Blending

Motivation

- Digital Images: Merging and Blending Images
- Different methods for Combing Multiple Images to Generate a New Image
  - Merging two images
    - How are the two signals blended?
  - Window sizes used for merging images
  - Advantages of a using the Fourier Domain
- Applications Panoramas

Blending Application: Panorama

Merging Two Images

Merging Two Images

- Grayscale

Merging Two Images

- 50/50 Blend
- Photoshop

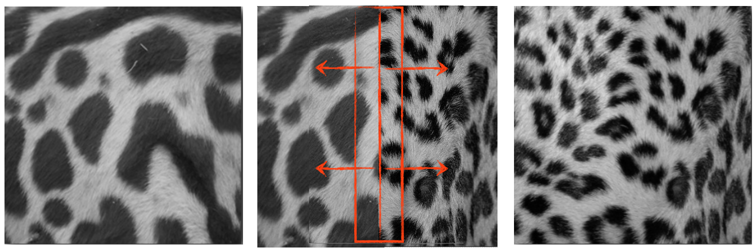
Merging Two Images

- Method 1: 50/50 Blend

Merging Two Images

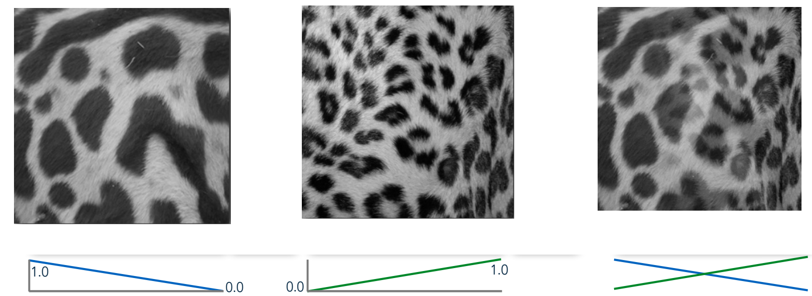
- Goal: Blend around the middle so they blend seamlessly together
  - Remove Sharp Line

Merging Two Images



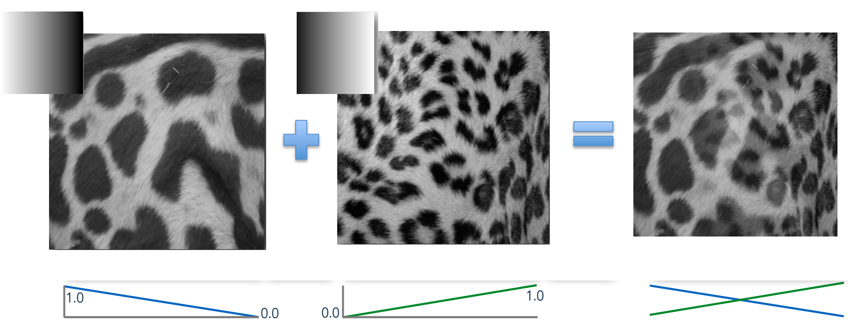
- Goal: Blend around the middle so they blend seamlessly together
  - Remove Sharp Line

