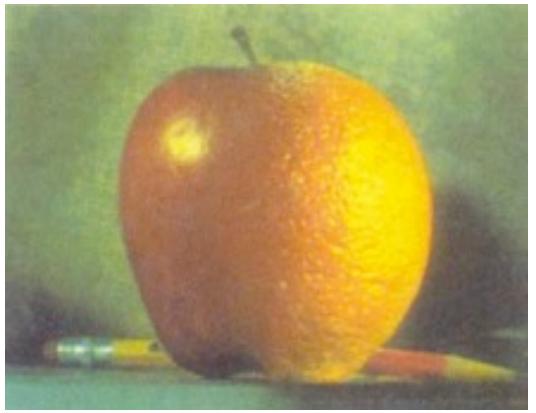
Interpolation & Optimization

Computer Vision (UW EE/CSE 576)

Richard Szeliski

Facebook & UW

Lecture 5 – Apr 14, 2020



Computer Vision (EE/CSE 576)

Instr.s

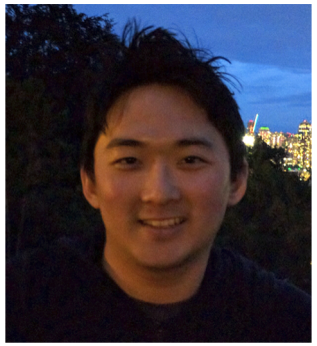
Steve Seitz

Rick Szeliski

Harpreet Sawhney

TAs

Aleksander Holynski



Keunhong Park

Svet Kolev

Slides are online here (calendar tab):

http://www.cs.washington.edu/education/courses/cse576/20sp/

Rick's book

Computer Vision: Algorithms and Applications, 2nd ed.

© 2020 Richard Szeliski, Facebook

http://szeliski.org/Book/2ndEdition.htm

Class calendar

Date	Lecture	Reading	Homework
March 31	Introduction	Szeliski, Chapter 1	
April 2	Human Vision, Color Spaces and Transforms	Brown, M. S. (2019). ICCV 2019 tutorial on understanding color and the in-camera image processing pipeline for computer vision.	HW1 assigned
April 7	Image coordinates, resizing	Szeliski, Chapter 2.1, 3.6	
April 9	Filters and convolutions	Szeliski, Chapter 3	HW1 due, HW2 assigned
April 14	Interpolation and Optimization	Szeliski, Chapter 4	
April 16	Machine Learning	Szeliski, Chapter 5.1-5.2	

Readings

Chapter 3

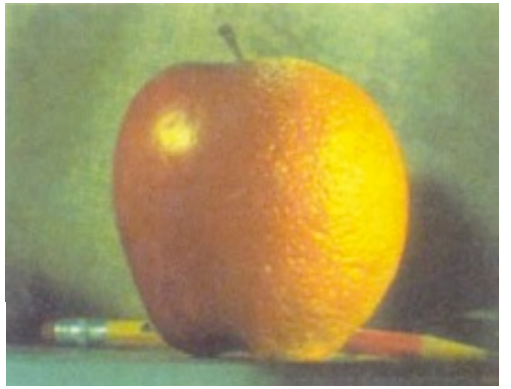
Image processing

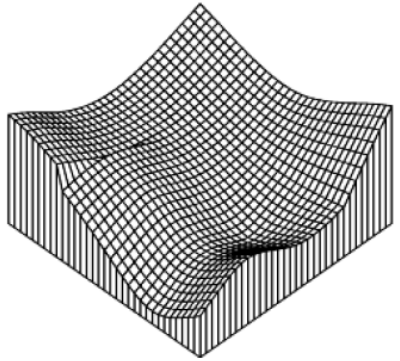
Chapter 4

Model fitting and optimization

Interpolation and optimization

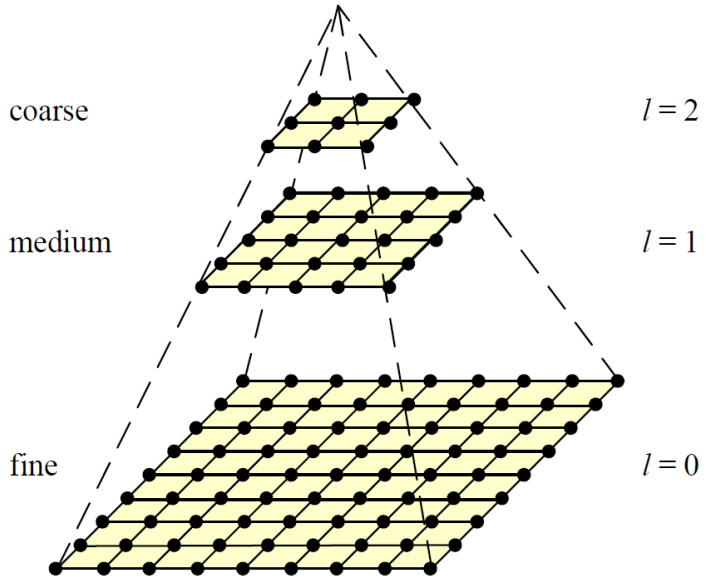
- Interpolation
- Pyramids
- Blending
- Resampling (rotations, etc.)
- Data Fitting
- Regularization and variational techniques
- Markov Random Fields



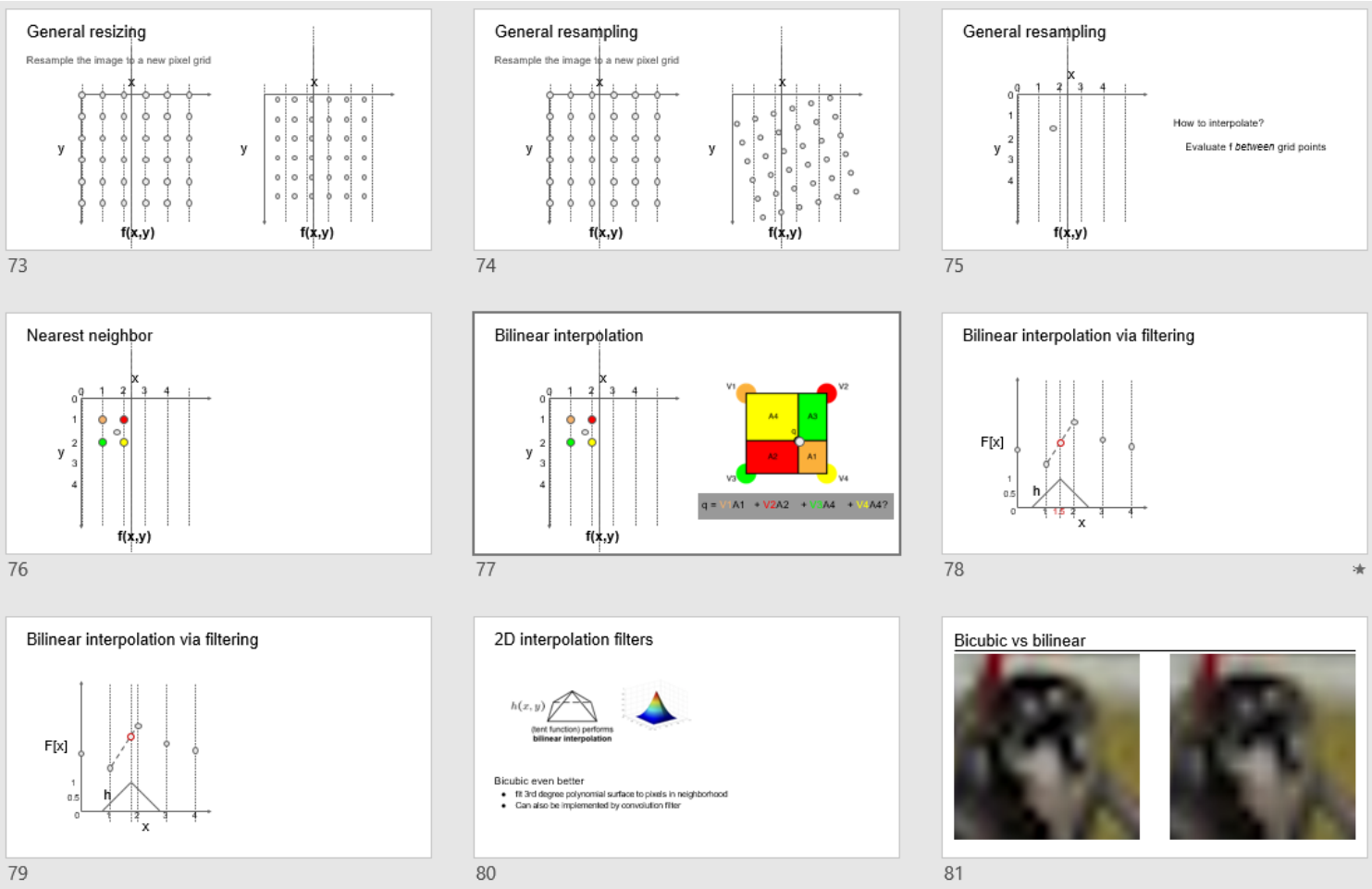


Pyramids and wavelets

- Interpolation: scaling up
- Decimation: scaling down
- Image pyramids
- (Invertible) difference pyramids
- ~~Wavelets~~



Recall from previous lecture



Interpolation

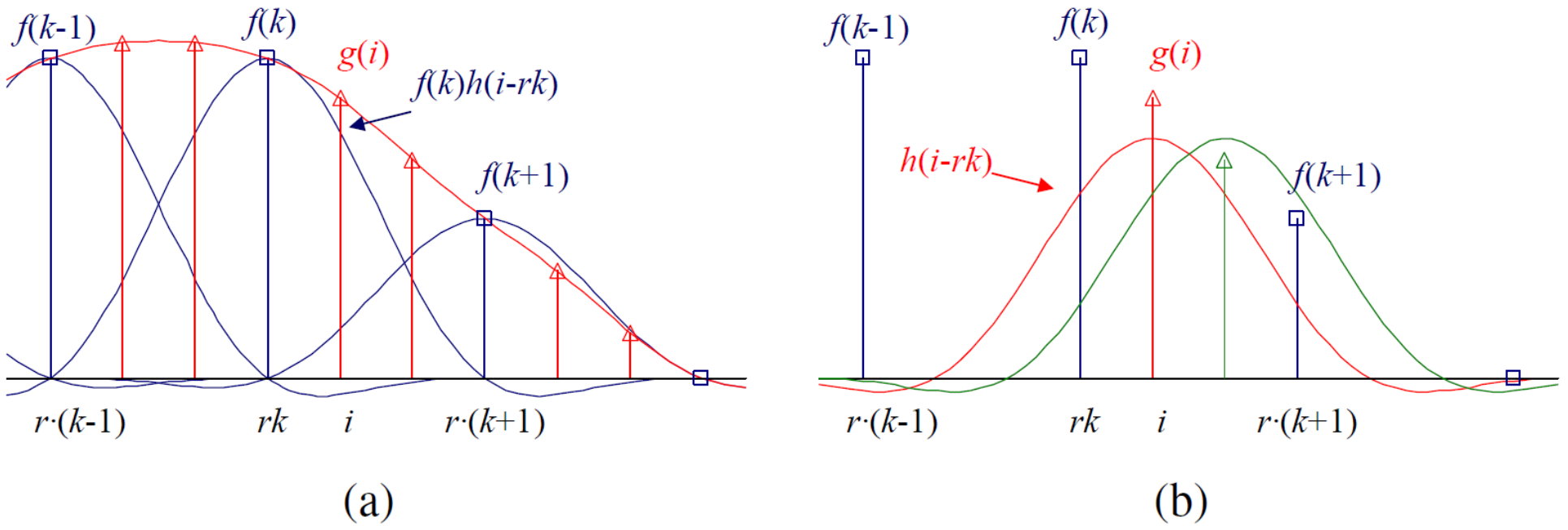


Figure 3.27 Signal interpolation, $g(i) = \sum_k f(k)h(i - rk)$: (a) weighted summation of input values; (b) polyphase filter interpretation.

Interpolation – bicubic, windowed sinc

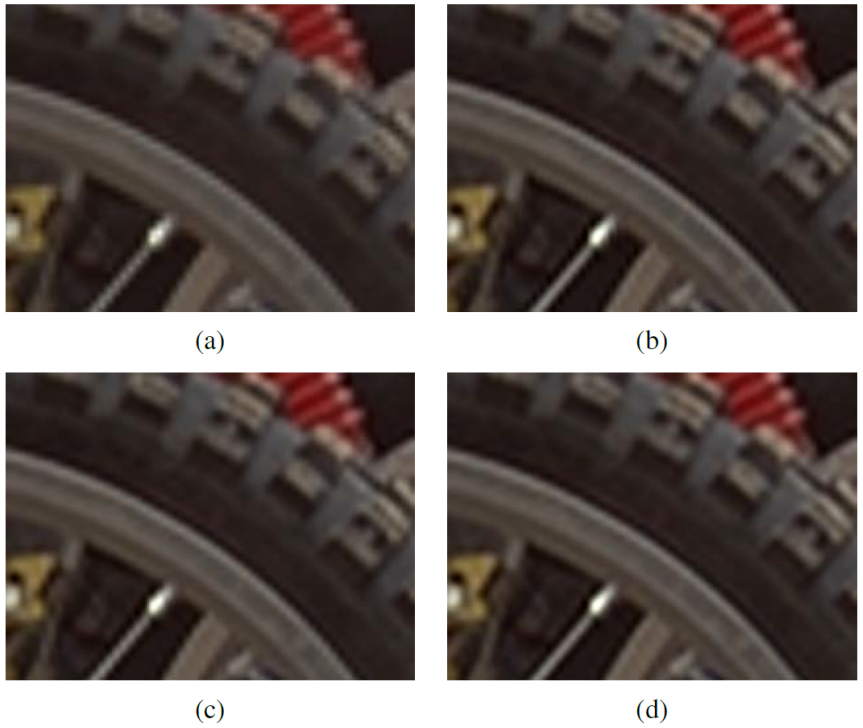


Figure 3.28 Two-dimensional image interpolation: (a) bilinear; (b) bicubic ($a = -1$); (c) bicubic ($a = -0.5$); (d) windowed sinc (nine taps).

More advanced topic: *super-resolution*
(Computational Photography lecture)

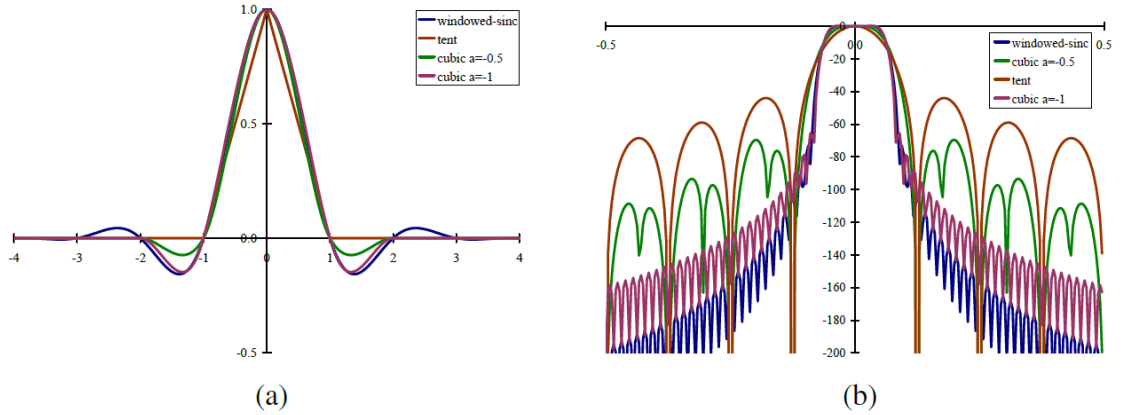


Figure 3.29 (a) Some windowed sinc functions and (b) their log Fourier transforms: raised-cosine windowed sinc in blue, cubic interpolators ($a = -1$ and $a = -0.5$) in green and purple, and tent function in brown. They are often used to perform high-accuracy low-pass filtering operations.

While most graphics cards use the bilinear kernel (optionally combined with a MIP-map—see Section 3.5.3), most photo editing packages use *bicubic* interpolation. The cubic interpolant is a C^1 (derivative-continuous) piecewise-cubic *spline* (the term "spline" is synonymous with "piecewise-polynomial")[14] whose equation is

$$h(x) = \begin{cases} 1 - (a+3)x^2 + (a+2)|x|^3 & \text{if } |x| < 1 \\ a(|x|-1)(|x|-2)^2 & \text{if } 1 \le |x| < 2 \\ 0 & \text{otherwise,} \end{cases} \tag{3.79}$$

where a specifies the derivative at $x = 1$ (Parker, Kenyon, and Troxel 1983). The value of

Decimation (downsampling)

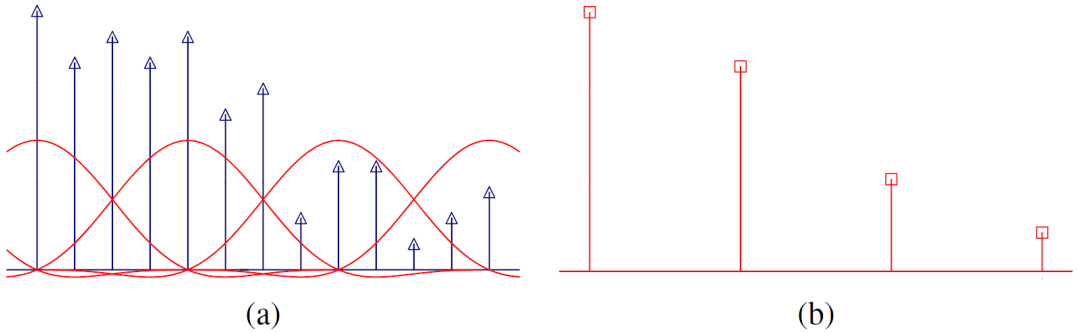


Figure 3.30 Signal decimation: (a) the original samples are (b) convolved with a low-pass filter before being downsampled.

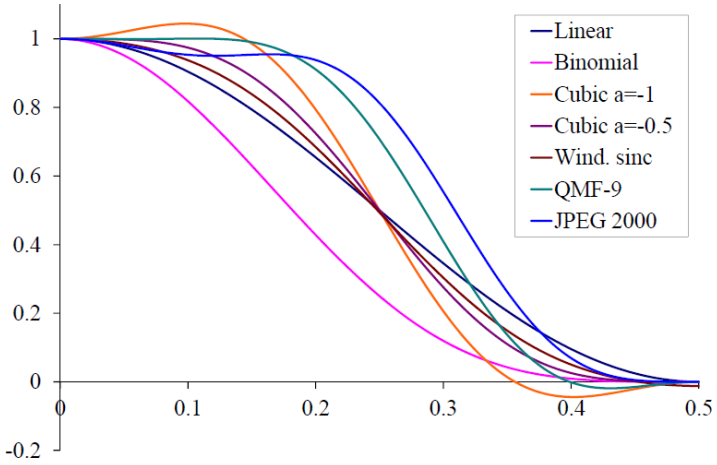


Figure 3.31 Frequency response for some $2\times$ decimation filters. The cubic $a = -1$ filter has the sharpest fall-off but also a bit of ringing; the wavelet analysis filters (QMF-9 and JPEG 2000), while useful for compression, have more aliasing.

