University of Nottingham
UK | CHINA | MALAYSIA

COMP3065 Computer Vision

Topic 5 – Image Stitching 2

Dr. Tianxiang Cui

2025 Spring

Image Stitching: The Idea

1. Take a sequence of images from the same position.
2. To stitch two images: compute transformation between second image and first.
3. Shift (warp) the second image to overlap with the first.
4. Blend the two together to create a mosaic.
5. If there are more images, repeat step 2 to 4.

Outline

- Image Warping
  - Forward Warping
  - Inverse Warping
- Image Blending
  - Feathering
  - Alpha Blending
  - Pyramid Blending
  - Multiband Blending
- Recognizing Panoramas

Image (parametric) Warping

- Change every pixel locations to create a new image (global)
- Examples of parametric warps:

translation

rotation

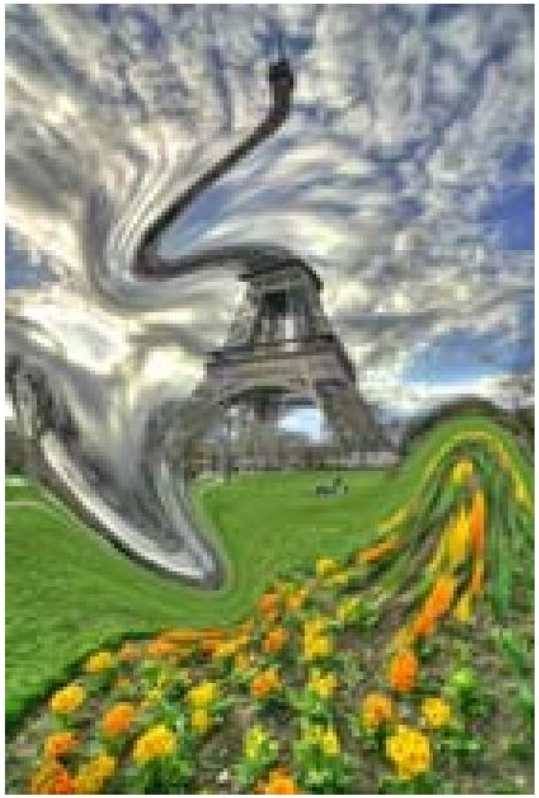
Scale

affine

projective

Nonparametric (local) Warping

Original



Warped

Image Warping

- Move pixels of an image
- Given a coordinate transform $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ and a source image $\boldsymbol{f}(\boldsymbol{x})$,
- We compute a transformed image $\boldsymbol{g}(\boldsymbol{x'}) = \boldsymbol{f}(\boldsymbol{h}(\boldsymbol{x}))$.
  - Change the domain of image function

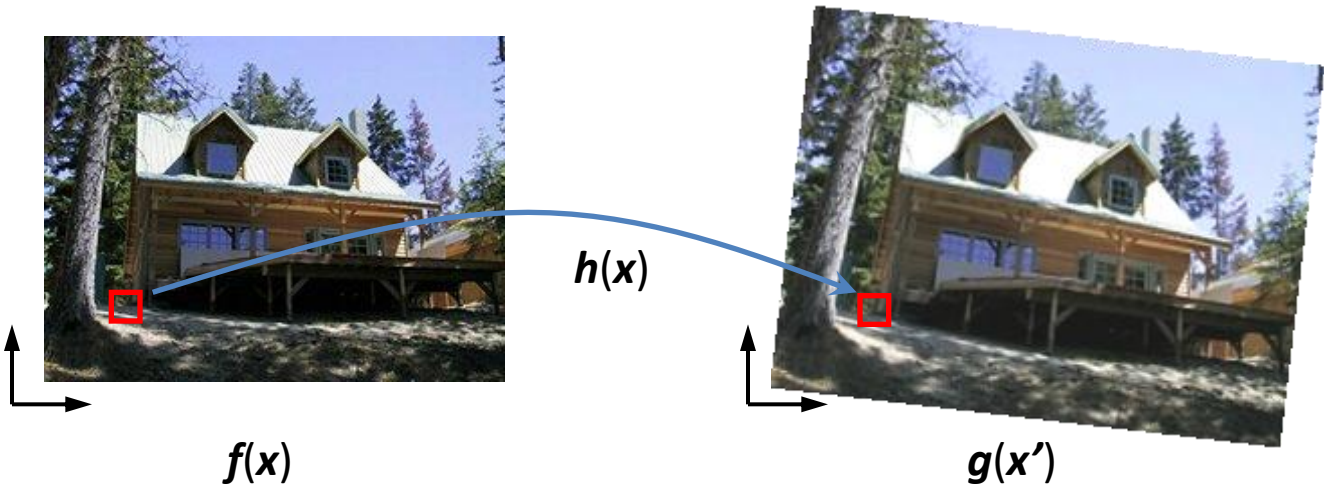


Image Warping

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

$$g(x) = h(f(x))$$

$h(y)=0.5y+0.5$

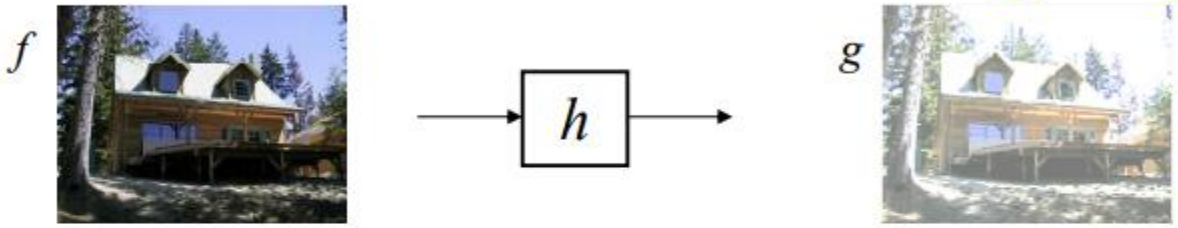


image warping: change ***domain*** of image

$$g(x) = f(h(x))$$

$h([x,y])=[x,y/2]$

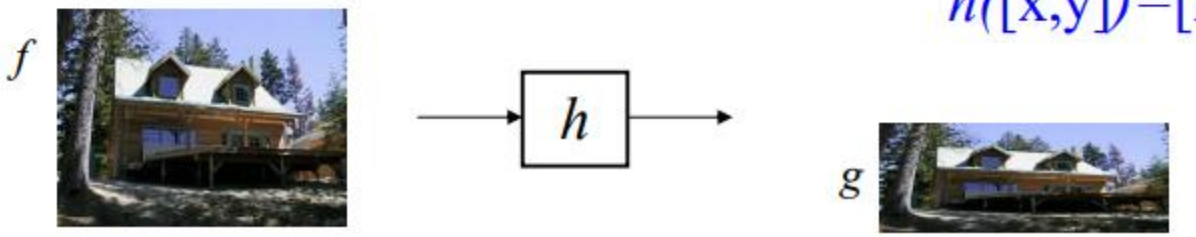


Image Warping

- How do we specify where every pixel goes?
  - Describe the destination pixel for every source, and vice-versa
  - **Mapping**

- How do we compute colors at dest pixels?
  - **Resampling**

Forward Warping

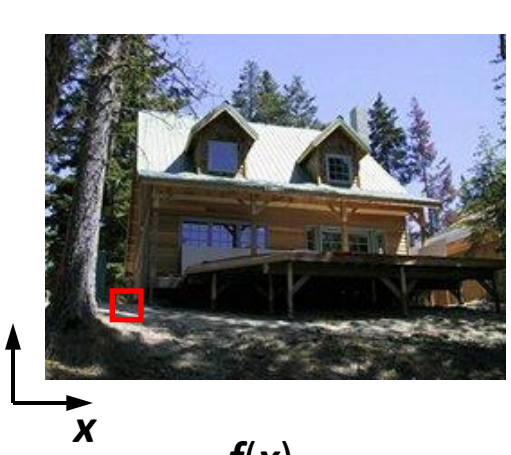
- Send each pixel $f(x)$ --the RGB value-- to its corresponding location in the dest image: $x' = h(x)$ in $g(x')$.

$f(x)$

$g(x')$

Forward Warping

- Send each pixel $f(x)$ --the RGB value-- to its corresponding location in the dest image: $x' = h(x)$ in $g(x')$.

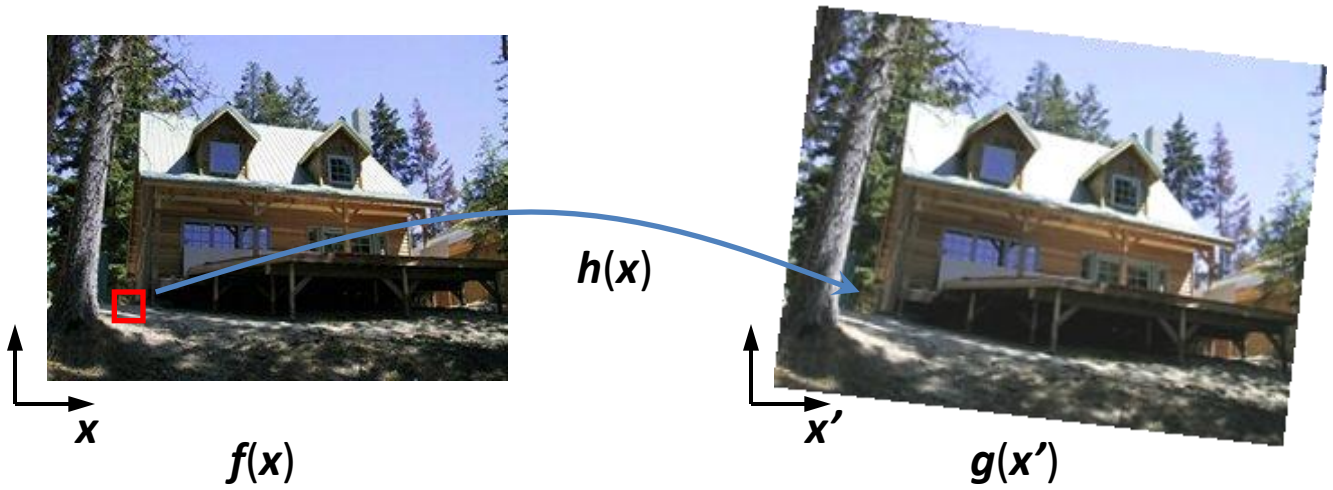


$f(x)$

$g(x')$

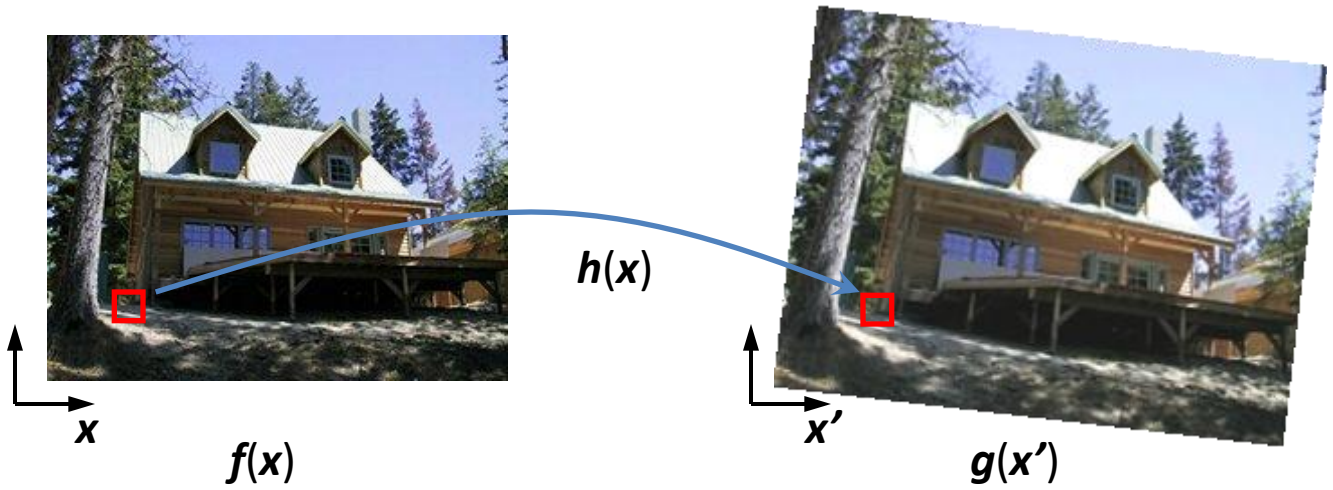
Forward Warping

- Send each pixel $f(x)$ --the RGB value-- to its corresponding location in the dest image: $x' = h(x)$ in $g(x')$.



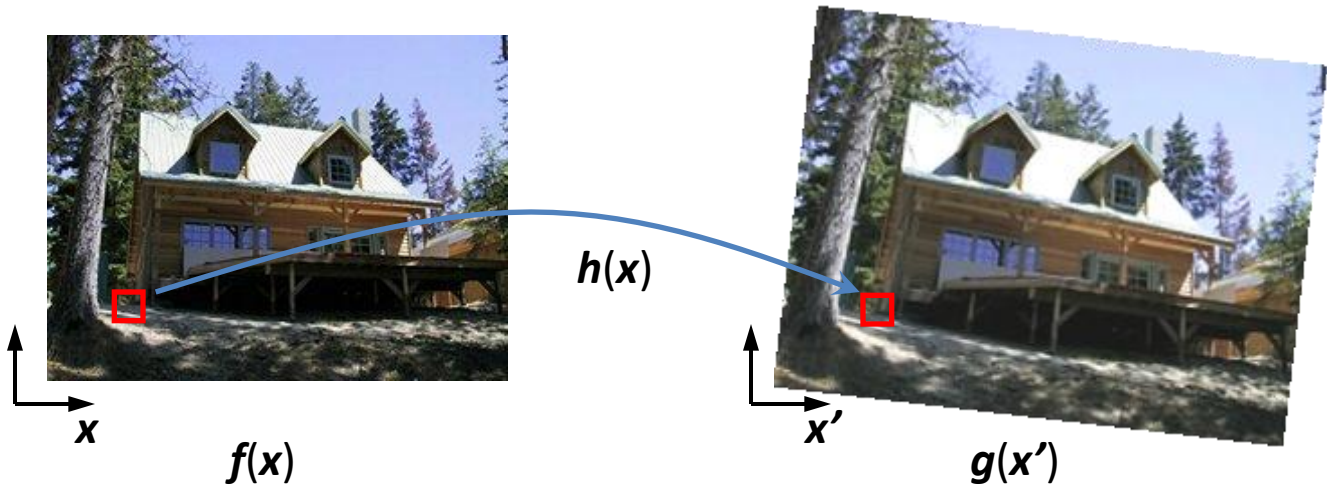
Forward Warping

- Send each pixel $f(x)$ --the RGB value-- to its corresponding location in the dest image: $x' = h(x)$ in $g(x')$.