Merging Two Images



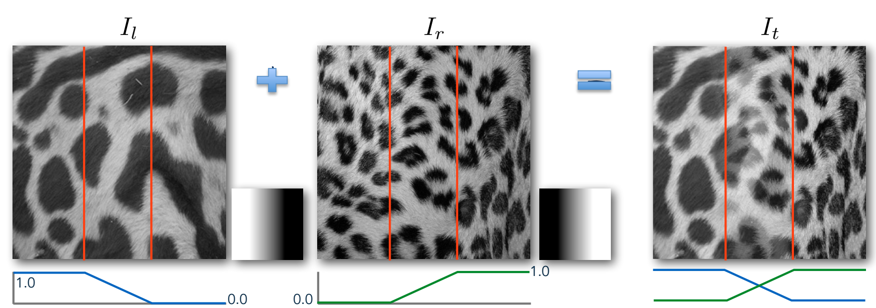
- Method 2: Cross Fading

Merging Two Images

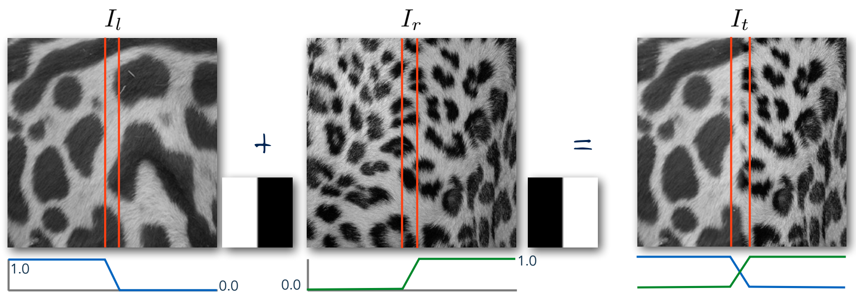


- Method 2: Cross Fading

Refinements

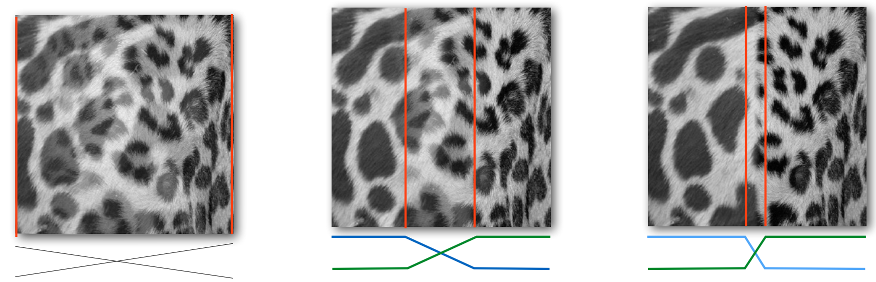


- Window Size



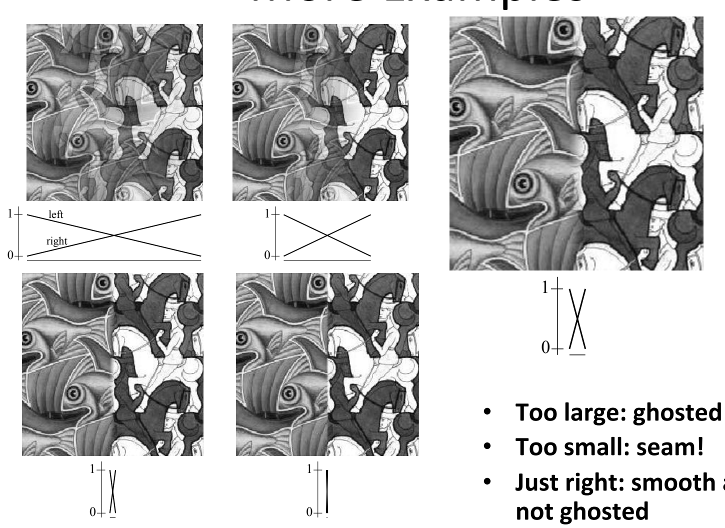
- Even Smaller
- Goal is the remove effect of obvious elending, edges, or artifacts

What is the Best Window Size



- **Too large** : Ghostly artifacts
- **Too small** : See edges or seam between two images.

More Examples



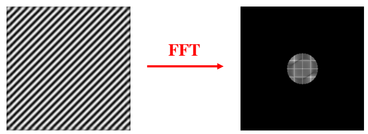
- **Too large: ghosted**
- **Too small: seam!**
- **Just right: smooth and not ghosted**

What is the Optimal Size?