Multi-resolution image pyramids

- Commonly used in coarse-to-fine matching, optical flow, stereo, blending, …
- … deep neural networks

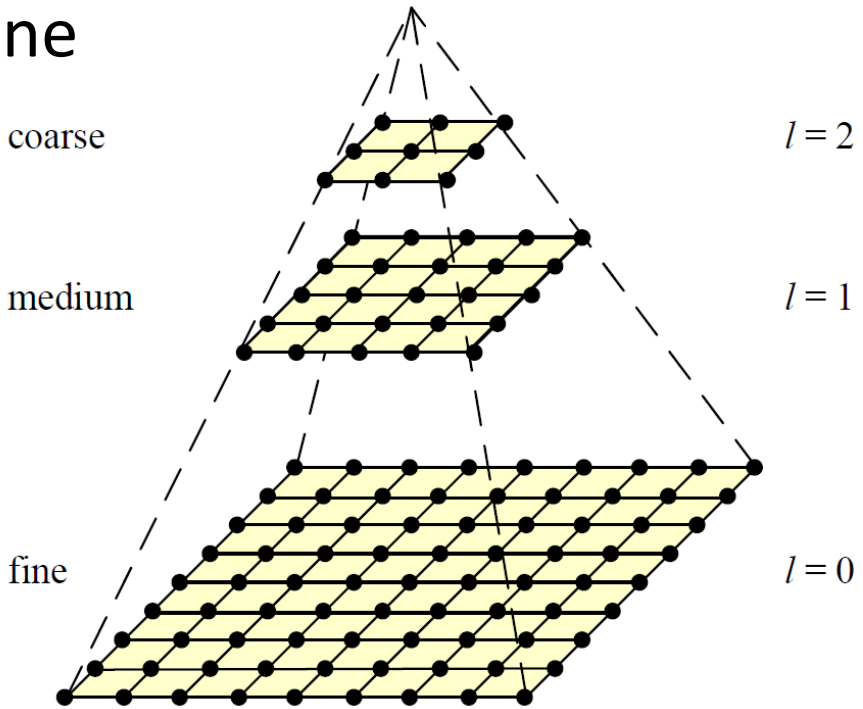


Figure 3.32 A traditional image pyramid: each level has half the resolution (width and height), and hence a quarter of the pixels, of its parent level.

“Gaussian” pyramid

- Uses “binomial” (not real Gaussian) downsampling kernel

$\frac{1}{256}$

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

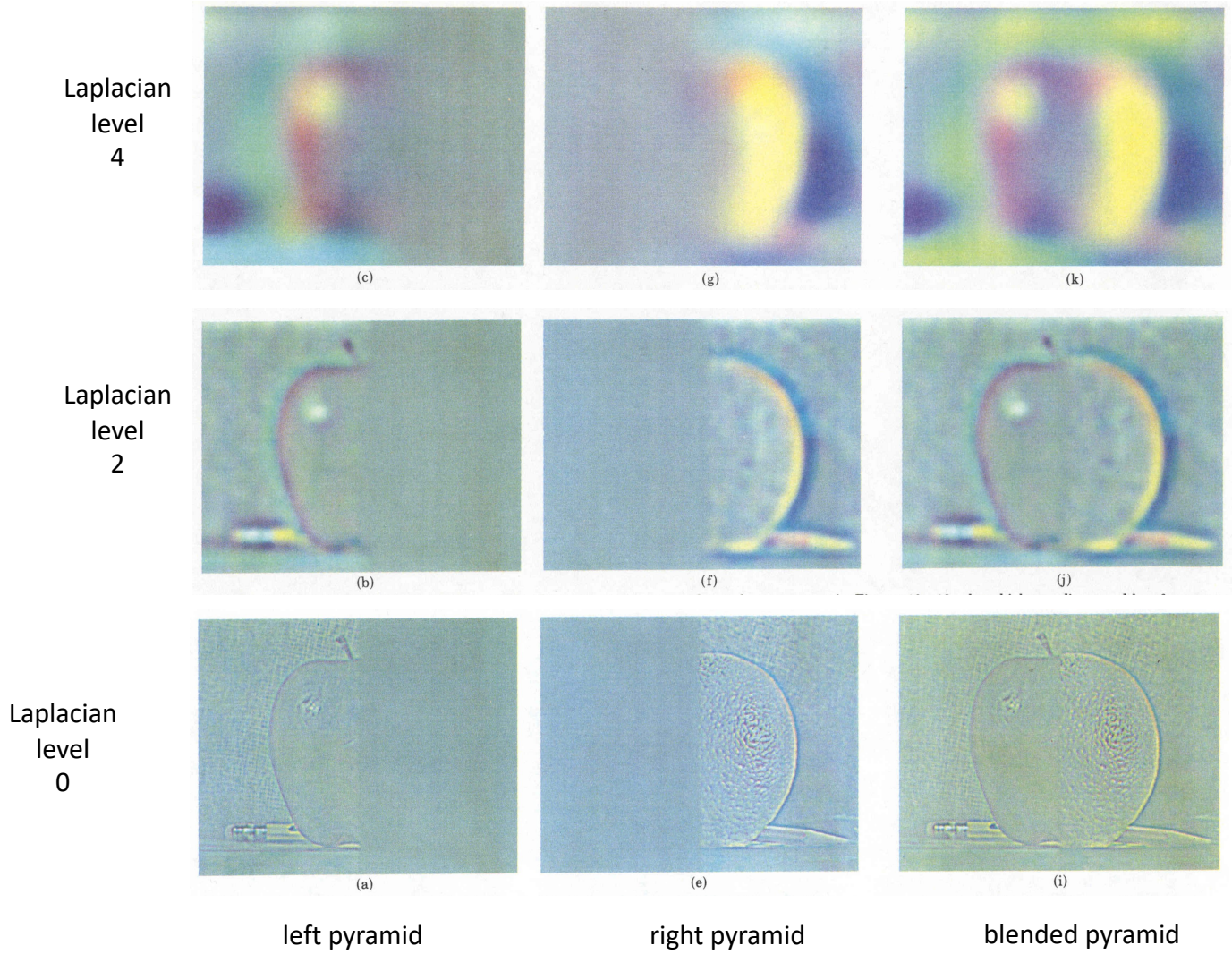
$\frac{1}{16}$

1	4	6	4	1

(c) “Gaussian”

“Laplacian” pyramid

- Burt & Adelson’s “Laplacian” is a difference of (interpolated) “Gaussians”



Laplacian pyramid coding (compression)

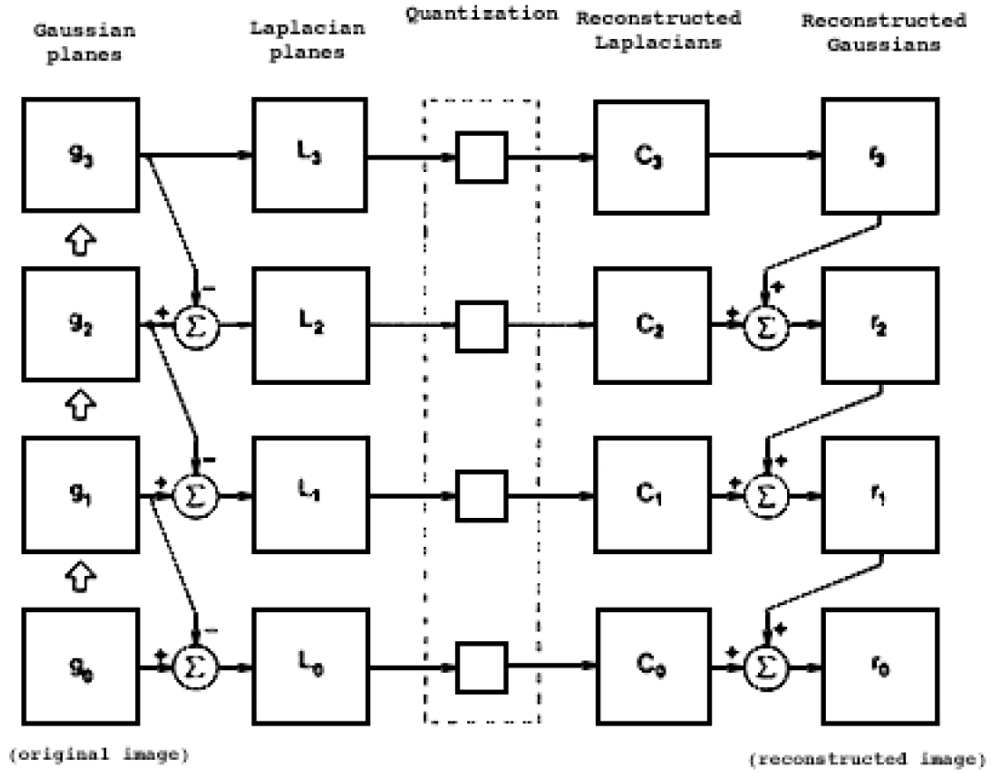


Fig. 10. A summary of the steps in Laplacian pyramid coding and decoding. First, the original image g_0 (lower left) is used to generate Gaussian pyramid levels g_1, g_2, … through repeated local averaging. Levels of the Laplacian pyramid L_0, L_1, … are then computed as the differences between adjacent Gaussian levels. Laplacian pyramid elements are quantized to yield the Laplacian pyramid code C_0, C_1, C_2, …. Finally, a reconstructed image r_0 is generated by summing levels of the code pyramid.

Laplacian pyramid

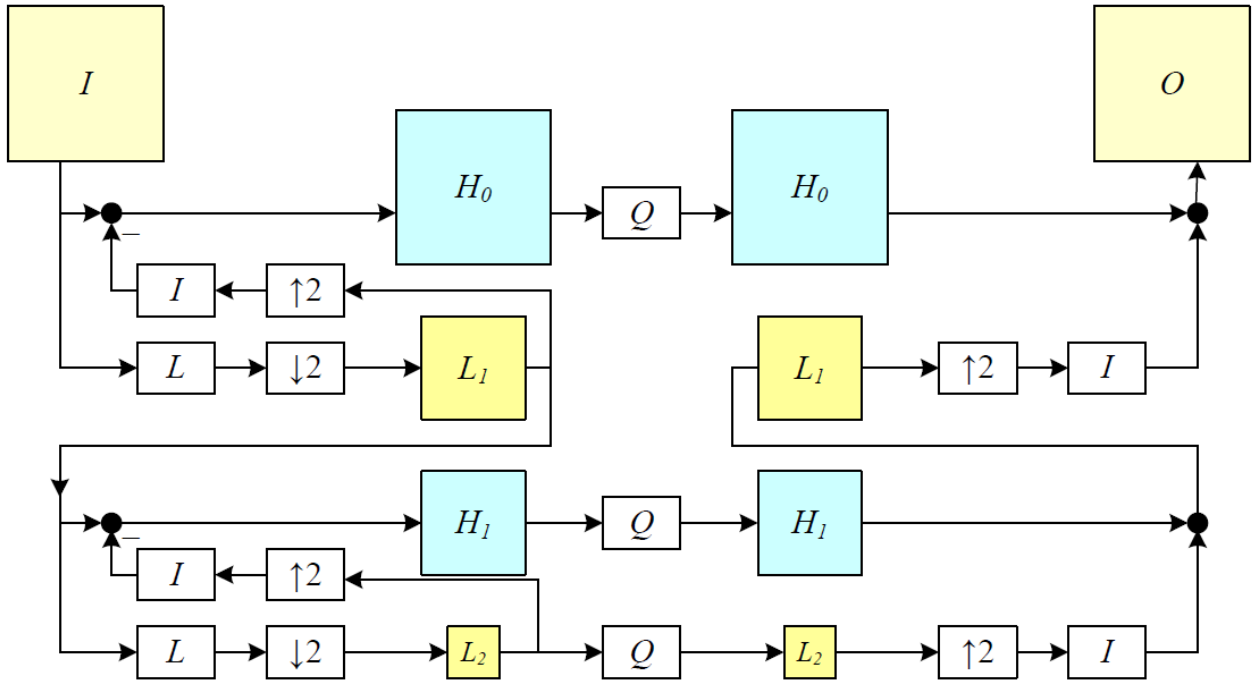


Figure 3.33 *The Laplacian pyramid. The yellow images form the* Gaussian *pyramid, which is obtained by successively low-pass filtering and downsampling the input image. The blue images, together with the smallest low-pass image, which is needed for reconstruction, form the* Laplacian *pyramid. Each band-pass (blue) image is computed by upsampling and interpolating the lower-resolution Gaussian pyramid image, resulting in a blurred version of that level's low-pass image, which is subtracted from the low-pass to yield the blue band-pass*

Laplacian pyramid blending

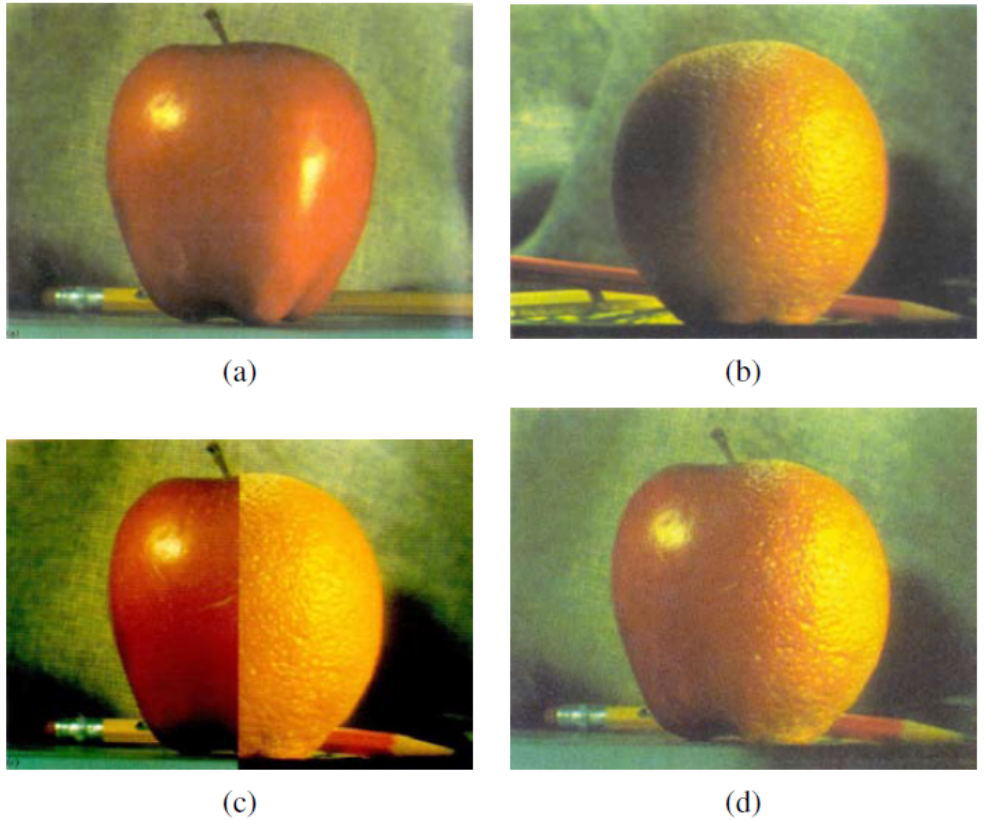
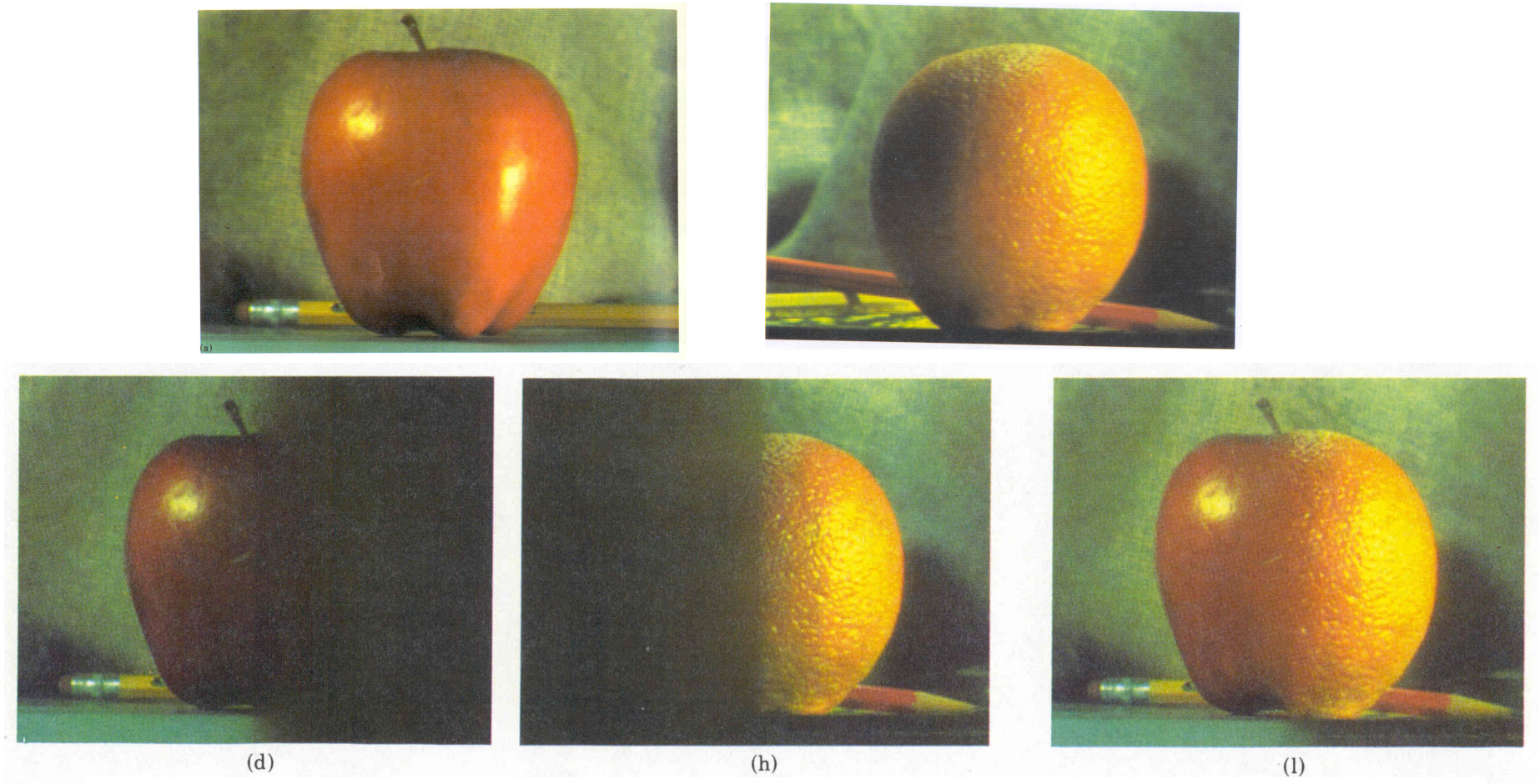


Figure 3.41 Laplacian pyramid blending (Burt and Adelson 1983b) © 1983 ACM: (a) original image of apple, (b) original image of orange, (c) regular splice, (d) pyramid blend.

