3 the size of the original image!

Gaussian pre-filtering



Gaussian 1/2

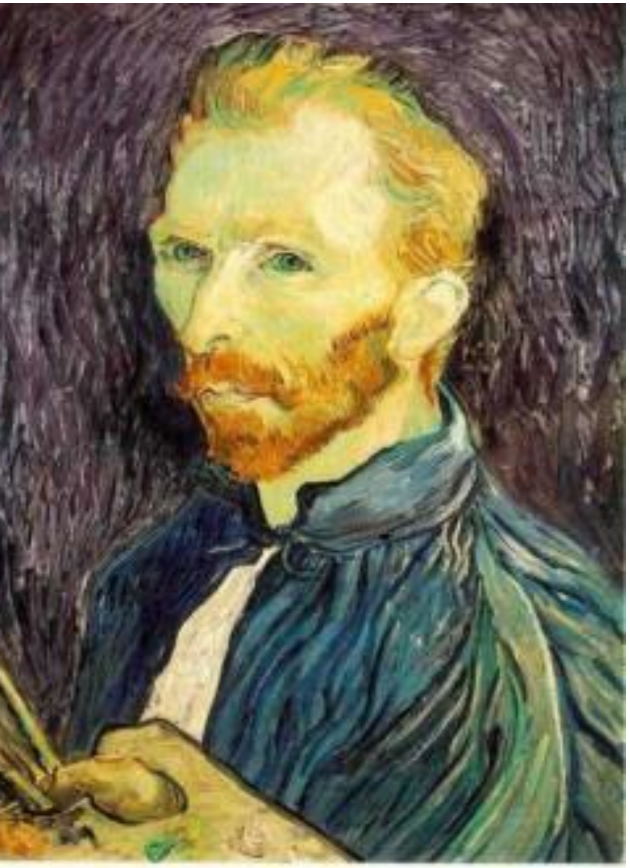
G 1/4

G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction.

Subsampling with Gaussian pre-filtering



Gaussian 1/2

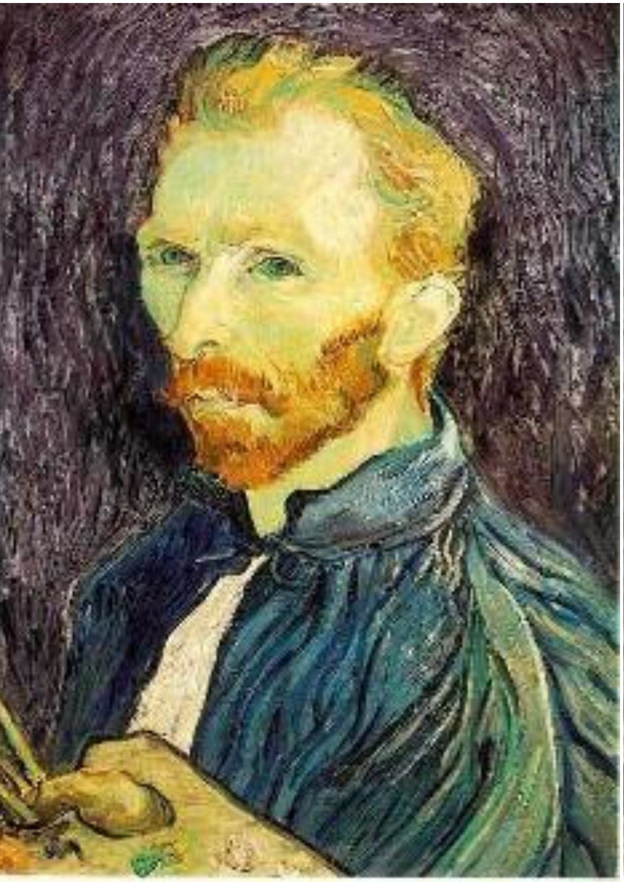
G 1/4

G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction.

Compare with...