- Avoid "Seams" of too small window sizes
  - Window size = the size of largest most prominent feature (window needs to fully contain it)
    - Expand it: Otherwise it looks cut off, so expand it to contain it.
- Avoid Ghostly Artifacts of too large window sizes
  - Window size <= 2*size of smallest prominent feature
    - Shrink it.
- Recast to related it to Frequency
  - **Image** frequency should occupy **one** octave (power of two)
  - **Largest** frequency <= 2*size of smallest frequency

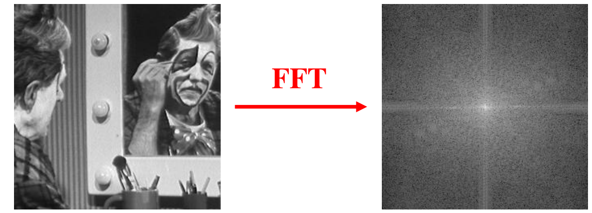
http://graphics.cs.cmu.edu/courses/15-463/2010_spring/Lectures/blending.pdf

- Avoid Seams: Image frequency content should occupy one "octave" (expand it to allow it)
- Avoid Ghosting: Largest frequency <= 2*size of smallest frequency (shrink it to disallow ghosting)



- **Idea**: Use the frequency domain to extract the window size

What if the frequency spread is too wide?



Idea (Burt and Adelson) 1983:

- Compute $F_{left} = FFT(I_{left})$, $F_{right} = FFT(I_{right})$
- Decompose Fourier image into octaves (bands)
  - $F_{left} = F_{left1} + F_{left2} + \dots$
- Feather corresponding octaves F_{lefti} with F_{righti}
- **Sum** feathered octave images in **frequency domain**

Feathering & Refinement

- Blur edges before 'blending; makes the blend smoother
- Compute $FFT(I_{left}) = F_{left}$, $FFT(I_{right}) = F_{right}$
- Decompose Fourier Images into Octaves
- **Compute Inverse FFT and feather corresponding octaves** F_{lefti} with F_{righti} **in spatial domain**
- **Return to Fourier Domain** and **Sum** Feathered octave images

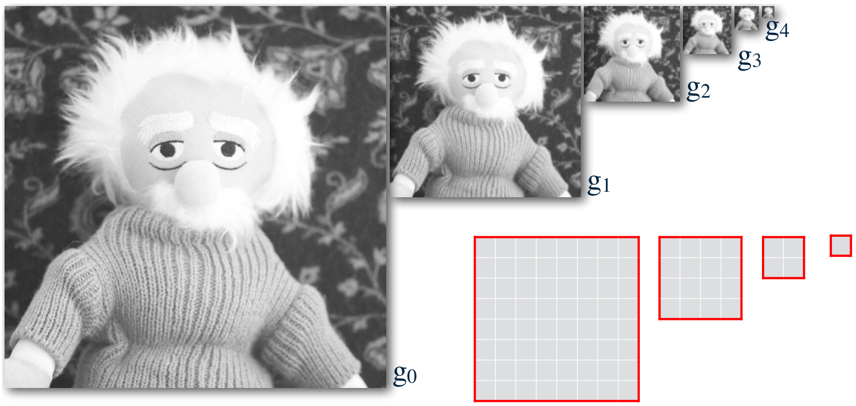
Recap

- **Key Insight:** Coarse structure should blend very slowly between images (lots of feathering), while fine details should transition more quickly.

- **Advantage**: Both ghosting and seams are blended
- **Disadvantage**: **Inefficient** going back and forth between the frequency and the spatial domains.