Pyramid Blending

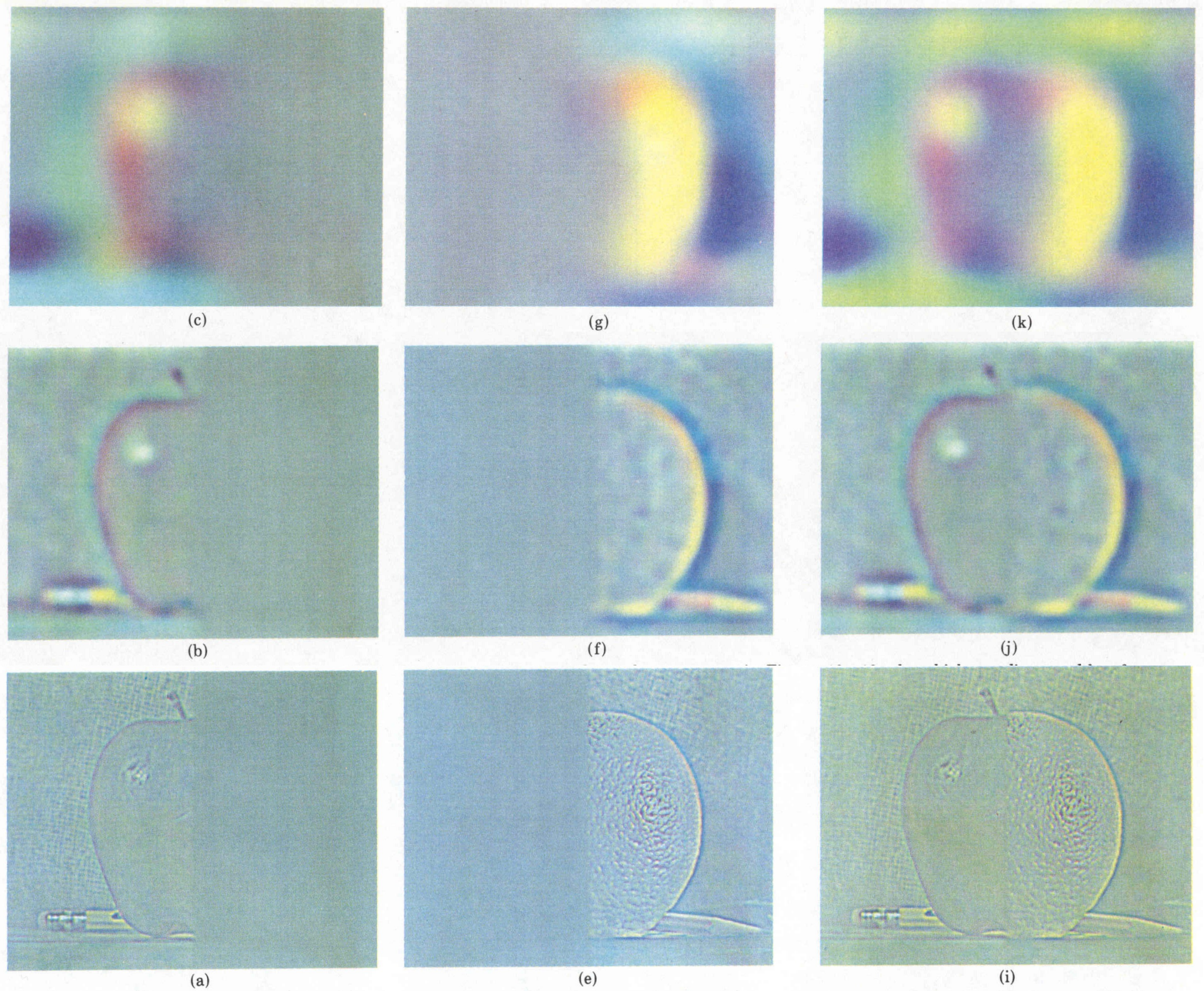


Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

Laplacian level 4

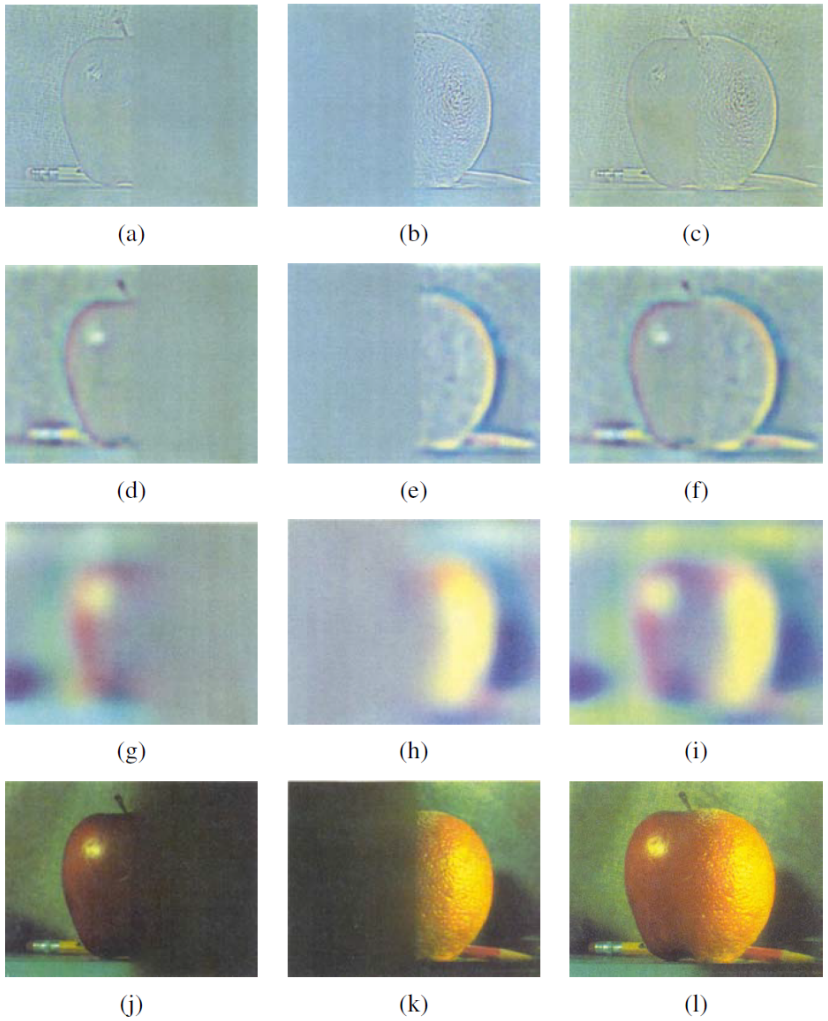
Laplacian level 2

Laplacian level 0



left pyramid right pyramid blended pyramid

Laplacian pyramid blending



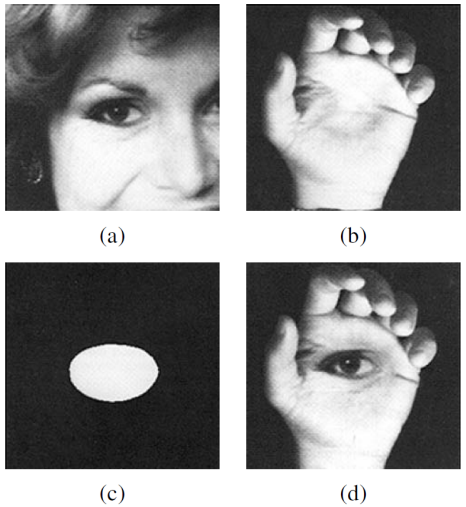
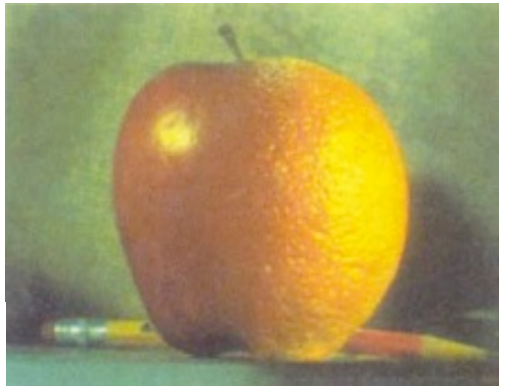


Figure 3.43 Laplacian pyramid blend of two images of arbitrary shape (Burt and Adelson 1983b) © 1983 ACM: (a) first input image; (b) second input image; (c) region mask; (d) blended image.

Interpolation and optimization

- Interpolation
- Pyramids
- Blending
- Resampling (rotations, etc.)
- Data Fitting
- Regularization and variational techniques
- Markov Random Fields



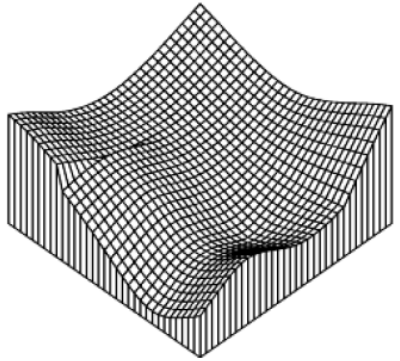
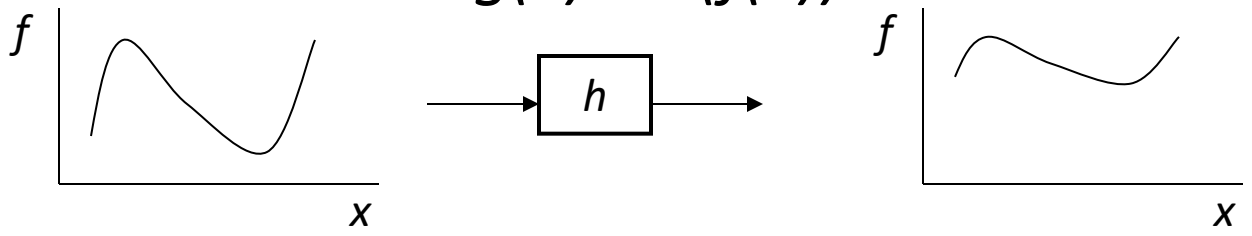


Image warping

Geometric Transforms

Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$



- image warping: change *domain* of image
  - $g(x) = f(h(x))$

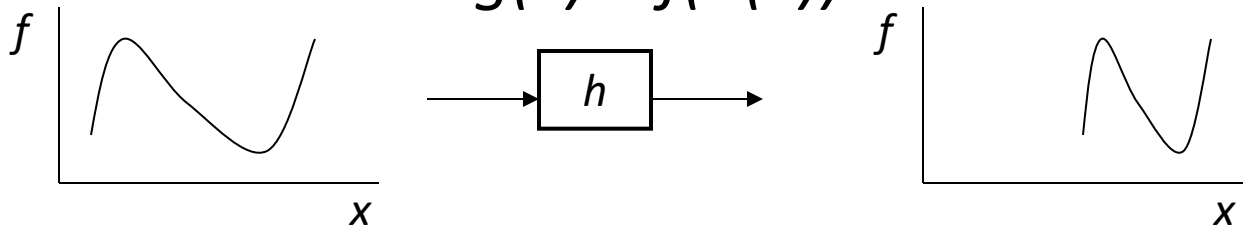
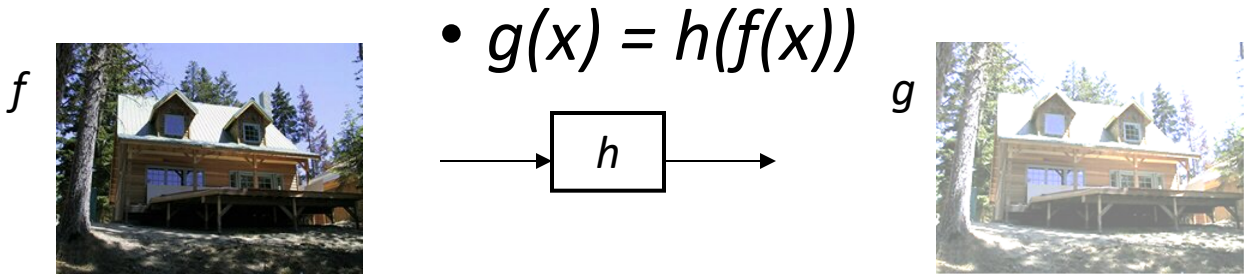
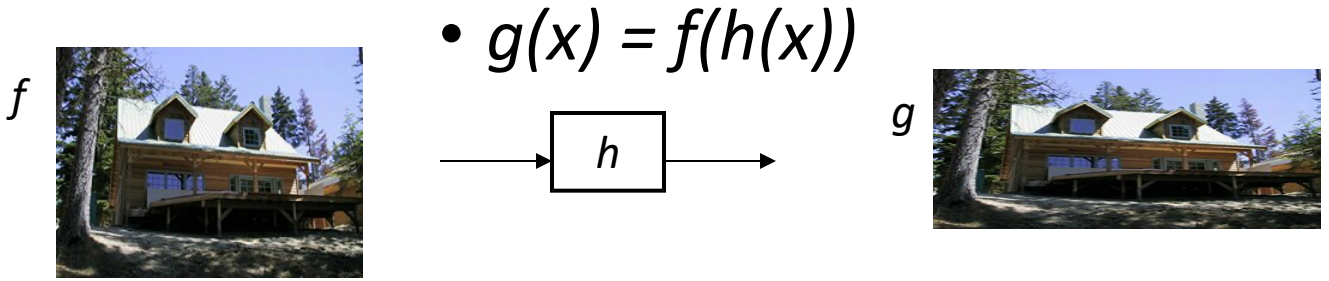


Image Warping

- image filtering: change *range* of image



- image warping: change *domain* of image



Parametric (global) warping

- Examples of parametric warps:

translation

rotation

aspect

affine

perspective

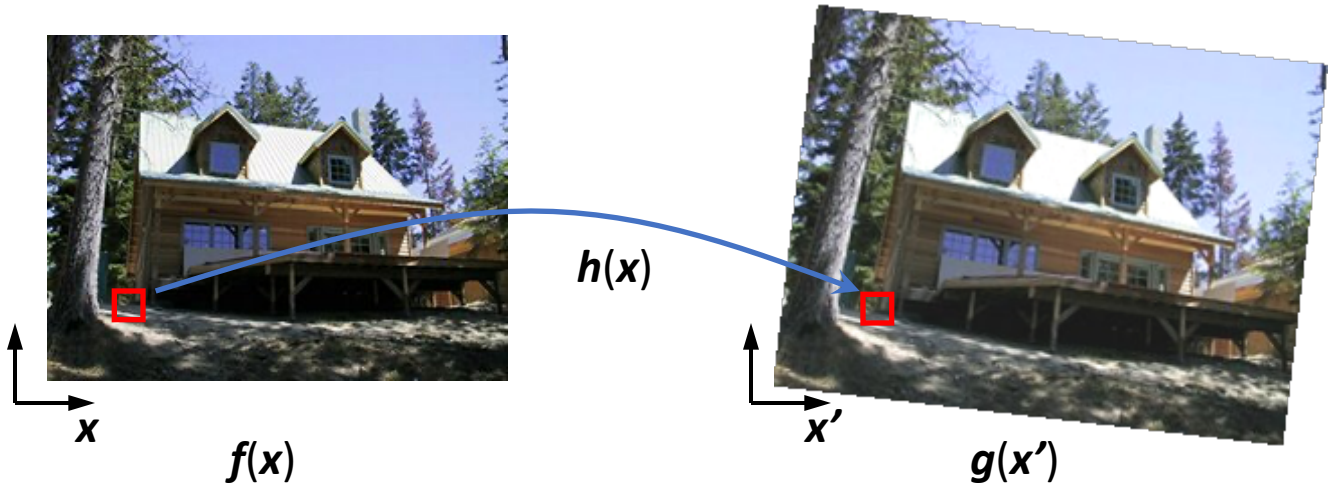
cylindrical

2D coordinate transformations

- translation: $\boldsymbol{x'} = \boldsymbol{x} + \boldsymbol{t}$ $\boldsymbol{x} = (x,y)$
- rotation: $\boldsymbol{x'} = \boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- similarity: $\boldsymbol{x'} = s\,\boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- affine: $\boldsymbol{x'} = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{t}$
- perspective: $\underline{\boldsymbol{x}}' \cong \boldsymbol{H}\,\underline{\boldsymbol{x}}$ $\underline{\boldsymbol{x}} = (x,y,1)$
  ($\underline{\boldsymbol{x}}$ is a *homogeneous* coordinate)
- These all form a nested *group* (closed w.r.t. inversion)
- *How many parameters for each one?*

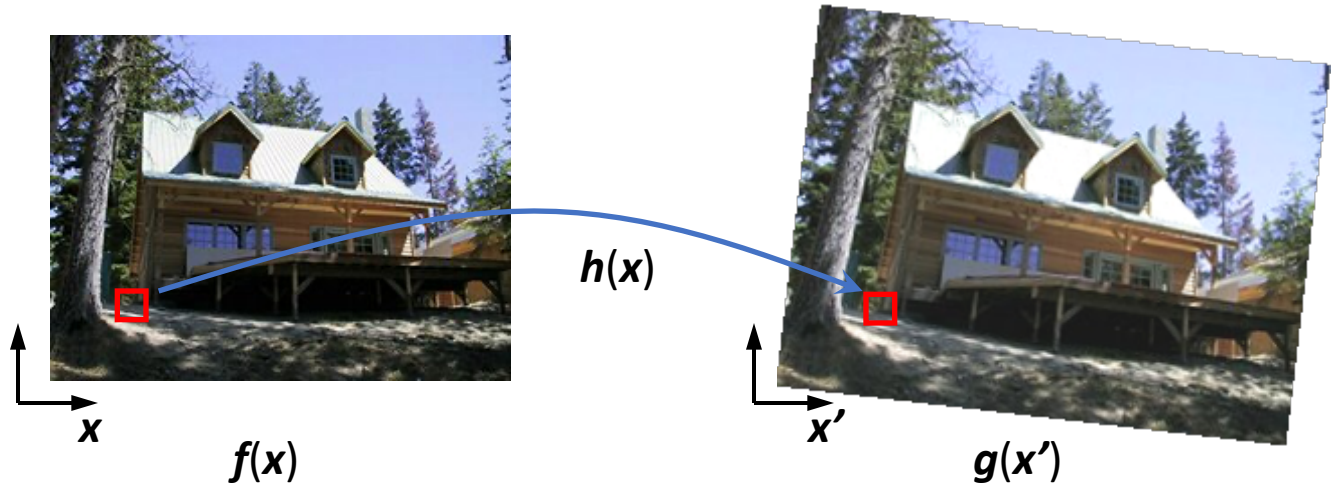
Image Warping

- Given a coordinate transform $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ and a source image $\boldsymbol{f}(\boldsymbol{x})$, how do we compute a transformed image $\boldsymbol{g}(\boldsymbol{x'}) = \boldsymbol{f}(\boldsymbol{h}(\boldsymbol{x}))$?



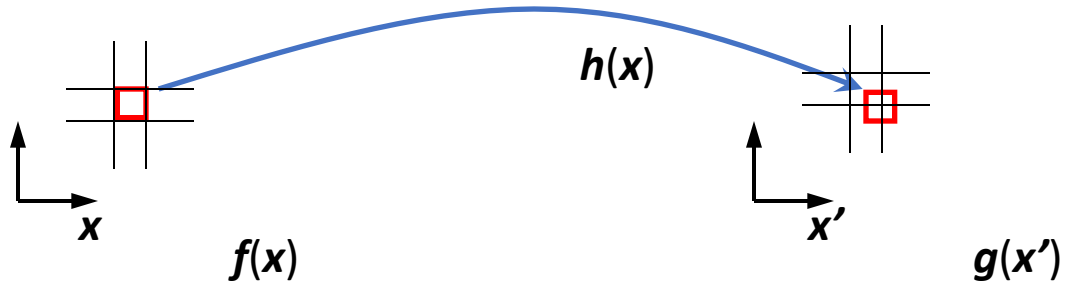
Forward Warping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
  - What if pixel lands "between" two pixels?