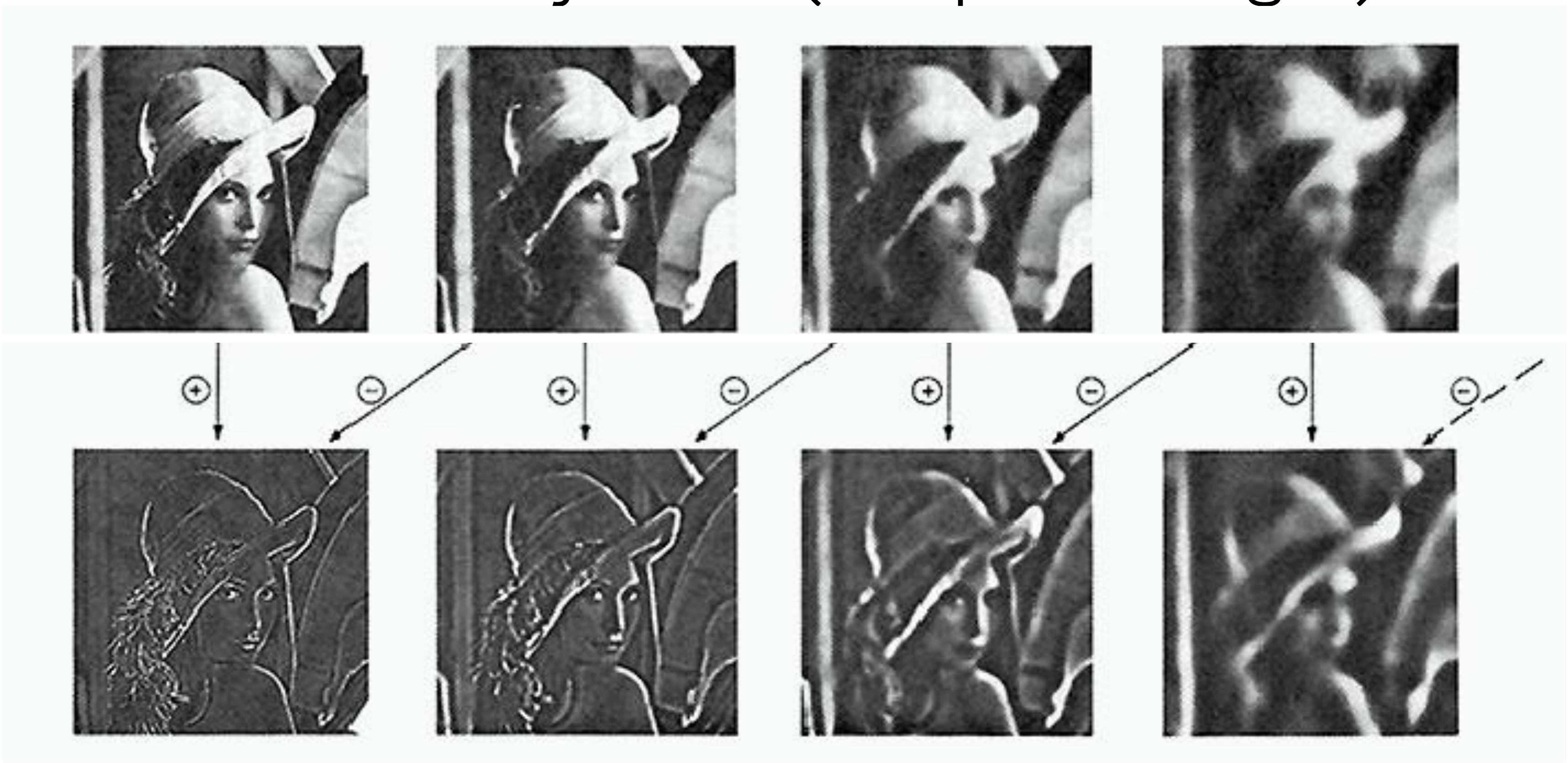
1/2

1/4 (2x zoom)

1/8 (4x zoom)