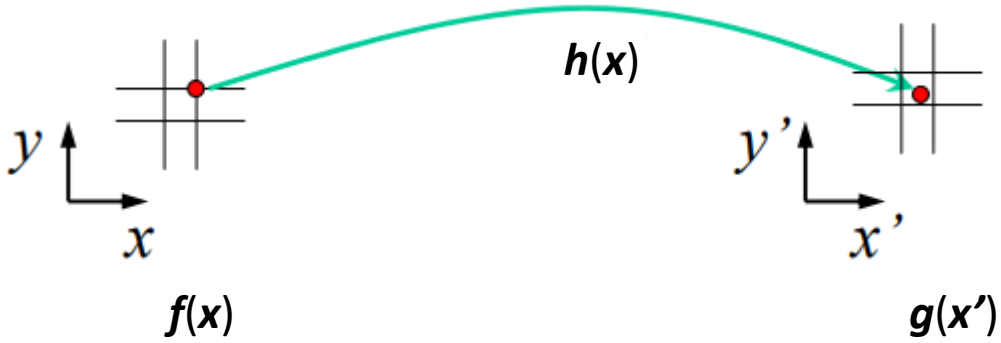
Forward Warping

- Send each pixel $f(x)$ --the RGB value-- to its corresponding location in the dest image: $x' = h(x)$ in $g(x')$.
  - What if pixel lands "between" two pixels?



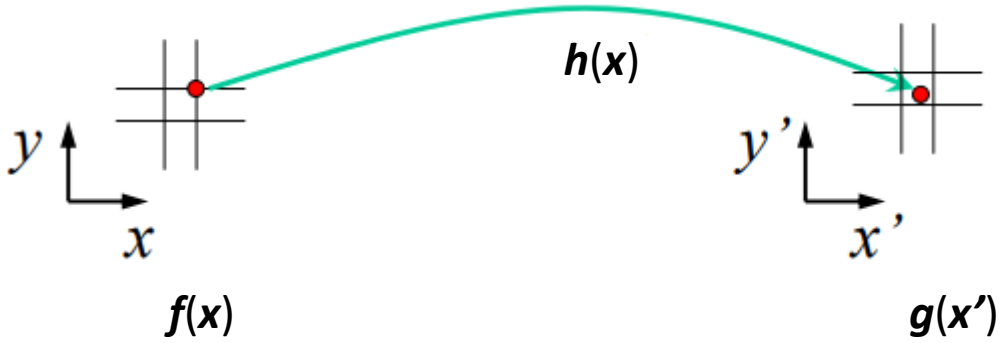
Forward Warping

- Send each pixel $f(x)$ --the RGB value-- to its corresponding location in the dest image: $x' = h(x)$ in $g(x')$.
  - What if pixel lands "between" two pixels?



Forward Warping

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



Inverse Warping

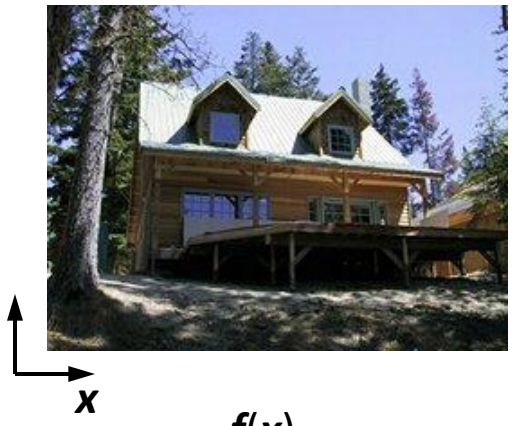
- Get each pixel $g(x')$ --the RGB value-- from its corresponding location in the source image: $x = h^{-1}(x')$ in $f(x)$.

$f(x)$

$g(x')$

Inverse Warping

- Get each pixel $g(x')$ --the RGB value-- from its corresponding location in the source image: $x = h^{-1}(x')$ in $f(x)$.

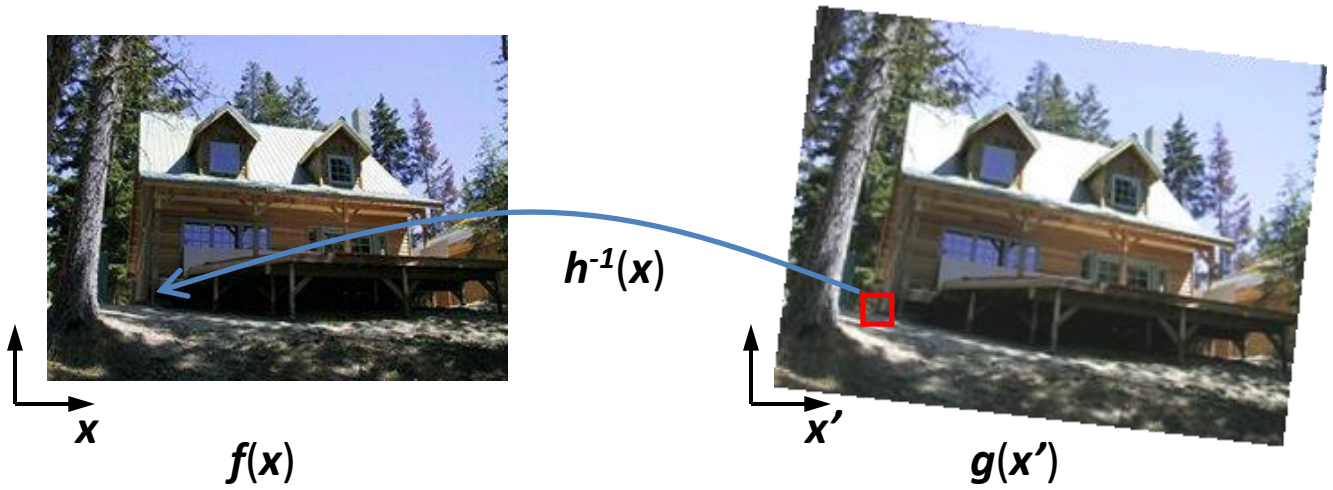


$f(x)$

$g(x')$

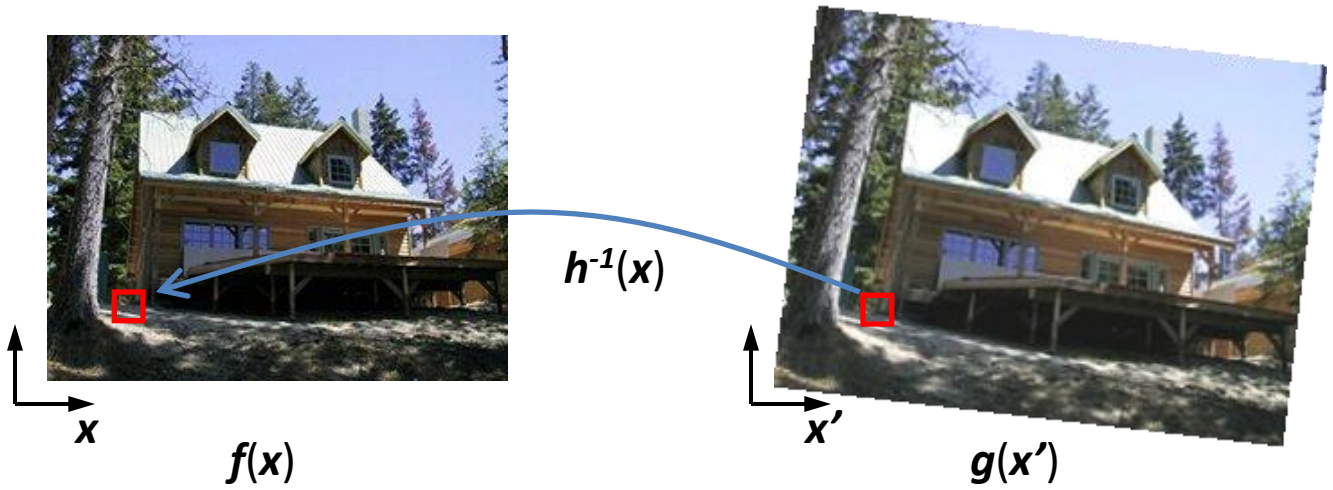
Inverse Warping

- Get each pixel $g(x')$ --the RGB value-- from its corresponding location in the source image: $x = h^{-1}(x')$ in $f(x)$.



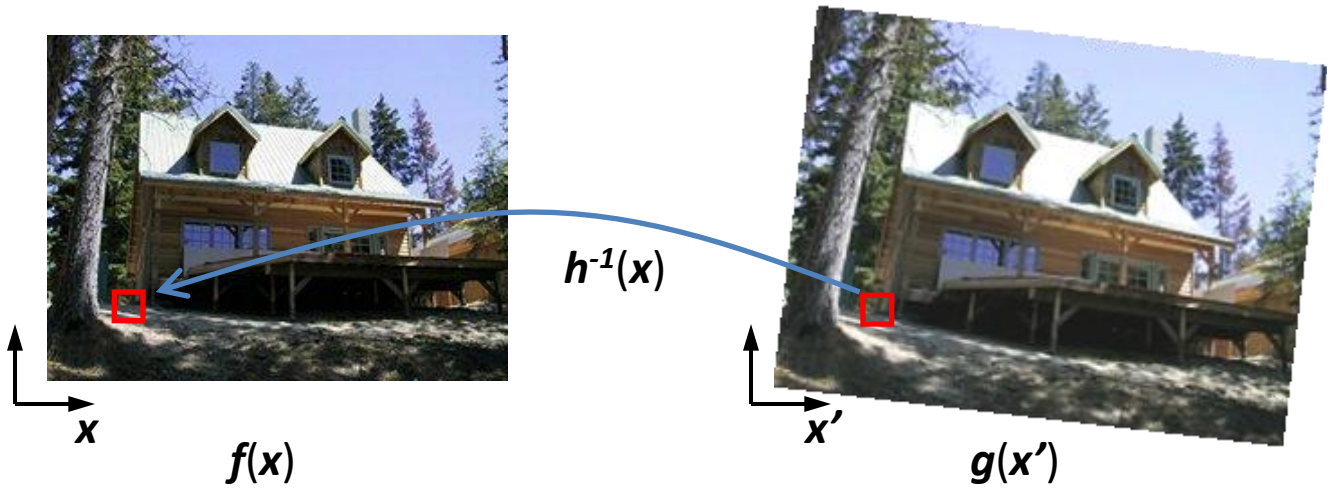
Inverse Warping

- Get each pixel $g(x')$ --the RGB value-- from its corresponding location in the source image: $x = h^{-1}(x')$ in $f(x)$.



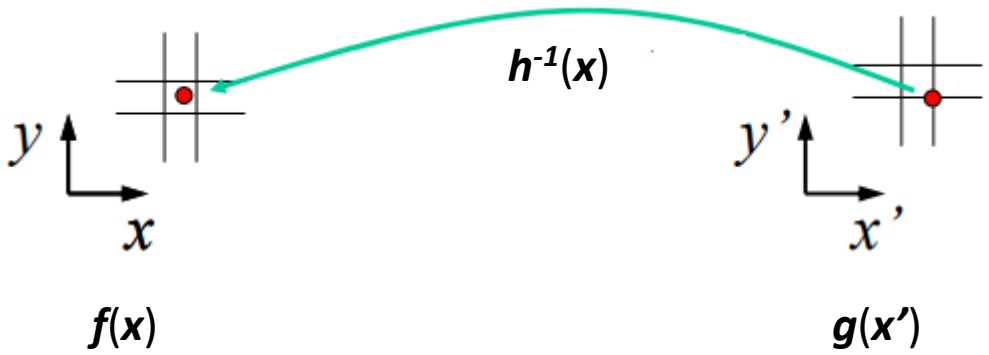
Inverse Warping

- Get each pixel $g(x')$ --the RGB value-- from its corresponding location in the source image: $x = h^{-1}(x')$ in $f(x)$.
  - What if pixel comes from "between" two pixels?



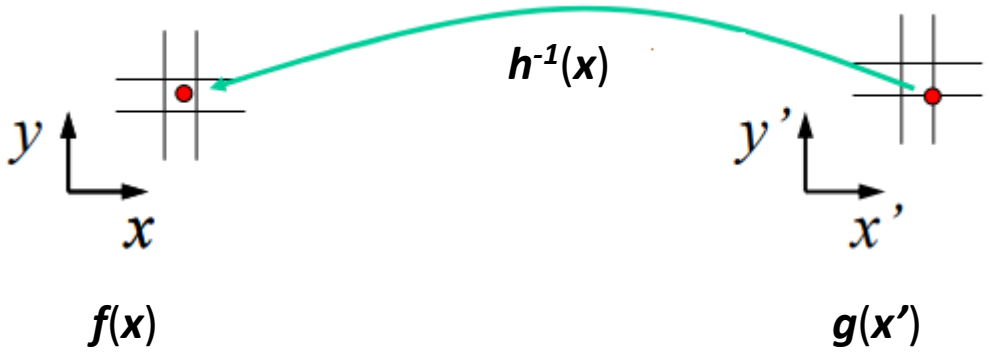
Inverse Warping

- Get each pixel $g(x')$ --the RGB value-- from its corresponding location in the source image: $x = h^{-1}(x')$ in $f(x)$.
  - What if pixel comes from "between" two pixels?



Inverse Warping

- Get each pixel $g(x')$ --the RGB value-- from its corresponding location in the source image: $x = h^{-1}(x')$ in $f(x)$.
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* source image.