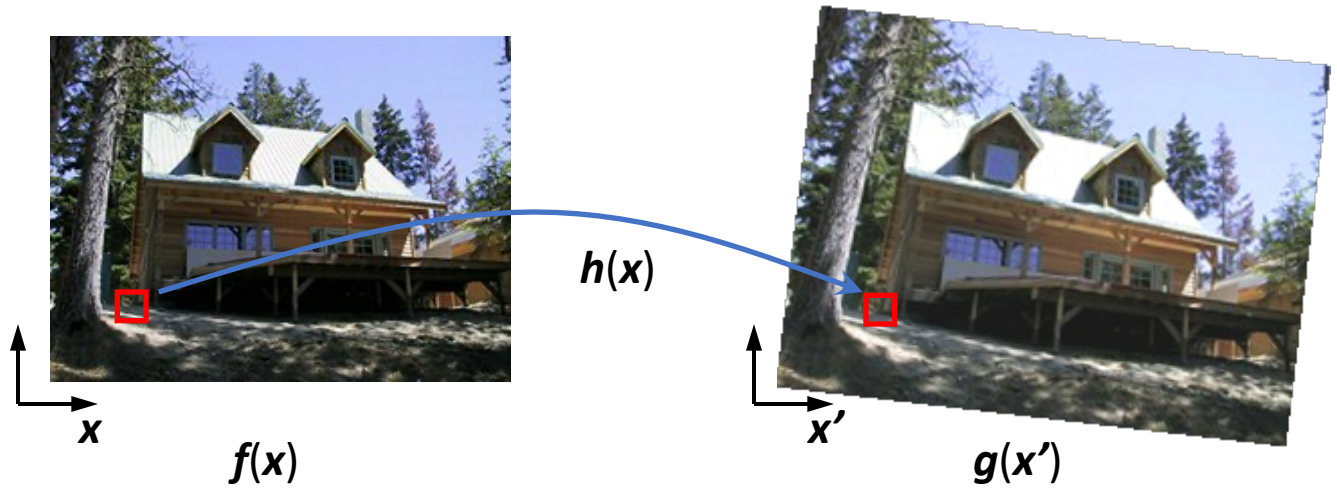
Forward Warping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)



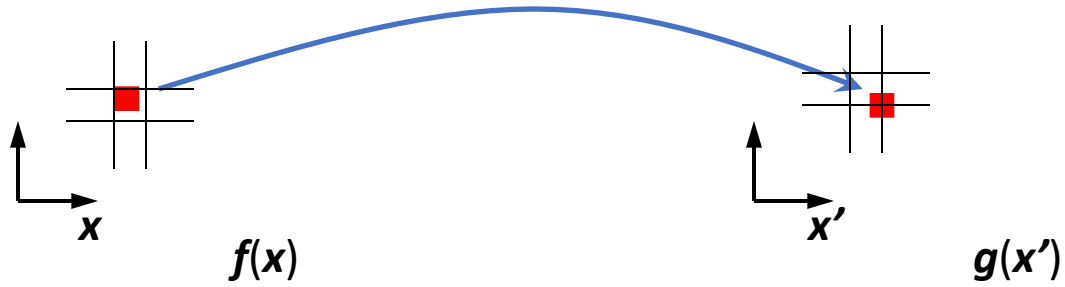
Inverse Warping

- Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$
  - What if pixel comes from "between" two pixels?



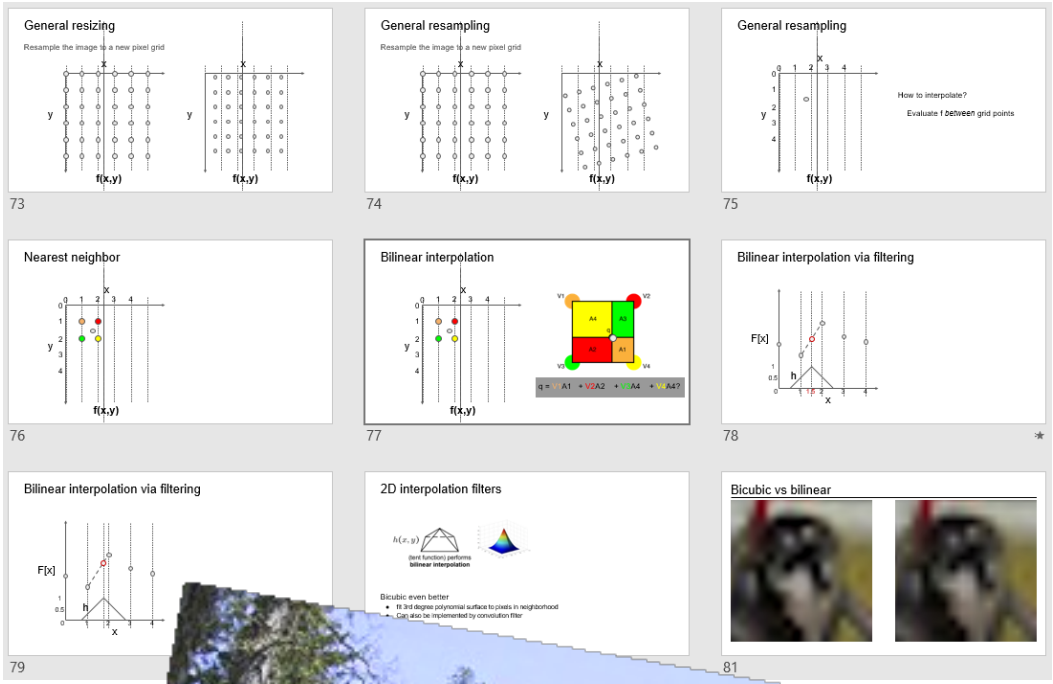
Inverse Warping

- Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image



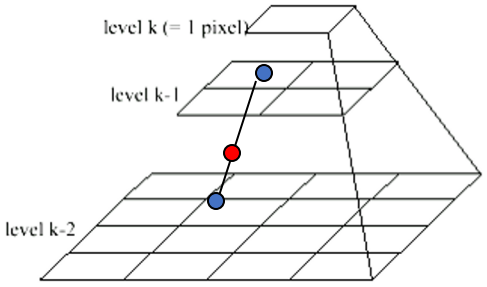
Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)
  - sinc / FIR (finite impulse resp.)
- Needed to prevent “jaggies” and “texture crawl”



Prefiltering

- Essential for *downsampling* (*decimation*) to prevent *aliasing*
- MIP-mapping [Williams'83]:
  1. build pyramid (but what decimation filter?):
     - block averaging
     - Burt & Adelson (5-tap binomial)
     - 7-tap wavelet-based filter (better)
  2. *trilinear* interpolation
     - bilinear within each 2 adjacent levels
     - linear blend *between* levels (determined by pixel size)



General mesh warping

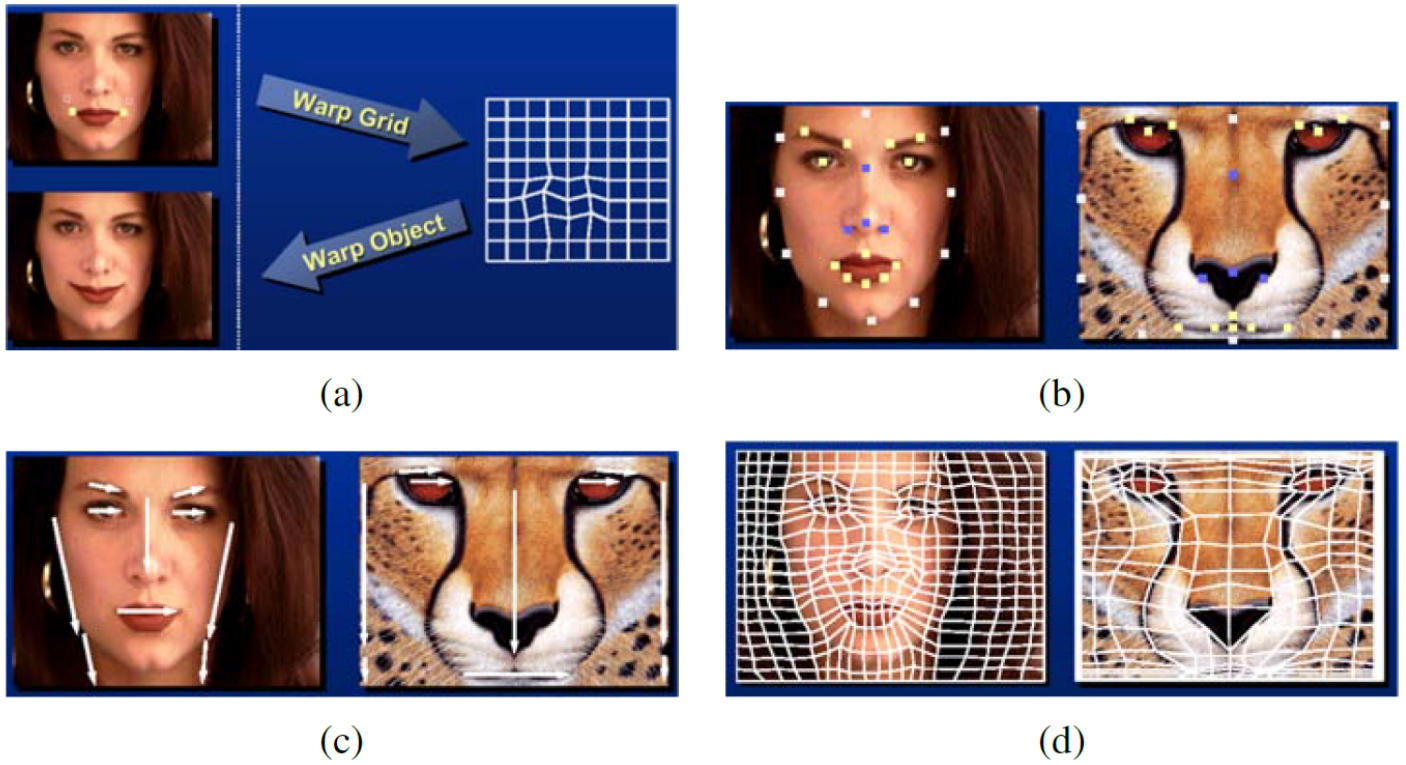


Figure 3.51 Image warping alternatives (Gomes, Darsa, Costa *et al.* 1999) © 1999 Morgan Kaufmann: (a) sparse control points ⟶ deformation grid; (b) denser set of control point correspondences; (c) oriented line correspondences; (d) uniform quadrilateral grid.

Image morphing

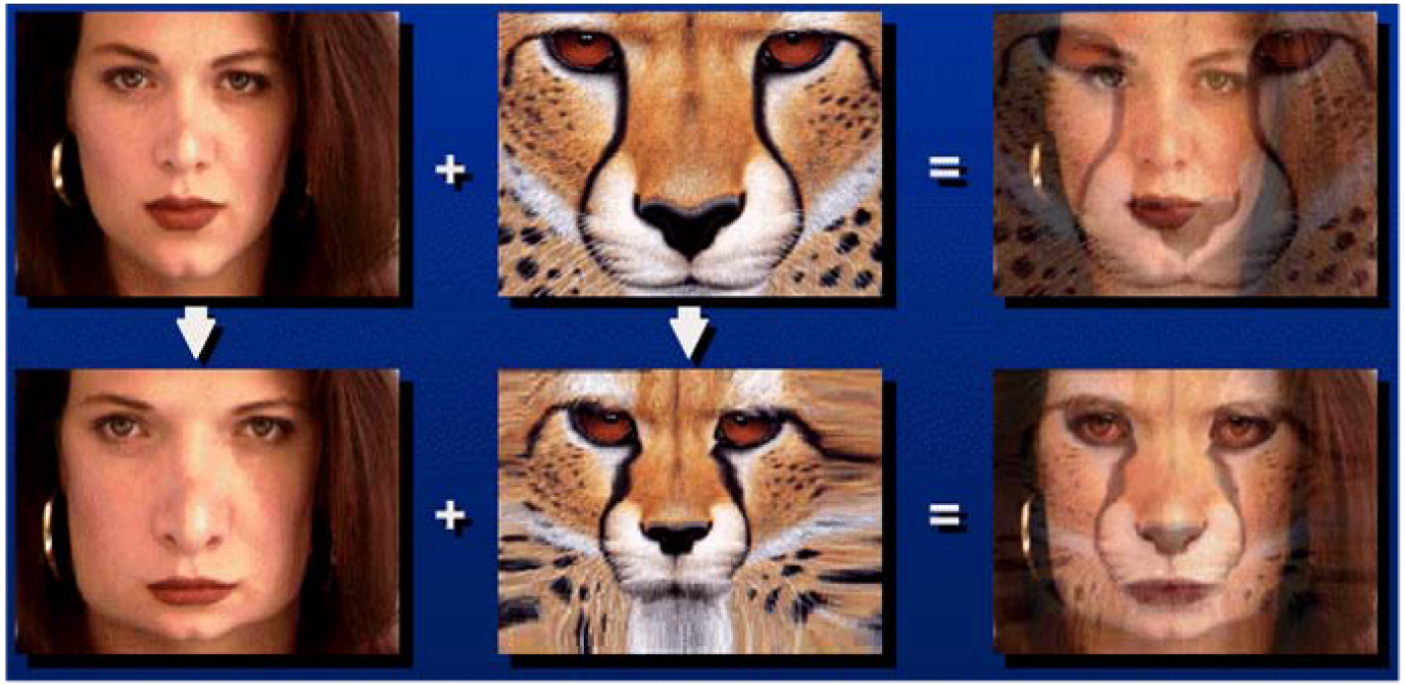


Figure 3.53 Image morphing (Gomes, Darsa, Costa *et al.* 1999) © 1999 Morgan Kaufmann. Top row: if the two images are just blended, visible ghosting results. Bottom row: both images are first warped to the same intermediate location (e.g., halfway towards the other image) and the resulting warped images are then blended resulting in a seamless morph.

Image morphing – *Black or White* video

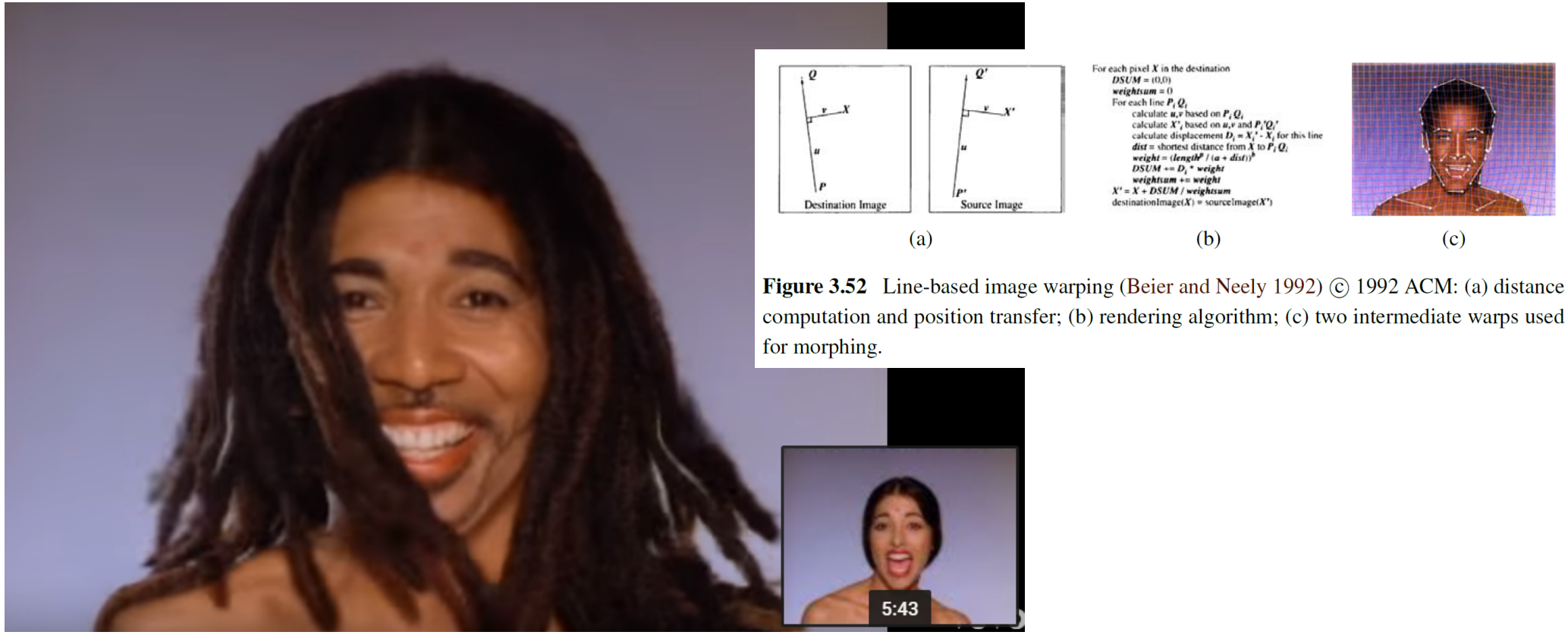


Figure 3.52 Line-based image warping (Beier and Neely 1992) © 1992 ACM: (a) distance computation and position transfer; (b) rendering algorithm; (c) two intermediate warps used for morphing.

https://www.youtube.com/watch?v=F2AitTPI5U0

Filtering, resampling, pyramids, blending

- Linear and separable filtering
- Non-linear filtering
- Pyramids
- Blending
- Resampling (rotations, etc.)
- Data Fitting
- Regularization and variational techniques
- Markov Random Fields

Data fitting and optimization

Chapter 4

Model fitting and optimization

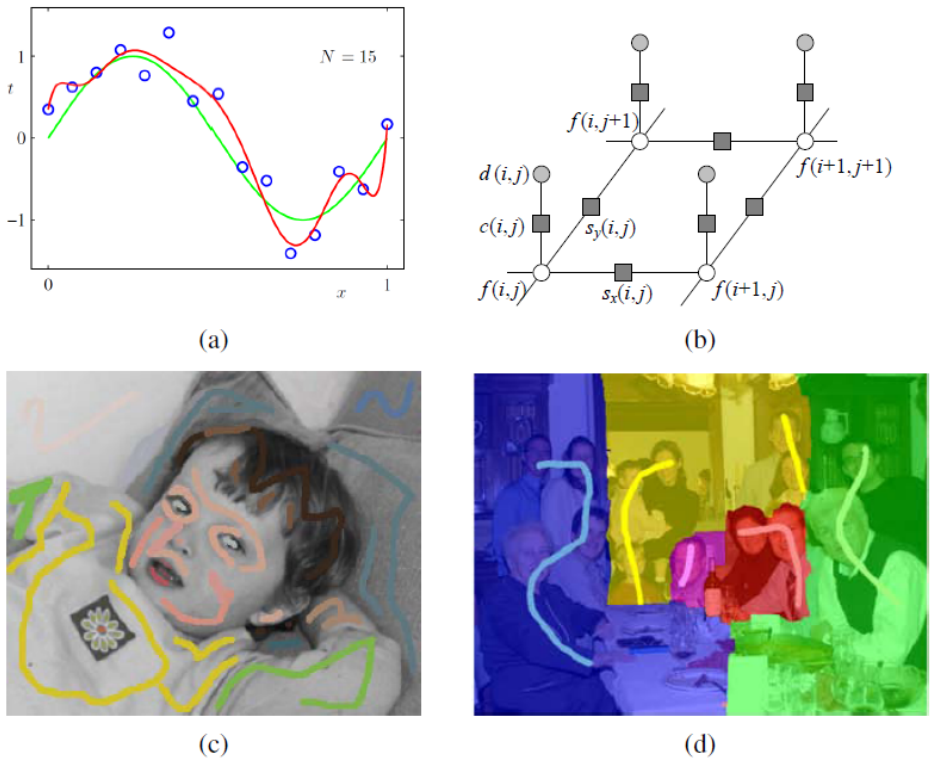
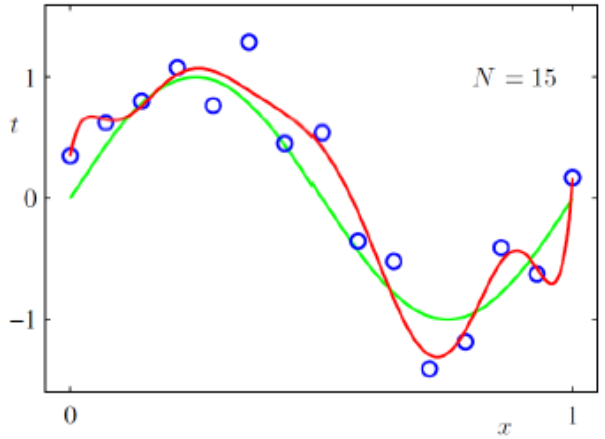


Figure 4.1 *Examples of data interpolation and global optimization: (a) scattered dat interpolation (curve fitting) (Bishop 2006) © 2006 Springer; (b) graphical model interpreta tion of first-order regularization; (c) colorization using optimization (Levin, Lischinski, an Weiss 2004) © 2004 ACM; (d) multi-image Photomontage formulated as an unordered labe MRF (Agarwala, Dontcheva, Agrawala* et al. *2004) © 2004 ACM.*

Scattered data interpolation / approximation

- Green: original (clean) curve
- Blue: noisy samples
- Red: fitted curve
  - Interpolation or approximation?
  - Underfitting or overfitting?