Band-pass filtering

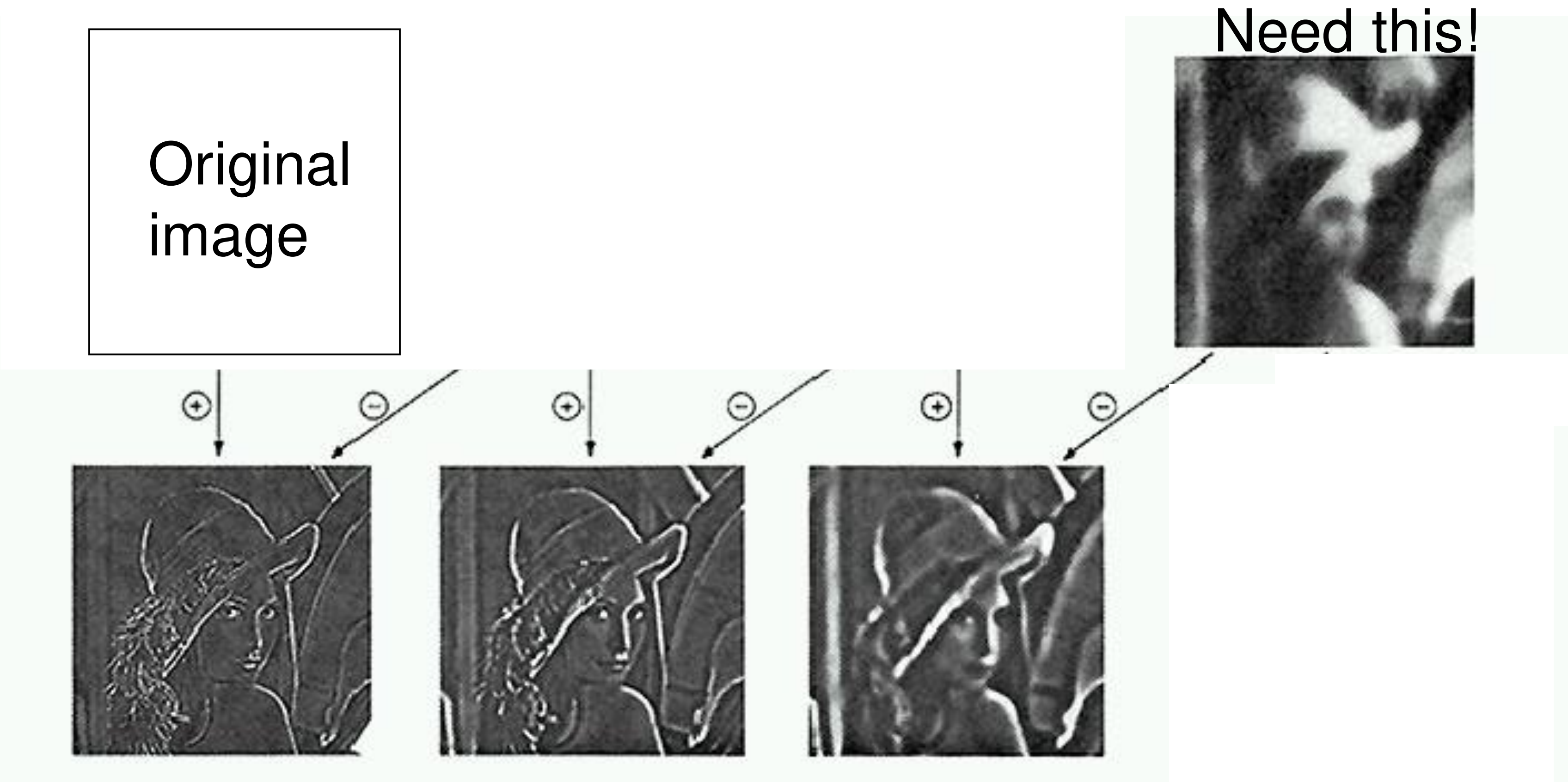
Gaussian Pyramid (low-pass images)