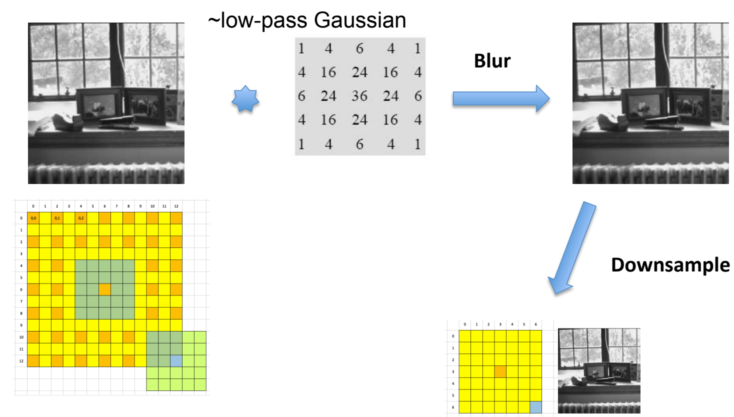
- Next Approach: Use Pyramids – avoid (reduce) dealing with lots of pixels at computational steps by 'modularizing' image into a set of images of different resolutions.

Gaussian Pyramid



- Convolution: Run a Gaussian 3x3 kernel over it to make a smaller image (create replacement value) and place it in the new 'smaller image'

Convolve & Down-sample



- **Refinement**: The Gaussian Kernel is approximated by a low pass 5x5 **binomial** filter.
  – Recall Pascal Triangle

Side Note: Binomial Coefficients

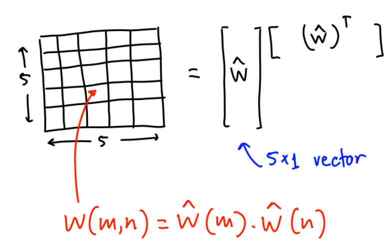
$$a_{nr} \equiv \frac{n!}{r!(n-r)!} \equiv \binom{n}{r}$$

n = number of elements in the 1D filter minus 1
r = position of element in the filter kernel (0, 1, 2…)

1 1
1 2 1 → [1 2 1]/4 - approximates Gaussian with sigma=1/sqrt(2)
1 3 3 1
1 4 6 4 1 → [1 4 6 4 1]/16 - approximates Gaussian with sigma=1
1 5 10 10 5 1
and so on…

Side Note: Separable Filter

- Can be split into a vertical component V (a vector) and horizontal components H (a vector) that can be applied independently.
- It be generated by convolving the components
  – Kernel = H * V
  – Result = I * Kernel is equivalent to
  – Result = (I*H) * V due to the associativity of convolution.
- More efficient



Golden Rules of Constructing the 'Gaussian' Kernel

w is a 5 element uni-modal vector (single maximum)

1. The 5 weights 'w' that are the same at each level
2. Normalized (applying w to a 'constant' image does not alter the image
   $$\sum_{m=-2}^{2} w(m) = 1$$
   $$a + 2b + 2c = 1$$
3. Symmetric about the center [c b a b c]
4. Equal Contribution of nodes from one level to another:
   a + 2c = 2b = 1/2

- To Satisfy 1-4 we have 2 equations & 3 unknowns ➔ a remains a free parameter

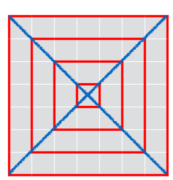
http://persci.mit.edu/pub_pdfs/pyramid83.pdf

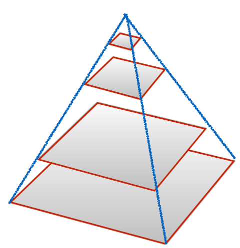
Generating the Weights

- w(2), w(1) w(0), w(-1), w(-2)
- W(2) = w(-2) = ¼ - a/2, w(-1)=w(1) = 1/4
- ¼ - a/2 ¼, a ¼, ¼ - a/2
- Usually a is [0.3, 0.6] a changes the peak of the gaussian distribution