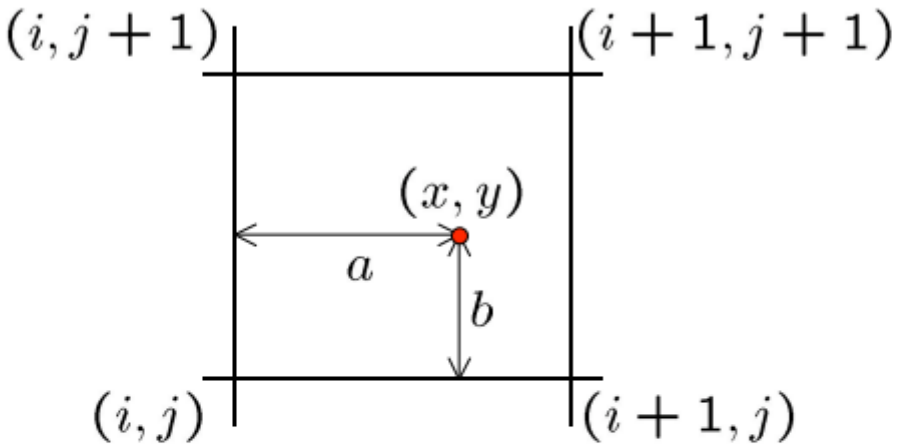


Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear

Bilinear Interpolation

Sampling at *f(x,y):*



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

Bilinear Interpolation

The solution can also be written as a weighted mean of the $f(Q)$:

$$f(x,y) \approx w_{11} f(Q_{11}) + w_{12} f(Q_{12}) + w_{21} f(Q_{21}) + w_{22} f(Q_{22}),$$

where the weights sum to 1 and satisfy the transposed linear system

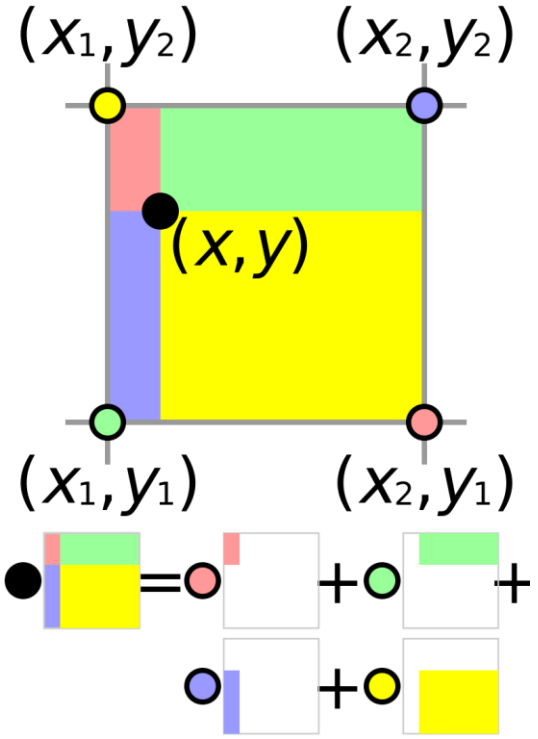
$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_1 & x_2 & x_2 \\ y_1 & y_2 & y_1 & y_2 \\ x_1 y_1 & x_1 y_2 & x_2 y_1 & x_2 y_2 \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{21} \\ w_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix},$$

yielding the result

$$\begin{bmatrix} w_{11} \\ w_{21} \\ w_{12} \\ w_{22} \end{bmatrix} = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 y_2 & -y_2 & -x_2 & 1 \\ -x_2 y_1 & y_1 & x_2 & -1 \\ -x_1 y_2 & y_2 & x_1 & -1 \\ x_1 y_1 & -y_1 & -x_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix},$$

which simplifies to

$$\begin{aligned} w_{11} &= (x_2 - x)(y_2 - y)/(x_2 - x_1)(y_2 - y_1), \\ w_{12} &= (x_2 - x)(y - y_1)/(x_2 - x_1)(y_2 - y_1), \\ w_{21} &= (x - x_1)(y_2 - y)/(x_2 - x_1)(y_2 - y_1), \\ w_{22} &= (x - x_1)(y - y_1)/(x_2 - x_1)(y_2 - y_1), \end{aligned}$$



Forward vs. inverse warping

- Which one is better?

- Usually inverse
  - But it is not always possible to get an invertible warp function

Image Warping

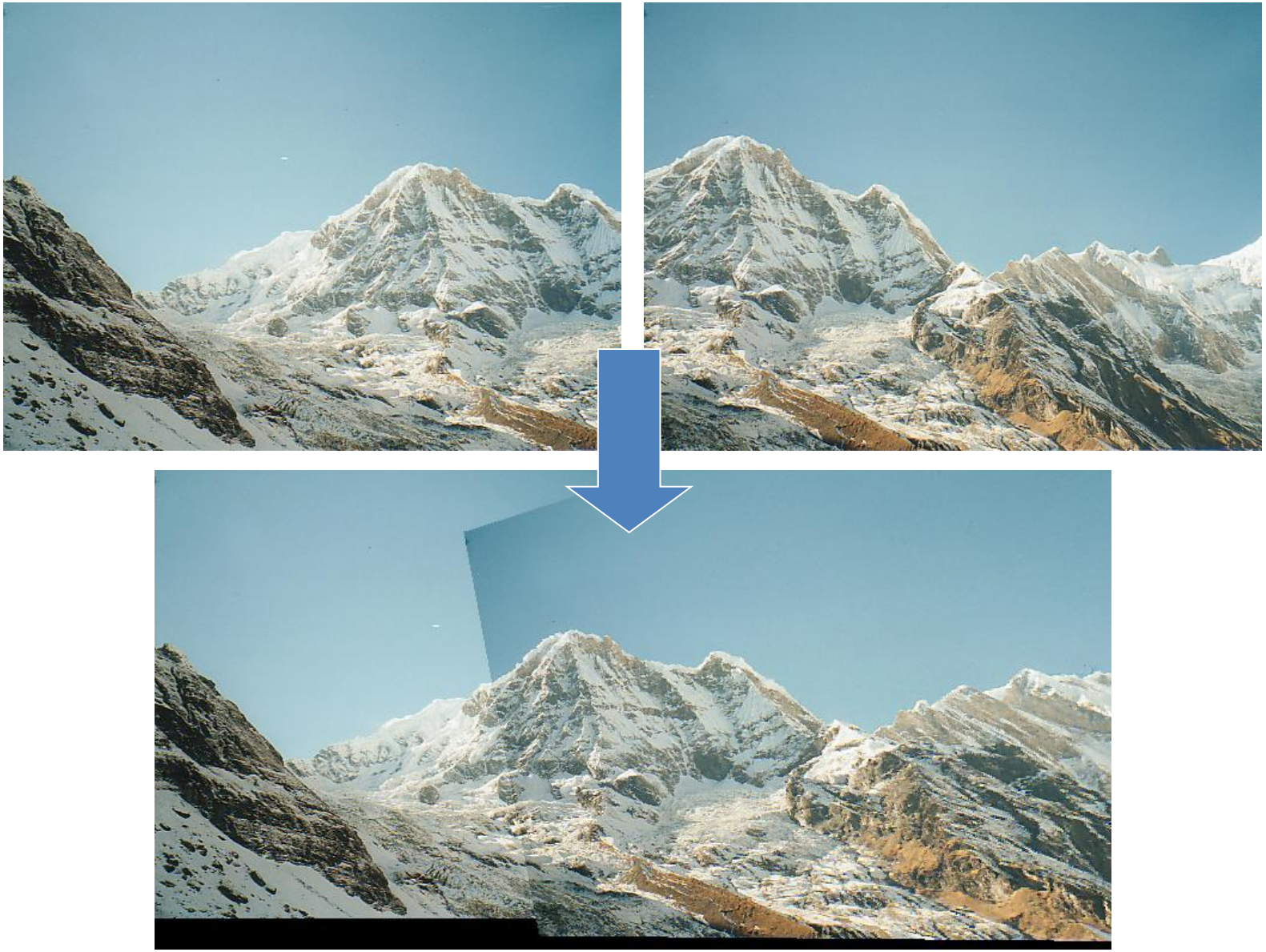


Image Blending

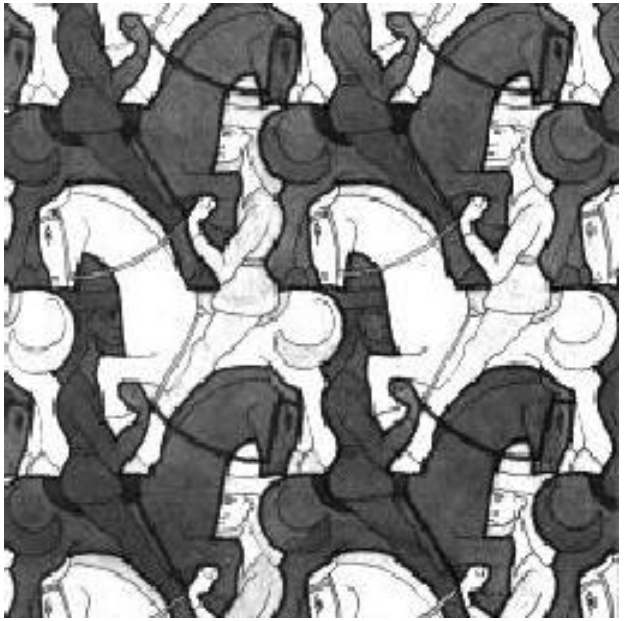
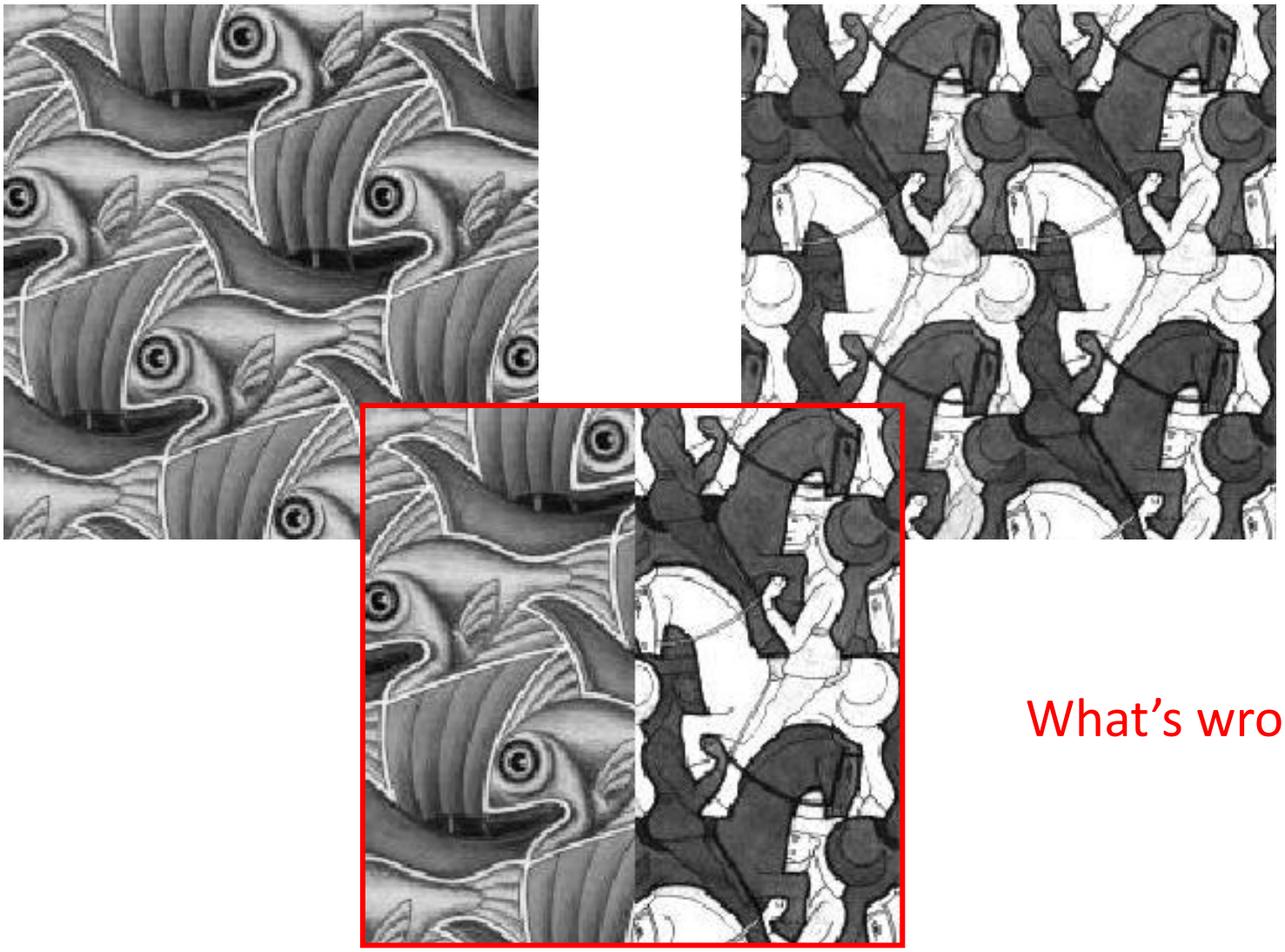
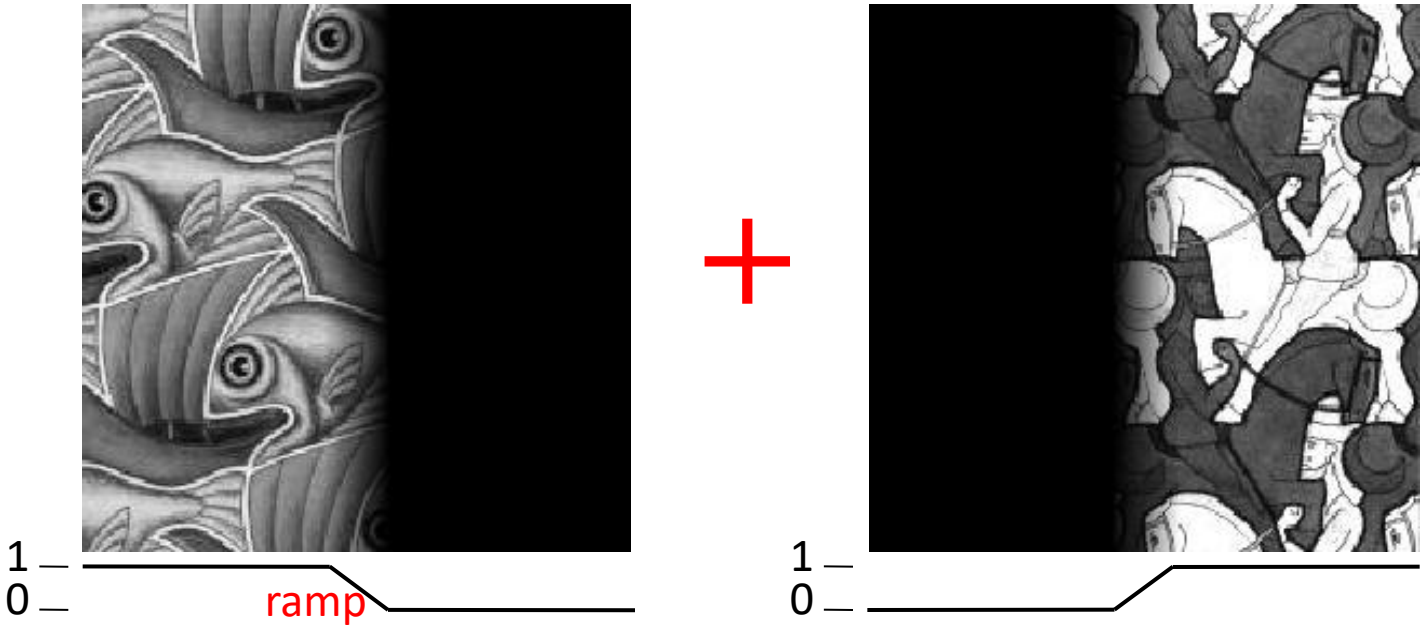


Image Blending



What's wrong?