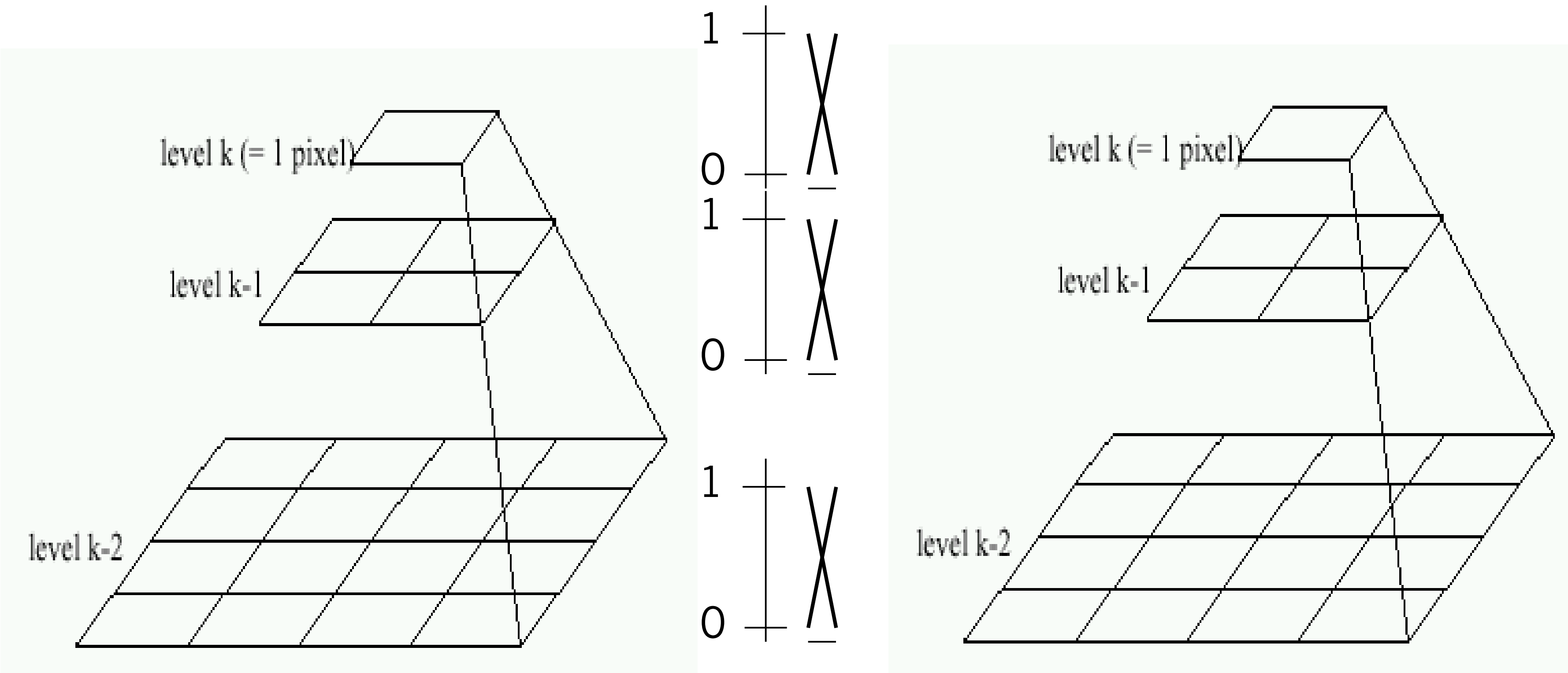
- Laplacian Pyramid (subband images)
- Created from Gaussian pyramid by subtraction

Laplacian Pyramid

- How can we reconstruct (collapse) this pyramid into the original image?