see paper for detail/

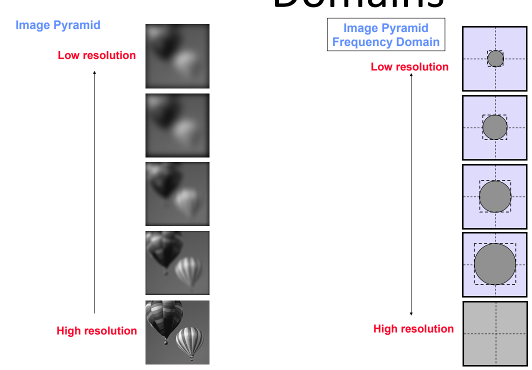
A Pyramid





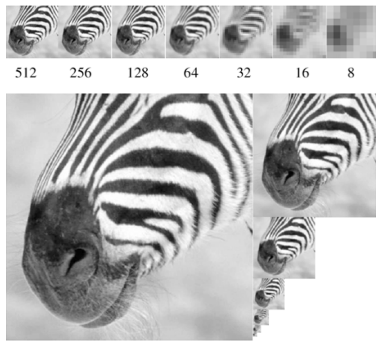
- End up with a pyramid of images of different levels: g_o, g_n of resolution
- Gaussian Pyramids (reduce)
- Laplacian Pyramids (expand)

Viewing the Pyramids In Different Domains



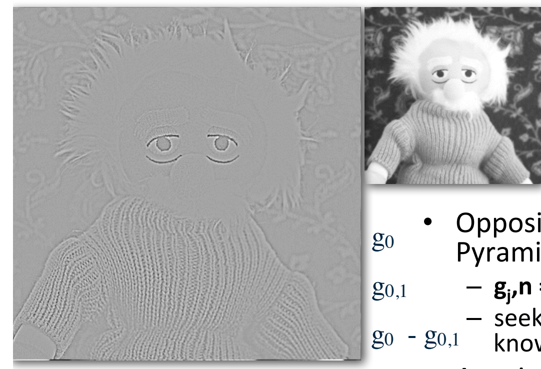
- Image Blurring → Low Pass Filter

Generating the Gaussian Pyramid

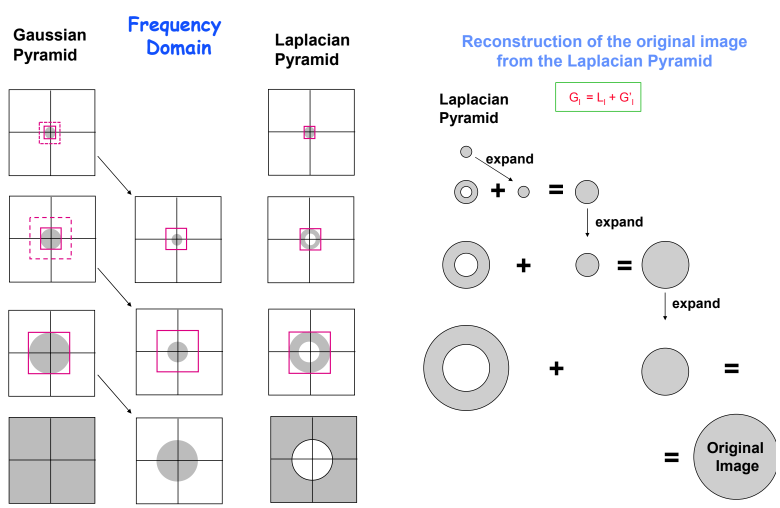


- gk = REDUCE(g(k - 1)) g0 is at the lowest level

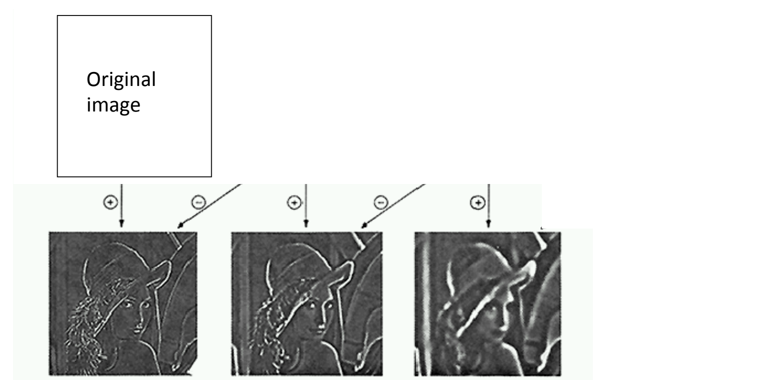
Pyramid Representation of Images (A Laplacian Pyramid)