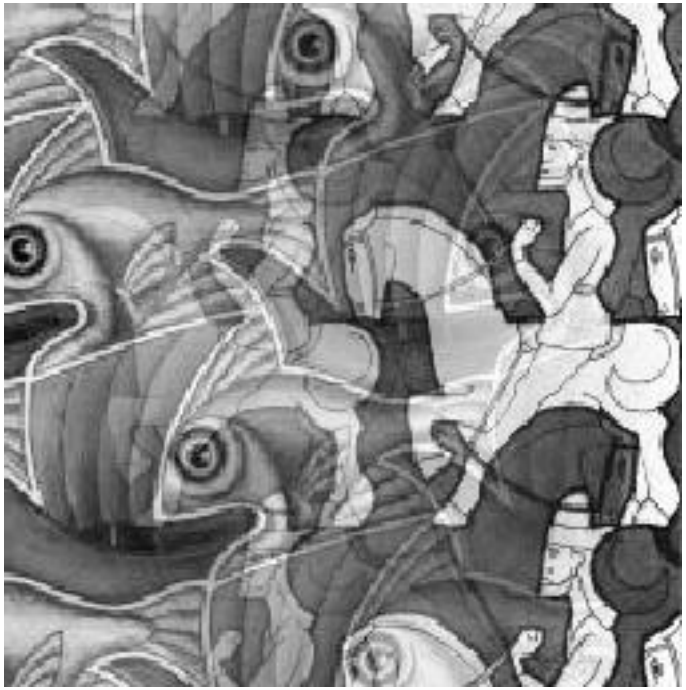
Feathering

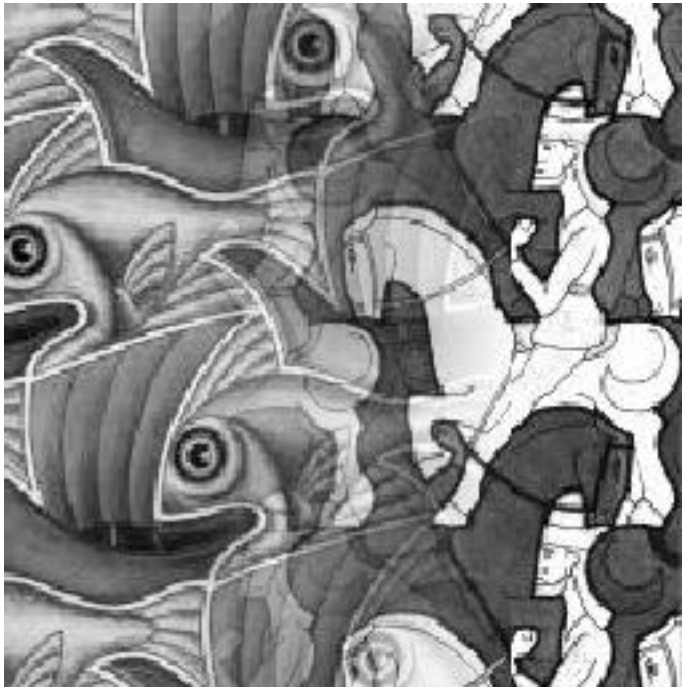


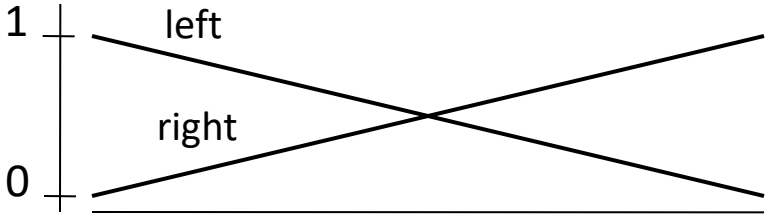
$$I_{blend} = c * I_{left} + (1 - c) * I_{right}$$

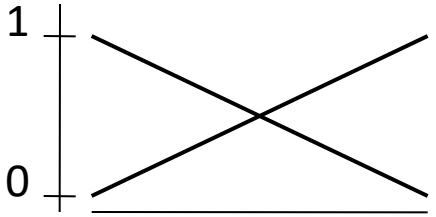
c can be 1 (object 1), and 0 (object 2), or somewhere in between

Effect of Window (ramp-width) Size

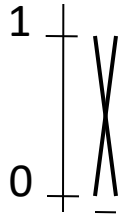


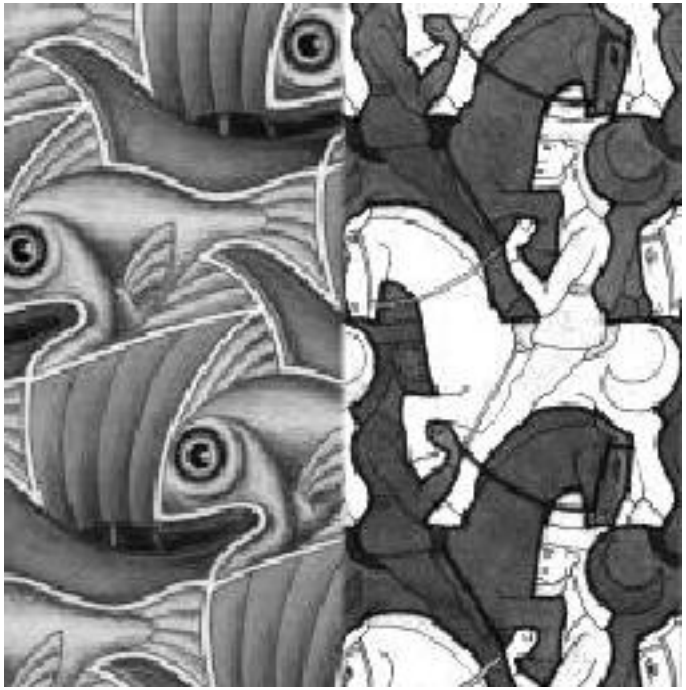


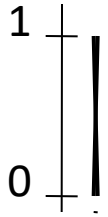




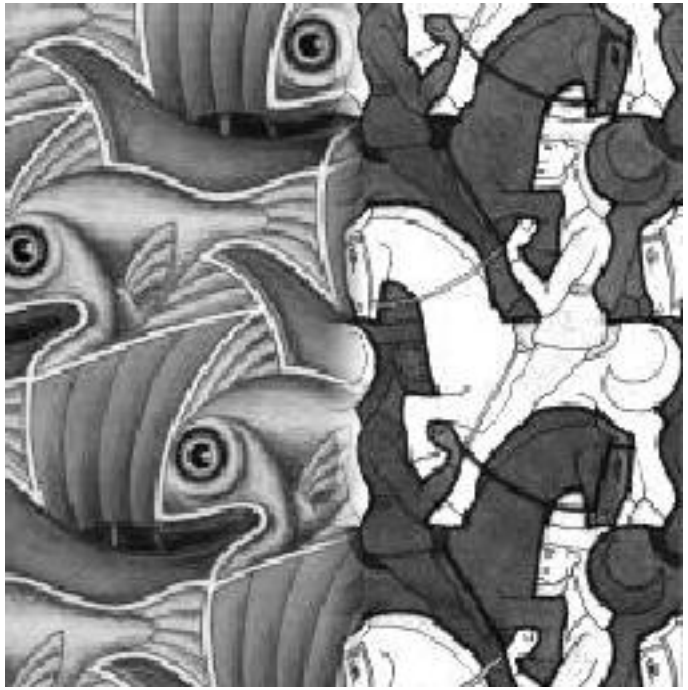
Effect of Window Size







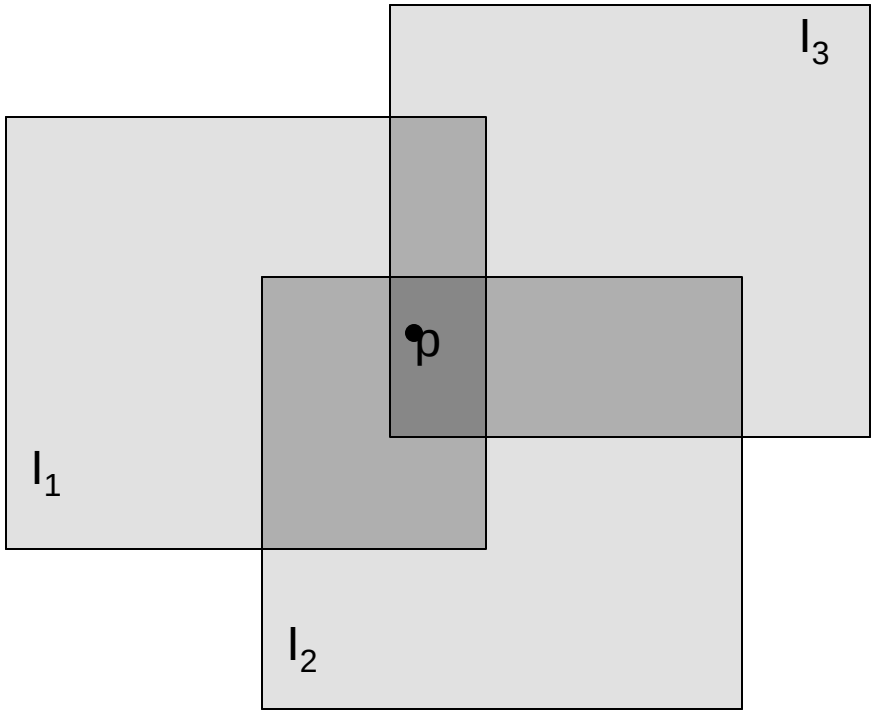
Good Window Size



“Optimal” window: smooth but not ghosted

- Doesn’t always work...

Alpha Blending



Encoding blend weights: $I(x,y) = (\alpha_1 R, \alpha_2 G, \alpha_3 B, \boldsymbol{\alpha})$

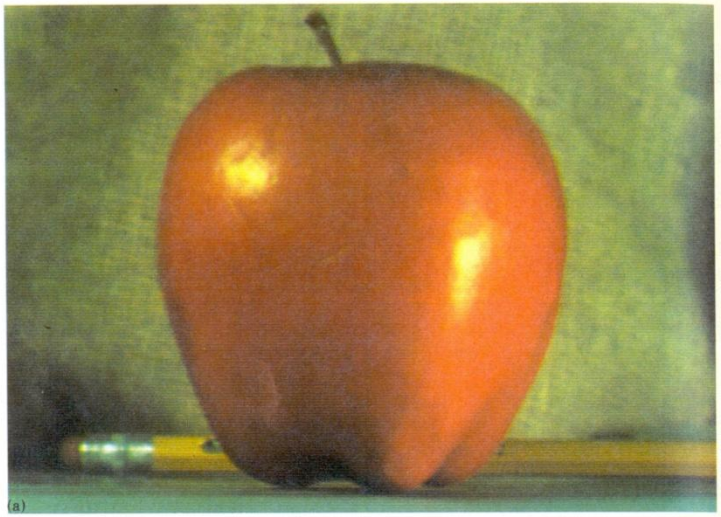
color at p = $$\frac{(\alpha_1 R_1,\ \alpha_1 G_1,\ \alpha_1 B_1) + (\alpha_2 R_2,\ \alpha_2 G_2,\ \alpha_2 B_2) + (\alpha_3 R_3,\ \alpha_3 G_3,\ \alpha_3 B_3)}{\alpha_1 + \alpha_2 + \alpha_3}$$

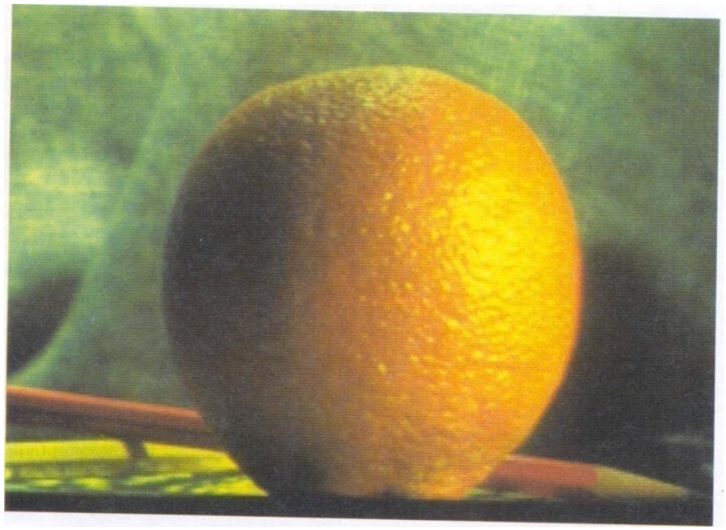
Implement this in two steps:

1. accumulate: add up the (α premultiplied) RGB values at each pixel.
2. normalize: divide each pixel's accumulated RGB by its α value.

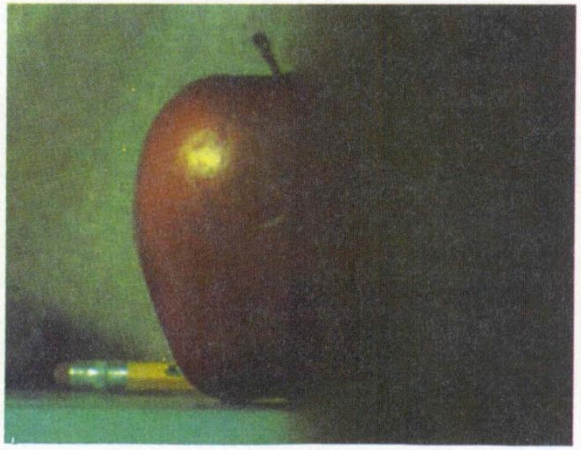
Pyramid Blending

apple

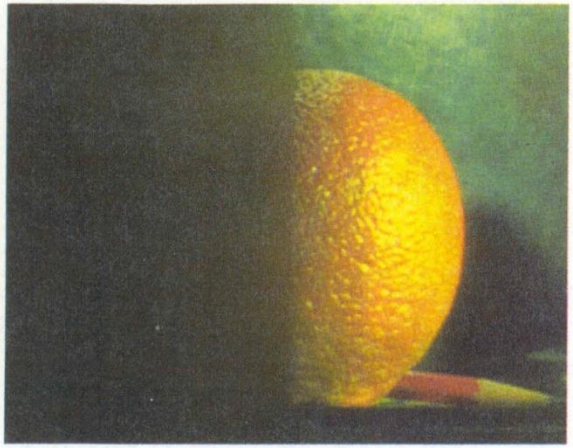




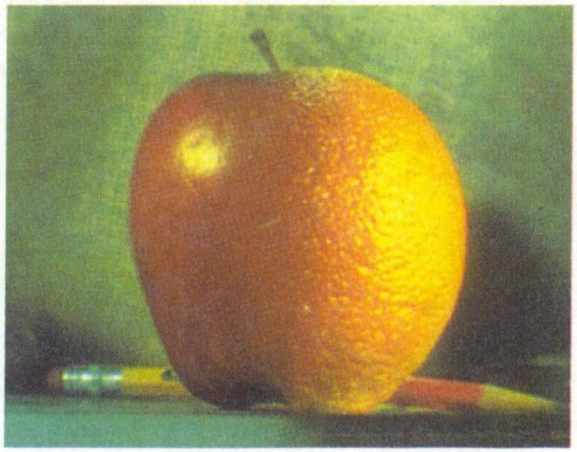
orange



(d)



(h)



(l)

Create a Laplacian pyramid, blend each level.

