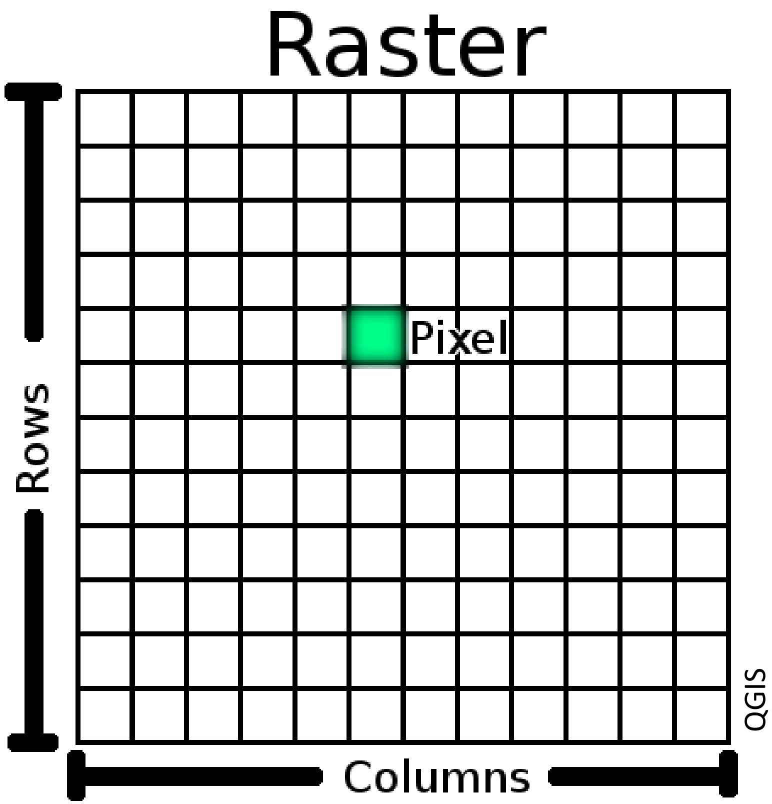
Research Methodology for Data Science

TU Delft

Last Week

Images

Rasterized images



Colorization using Optimization

Approach

- Given grayscale image and color annotations create colored image
- The scribbled colors are propagated to all pixels

Levin et al.

Formulating the Optimization Problem

Generic form

Find minimizer of the objective among all candidates satisfying the constraints!

- Variables: I_i (with $i \notin C$)
- Objective:

$$E(I) = \frac{1}{2}\sum_{i=1}^{n}\left(\sum_{j\in N(i)} w_{ij}(I_i - I_j)\right)^2$$

- Constraints: $I_i = \bar{I}_i$ for all $i \in C$

How can such a problem be solved?

Euler Lagrange Equation

A critical points

- The derivative vanishes
- Example in $\mathbb{R}^n$

$$f(x) = \frac{1}{2} x^T A x + b^T x + c$$

$$\frac{d}{dx} f(x) = Ax + b$$

- The critical point satisfies the linear equation

$$Ax + b = 0$$

- Solution x is a minimum if A is positive definite (only positive eigenvalues) and a maximum if A is negative definite (only negative eigenvalues)

Back to Colorization

Objective:

$$E(I) = \frac{1}{2}\sum_{i=1}^{n}\left(\sum_{j\in N(i)} w_{ij}(I_i - I_j)\right)^2$$

- Consider matrix $L \in \mathbb{R}^{n\times n}$ with
  - $L_{ii} = \sum_{j\in N(i)} w_{ij}$ diagonal entries
  - $L_{ij} = -w_{ij}$ for $j \in N(i)$
  - $L_{ij} = 0$ other entries

Then $(L\,I)_i = \sum_{j\in N(i)} w_{ij}(I_i - I_j)$

$$E(I) = \frac{1}{2}\|L\,I\|^2 = \frac{1}{2}I^T L^T L I$$

Summary of Algorithm

Colorization

- Load image and scribbles
  - Convert to YIQ color space
- Construct matrices L,S and vectors $\bar{I}$ and $\bar{Q}$
- Construct linear systems for I and Q and solve them

$$(L^T L + aS^T S)I = aS^T \bar{I}$$

- Save resulting image
  - Convert resulting image to RGB color space

Today

Overview

- Image gradients
- Gradient-based image sharpening
- Gradient-based image blending
- Sparse matrices
- The assignment
- Matlab tips
- Linear solvers