- Opposite (inverse) of the Gaussian Pyramid
  - **$g_{j,n}$ = EXPAND(g_j,n-1)**
  - seeks to add new values in between known ones. $g_{j,n}$ is g_j expanded n
- A series of "error" images,
- A *difference between two levels of a Gaussian Pyramid*



Laplacian Pyramid



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

Pyramid Blending with Regions (Mask)

Given two images A and B, and a mask M

1. Build Laplacian Pyramids LA and LB
2. Build a Gaussian pyramid GM from selected the region M
3. Build a third combined Laplacian pyramid LC from LA and LB using nodes of GM as weights:
   where for each level k, and i, j:

$$LC_k(i,j) = GM_k(i,j,)*LA_k(i,j) + (1-GM_k(i,j))*LB_k(i,j)$$

4. Obtain the image C by expanding and summing the levels of LC.

Don’t blend, Cut.

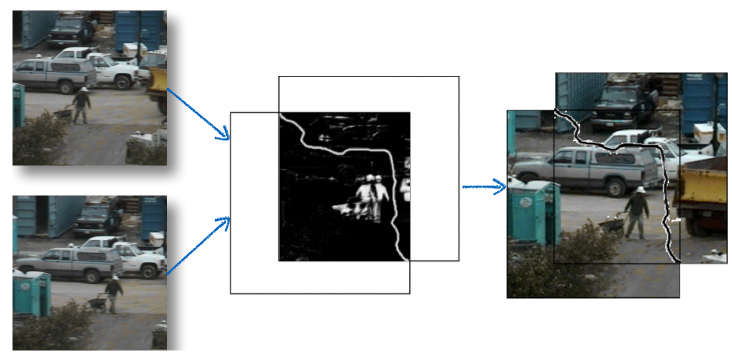
Moving objects become ghosts

- So far we only tried to blend between two images. What about finding an optimal seam?

Davis, J. 1998. Mosaics of scenes with moving objects. In Proceedings of CVPR.

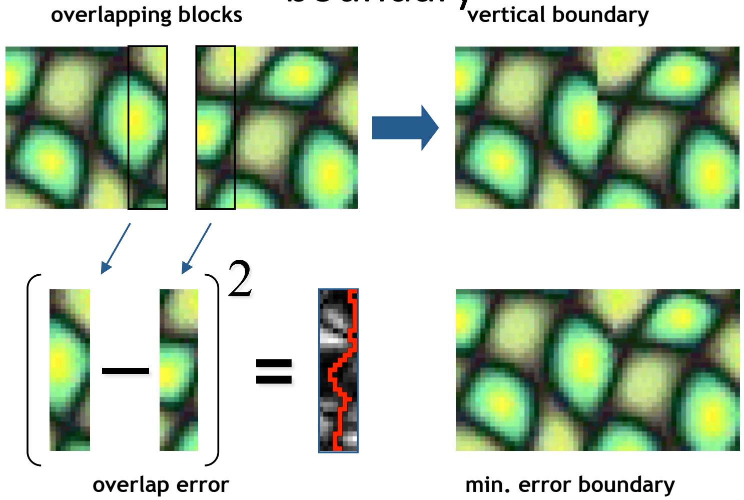
Don’t Blend Cut

- Segment the mosaic
  – Avoid artifacts along boundaries

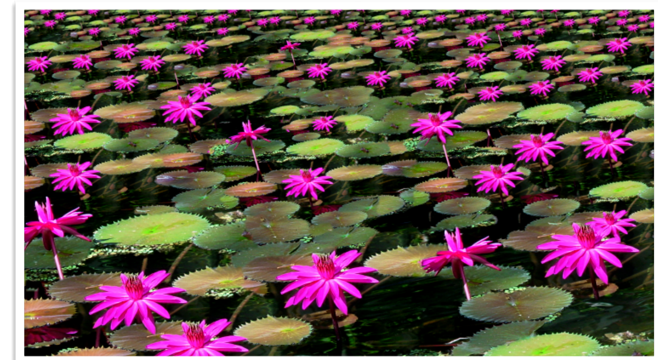


- Moving objects cause “ghosting”
- Find an optimal seam as opposed to blend between images
  – Idea minimal error cut.
- Final has exact pixels from an Image