Forming a Gaussian Pyramid

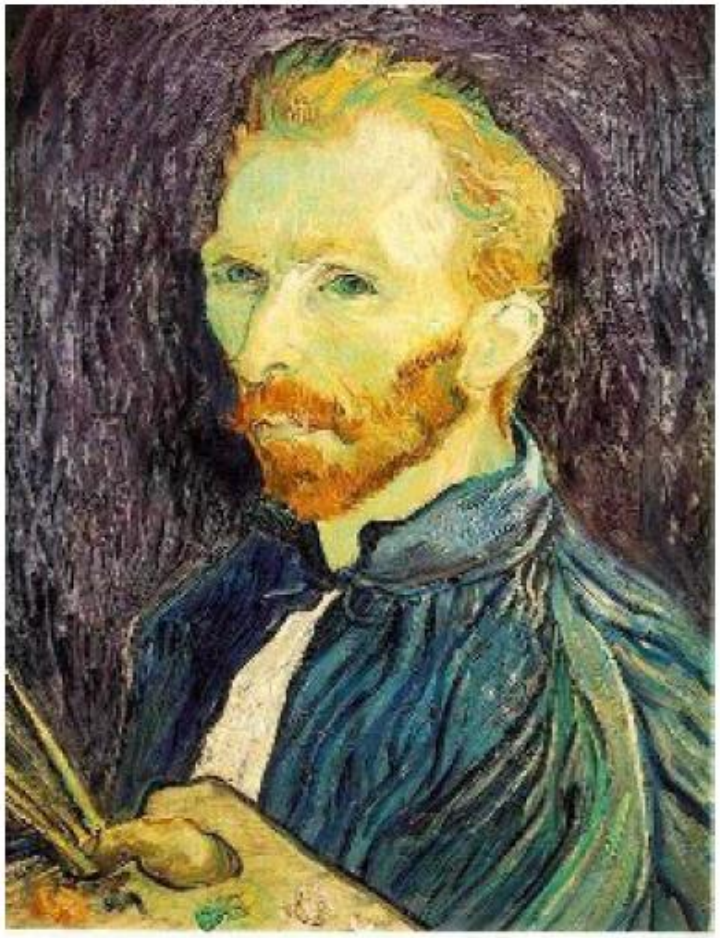
- Start with the original image G_0 .
- Perform a Gaussian filtering about each pixel, down sampling so that the result is a reduced image of half the size in each dimension.
- Do this all the way up the pyramid.

$G_l = \text{REDUCE}(G_{l-1})$

Forming a Gaussian Pyramid

Gaussian Pyramid

Image Sub-Sampling

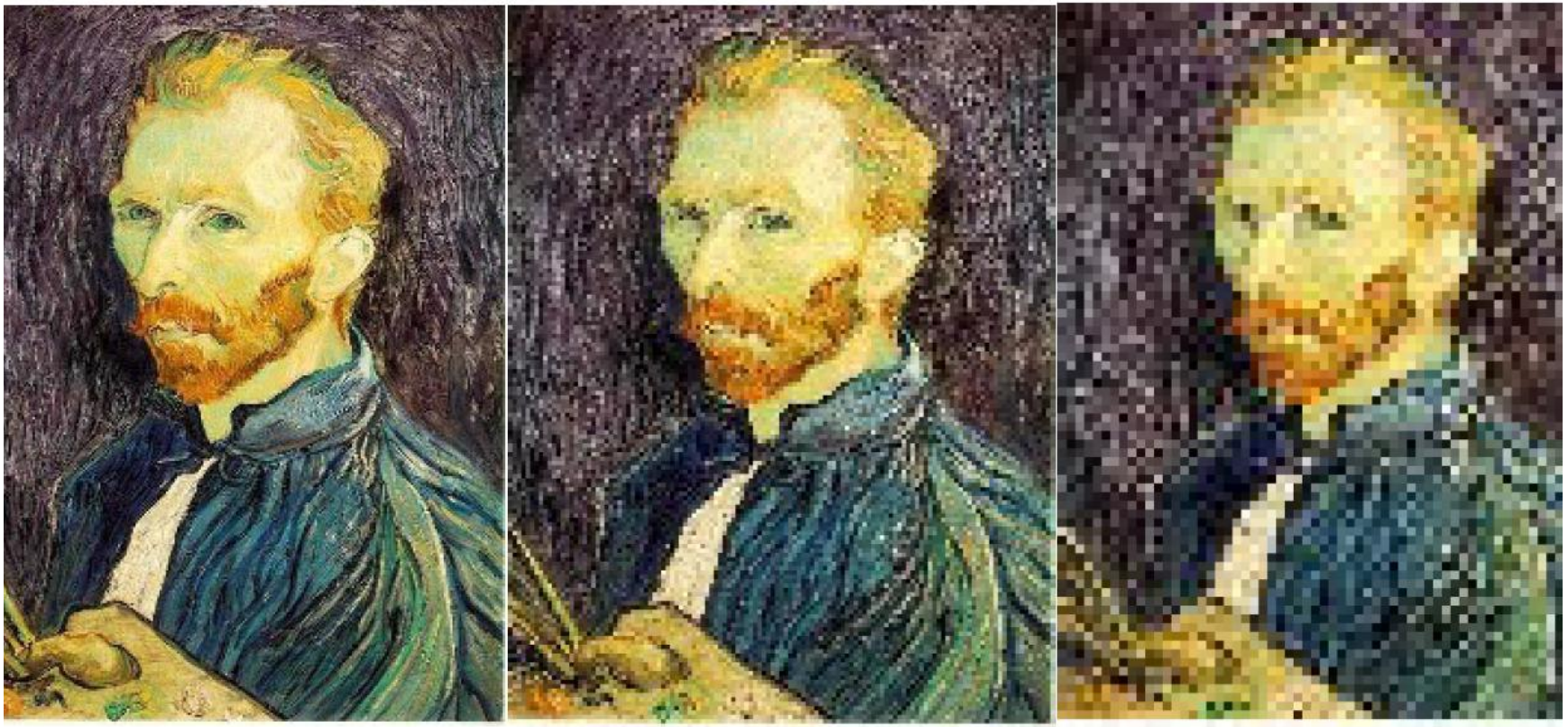


1/4

1/8

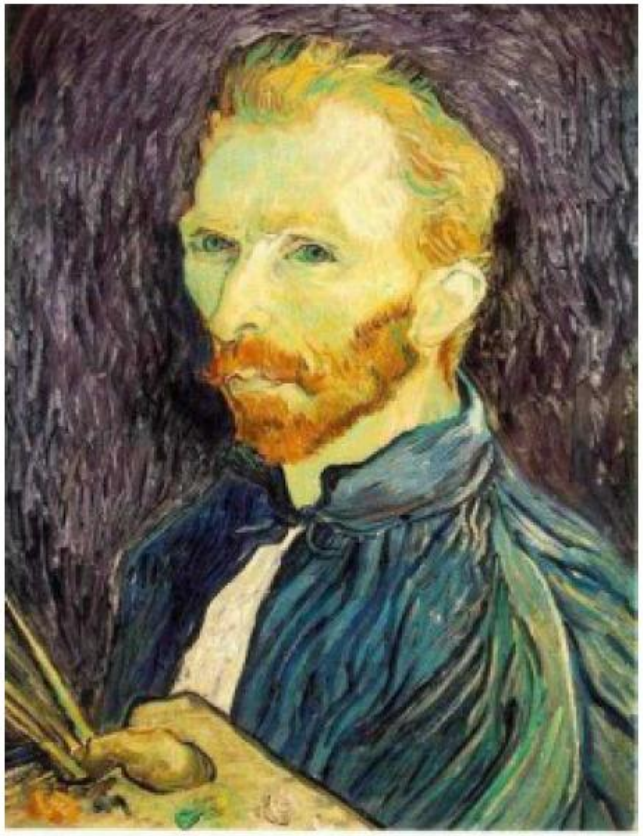
Throw away every other row and column to create a half size image

Image Sub-Sampling

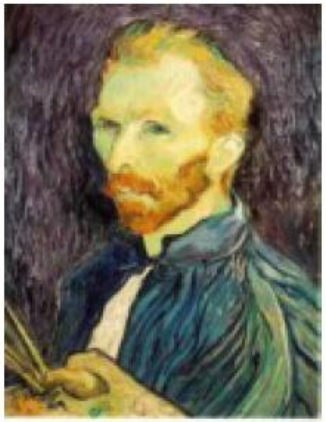


1/2 1/4 (2x zoom) 1/8 (4x zoom)

Gaussian Pre-filtering



Gaussian 1/2

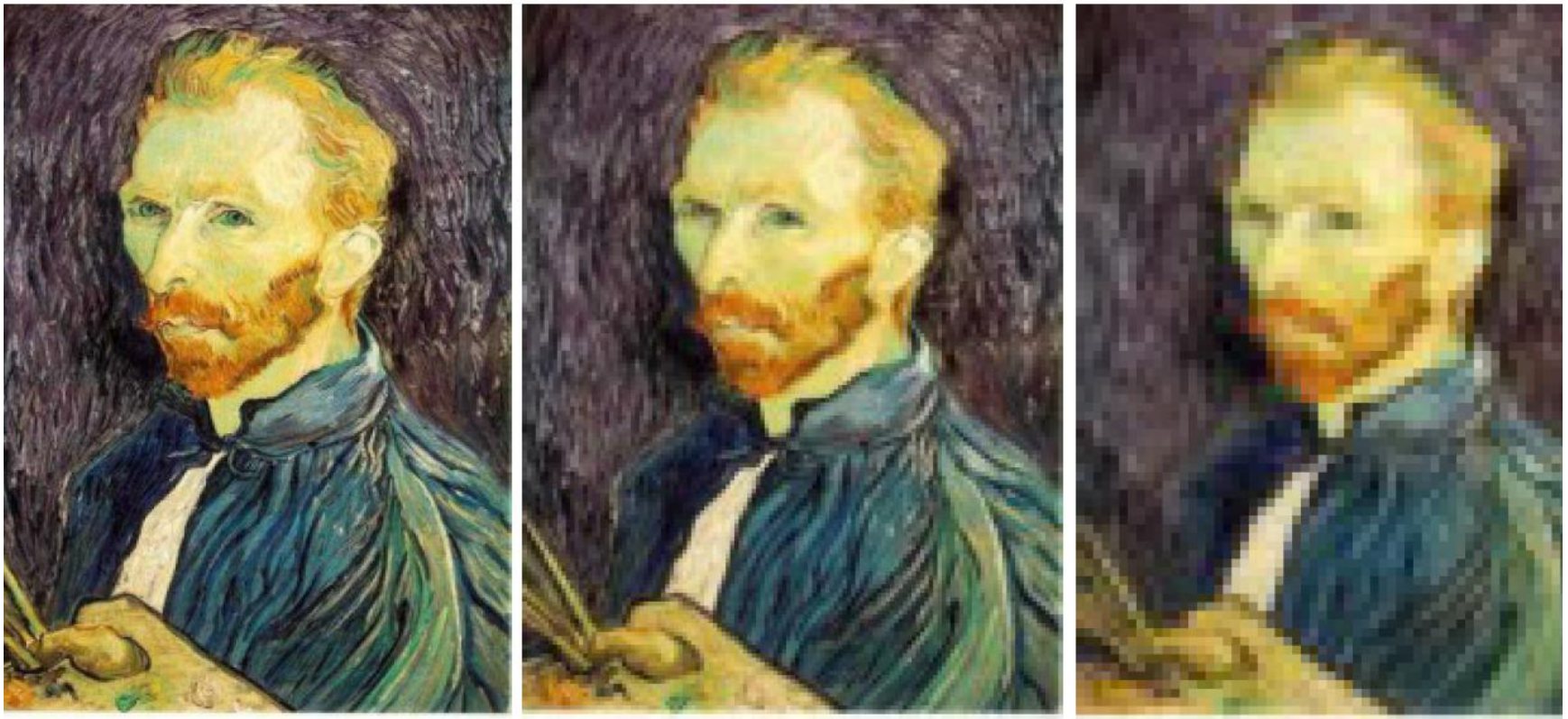


G 1/4

G 1/8

Filter the image, then subsample

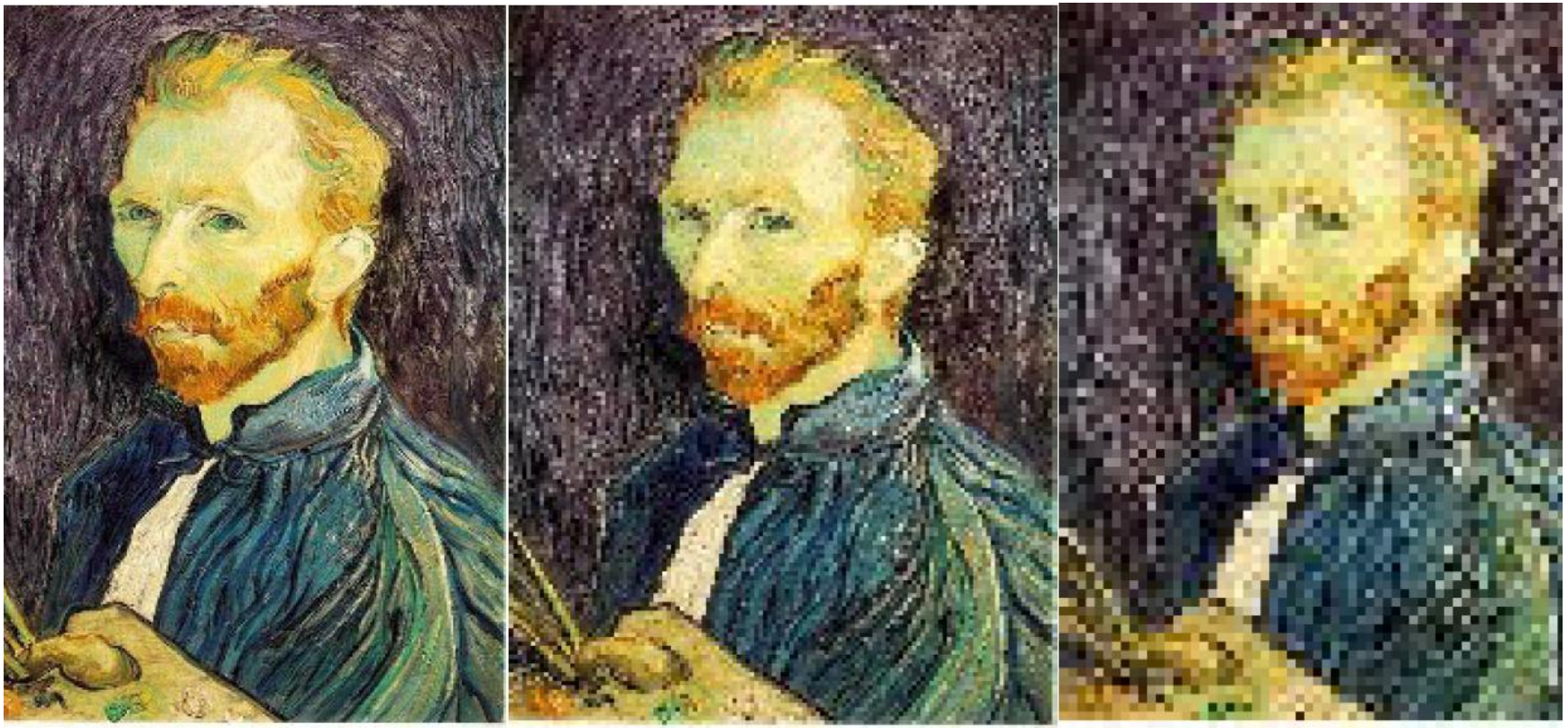
Gaussian Pre-filtering



Gaussian 1/2 G 1/4 G 1/8

Filter the image, then subsample

Compared with…



1/2 1/4 (2x zoom) 1/8 (4x zoom)

Making the Laplacians

- We want to subtract each level of the pyramid from the next lower one.
- But they are different sizes!
- In order to do the subtraction, we perform an interpolation process.
- We interpolate new samples between those of a given image to make it big enough to subtract.
- The operation is called EXPAND.