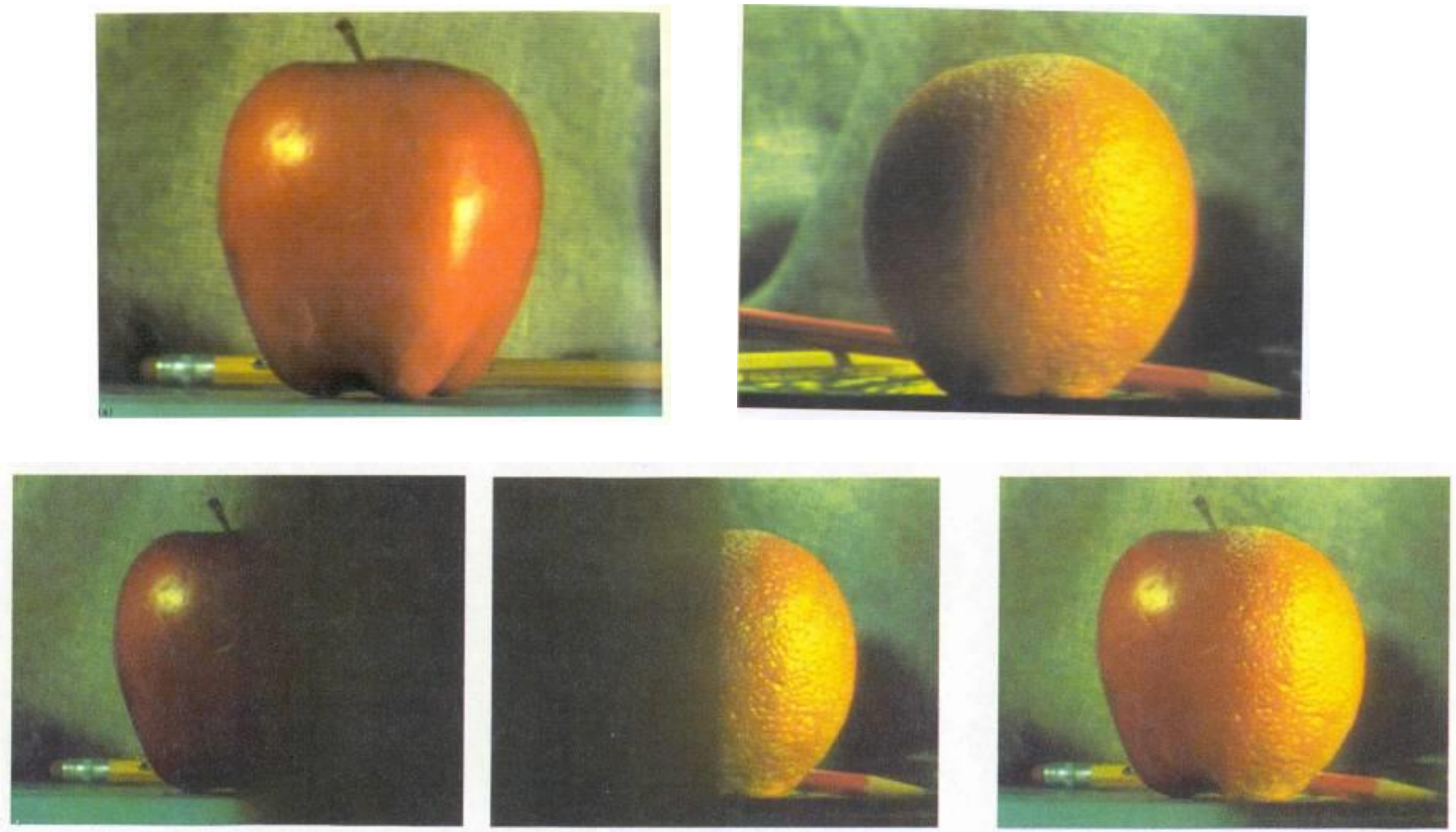


Pyramid Blending



Left pyramid blend Right pyramid