Finding the Seam Minimal error boundary

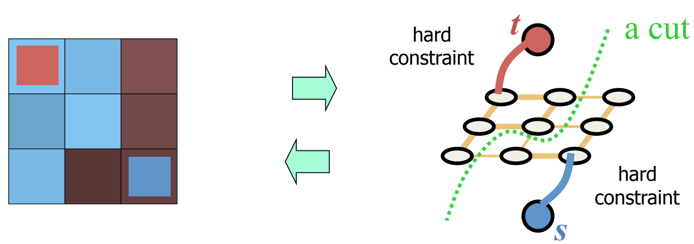


Extension: Creating New Images



- Kwatra, Schödl, Essa, Turk, Bobick (2003), SIGGRAPH

Extensions: Efficient Graph cuts



Minimum cost cut can be computed in polynomial time

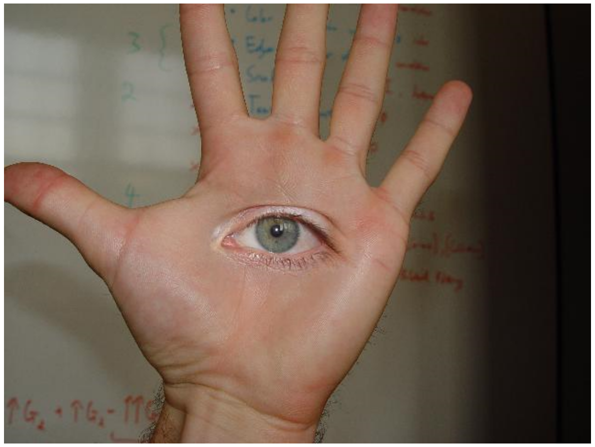
(max-flow/min-cut algorithms)

Boykov&Jolly, ICCV'01

Summary:

- Compositing images
  - Have a clever blending function
    - Feathering
    - Blend different frequencies differently
      - pyramids
  - Choose the right pixels from each image
    - Seams
    - Finding Seams Efficiently: Graph-cuts

Another Result: Horror Photo



Courtesy: prof. dmartin ©

Resources

Blending/Feathering:

- https://en.wikipedia.org/wiki/Pyramid_(image_processing)
- Burt & Adelson's, 1st Pyramid Paper, 1983 (here)
- Burt & Adelson's, 2nd Pyramid Paper, 1983 (includes Mask pyramid) (here)
- Brown & Lowe, Resent Blending using 2 bands, low and high frequency (here)

Cuts/Seams:

- Davis, 1998, Mosaics of scenes with moving objects
  - https://users.soe.ucsc.edu/~davis/panorama (link needs to be typed in)
- Efros & Freeman 2001, Image Quilting
  - http://www.eecs.berkeley.edu/Research/Projects/CS/vision/papers/efros-siggraph01.pdf
- Seam Carving for Content-Aware Image Resizing Seam
  - Shai Avidan, Ariel Shamir, 2007 (here)
  - https://en.wikipedia.org/wiki/Seam_carving
- Boykov & Jollym, 2001, Interactive Graph Cuts, ICCV'01
  - https://people.eecs.berkeley.edu/~efros/courses/AP06/Papers/boykov-iccv-01.pdf
- Kwatra, Schödl, Essa, Turk, Bobick (2003), Graph Cut Textures
  - http://www.cc.gatech.edu/gvu/perception/projects/graphcuttextures/gc-final.pdf