Forming a Gaussian Pyramid

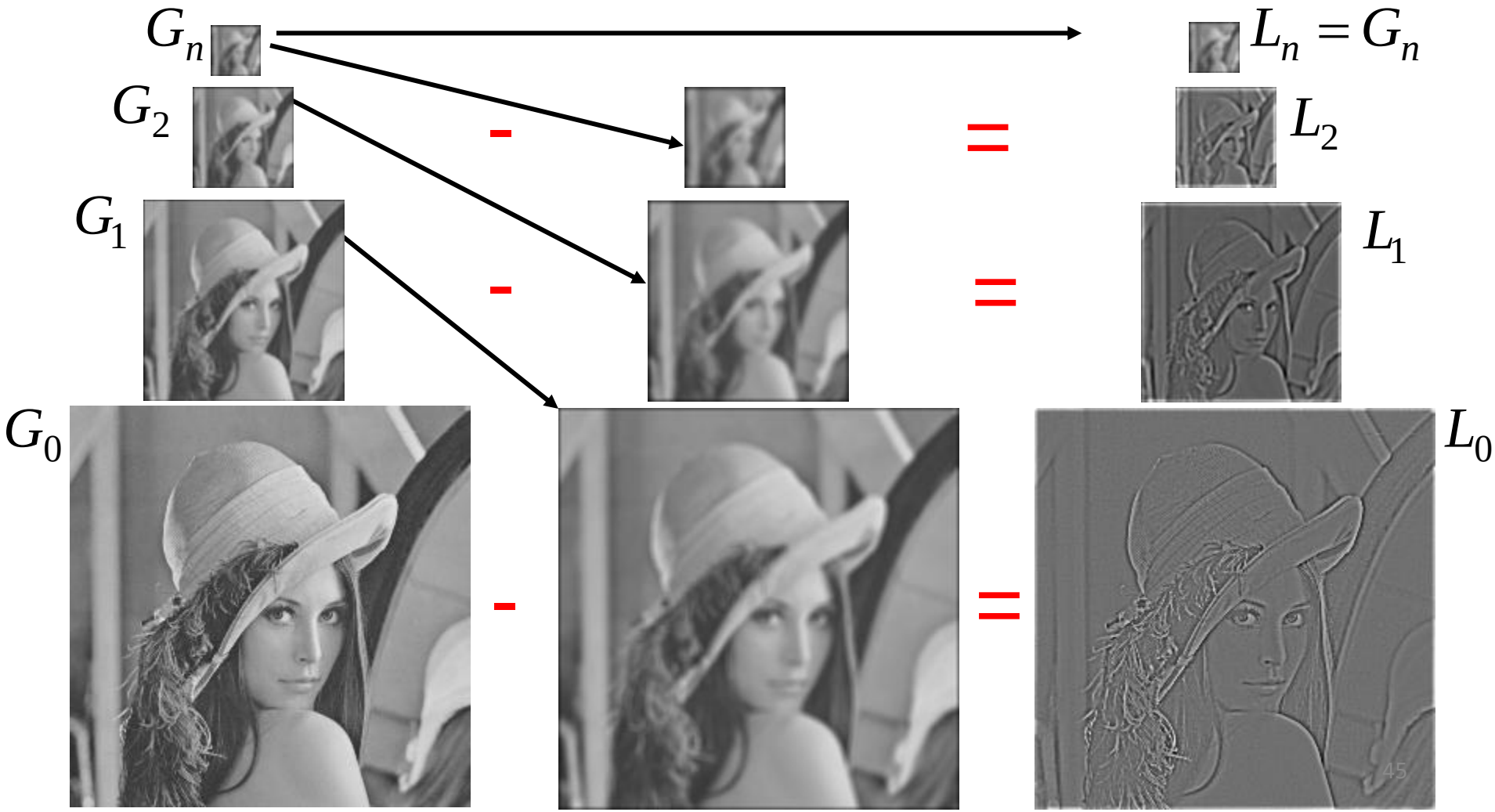
Gaussian Pyramid

The Laplacian Pyramid

$$L_i = G_i - \text{expand}(G_{i+1})$$

Gaussian Pyramid

Laplacian Pyramid



Forming the New Pyramid

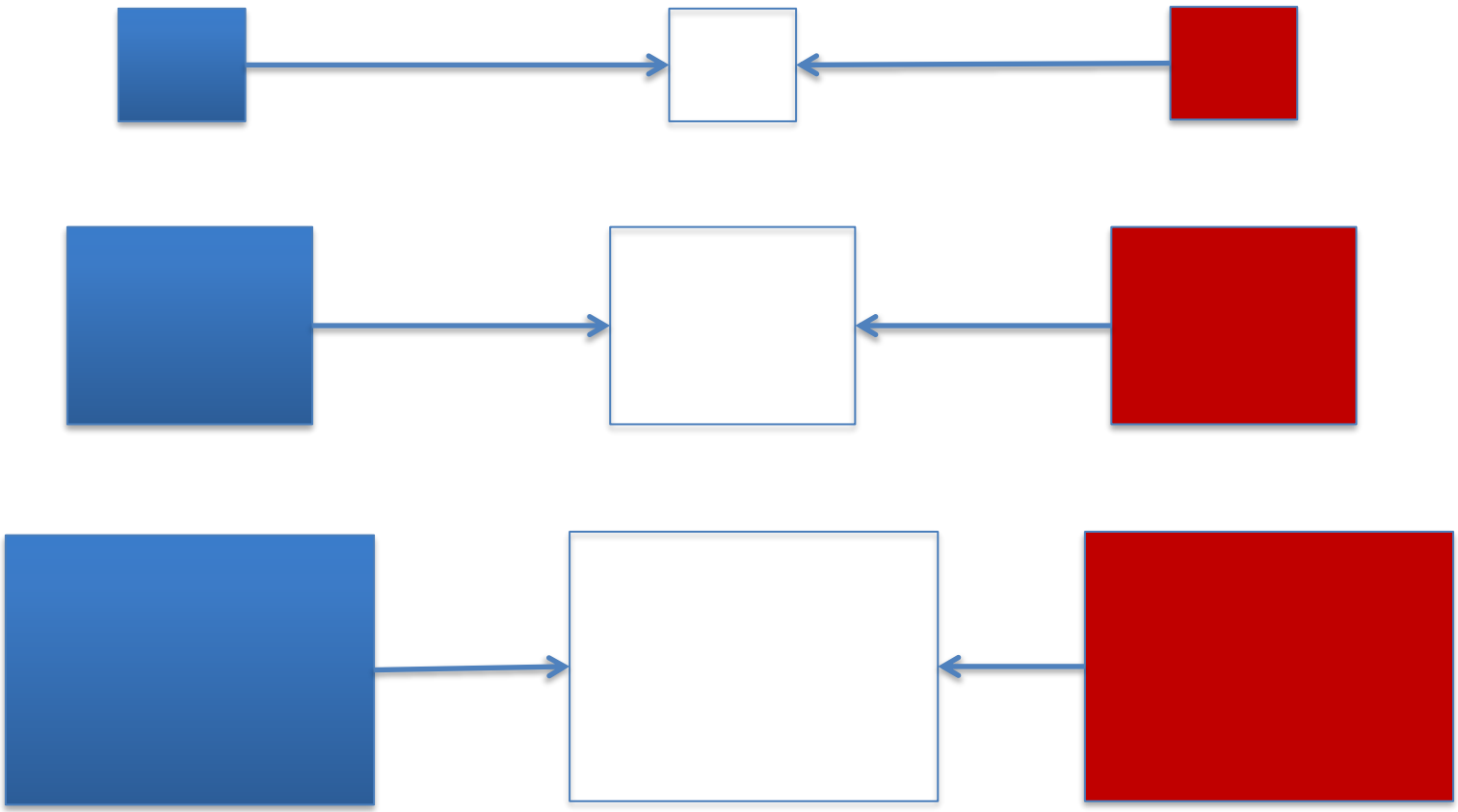
- Laplacian pyramids LA and LB are constructed for images A and B, respectively.
- A third Laplacian pyramid LS is constructed by copying nodes from the left half of LA to the corresponding nodes of LS and nodes from the right half of LB to the right half of LS.
- Nodes along the center line are set equal to the average of corresponding LA and LB nodes

Combining two Laplacian Pyramids for Blending

Laplacian Pyramid LA

Laplacian Pyramid LS
to be filled in

Laplacian Pyramid LB



Using the New Laplacian Pyramid

- Use the new Laplacian pyramid with the reverse of how it was created to create a Gaussian pyramid.

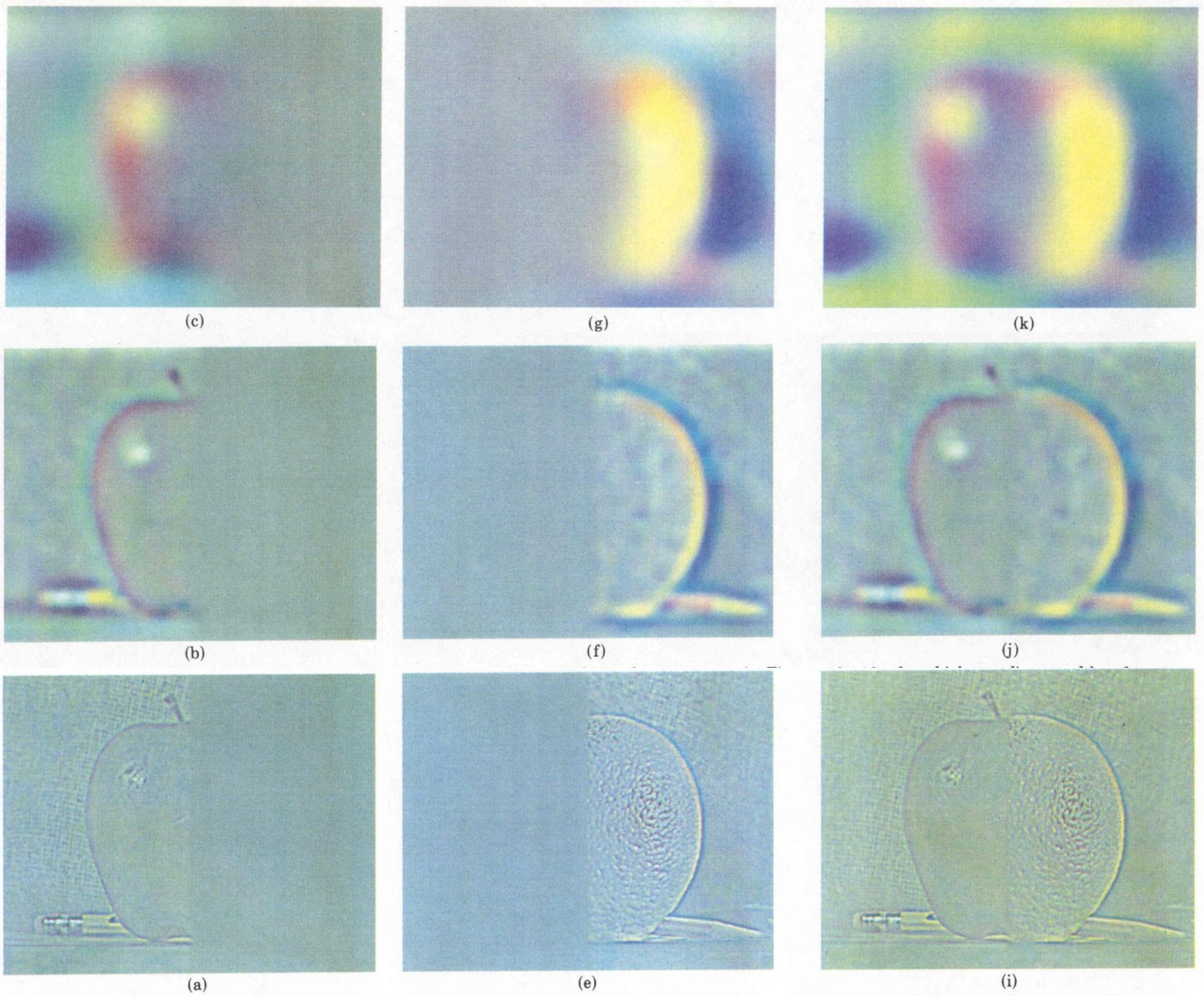
$$G_i = L_i + \text{expand}(G_{i+1})$$

- Which level of the new Gaussian pyramid gives the final result?