Pyramid Blending

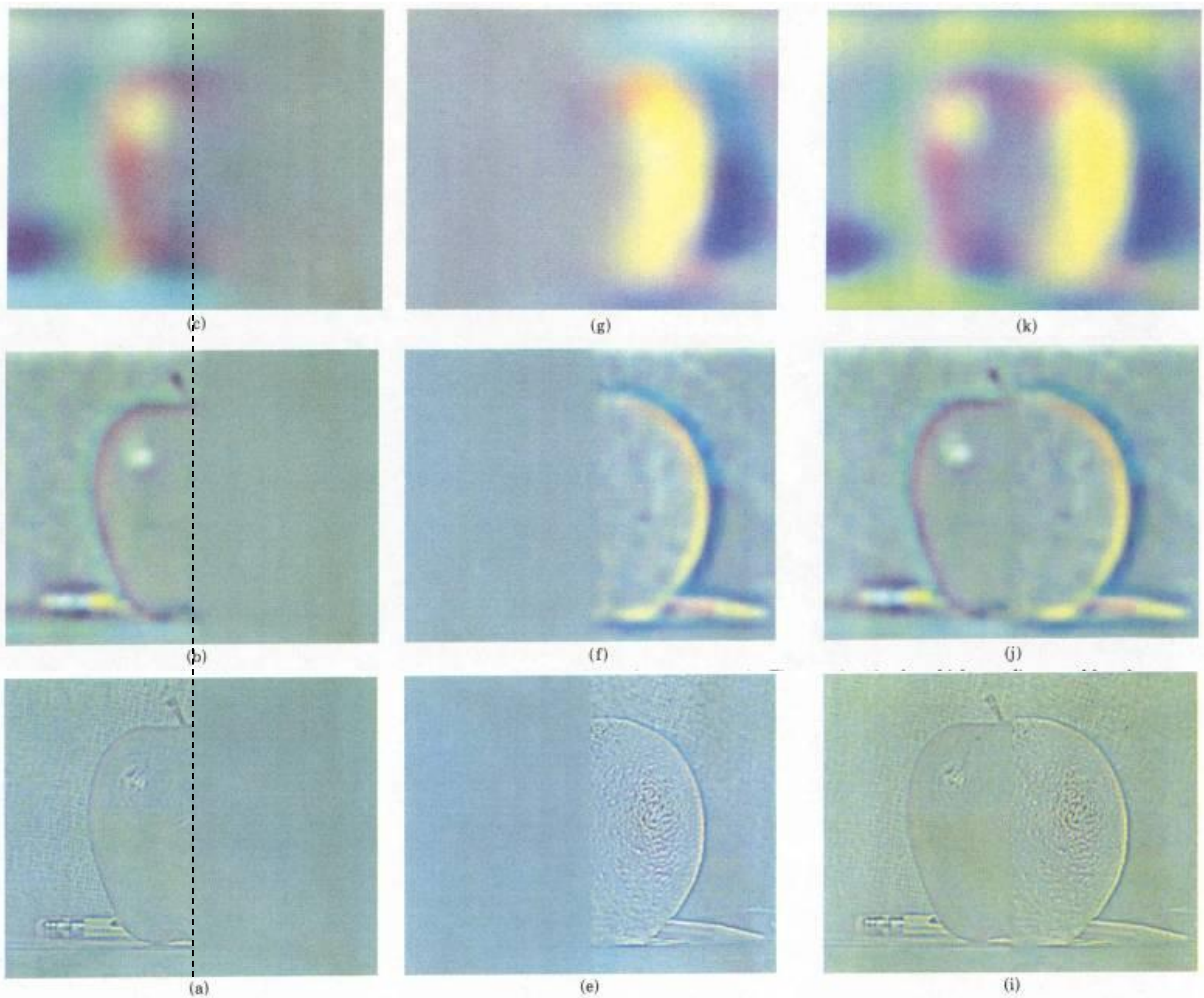


(d) (h) (l)