More formally

4.1 Scattered data interpolation

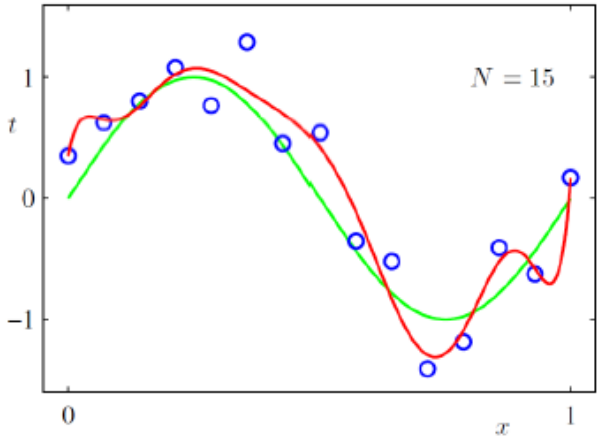
The goal of *scattered data interpolation* is to produce a (usually continuous and smooth) function $\mathbf{f}(\mathbf{x})$ that passes *through* a set of data points $\mathbf{d}_k$ placed at locations $\mathbf{x}_k$ such that

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

The related problem of *scattered data approximation* only requires the function to pass *near* the data points (Amidror 2002; Wendland 2004; Anjyo, Lewis, and Pighin 2014). This is usually formulated using a penalty function such as

$$E_{\mathrm{D}} = \sum_k \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2, \tag{4.2}$$

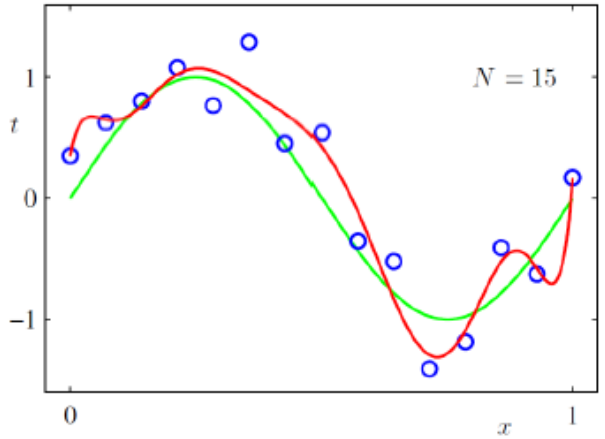
with the squared norm in the above formula sometime replaced by a different norm or robust function (Section 4.1.3). In statistics and machine learning, the problem of predicting an output function given a finite number of samples is called *regression* (Section 5.1), the $\mathbf{x}$ vectors are called the *inputs*, and the outputs $\mathbf{y}$ are called the *targets*. Figure 4.1a shows an example of one-dimensional scattered data interpolation, while Figures 4.2 and 4.9 show some two-dimensional examples.



Scattered data interpolation / approximation

- Green: original (clean) curve
- Blue: noisy samples
- Red: fitted curve
  - Underfitting or overfitting?

- What is it good for?

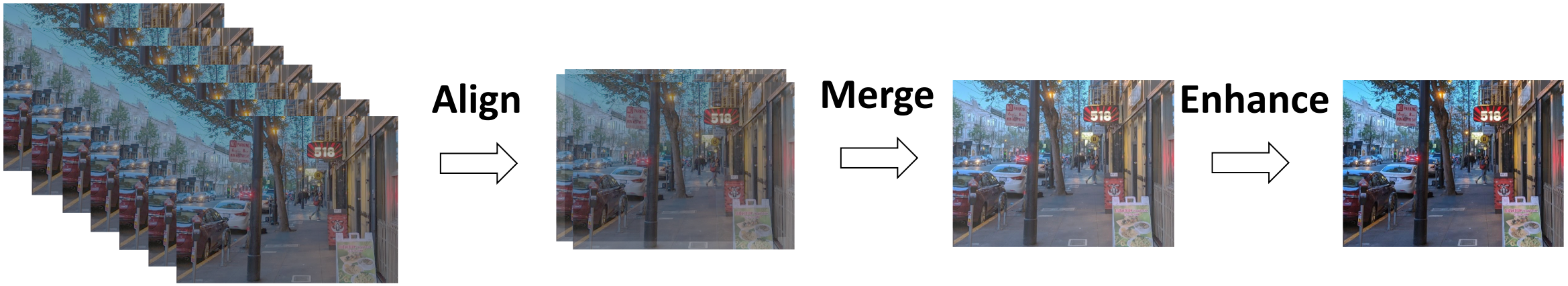


CVPR'19 NTIRE workshop talk

https://data.vision.ee.ethz.ch/cvl/ntire19/speakers/PeymanMilanfar_Computation_and_Photography.pdf

Modern Mobile Imaging: Burst Photography

Exposure control

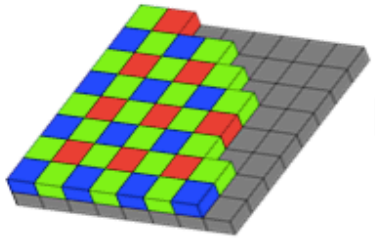


Align: Reliable Optical Flow – Scene is never stationary

Merge: Artifact-free Fusion – Alignment failures, occlusion, ...

Enhance: Denoise, Sharpen, Contrast, Dynamic Range

Classic Camera Image Processing Pipeline



Sensor with color filter array
(CCD/CMOS)

Gain Control
A/D Converter
Possible LUT

White
Balance

Demosaic

"Merge"

Noise
Reduction/
Sharpening

Color Space
Transform +
Color Preferences

Tone
Reproduction

"Enhance"

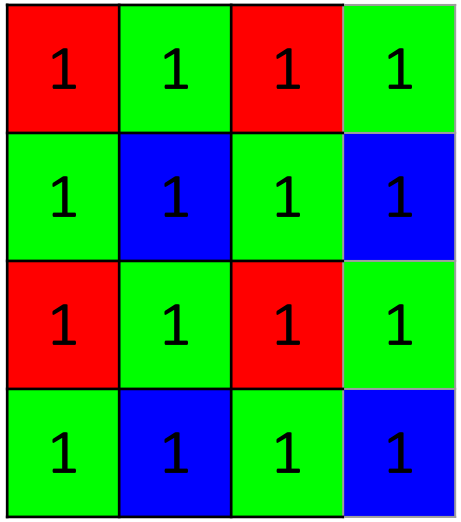
JPEG
Compression

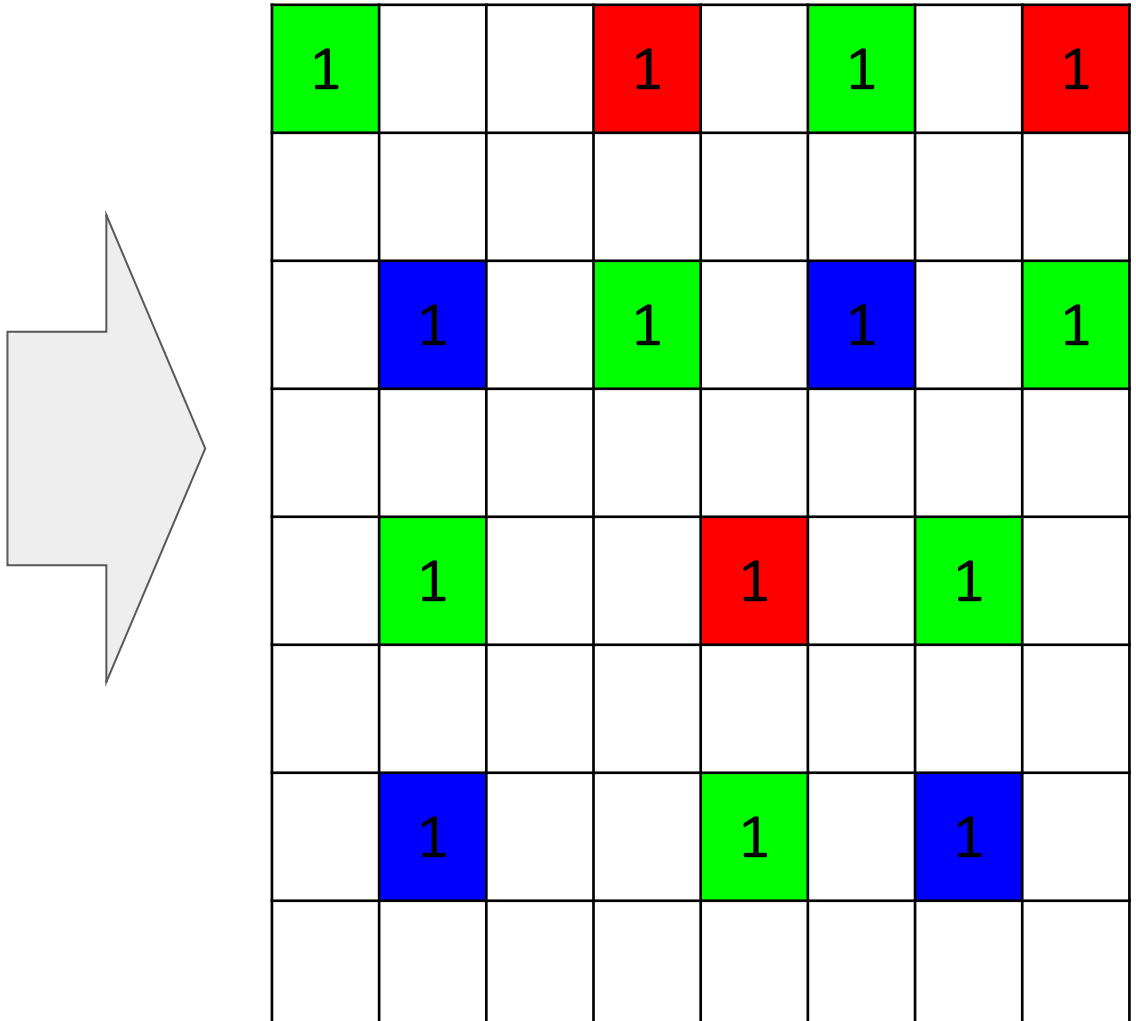
Demosaicing

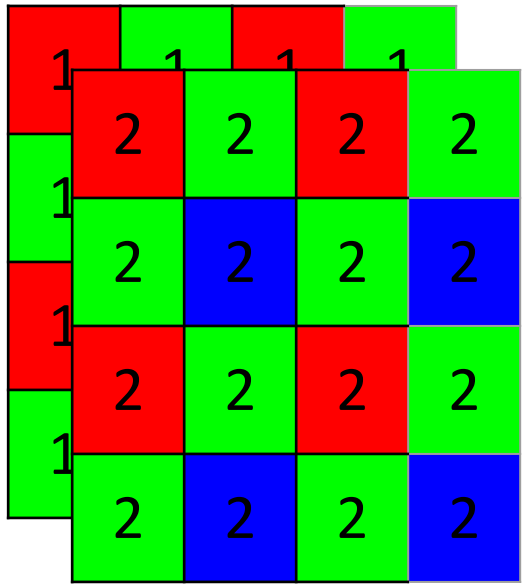
Demosaicing …. Kills Details and Produces Artifacts

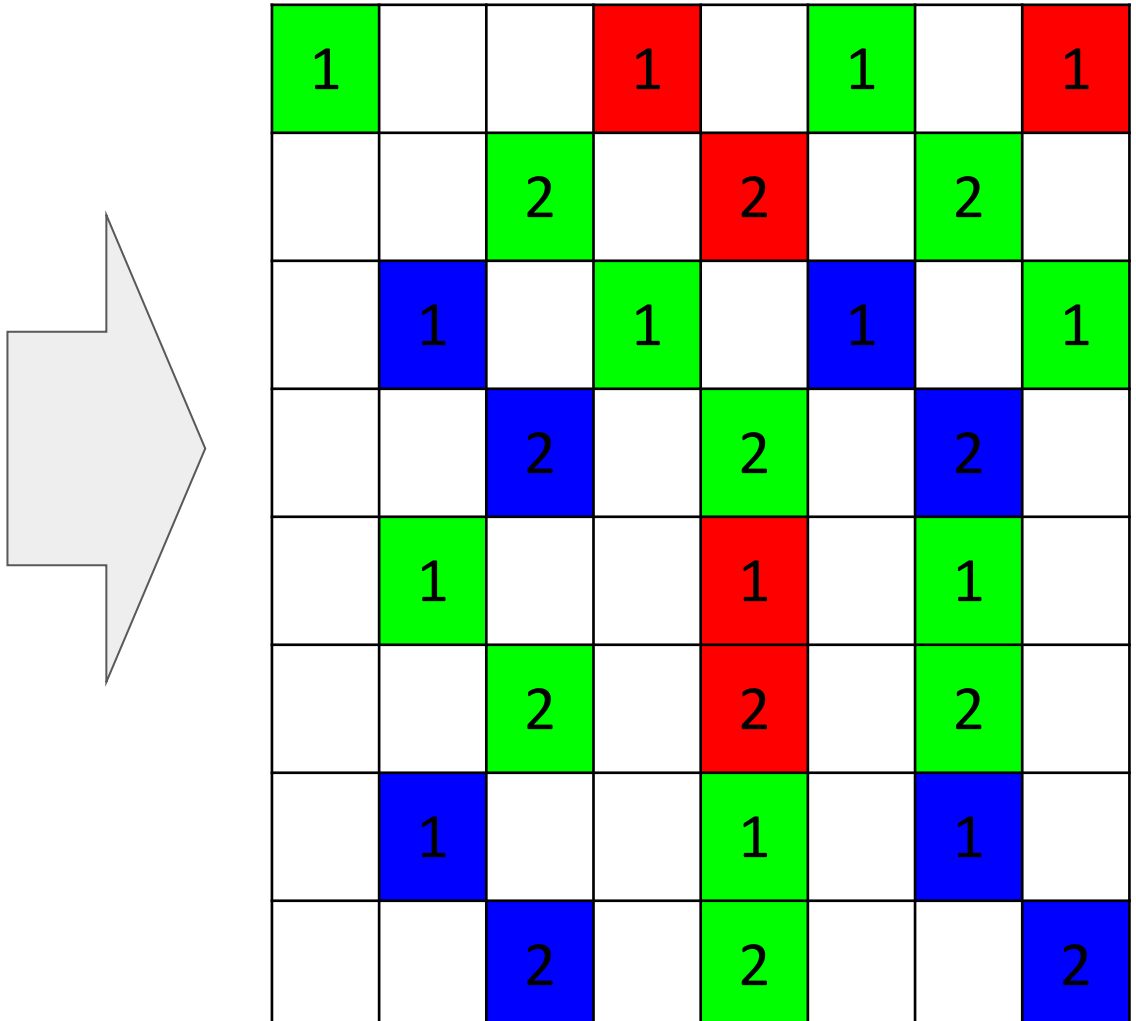
Instead ….. Replace demosaicing with multiple frames