Laplacian level 4

Laplacian level 2

Laplacian level 0



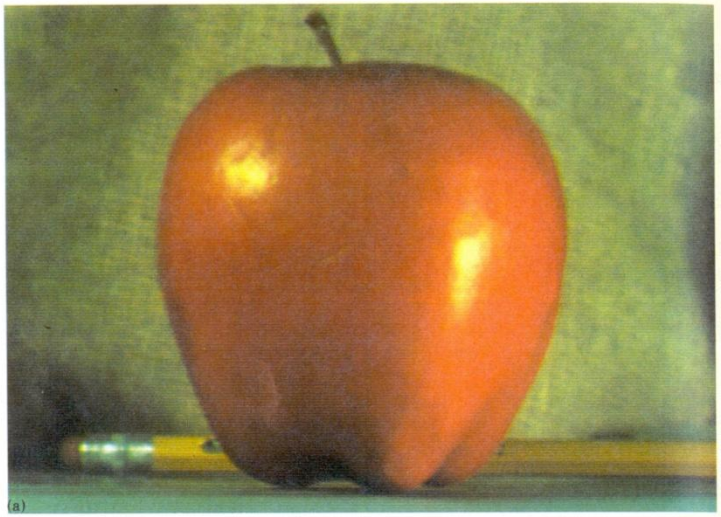
left pyramid

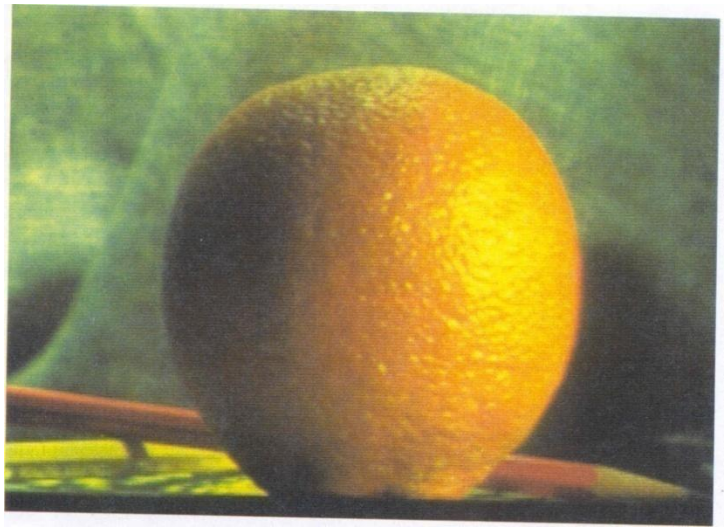
right pyramid

blended pyramid

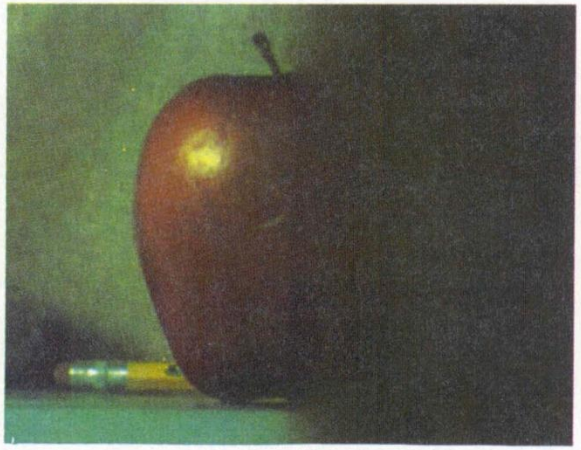
Pyramid Blending

apple

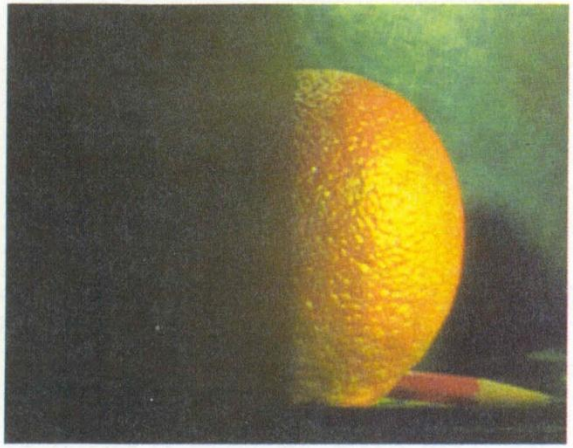




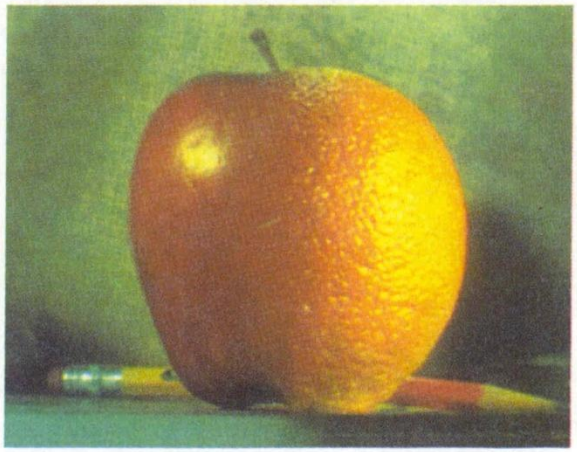
orange



(d)



(h)



(l)

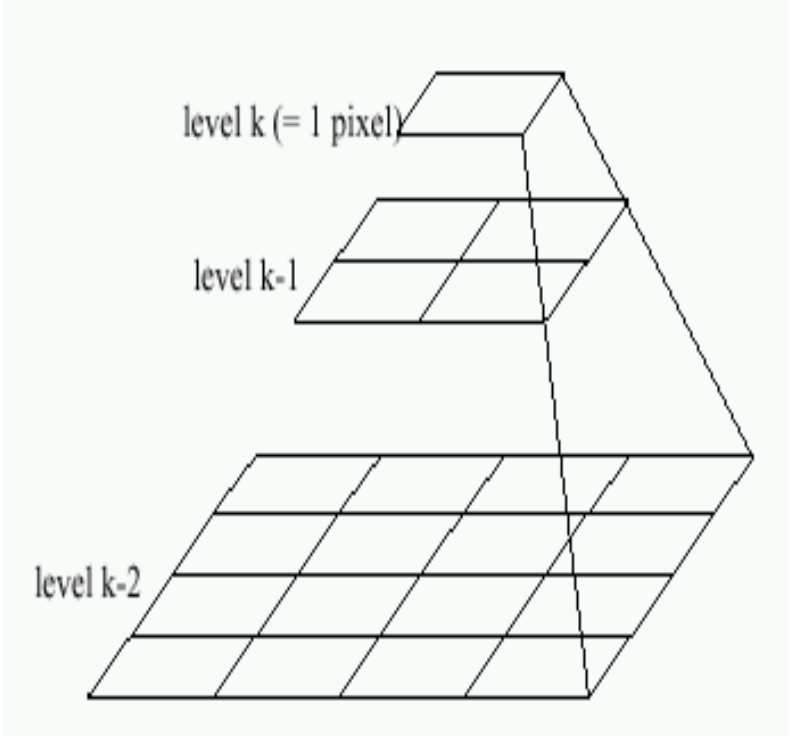
Create a Laplacian pyramid, blend each level.

Multiband blending

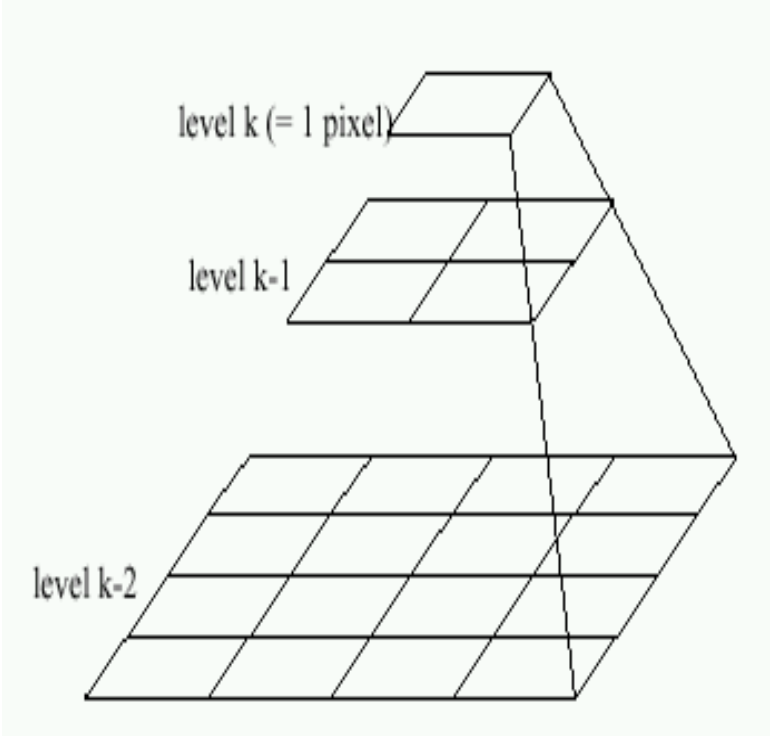
- Decompose the image into multi-band frequency
- Blend each band appropriately
- At low frequencies, blend slowly
  - Image regions that are “smooth”
- At high frequencies, blend quickly
  - Image regions have a lot of pixel intensity variation

Multiband Blending with Laplacian Pyramid

- At low frequencies, blend slowly
- At high frequencies, blend quickly



Left pyramid



Right pyramid

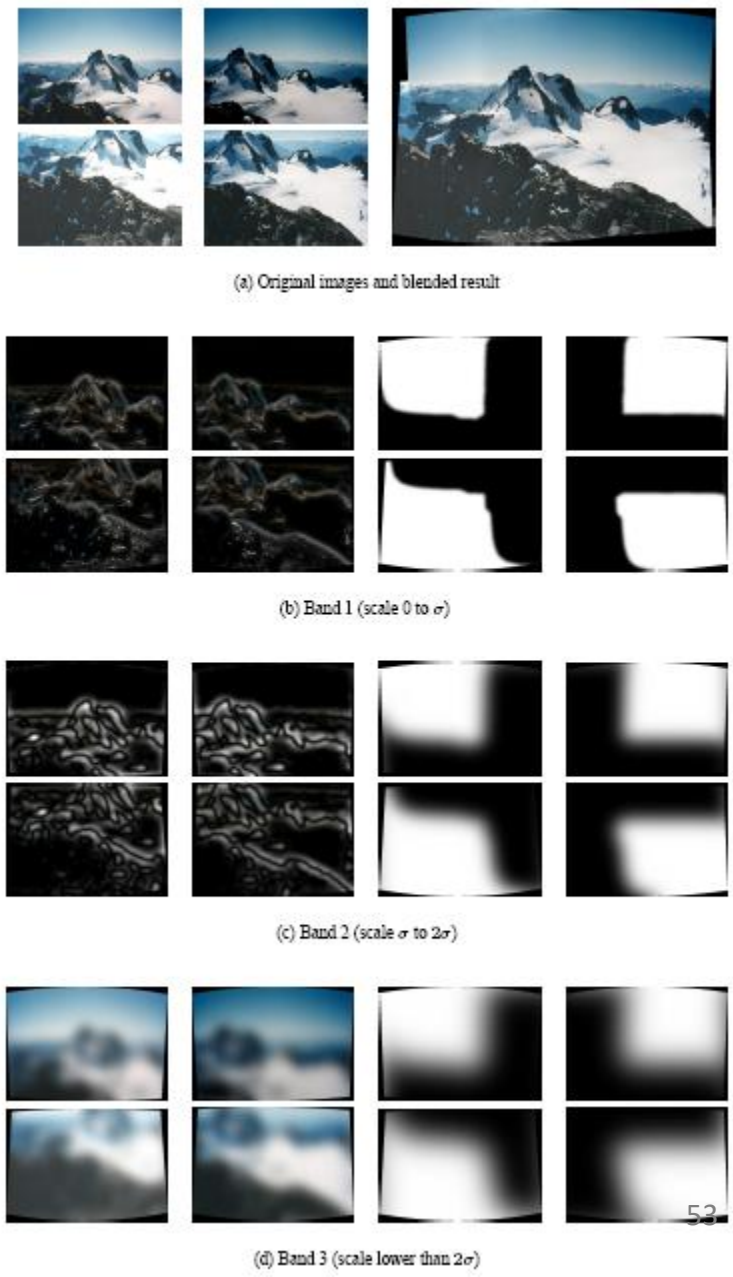
Multiband blending

Laplacian pyramids

1. Compute Laplacian pyramid of images and mask.

2. Create blended image at each level of pyramid.

3. Reconstruct complete image.



(a) Original images and blended result

(b) Band 1 (scale 0 to σ)

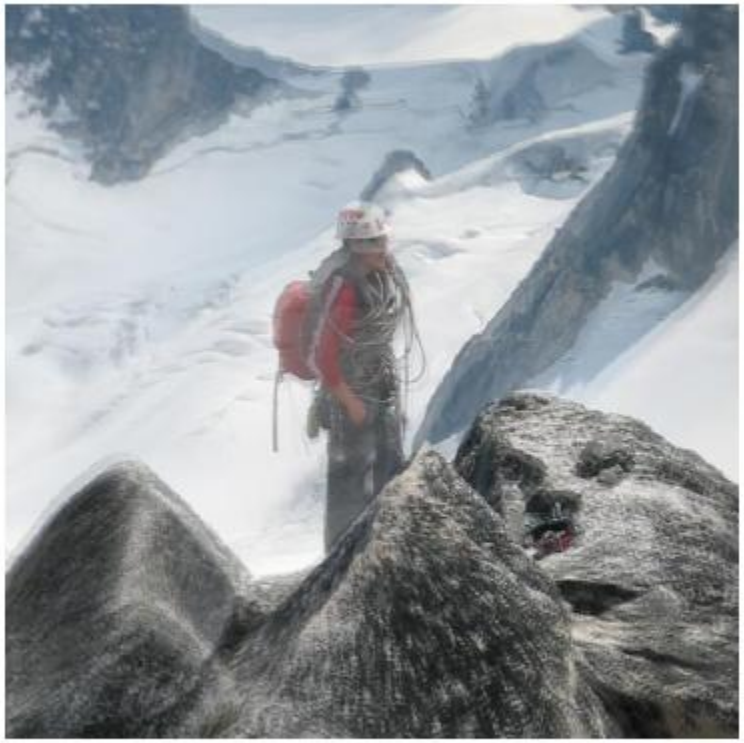
(c) Band 2 (scale σ to 2σ)

(d) Band 3 (scale lower than 2σ)

Simplification: Two-band Blending

Simplification: Two-band Blending

Blending Comparison (IJCV 2007)

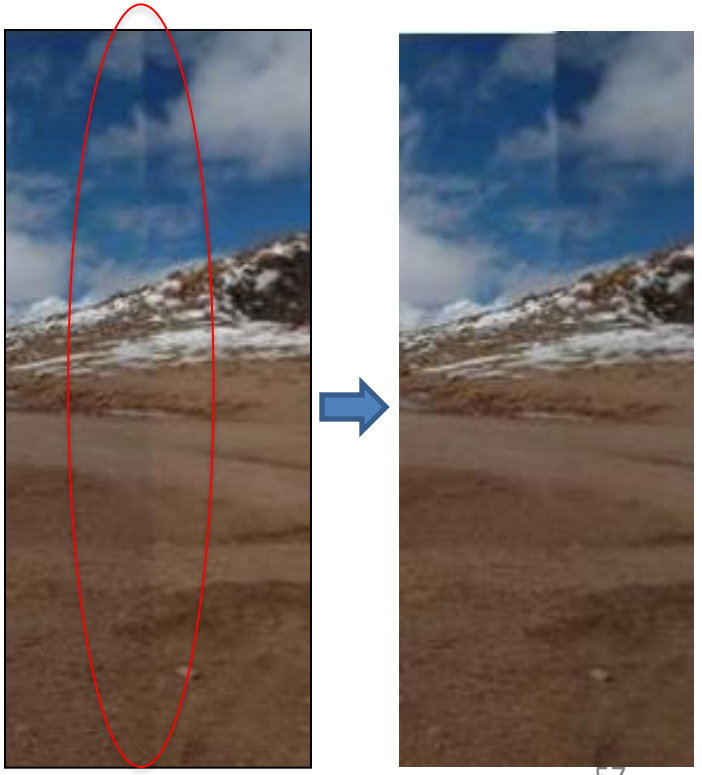


(a) Linear blending

(b) Multi-band blending

Gain Compensation: Getting Rid of Artifacts

- Simple gain adjustment
  - Compute average RGB intensity of each image in overlapping region
  - Normalize intensities by ratio of averages