Laplacian level 4

Laplacian level 2

Laplacian level 0



left pyramid

right pyramid

blended pyramid

Simplification: Two-band Blending

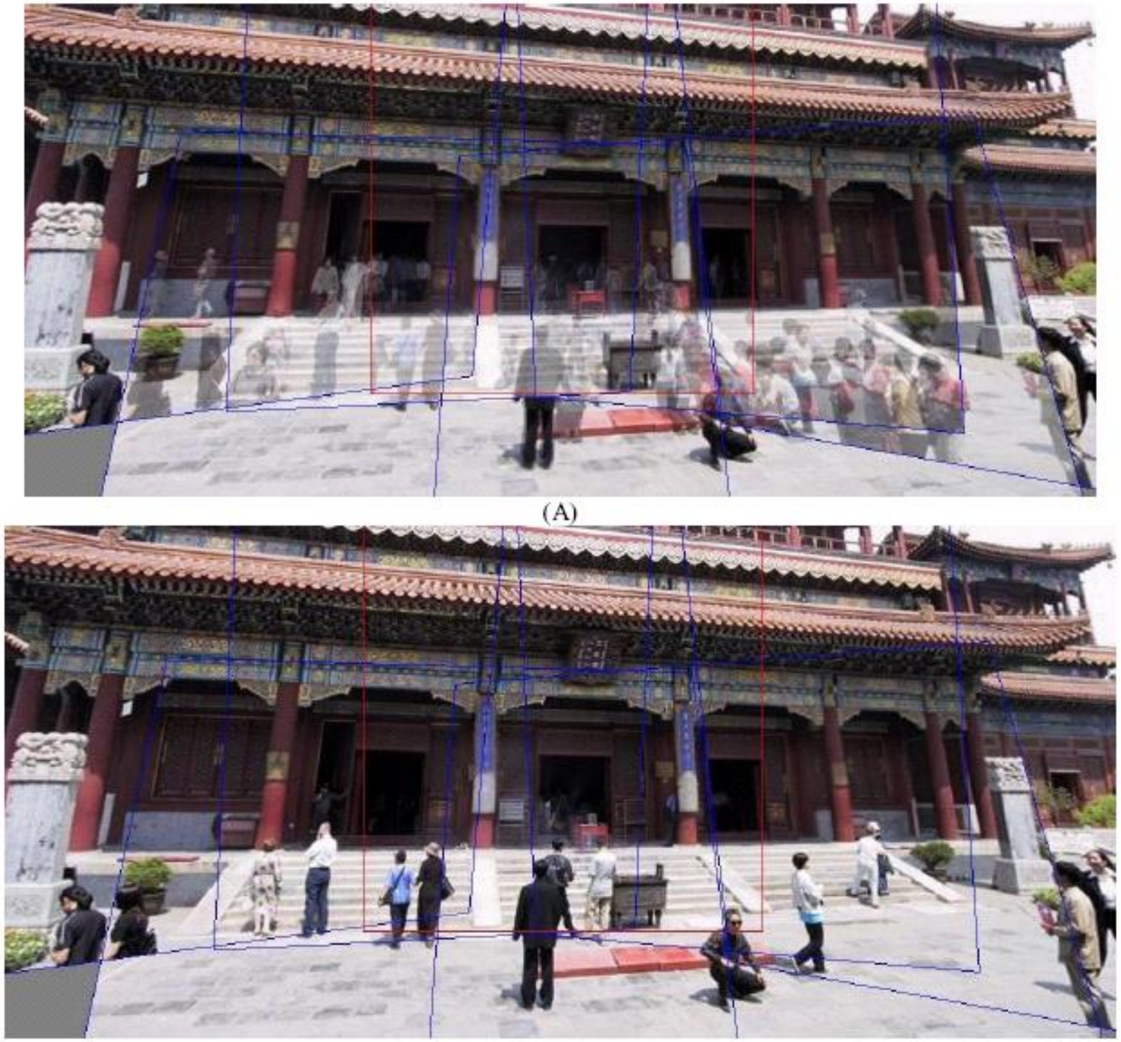
- Brown & Lowe, 2003
  - Only use two bands: high freq. and low freq.
  - Blends low freq. smoothly
  - Blend high freq. with no smoothing: use binary mask

Still Some Artifacts Left...

- Ghosting—objects move in the scene.
- Differing exposures between images.
    - Pyramid blending does not solve this.

De-Ghosting

•In regions with differences don't blend – choose one.



[Uyttendaele et al. 2001]

Gradient domain blending

- In pyramid blending, we decomposed our image into 2nd derivatives (Laplacian) and a low-res image
- Let us now look at 1st derivatives (gradients):
- No need for low-res image
  - captures everything (up to a constant)
  - easy to deal with low-frequency differences
- Idea:
  - Differentiate
  - Blend
  - Reintegrate

Gradient domain blending (2D)

- Trickier in 2D:
  - Take partial derivatives dx and dy (gradient field)
  - Modify them (smooth, blend, feather, etc)

  - Reintegrate
    - But now integral(dx) might not equal integral(dy)
  - Find the most agreeable solution
    - Equivalent to solving Poisson equation
    - Can use FFT, deconvolution, multigrid solvers, etc.

Gradient domain fusion

Compare [Agarwala 2004]

1. Given:
   a. intended gradient field: $\nabla_x I(x, y)$, $\nabla_y I(x, y)$
   b. boundary conditions: $I_{\text{fix}}(x, y)_j$
      fixed color at a couple of pixels $(x, y)_j$
2. Assemble gradient vector v and corresponding matrix A. Goal image g (as vector) $\rightarrow Ag = v$
   a. constraint vector

$$v = \begin{pmatrix} \nabla_x I \\ \nabla_y I \\ I_{\text{fix}} \end{pmatrix} \quad \begin{array}{l} \text{(given gradients)} \\ \\ \text{(given colors)} \end{array}$$