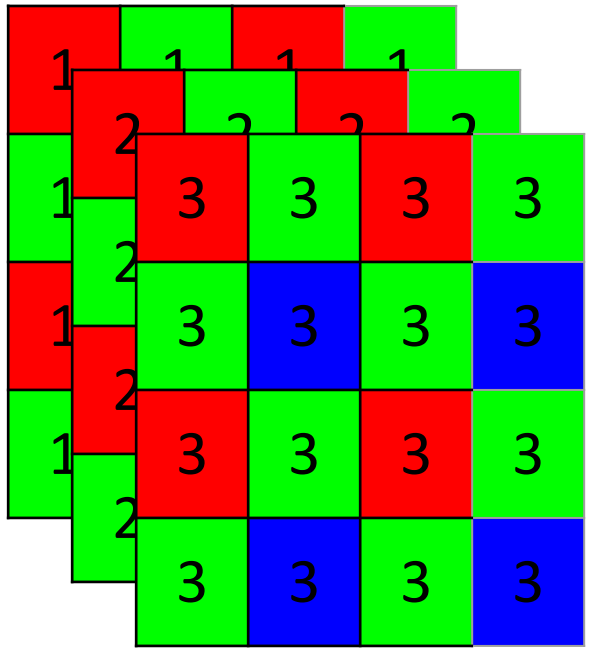
Pixel shifting



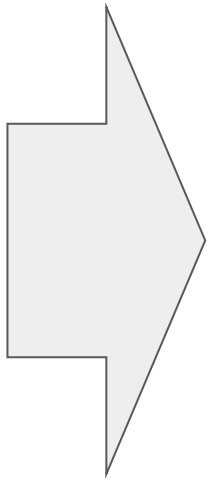


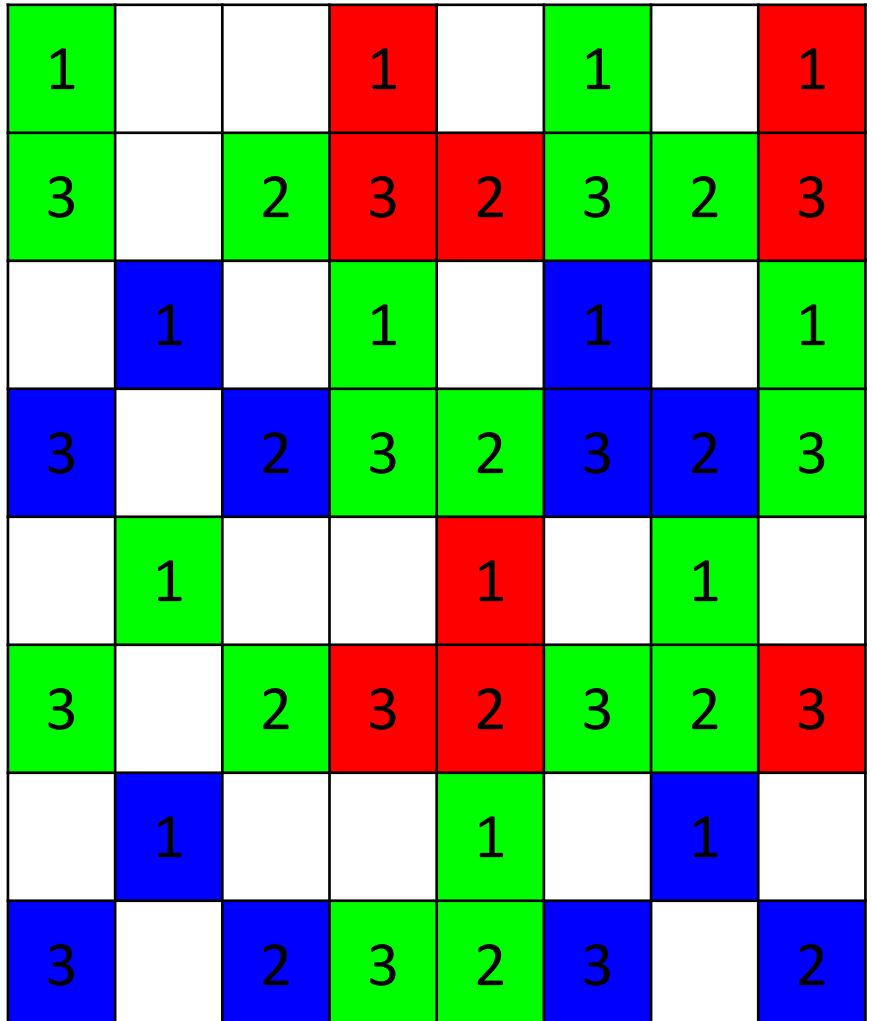

1 1 1 1
2 2 2 2
1
2 2 2 2
1
2 2 2 2
1
2 2 2 2


1 1 1 1
2 2 2
1 1 1 1
2 2 2
1 1 1 1
2 2 2
1 1 1
2 2 2 2

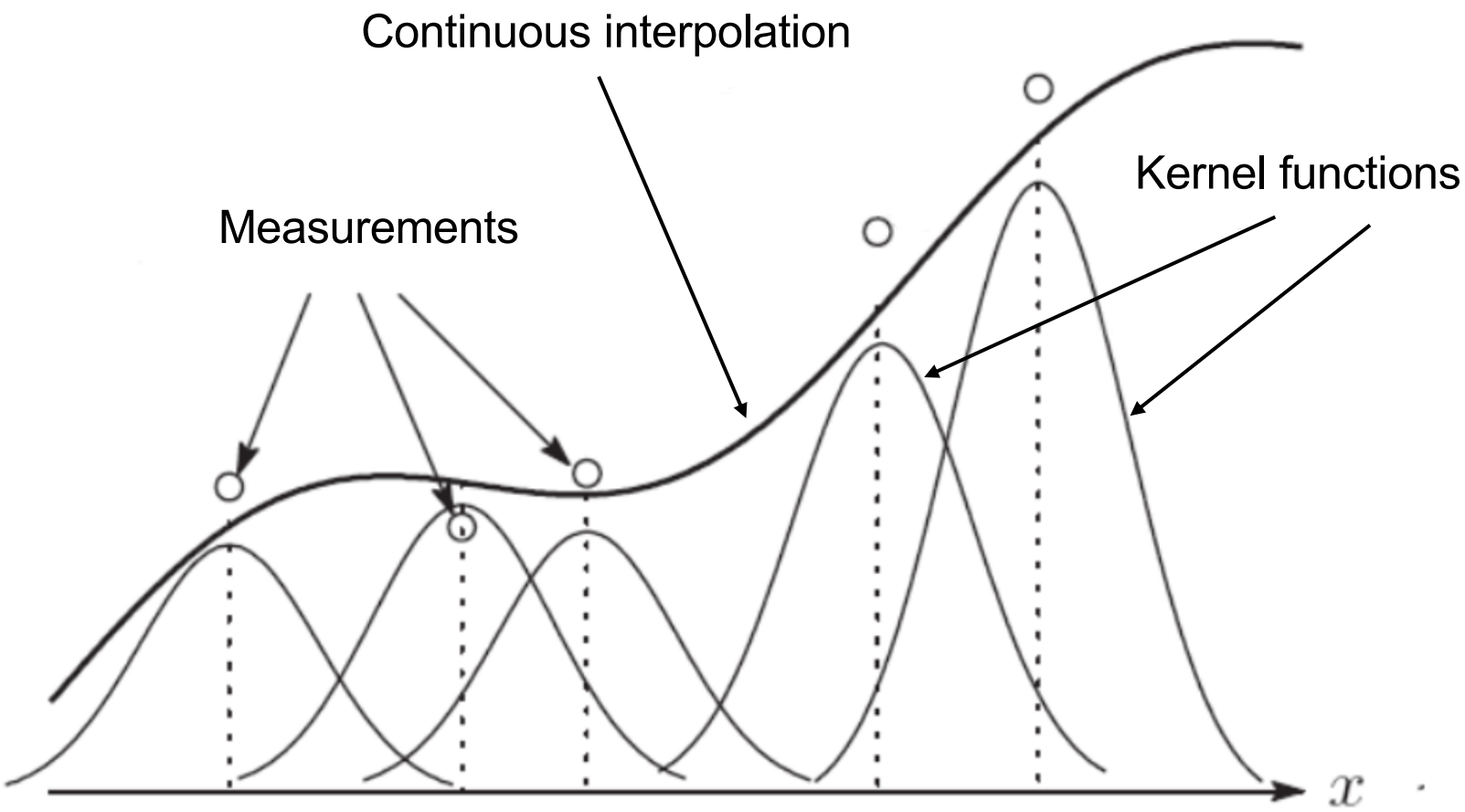


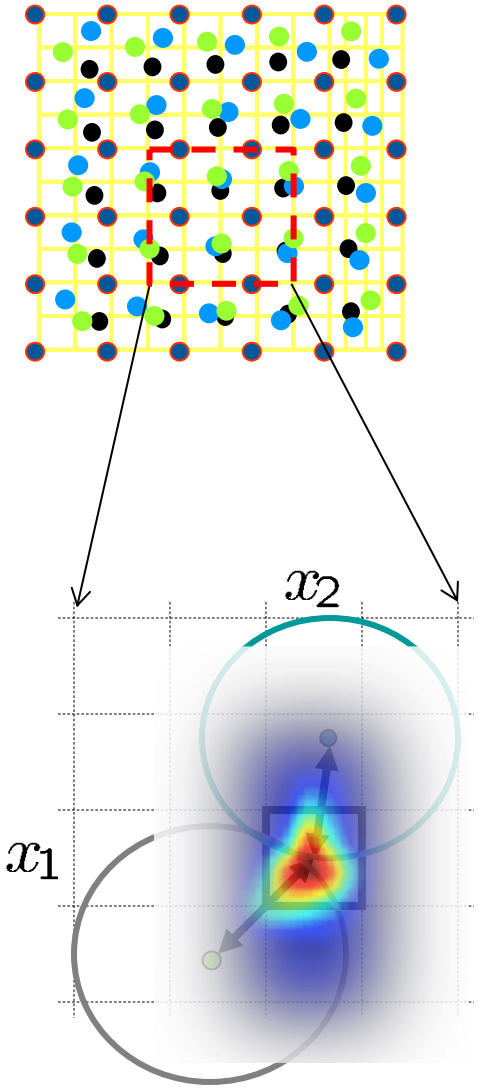
How do you merge these samples
and fill in the missing data?



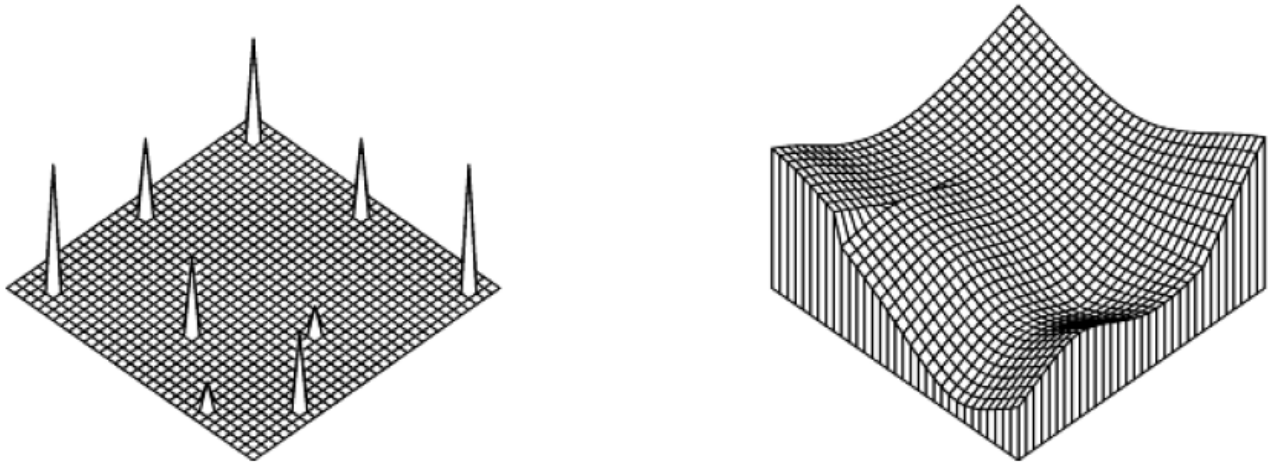


Merge: Nonlinear Kernel Regression





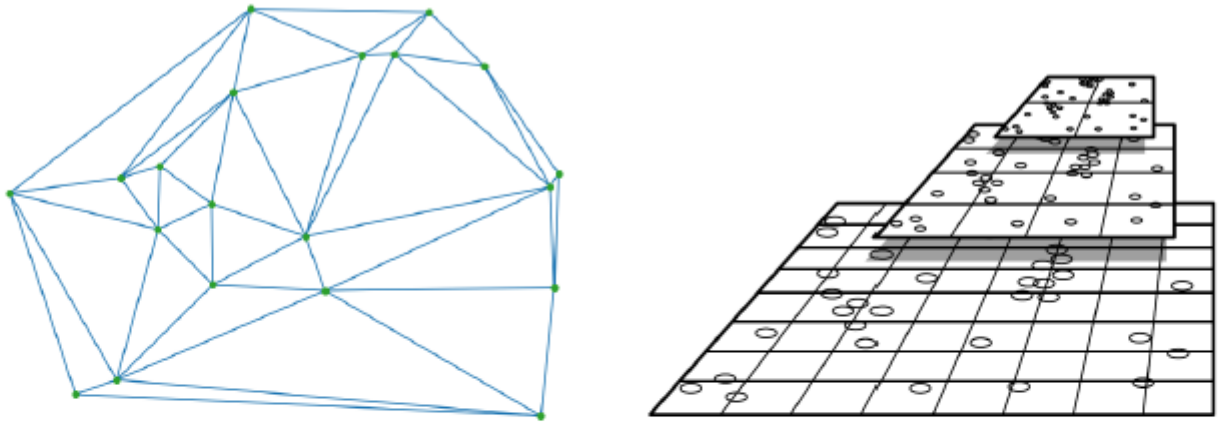
How can we interpolate scattered data?



(a) (b)

Figure 4.9 *A simple surface interpolation problem: (a) nine data points of various height scattered on a grid; (b) second-order, controlled-continuity, thin-plate spline interpolator, with a tear along its left edge and a crease along its right (Szeliski 1989)* © *1989 Springer.*

Triangulations and pyramids



(a) (b)

Figure 4.2 *Some simple scattered data interpolation and approximation algorithms: (a) a Delaunay triangularion defined over a set of data point locations; (b) data structure and intermediate results for the* pull-push *algorithm (Gortler, Grzeszczuk, Szeliski* et al. *1996)* © *1996 ACM.*

Underfitting and overfitting

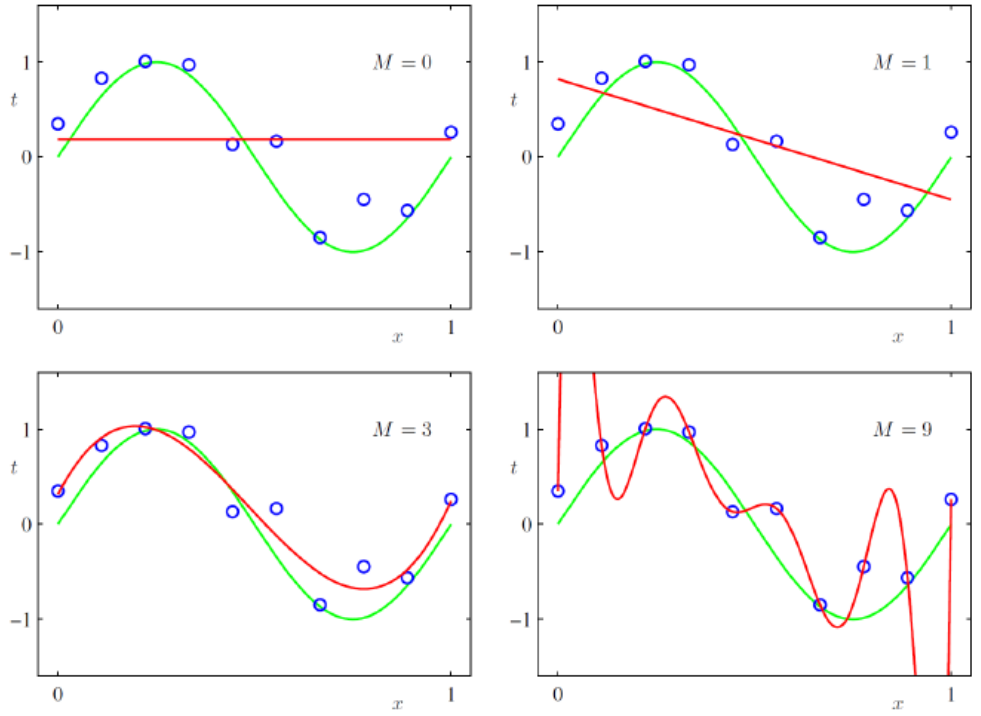


Figure 4.3 *Polynomial curve fitting to the blue circles, which are noisy samples from the green sine curve (Bishop 2006) © 2006 Springer. The four plots show the 0th order constant function, the first order linear fit, the $M = 3$ cubic polynomial, and the 9th degree polynomial. Notice how the first two curves exhibit* underfitting, *while the last curve exhibits* overfitting, *i.e., excessive wiggle.*

Underfitting and overfitting

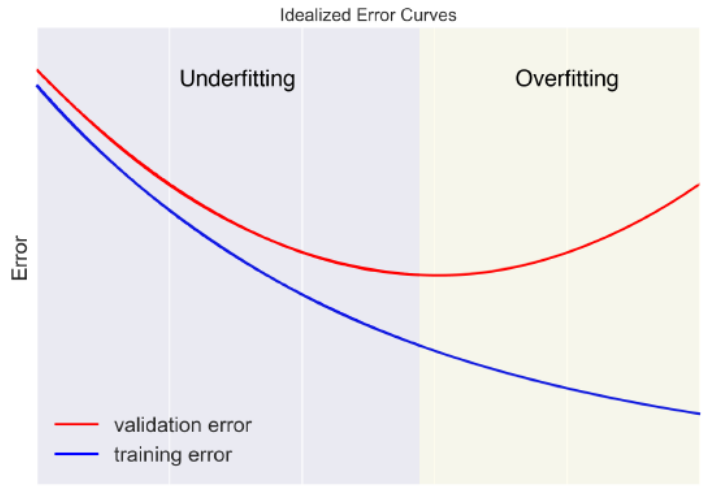


Figure 4.5 *Fitting (training) and validation errors as a function of the amount of regularization or smoothing (Glassner 2018) © 2018 Andrew Glassner. The less regularized solutions on the right, while exhibiting lower fitting error, perform less well on the validation data.*

Bias / variance tradeoff

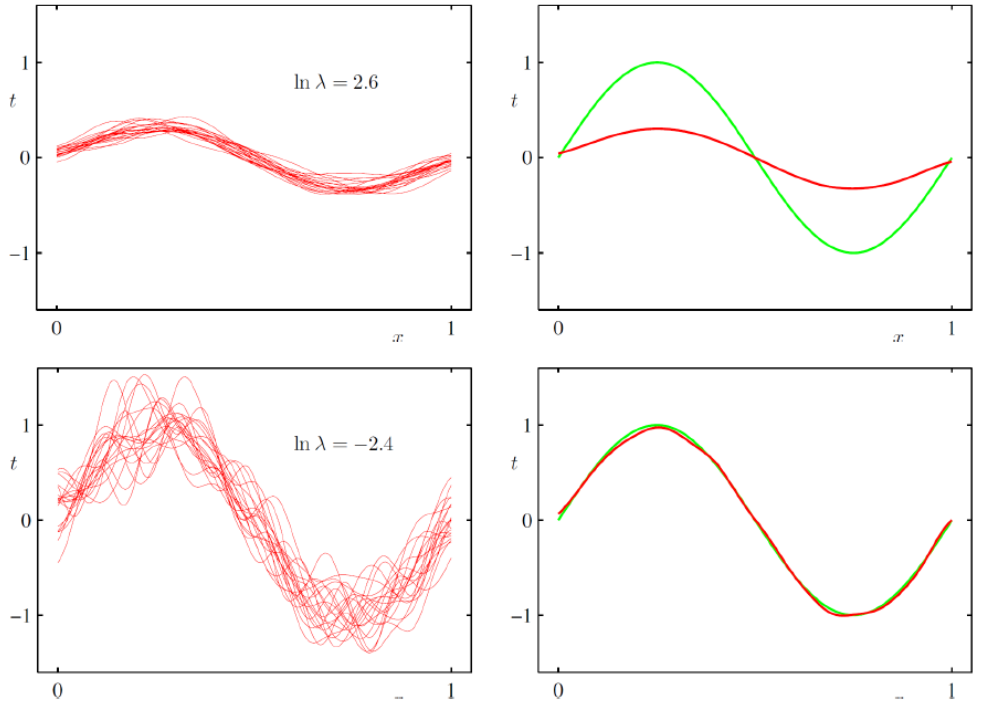
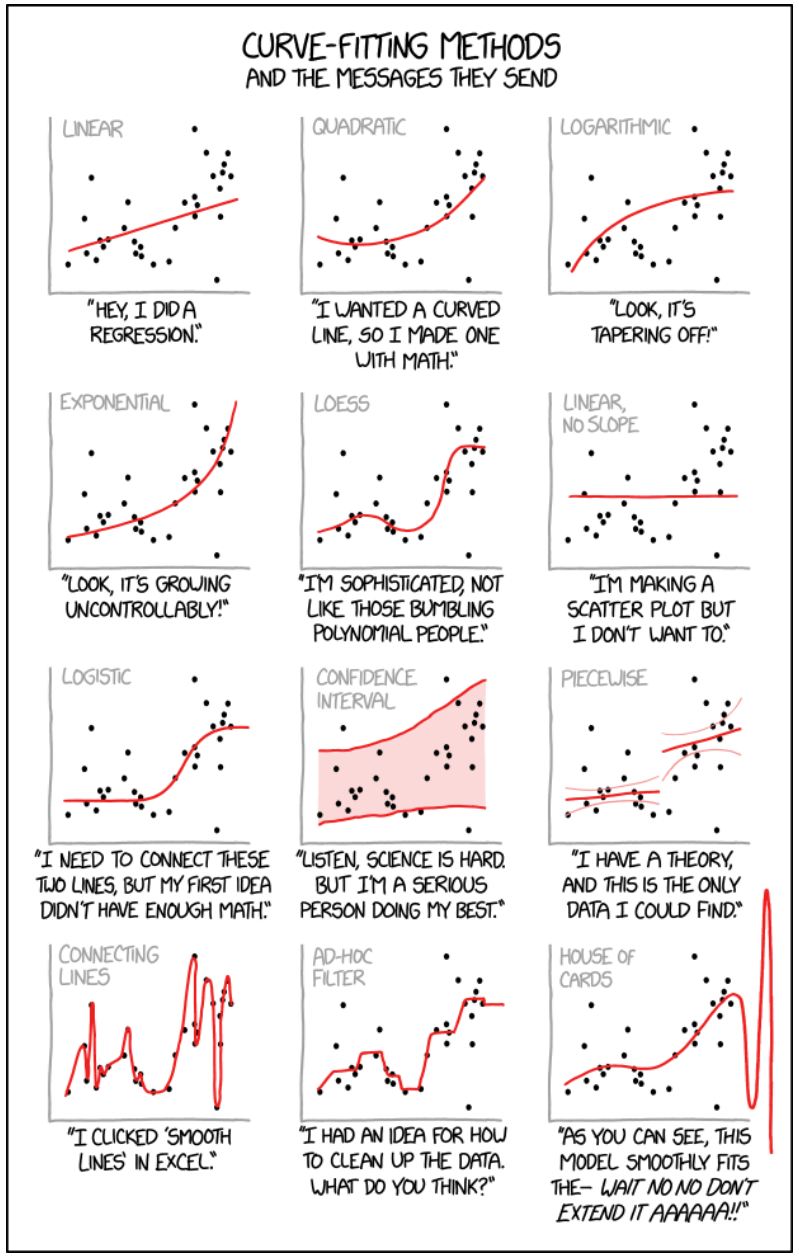


Figure 4.6 *The more heavily regularized solution* $\log \lambda = 2.6$ *exhibits higher bias (deviation from original curve) than the less heavily regularized version* ($\log \lambda = -2.4$)*, which has much higher variance (Bishop 2006)* © *2006 Springer. The red curves on the right a* $M = 24$ *Gaussian basis fits to 25 randomly sampled points on the green curve. The red curve on the right is their mean.*

Curve fitting



https://xkcd.com/2048/

L2 interpolation / approximation

4.1 Scattered data interpolation

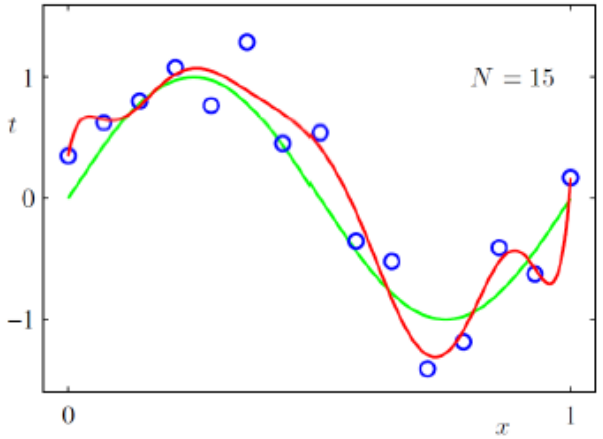
The goal of *scattered data interpolation* is to produce a (usually continuous and smooth) function $\mathbf{f}(\mathbf{x})$ that passes *through* a set of data points $\mathbf{d}_k$ placed at locations $\mathbf{x}_k$ such that

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

The related problem of *scattered data approximation* only requires the function to pass *near* the data points (Amidror 2002; Wendland 2004; Anjyo, Lewis, and Pighin 2014). This is usually formulated using a penalty function such as

$$E_{\mathrm{D}} = \sum_k \|\mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k\|^2, \tag{4.2}$$

with the squared norm in the above formula sometime replaced by a different norm or robust function (Section 4.1.3). In statistics and machine learning, the problem of predicting an output function given a finite number of samples is called *regression* (Section 5.1), the $\mathbf{x}$ vectors are called the *inputs*, and the outputs $\mathbf{y}$ are called the *targets*. Figure 4.1a shows an example of one-dimensional scattered data interpolation, while Figures 4.2 and 4.9 show some two-dimensional examples.