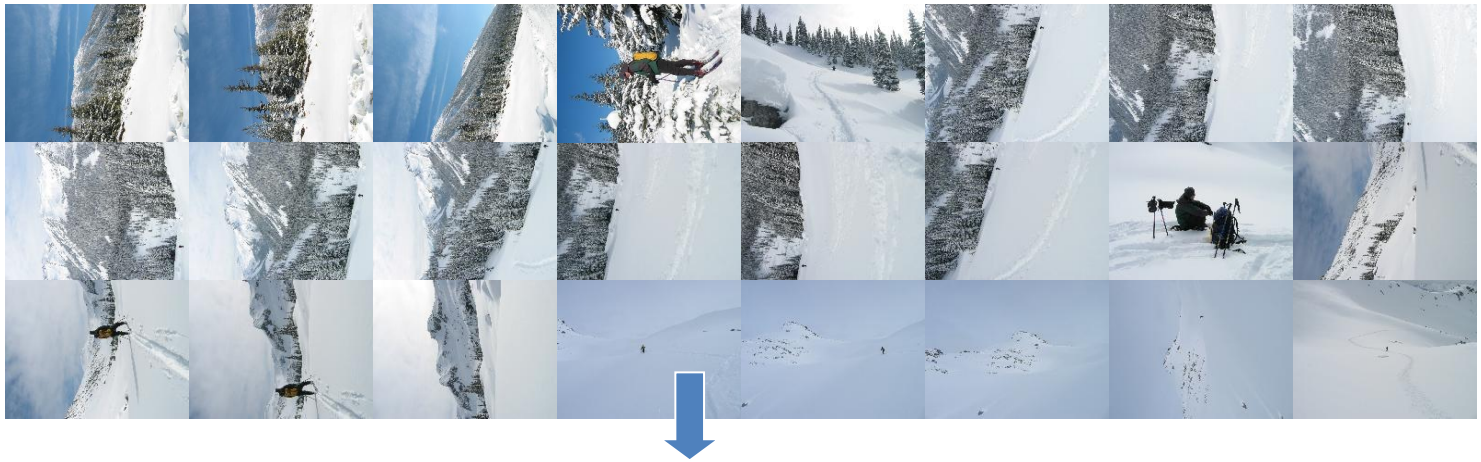
Blending Comparison

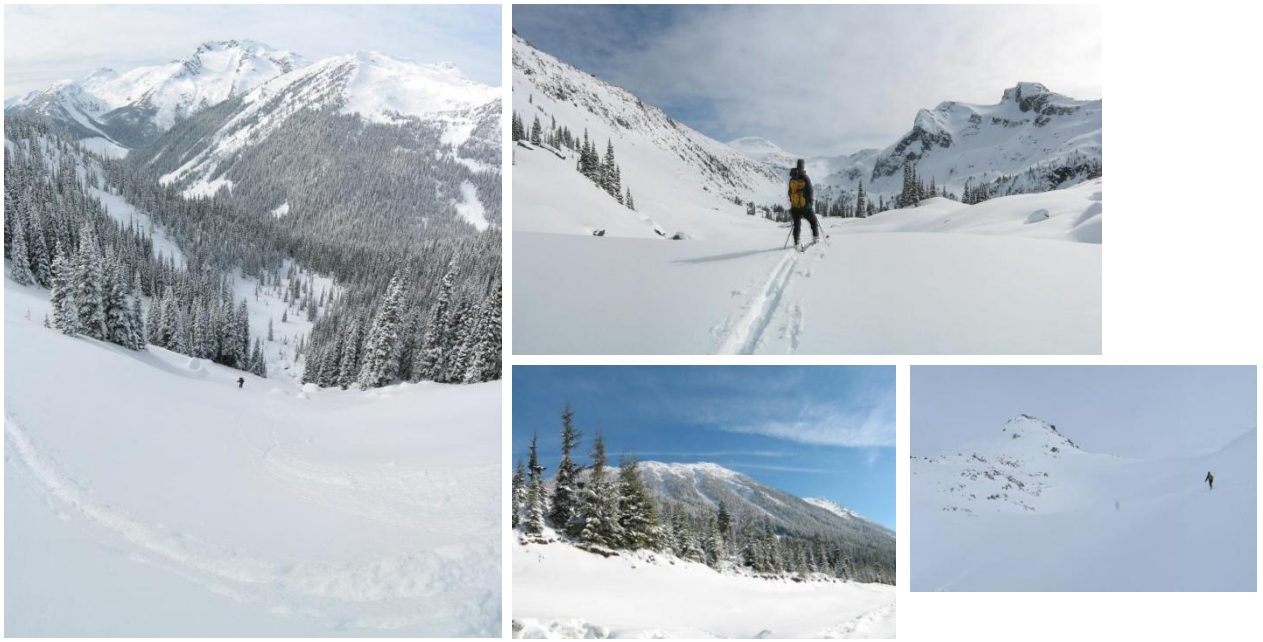
(b) Without gain compensation

(c) With gain compensation

(d) With gain compensation and multi-band blending

Recognizing Panoramas





Some of following material from Brown and Lowe 2003 talk

Brown and Lowe 2003, 2007

Recognizing Panoramas

Input: N images

1. Extract SIFT points, descriptors from all images.
2. Find K-nearest neighbors for each point (K=4).
3. For each image
   a) Select M candidate matching images by counting matched keypoints (M=6).
   b) Solve homography $\mathbf{H}_{ij}$ for each matched image.

Recognizing Panoramas

Input: N images

1. Extract SIFT points, descriptors from all images
2. Find K-nearest neighbors for each point (K=4)
3. For each image
   a) Select M candidate matching images by counting matched keypoints (M=6)
   b) Solve homography $\mathbf{H}_{ij}$ for each matched image
   c) Decide if match is valid ($n_i > 8 + 0.3\ n_f$)

inliers

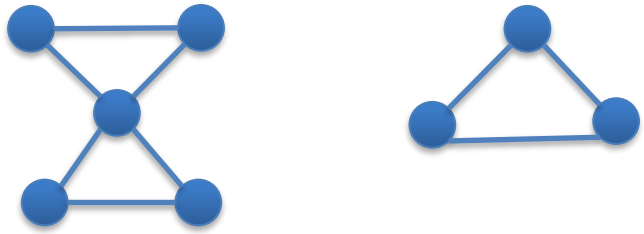
keypoints in overlapping area

Recognizing Panoramas (cont.)

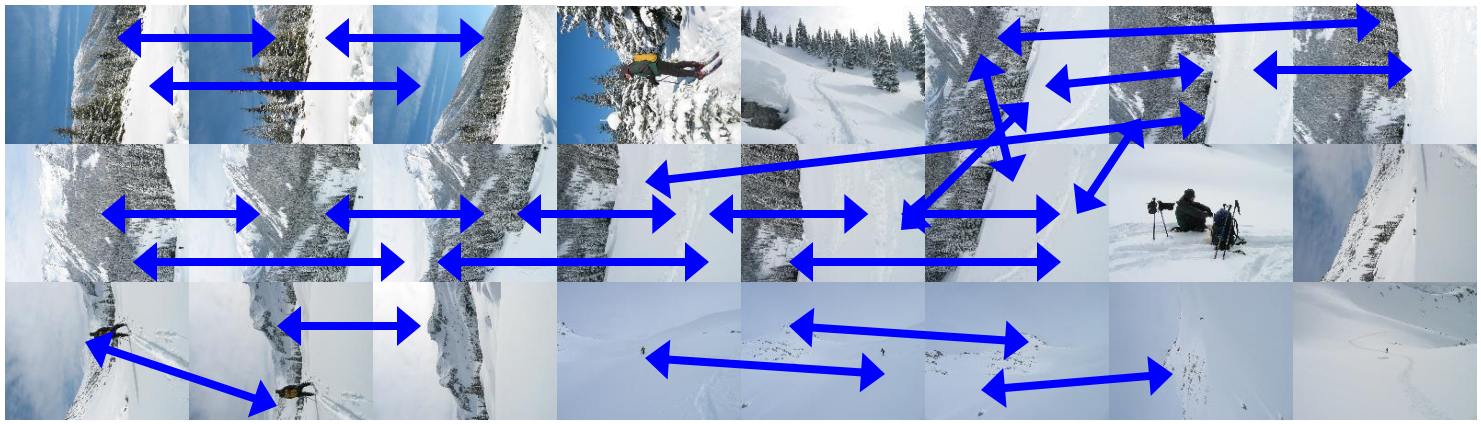
(now we have matched pairs of images)

4. Make a graph of matched pairs

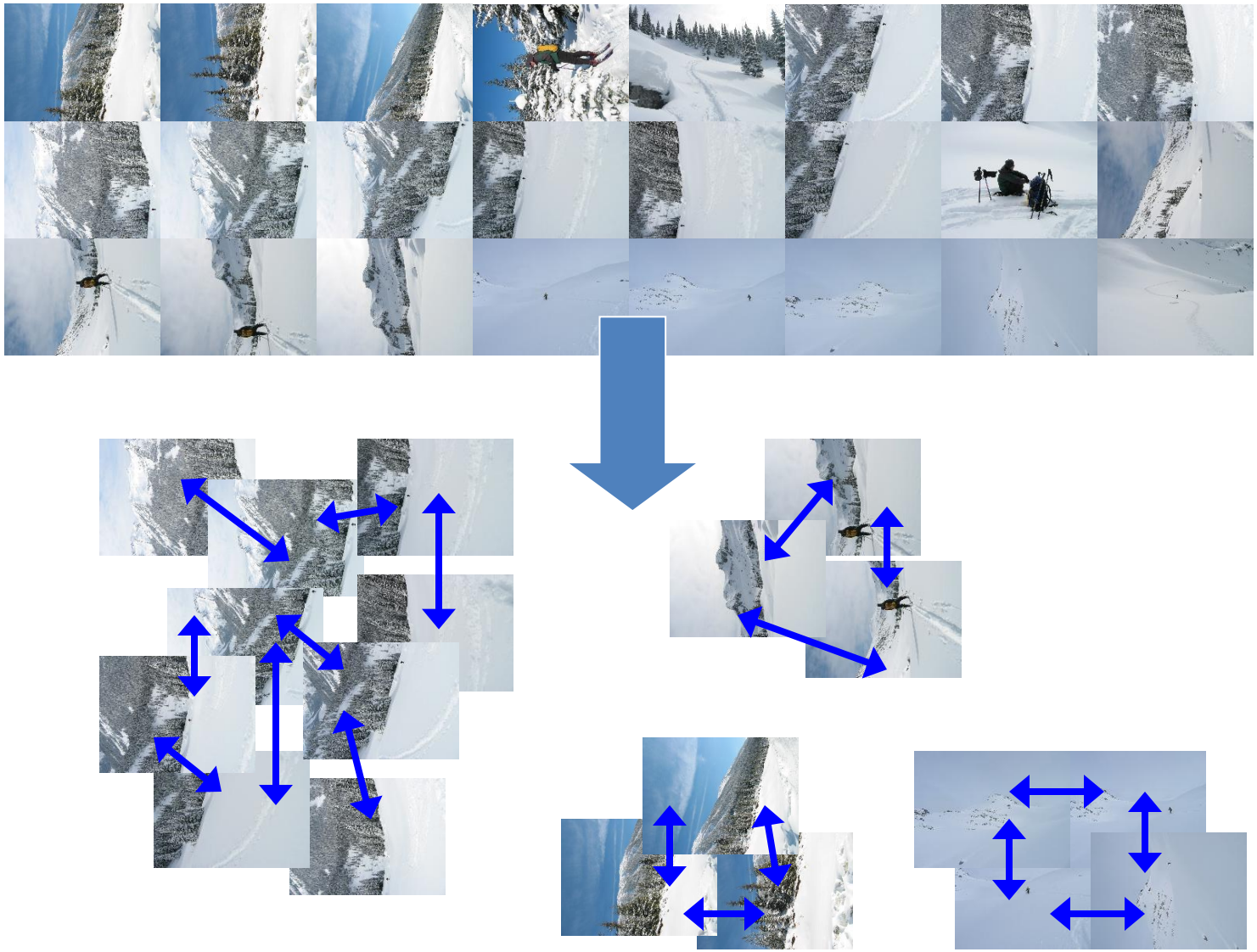
Find connected components of the graph



Finding the panoramas



Finding the panoramas

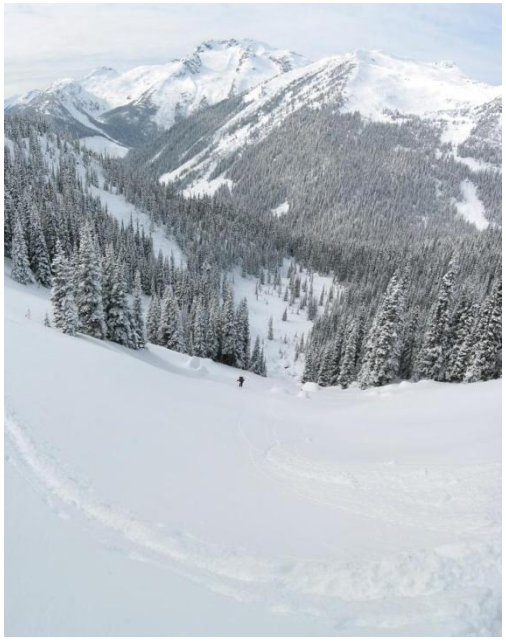


Recognizing Panoramas (cont.)

(now we have matched pairs of images)

4. Find connected components
5. For each connected component
   a) Solve for rotation and f (camera parameters)
   b) Project to a surface (plane, cylinder, or sphere)
   c) Render with multiband blending

Finding the Panoramas



Conclusion

- Image warping
  - Once we found the transformation,
  - How do we shift one image according to the transformation.
  - Forward warping.
  - Inverse warping.
- Image blending
  - Feathering with ramp, alpha blending.
  - Pyramid blending.
  - Multiband blending.
- Panorama.