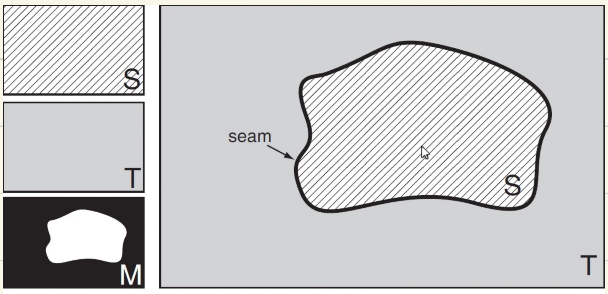
image blending



(a) (b)

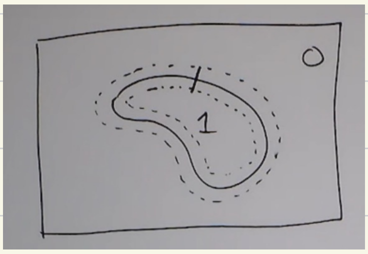
hard compositing:

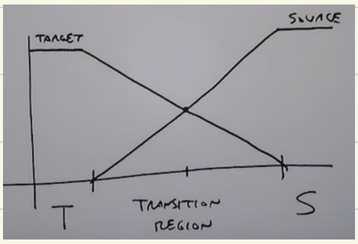
$I(x,y) = M(x,y)S(x,y) + (1-M(x,y))T(x,y)$

$$= \begin{cases} S(x,y), & M(x,y) = 1 \\ T(x,y), & M(x,y) = 0 \end{cases}$$

generally bad: seam/matte line is visibal

weighted transition region





better idea: multi-resolution blending with a laplacian pyramid

idea: wide transition regions for low-frequency components; narrow transition regions for high-frequency component (edges)

Gaussian pyramid

- K = 5*5 Gaussian filter
- G_0= original image (full resolution)
- $G_i = (K \underset{conv}{*} G_{i-1})_{\downarrow 2}$ downsample by 2 in both dimension

Differences of Gaussian at each scale:

$$L_i = \underbrace{G_i}_{\text{Gaussian pyramid image at scale i}} - \underbrace{(K * G_i)}_{\text{blurred version of that image}}$$

High-pass image at scale i

$\{L_i\}$ form a laplacian pyramid

we can recover the original image as

$$I = \sum_{i=0}^{N} (L_i)_{\uparrow \text{ to full size}}$$

$$G_N = L_N$$

i.e add back all the edges at difference scales; base image is smallest, blurriest image

To do image composition,

compute laplacian pyramids for S,T, L^S and L^T

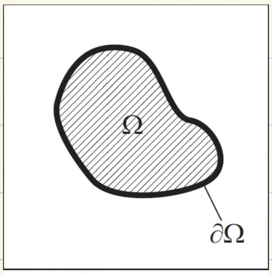
compute Gaussian pyramids for mask M, G

Laplacian pyramids for composite:

$$L_i^I = G_i L_i^S + (1-G_i) L_i^T \quad i = 0, \dots, N$$

then add up to get final composite

A better idea: Poisson image editing



Source image

idea: to reduce colour mismatch between source and target, create composite in gradient domain

we want gradient of the composite inside Ω to look as close as possible to the source image gradient.

The composite must match target image on the boundary ∂Ω

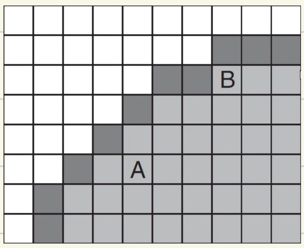
$$\min_{I(x,y) \in \Omega} \iint_{\Omega} \| \nabla I(x,y) - \nabla S(x,y) \|^2 \, dx\, dy$$

$$\text{s.t } I(x,y) = T(x,y) \text{ on } \partial\Omega$$

solution to this problem:

$$\nabla^2 I(x,y) = \nabla^2 S(x,y) \text{ in } \Omega$$

$$I(x,y) = T(x,y) \text{ on } \partial\Omega$$ poisson equation



discretizing and solving the problem:

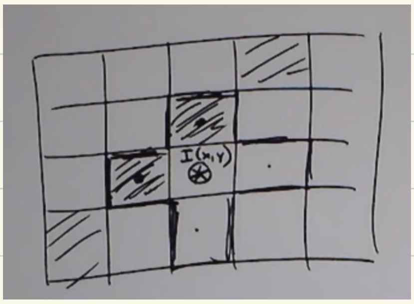
1) for a pixel p inside Ω,

$$\nabla^2 I(x,y) = \nabla^2 S(x,y)$$

	1	
1	-4	1
	1	

$I(x+1,y)+I(x-1,y)+I(x,y-1)+I(x,y+1)$
$- 4I(x,y) = S(x+1,y)+S(x-1,y)+S(x,y-1)$
$+S(x,y+1) - 4S(x,y)$

2) for a pixel p whose neighbourhood is not fully inside Ω



$-4I(x,y)+I(x,y+1)+I(x+1,y)+T(x,y-1)+T(x-1,y)$

target must agree with composite of boundary

$= -4S(x,y)+S(x+1,y)+S(x-1,y)+S(x,y+1)+S(x,y-1)$

another equation in unknown I
big linear system

Mixed-gradient compositing:

$$\nabla^2 I(x,y) = \nabla^2 S(x,y)$$

more general:

$$\nabla^2 I(x,y) = \operatorname{div}\begin{bmatrix} S_x(x,y) \\ S_y(x,y) \end{bmatrix} = \frac{\partial S_x}{\partial x} + \frac{\partial S_y}{\partial y}$$

difference: right hand side need not arise from the laplacian of a real image (non-conservative vector field)

e.g

$$\begin{bmatrix} S_x(x,y) \\ S_y(x,y) \end{bmatrix} = \begin{cases} \nabla T(x,y), & \text{if } \|\nabla T(x,y)\| > \|\nabla S(x,y)\| \\ \nabla S(x,y), & \text{otherwise} \end{cases}$$

can also use an algorithm to automatically find a good boundary for compositing