Gradient domain fusion

- 2. Assemble gradient vector v and corresponding matrix A. Goal image g (as vector) → $Ag = v$

    b. assemble matrix A according to

$$g(x+1, y) - g(x, y) = \nabla_x I(x, y)$$
$$g(x, y+1) - g(x, y) = \nabla_y I(x, y)$$

structure similar to $A \approx \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$ always one „+1“ and „-1“ per row

Gradient domain fusion

- 2. Assemble gradient vector v and corresponding matrix A. Goal image g (as vector) $Ag = v$
  c. Extend matrix A with boundary conditions
  $$g(x,y) = I_{\text{fix}}(x,y)$$

structure similar to

$$A \approx \left(\begin{array}{cccc} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right)$$

always one „+1“ per row

Gradient domain fusion

- 3. Solve for g: $Ag = v$
  - As gradients might be contradicting, use least-squares solution

$$A^T A g = A^T v \quad \text{(normal equation)}$$

$$\Rightarrow \quad g = (A^T A)^{-1} A^T v$$

- Implicitly done by QR decomposition, SVD, etc.
- Alternatively, use conjugate gradient methods

e.g. http://www.eecs.berkeley.edu/~demmel/cs267/lecture24/lecture24.html

Comparisons [Levin et al 2004]

Pyramid blending

Pyramid blending on the gradients

Feathering

More common: Poisson editing

- Use **Laplacian operator** $\Delta = \nabla^2$ instead of $\nabla_{x,y}$
- Discrete Laplacian: convolution with 2D kernel $\begin{bmatrix} & -1 & \\ -1 & +4 & -1 \\ & -1 & \end{bmatrix}$
- This leads to convolution matrix:

$$M_\Delta = \begin{bmatrix} \begin{bmatrix} +4 & -1 & \\ -1 & \ddots & -1 \\ & -1 & +4 \end{bmatrix} & \begin{bmatrix} -1 & & \\ & \ddots & \\ & & -1 \end{bmatrix} & \\ \begin{bmatrix} -1 & & \\ & \ddots & \\ & & -1 \end{bmatrix} & \ddots & \begin{bmatrix} -1 & & \\ & \ddots & \\ & & -1 \end{bmatrix} \\ & \begin{bmatrix} -1 & & \\ & \ddots & \\ & & -1 \end{bmatrix} & \begin{bmatrix} +4 & -1 & \\ -1 & \ddots & -1 \\ & -1 & +4 \end{bmatrix} \end{bmatrix}$$

read, “output pixel $(i,j) = 4 \cdot$ input pixel $(i,j) - \sum$ input pixels $(i \pm 1, j \pm 1)$”

For a $w \times h$ image, M has $w \times w$ blocks of $h \times h$ elements each
e.g., 307200×307200 matrix for a 640×480 image.
Use sparse matrices, or die of memory exhaustion

Poisson editing continued

- Construct M_Δ
- Compute $v_{1,2} = M_\Delta I_{1,2}$
- Blend $v_{1,2}$ using method of choice to obtain combined v
- Reconstruct I by solving $M_\Delta I = v$

Add constraints as before:

- Constraint matrix M_c with 1 entry per row
- Constraint vector v_c with prescribed pixel values

- Solve $\begin{bmatrix} M_\Delta \\ M_\mathrm{c} \end{bmatrix} I = \begin{bmatrix} v \\ v_\mathrm{c} \end{bmatrix}$ in least-squares sense

Panoramas and Image blending

- A multiresolution spline with application to image mosaics
  P. J. Burt, E. H. Adelson.
  ACM Transactions on Graphics. *2(4)*, pp. 217-236, 1983.

- Recognising Panoramas. M. Brown and D. G. Lowe. In Proceedings of the *9th International Conference on Computer Vision (ICCV2003)*, pages 1218-1225, Nice, France, 2003.

- Multiple View Geometry in Computer Vision (First Edition). Richard Hartley and Andrew Zisserman, Cambridge University Press, March 2001

- Seamless Image Stitching in the Gradient Domain. A. Levin, A. Zomet, S. Peleg and Y. Weiss, In Proc. ECCV 2004.

- Interactive Digital Photomontage. A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, *M. Cohen, In SIGGRAPH 2004.*