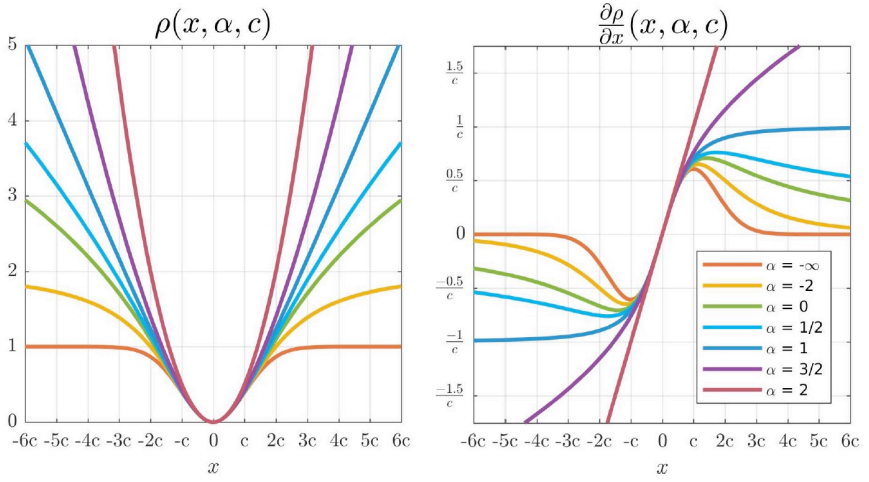
Robust fitting – dealing with outliers

This same idea can be applied to data terms such as (4.2), where instead of using a quadratic penalty, we can use a *robust loss function* $\rho()$,

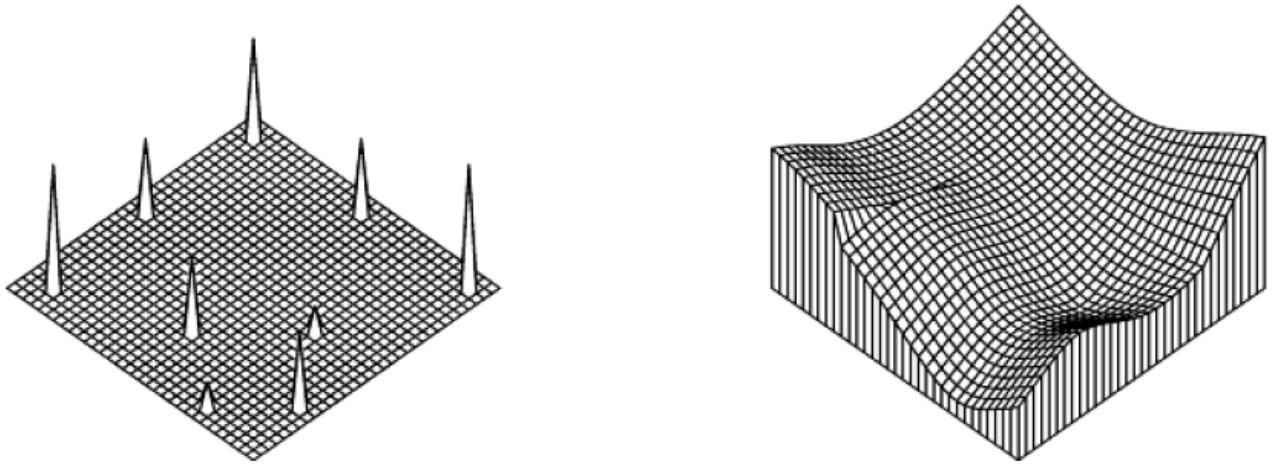
$$E_{\mathrm{R}} = \rho(\|\mathbf{r}_k\|), \quad \text{with } \mathbf{r}_k = \mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k, \tag{4.15}$$

which gives lower weights to larger data fitting errors, which are more likely to be outlier measurements. (The fitting error term $\mathbf{r}_k$ is called the *residual error*.)

Some examples of loss functions from (Barron 2019) are shown in Figure 4.8 along with their derivatives. The regular quadratic quadratic ($\alpha = 2$) penalty gives full (linear) weight to each error, whereas the $\alpha = 1$ loss gives equal weight to all larger residuals, i.e., it behaves as an L_1 loss for large residuals, and L_2 for small ones. Even larger values of α discount large errors (outliers) even more, although they result in optimization problems that are *non-convex*, i.e., that can have multiple local minima. We will discuss techniques for finding good inital guesses for such problems later on in Section 9.1.4.



Controlled continuity surface fitting



(a) (b)

Figure 4.9 *A simple surface interpolation problem: (a) nine data points of various height scattered on a grid; (b) second-order, controlled-continuity, thin-plate spline interpolator, with a tear along its left edge and a crease along its right (Szeliski 1989)* © *1989 Springer.*

Variational methods

(Here, we use subscripts to denote differentiation.) Such energy measures are examples of *functionals*, which are operators that map functions to scalar values. They are also often called *variational methods*, because they measure the variation (non-smoothness) in a function.

In two dimensions (e.g., for images, flow fields, or surfaces), the corresponding smoothness functionals are

$$\mathcal{E}_1 = \int f_x^2(x,y) + f_y^2(x,y)\,dx\,dy = \int \|\nabla f(x,y)\|^2\,dx\,dy \tag{4.18}$$

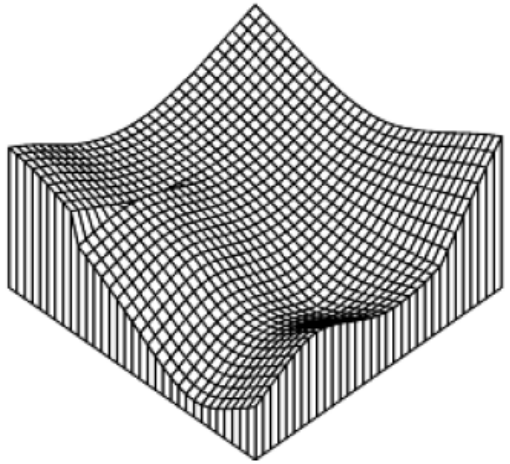
and

$$\mathcal{E}_2 = \int f_{xx}^2(x,y) + 2f_{xy}^2(x,y) + f_{yy}^2(x,y)\,dx\,dy, \tag{4.19}$$

where the mixed $2f_{xy}^2$ term is needed to make the measure rotationally invariant (Grimson 1983).

The first derivative norm is often called the *membrane*, since interpolating a set of data points using this measure results in a tent-like structure. (In fact, this formula is a small-deflection approximation to the surface area, which is what soap bubbles minimize.) The second-order norm is called the *thin-plate spline*, since it approximates the behavior of thin plates (e.g., flexible steel) under small deformations. A blend of the two is called the *thin-*

Controlled continuity splines

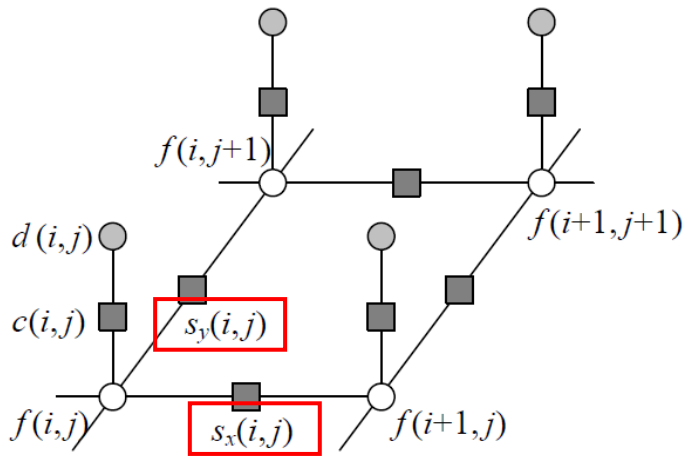


To better model such functions, Terzopoulos (1986b) introduced *controlled-continuity splines*, where each derivative term is multiplied by a local weighting function,

$$\mathcal{E}_{\mathrm{CC}} = \int \rho(x,y)\{[1-\tau(x,y)][f_x^2(x,y)+f_y^2(x,y)] + \tau(x,y)[f_{xx}^2(x,y)+2f_{xy}^2(x,y)+f_{yy}^2(x,y)]\}\,dx\,dy. \quad (4.20)$$

Here, $\rho(x,y) \in [0,1]$ controls the *continuity* of the surface and $\tau(x,y) \in [0,1]$ controls the local *tension*, i.e., how flat the surface wants to be. Figure 4.9 shows a simple example of

Discrete energy minimization



Fortunately, for both the first-order and second-order smoothness functionals, the judicious selection of appropriate finite elements results in particularly simple discrete forms (Terzopoulos 1983). The corresponding *discrete* smoothness energy functions become

$$E_1 = \sum_{i,j} s_x(i,j)[f(i+1,j) - f(i,j) - g_x(i,j)]^2 + s_y(i,j)[f(i,j+1) - f(i,j) - g_y(i,j)]^2 \quad (4.24)$$

The data values $g_x(i,j)$ and $g_y(i,j)$ are gradient data terms (constraints) used by algorithms, such as photometric stereo (Section 14.1.1), HDR tone mapping (Section 11.2.1) (Fattal, Lischinski, and Werman 2002), Poisson blending (Section 9.4.4) (Pérez, Gangnet, and Blake 2003), gradient-domain blending (Section 9.4.4) (Levin, Zomet, Peleg *et al.* 2004), and

Interactive edge-preserving colorization

(a) (b) (c)

Figure 4.11 *Colorization using optimization (Levin, Lischinski, and Weiss 2004) © 2004 ACM: (a) grayscale image some color scribbles overlaid; (b) resulting colorized image; (c) original color image from which the grayscale image and the chrominance values for the scribbles were derived. Original photograph by Rotem Weiss.*

Speeding up the solver

Figure 4.12 *Speeding up the inhomogeneous least squares colorization solver using locally adapted hierarchical basis preconditiong (Szeliski 2006b) © 2006 ACM: (a) input gray image with color strokes overlaid; (b) solution after 20 iterations of conjugate gradient; (c) using 1 iteration of hierarchical basis function preconditioning; (d) using 1 iteration of locally adapted hierarchical basis functions.*

Hierarchical basis preconditioning

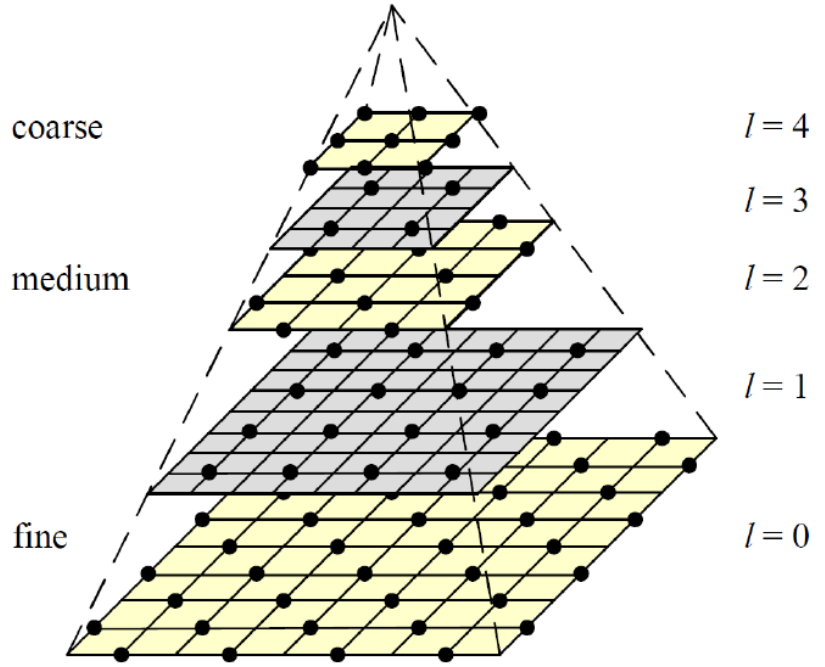
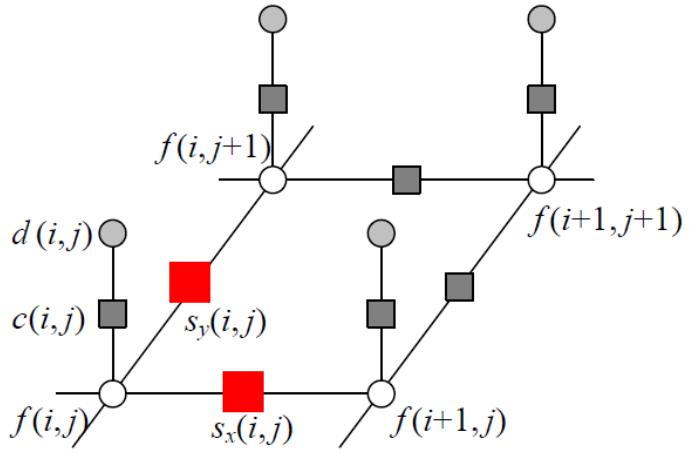


Figure A.3 *Multiresolution pyramid with half-octave (quincunx sampling (odd levels are colored gray for easier visibility) (Szeliski 2006b) © 2006 ACM. Hierarchical basis function control variables are shown as black dots.*

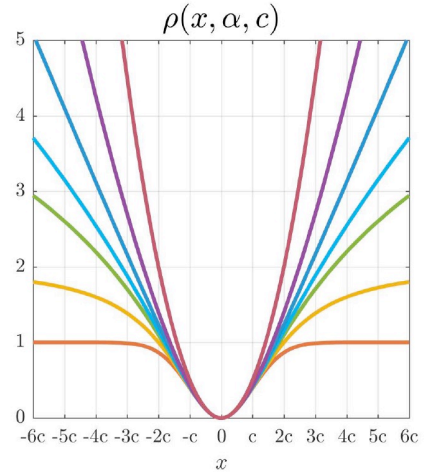
Robust regularization



Robust regularization

While regularization is most commonly formulated using quadratic (L_2) norms i.e., the squared derivatives in (4.16–4.19) and squared differences in (4.24–4.25), it can also be formulated using the non-quadratic *robust* penalty functions first introduced in Section 4.1.3 and discussed in more detail in Appendix B.3. For example, (4.24) can be generalized to

$$E_{\mathrm{1R}} = \sum_{i,j} s_x(i,j)\rho(f(i+1,j) - f(i,j)) + s_y(i,j)\rho(f(i,j+1) - f(i,j)), \tag{4.29}$$



Bilateral solver

uating weighted errors between neighboring pixels. As we saw previously in our discussion of bilateral filtering in Section 3.3.2, we can often get better results by looking at a larger spatial neighborhood and combining pixels with similar colors or grayscale values. To extend this idea to a variational (energy minimization) setting, Barron and Poole (2016) propose replacing the usual first-order nearest-neighbor smoothness penalty (4.24) with a bilaterally weighted version

$$E_{\mathrm{B}} = \sum_{i,j} \sum_{k,l} \hat{w}(i,j,k,l)[f(k,l) - f(i,j)]^2, \tag{4.31}$$

where

$$\hat{w}(i,j,k,l) = \frac{w(i,j,k,l)}{\sum_{m,n} w(i,j,m,n)}, \tag{4.32}$$

is the *bistochastized* (normalized) version of the *bilateral weight function* given in (3.37), which may depend on an input guide image, but not on the estimated values of f.[5]

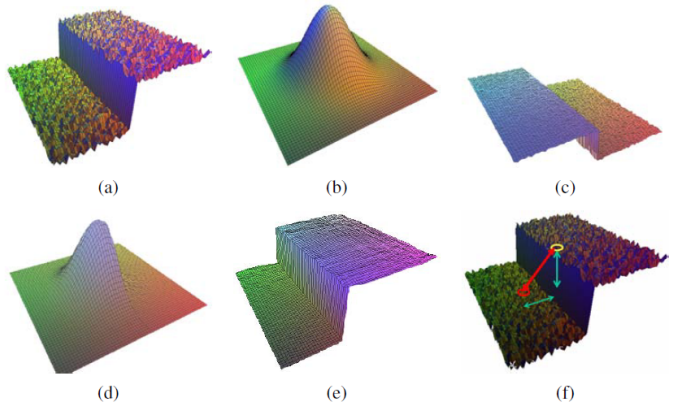


Figure 3.20 Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.

Depth from Motion for Smartphone AR

JULIEN VALENTIN, ADARSH KOWDLE, JONATHAN T. BARRON, NEAL WADHWA, MAX DZITSIUK, MICHAEL SCHOENBERG, VIVEK VERMA, AMBRUS CSASZAR, ERIC TURNER, IVAN DRYANOVSKI, JOAO AFONSO, JOSE PASCOAL, KONSTANTINE TSOTSOS, MIRA LEUNG, MIRKO SCHMIDT, ONUR GULERYUZ, SAMEH KHAMIS, VLADIMIR TANKOVITCH, SEAN FANELLO, SHAHRAM IZADI, and CHR RHEMANN, Google Inc.

