Gradient Blending
Compositing Procedure

1. Extract Sprites (e.g. using *Intelligent Scissors* in Photoshop)

2. Blend them into the composite (in the right order)
How to blend two images?

image blending: image surgery…
- cutting from one image (which we will cover in details on segmentation)
- reconstructing onto the new image
Blend = Cut and Paste images

Image blending is an art of …
  faking images, hiding evidence of image surgery, making it looks natural
Direct Copy and Paste

Direct attempt: not so good!
-- we created an artificial boundary between the pasted region
Challenge: color and brightness mismatch

Blue channel value of the vertical line

Big change in intensity creates new image boundary…
- any ideas on how to remove that boundary?
Just replacing pixels rarely works

Problems: boundaries & transparency (shadows)

Binary mask
Two Problems:

Semi-transparent objects

Pixels too large
Add one more channel:  
- Image(R,G,B,alpha)  

**Solution: alpha channel**

Encodes transparency (or pixel coverage):
- Alpha = 1: opaque object (complete coverage)
- Alpha = 0: transparent object (no coverage)
- 0<Alpha<1: semi-transparent (partial coverage)

Example: alpha = 0.3

Partial coverage or semi-transparency
Alpha Blending

\[ I_{\text{comp}} = \alpha I_{\text{fg}} + (1-\alpha)I_{\text{bg}} \]
Multiple Alpha Blending

So far we assumed that one image (background) is opaque. If blending semi-transparent sprites (the “A over B” operation):

\[
I_{comp} = \alpha_a I_a + (1-\alpha_a)\alpha_b I_b
\]

\[
\alpha_{comp} = \alpha_a + (1-\alpha_a)\alpha_b
\]

Note: sometimes alpha is premultiplied:

im(\alpha R, \alpha G, \alpha B, \alpha):

\[
I_{comp} = I_a + (1-\alpha_a)I_b
\]

(same for alpha!)
Multiple Alpha Blending

So far we assumed that one image (background) is opaque. If blending semi-transparent sprites (the “A over B” operation):

\[ I_{\text{comp}} = \alpha_a I_a + (1-\alpha_a)\alpha_b I_b \]

\[ \alpha_{\text{comp}} = \alpha_a + (1-\alpha_a)\alpha_b \]

Note: sometimes alpha is premultiplied:

\[ \text{im}(\alpha R, \alpha G, \alpha B, \alpha) : \]

\[ I_{\text{comp}} = I_a + (1-\alpha_a)I_b \]

(same for alpha!)
Alpha blending

Alpha channel encodes the transparency of the object
Alpha blending
Alpha blending
Alpha blending

Copy - paste is a special kind of alpha blending – binary mask
copy - paste

alpha blending
Alpha blending can deal with transparent objects

copy - paste

alpha blending
Alpha Hacking…

No physical interpretation, but it smoothes the seams
Feathering

Encoding as transparency

\[ I_{\text{blend}} = \alpha I_{\text{left}} + (1-\alpha)I_{\text{right}} \]
Affect of Window Size
Affect of Window Size
Good Window Size

“Optimal” Window: smooth but not ghosted
Setting alpha: simple averaging

Alpha = .5 in overlap region
Setting alpha: center seam

Distance transform

Alpha = logical(dtrans1 > dtrans2)
Setting alpha: blurred seam

Distance transform

Alpha = blurred
Setting alpha: center weighting

Distance transform

Ghost!

Alpha = dtrans1 / (dtrans1+dtrans2)
Alpha blending hacking
Alpha blending
Alpha Blending

Blue channel value of the vertical line

Continuous change,
but still a visually un-natural pattern in intensity
Solution: Copy only gradient + Recreate
Gradient Blending:
No more intensity change!

Blue channel value of the vertical line

Smooth transition
Gradient Domain blending (1D)

Two signals

Regular blending

Blending derivatives

bright
dark
Image Blending

copy - paste
alpha blending
Gradient blending
Results of gradient blending

<table>
<thead>
<tr>
<th>Source (figure)</th>
<th>Target (background)</th>
<th>Result</th>
</tr>
</thead>
</table>

Color of the hat blends into the background
-- successful image surgery… with interesting side-effect
Perez et al, 2003

Limitations:

- Can’t do contrast reversal (gray on black -> gray on white)
- Colored backgrounds “bleed through”
- Images need to be very well aligned
Don’t blend, CUT!

Moving objects become ghosts

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

- Single source image per segment
- Avoid artifacts along boundaries
  - Dijkstra’s algorithm
Gradient Blending