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 look...
Direct attempt: not so good!
-- we created an artificial boundary between the pasted regions.
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)
Two Problems:

Semi-transparent objects

Pixels too large
Add one more channel: Solution: alpha channel

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

Example: \(\text{Alpha} = 0.3\)

Partial coverage or semi-transparency
Alpha Blending

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

So far we assumed that one image (background) is opaque.

If blending semi-transparent sprites (the “A over B” operation):

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

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

Note: sometimes alpha is premultiplied:

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

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

(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):

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

Note: sometimes alpha is premultiplied:

\[ \text{im}(\alpha R, \alpha G, \alpha B, \alpha) : \]
\[ I_{comp} = I_a + (1-\alpha_a)I_b \]
(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
shadow not real
copy - paste

alpha blending
Alpha blending can deal with transparent objects

copy - paste

alpha blending
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
Good Window Size

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

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

Distance transform

$\text{Alpha} = \text{logical}(d\text{trans}1 > d\text{trans}2)$
Setting alpha: blurred seam

Distance transform

Alpha = blurred
Setting alpha: center weighting

\[ \text{Alpha} = \frac{\text{dtrans1}}{\text{dtrans1} + \text{dtrans2}} \]
Alpha blending hacking

\[ \times \quad + \quad \times \quad = \]
copy - paste

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

Source (figure)  Target (background)  Result

Color of the hat blends into the background
-- successful image surgery… with interesting side-affect
Perez et al., 2003
Gradient Blending
Gradient Blending
Gradient Blending
Why use the gradients to recreate images
Gradient magnitude
Gradients domain

- Gradient captures everything important about shape and shading
- It contains the microscope texture of the object.
- It encodes subtle changes of illumination.
- In Pyramid Blending, we decomposed our image into 2nd derivatives (Laplacian) which encodes the shape
• We blend the gradient magnitude, to create a seamless ‘edge’ image
We blend the gradient angle images, to make sure both the microscopic texture, shape of object boundary, and the illumination changes are smoothly integrated.
Image gradients blending

\[ \nabla_x l = l \otimes g_x \]

\[ \nabla_y l = l \otimes g_y \]
How to recreate the original image from gradient?

\[ \nabla_x I = I \otimes g_x \]

\[ \nabla_y I = I \otimes g_y \]
zoom in a small patch

\[ \nabla_x l(a) \]

\[ \nabla_y l(a) \]

\[
g
\]

\[
a - b =
\]

\[
a - c =
\]
2 equations with 3 unknowns, need constraints

\[
\begin{align*}
\nabla_x l(a) = 1 & -1 \\
\nabla_y l(a) = 1 & -1 \\
\end{align*}
\]
Boundary condition

\( l_{\text{background}} \)

\[ \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix} = \nabla_x l \]

\[ \nabla_y l \]

- Keep the value on boundary \( \partial \Omega \) the same
Least Square Problem

Minimize the loss with respect to all pixels in the region $\Omega$
Keep the boundary $\partial \Omega$ the same
Least square solution
Solution on Pixels

- The convolutional kernel for Laplacian operator

Given: [Image]
Unknown: [Image]

\[ \nabla_x I \]

\[ \nabla_y I \]

\[ \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \]
Special case

• When the \( \nabla_x \) and \( \nabla_y \) are directly computed from the image (without cutting)

\[
\begin{bmatrix}
\begin{array}{ccc}
-1 & 4 & -1 \\
-1 & & \\
\end{array}
\end{bmatrix}
\]

• Keep the value on the boundary \( \partial \Omega \) the same

• Solve the equation for each channel (RGB) separately

• There is one equation matching the Laplacian values from the source to the target
Solving the inverse Laplacian problem using convolutional neural network

Blend Location (x1,y1), (x2,y2), ...

Generator

Z ~ N(0,1)

Pixel Loss

Ground Truth Gradient Map

Gradient Loss

Source Image 1  Mask Image 1  Source Image 2  Mask Image 2  Background Image
Reconstruction Process
Copy and paste

Analytical Solution (OpenCV)

Deep Learning Solution (Ours)
Gradient Blending
How to solve this inverse Laplacian problem?
An example

For the purpose of displaying, we use alpha mask for $f$ and the background of $g$.

The RGB value for $f$ is $(255, 0, 0)$, for the background of $g$ is $(0, 0, 0)$.
Mask for merging

Region $\Omega$ to be merged
Direct copy and paste

f
g
Keeping $f$ the same on the boundary

Region boundary $\partial \Omega$

Keep $f$ the same on the boundary $\partial \Omega$
\[ \Delta \]

Laplacian of the source
Before copying, we first index the pixels
Copying and *reshaping* the Laplacian of source

\[ \Delta g \]
\[
\begin{cases}
\Delta f = b \text{ in } \Omega \\
f|_{\partial \Omega} \text{ keeps same}
\end{cases}
\]
\[ \Delta f = b \]

\( \Delta \) is a linear operator…
Af = b

…We can use a matrix A to encode it!
Indexing the unknowns
Laplacian as a matrix?

\[ A? \]
Matrix A encoding the Laplacian operator

When the pixel is in the region
Matrix $A$ encoding the Laplacian operator

When the pixel is in the region
Matrix $A$: Laplacian operator

When the pixel is on the boundary
How to deal with the knowns Boundary Values?
Move the boundary value of knowns to b side!
Matrix $A$ is fixed, encoding the Laplacian Operator.
Guess what’s our blended image?
Let’s summarize with a real image
Step 1. Indexing the unknowns in Target

N is the number of unknowns
Step 2. Copying the gradient from source

Gradient magnitude

Gradient angle
Step 2. Copying the gradient from source

\[ \nabla_x I \quad \nabla_y I \]
Step2. Compute the Laplacian from source image.

Source image

Laplacian
Step 2. Copying the gradient from the source.
Step 2. Cropping the masked Laplacian
Step 3. Constructing matrix $A$ from the Laplacian operator.

For pixel in the region

$$\begin{align*}
A(18272, 18272) &= 4 \\
A(18272, 18025) &= -1 \\
A(18272, 18519) &= -1 \\
A(18272, 18271) &= -1 \\
A(18272, 18273) &= -1
\end{align*}$$
Step 3. Constructing matrix $A$ from the Laplacian operator

For pixel on the boundary
Step4. Solving the linear equation and copy the values back to target
Where is waldo
Where is waldo
Where is waldo
Where is waldo
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