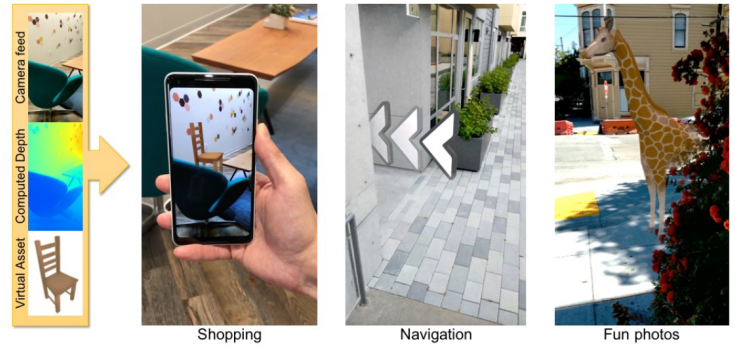
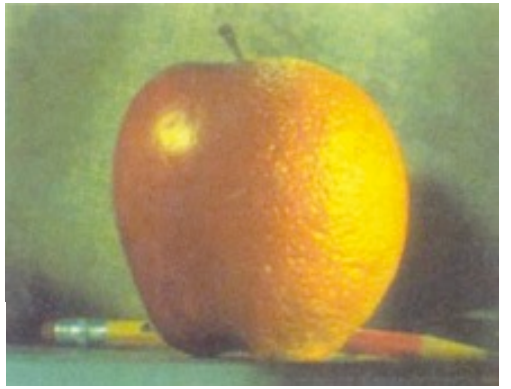


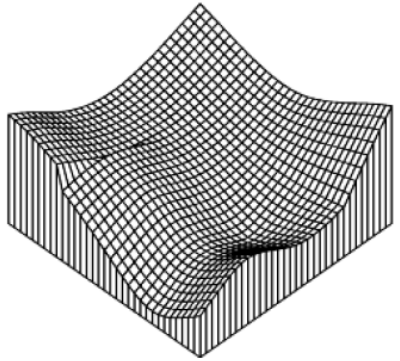
Fig. 1. AR occlusions. Estimating the depth of the scene is crucial to render virtual objects such that they realistically blend into the real context. We provide the first system capable of providing dense, low latency depth maps at 30Hz on a single mobile CPU core, using only the standard color camera found on most smartphones. Applications include AR shopping, navigation and creative photo apps.

(future lecture)?

Interpolation and optimization

- Interpolation
- Pyramids
- Blending
- Resampling (rotations, etc.)
- Data Fitting
- Regularization and variational techniques
- Markov Random Fields





Markov Random Fields

According to Bayes' Rule (Appendix B.4), the *posterior* distribution for a given set of measurements $\mathbf{y}$, $p(\mathbf{y}|\mathbf{x})$, combined with a prior $p(\mathbf{x})$ over the unknowns $\mathbf{x}$, is given by

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}, \tag{4.33}$$

where $p(\mathbf{y}) = \int_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$ is a normalizing constant used to make the $p(\mathbf{x}|\mathbf{y})$ distribution *proper* (integrate to 1). Taking the negative logarithm of both sides of (4.33), we get

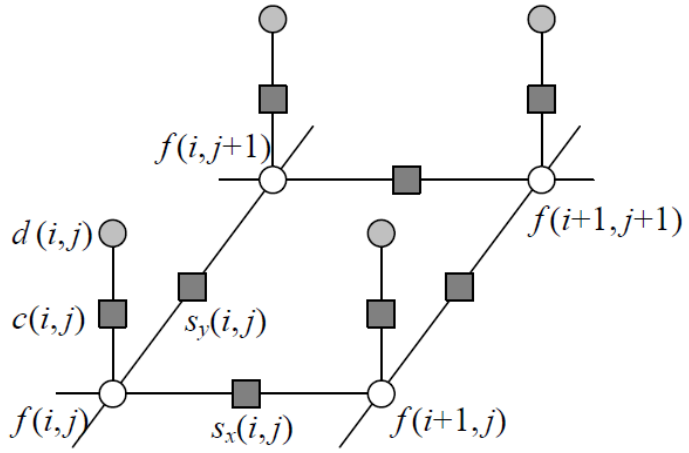
$$-\log p(\mathbf{x}|\mathbf{y}) = -\log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{x}) + C, \tag{4.34}$$

which is the *negative posterior log likelihood*.

To find the most likely (*maximum a posteriori* or MAP) solution $\mathbf{x}$ given some measurements $\mathbf{y}$, we simply minimize this negative log likelihood, which can also be thought of as an *energy*,

$$E(\mathbf{x}, \mathbf{y}) = E_{\mathrm{D}}(\mathbf{x}, \mathbf{y}) + E_{\mathrm{P}}(\mathbf{x}). \tag{4.35}$$

(We drop the constant C because its value does not matter during energy minimization.) The first term $E_{\mathrm{D}}(\mathbf{x}, \mathbf{y})$ is the *data energy* or *data penalty*; it measures the negative log likelihood that the data were observed given the unknown state $\mathbf{x}$. The second term $E_{\mathrm{P}}(\mathbf{x})$ is the *prior energy*; it plays a role analogous to the smoothness energy in regularization. Note



Why Bayesian (probabilistic) modeling?

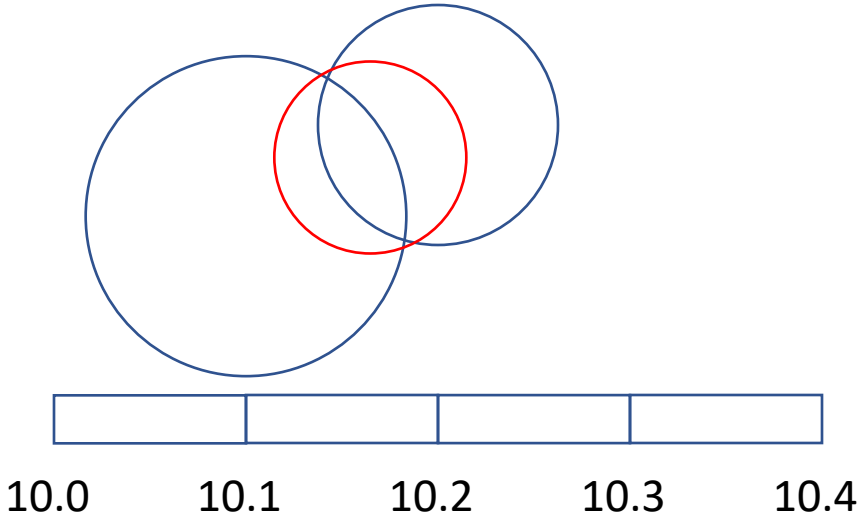
- Merge uncertain measurement in an optimal way
- Estimate the uncertainty in the final answer
- Build in (quantitative) prior assumptions

Appendix B

Bayesian modeling and inference

Example: merge GPS readings

- *Measurement 1*: 10.1 ± 0.1667 (1/6) Gaussian noise
- *Measurement 2*: 10.2 ± 0.1250 (1/8) Gaussian noise
- What is the optimal combined measurement?

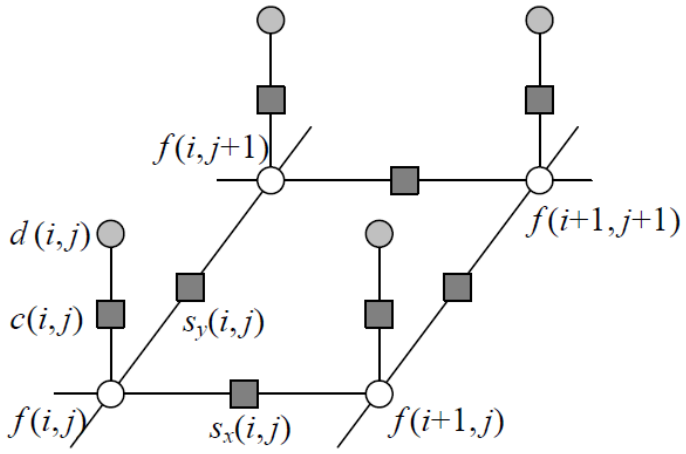


Exercise for next lecture

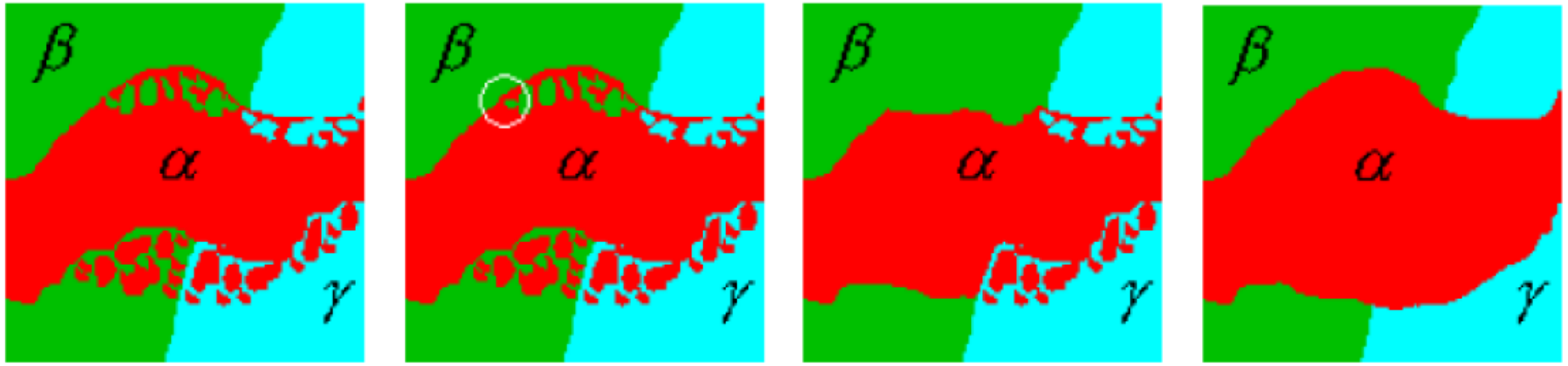
(send me your answer on Slack)

Why a Bayesian formulation?

- Wider range of probability distributions
- Learn distributions from data
- Wider range of inference algorithms [Kappes, Andres, *et al.*, IJCV 2015]



Energy minimization



(a) initial labeling (b) standard move (c) α-β-swap (d) α-expansion

Figure 4.15 *Multi-level graph optimization from (Boykov, Veksler, and Zabih 2001) © 2001 IEEE: (a) initial problem configuration; (b) the standard move only changes one pixel; (c) the α-β-swap optimally exchanges all α and β-labeled pixels; (d) the α-expansion move optimally selects among current pixel values and the α label.*

Digital Photomontage

Figure 4.17 *An unordered label MRF (Agarwala, Dontcheva, Agrawala* et al. *2004)* © *2004 ACM: Strokes in each of the source images on the left are used as constraints on an MRF optimization, which is solved using graph cuts. The resulting multi-valued label field is shown as a color overlay in the middle image, and the final composite is shown on the right.*

Markov Random Fields

According to Bayes' Rule (Appendix B.4), the *posterior* distribution for a given set of measurements $\mathbf{y}$, $p(\mathbf{y}|\mathbf{x})$, combined with a prior $p(\mathbf{x})$ over the unknowns $\mathbf{x}$, is given by

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}, \tag{4.33}$$

where $p(\mathbf{y}) = \int_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$ is a normalizing constant used to make the $p(\mathbf{x}|\mathbf{y})$ distribution *proper* (integrate to 1). Taking the negative logarithm of both sides of (4.33), we get

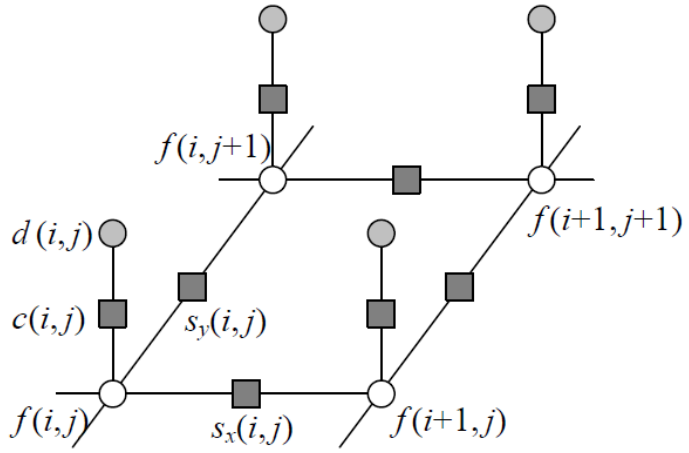
$$-\log p(\mathbf{x}|\mathbf{y}) = -\log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{x}) + C, \tag{4.34}$$

which is the *negative posterior log likelihood*.

To find the most likely (*maximum a posteriori* or MAP) solution $\mathbf{x}$ given some measurements $\mathbf{y}$, we simply minimize this negative log likelihood, which can also be thought of as an *energy*,

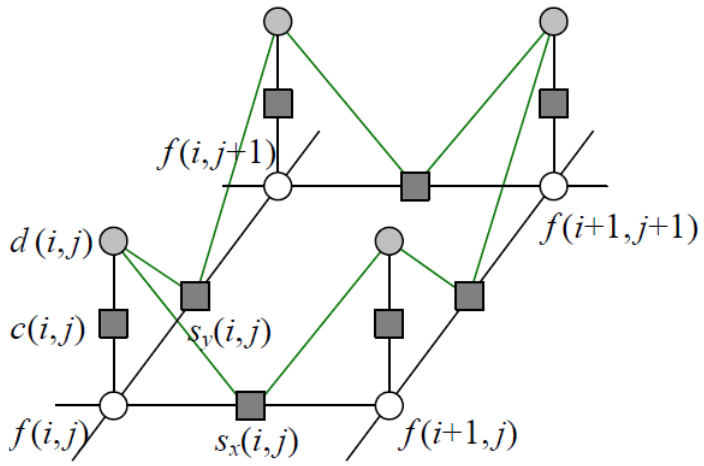
$$E(\mathbf{x}, \mathbf{y}) = E_{\mathrm{D}}(\mathbf{x}, \mathbf{y}) + E_{\mathrm{P}}(\mathbf{x}). \tag{4.35}$$

(We drop the constant C because its value does not matter during energy minimization.) The first term $E_{\mathrm{D}}(\mathbf{x}, \mathbf{y})$ is the *data energy* or *data penalty*; it measures the negative log likelihood that the data were observed given the unknown state $\mathbf{x}$. The second term $E_{\mathrm{P}}(\mathbf{x})$ is the *prior energy*; it plays a role analogous to the smoothness energy in regularization. Note



Conditional Random Fields

- Interaction potentials depend on guide image (sound familiar?)



Since the smoothness term now depends on the data, Bayes' Rule (4.44) no longer applies. Instead, we use a direct model for the posterior distribution $p(\mathbf{x}|\mathbf{y})$, whose negative log likelihood can be written as

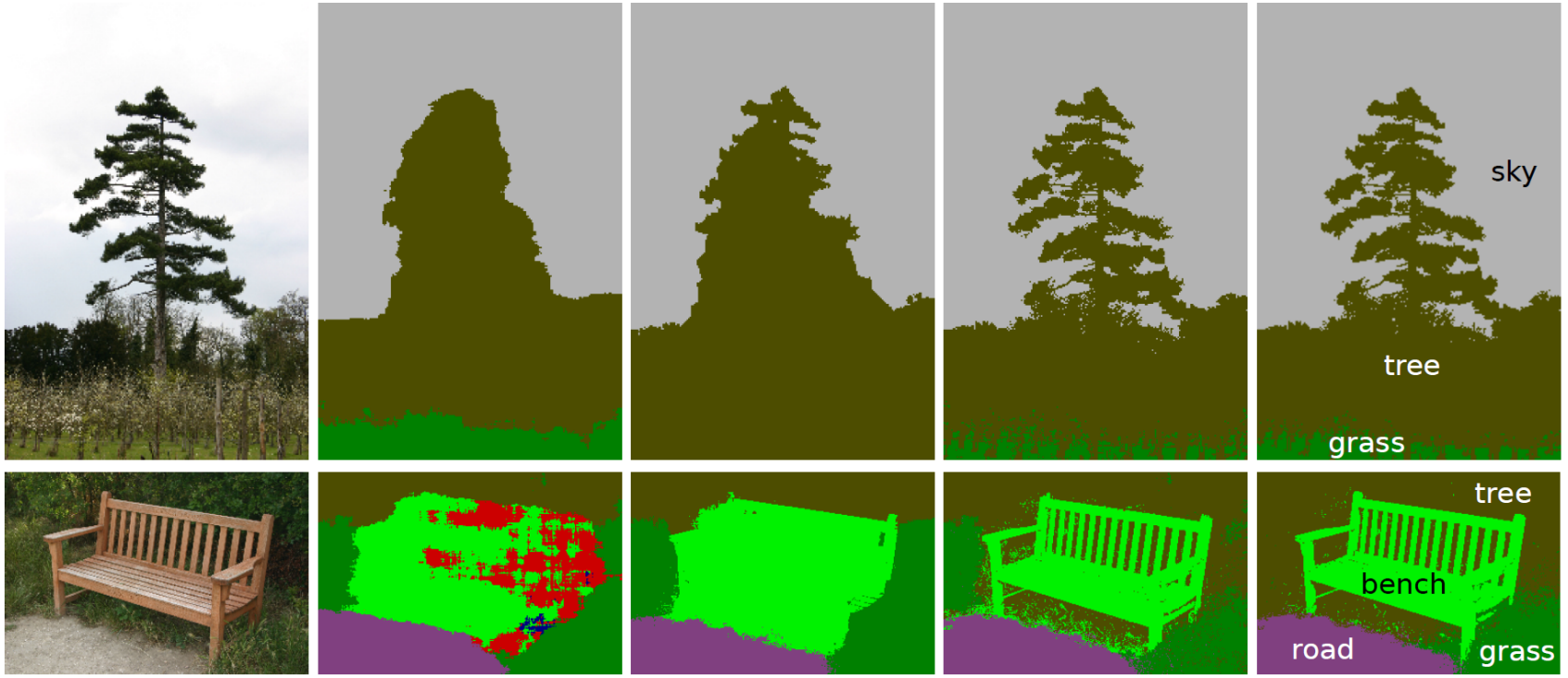
$$E(\mathbf{x}|\mathbf{y}) = E_{\mathrm{D}}(\mathbf{x},\mathbf{y}) + E_{\mathrm{S}}(\mathbf{x},\mathbf{y})$$
$$= \sum_p V_p(x_p,\mathbf{y}) + \sum_{(p,q)\in\mathcal{N}} V_{p,q}(x_p,x_q,\mathbf{y}), \qquad (4.45)$$

using the notation introduced in (4.43). The resulting probability distribution is called a *conditional random field* (CRF) and was first introduced to the computer vision field by Kumar and Hebert (2003), based on earlier work in text modeling by Lafferty, McCallum, and Pereira (2001).

Figure 4.18 shows a graphical model where the smoothness terms depend on the data values. In this particular model, each smoothness term depends only on its adjacent pair of data values, i.e., terms are of the form $V_{p,q}(x_p,x_q,y_p,y_q)$ in (4.45).

Dense CRF [Krähenbühl and Koltun 2011]

- Use a wide color-based support region (like bilateral solver)



(a) Image (b) Unary classifiers (c) Robust P^n CRF (d) Fully connected CRF, MCMC inference, 36 hrs (e) Fully connected CRF, our approach, 0.2 seconds

Wrapping up

- Interpolation
- Pyramids
- Blending
- Resampling (rotations, etc.)
- Data Fitting
- Regularization and variational techniques
- Markov Random Fields