Image Gradients

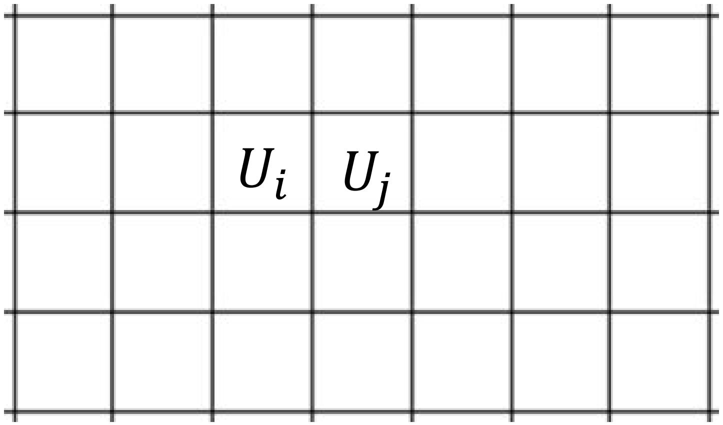
Image Gradients

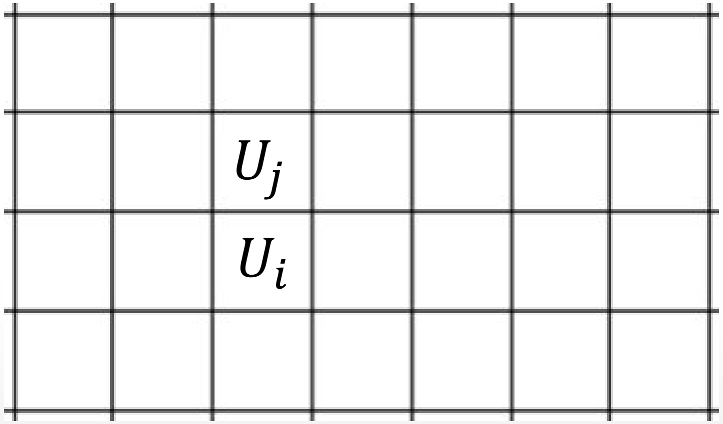
Image gradients

- Difference of neighboring pixel values
  - right minus left
  - up minus down

$$g_k = \frac{1}{2}(U_j - U_i)$$

  - In color images: three difference values for every pair of neighboring pixels





- Every such difference g_k gets an index $k \in \{1,2,\ldots,m\}$
- Question: For an image of width w and height h, what is m?

Gradient Vector and Matrix

Gradient vector

- We denote the m-dim. vector listing all the g_k by g

$$g = \begin{pmatrix} g_1 \\ \vdots \\ g_m \end{pmatrix}$$

- In color images, we have three such vectors g (one for every channel)

Gradient Matrix

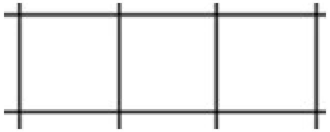
- The gradient matrix G is the $m \times n$ matrix that maps pixel values to their gradient vector

$$G\,U = g$$

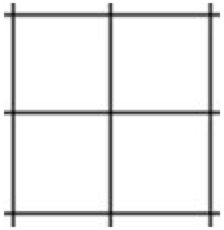
Example

What is the gradient matrix G?

1. **For an image with only 3 pixels and 1 row?**



2. **For an image with 4 pixels and 2 rows?**



Gradient-based
Image Sharpening

Gradient-based Image Sharpening

Image sharpening

- Idea: Compute gradients g_k re-scale them and construct a new (sharper) image whose gradients best match the rescaled gradients.

Remark: For color image repeat the procedure for every channel

Algorithm (Image sharpening)

Input: $\bar{U}$ (Pixel values of input image), c_s, $c_{\bar{U}}$ (parameter)

1. Construct gradient matrix G
2. Compute gradients of input image $\bar{g} = G\bar{U}$
3. Solve
$$\min_U(\|GU - c_s\,\bar{g}\|^2 + c_{\bar{U}}\|U - \bar{U}\,\|^2)$$
4. Return minimizer U

Result

Example

Input

Result with

$c_s = 3., c_{\bar{U}} = .5$

Solving the Optimization Problem

Quadratic Polynomial

$$f(U) = \|GU - c_s\bar{g}\|^2 + c_{\bar{U}}\|U - \bar{U}\|^2$$
$$= (GU - c_s\bar{g})^T(GU - c_s\bar{g}) + c_{\bar{U}}(U - \bar{U})^T(U - \bar{U})$$
$$= U^T G^T G U - 2c_s U^T G^T \bar{g} + {c_s}^2 \bar{g}^T \bar{g} + c_{\bar{U}}(U^T U - 2U^T \bar{U} + \bar{U}^T \bar{U})$$
$$= U^T(G^T G + c_{\bar{U}} Id)U - 2U^T(c_s G^T \bar{g} + c_{\bar{U}}\bar{U}) + {c_s}^2 \bar{g}^T \bar{g} + \bar{U}^T \bar{U}$$

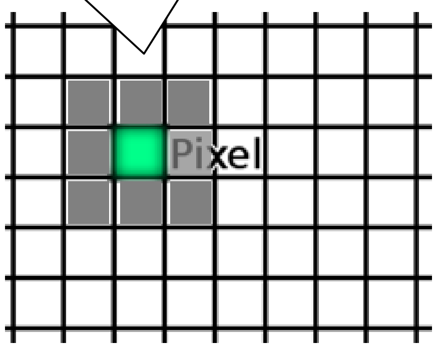
Minimum is the solution of the linear system

$$(G^T G + c_{\bar{U}} Id)U = c_s G^T \bar{g} + c_{\bar{U}}\bar{U}$$

Identity Matrix

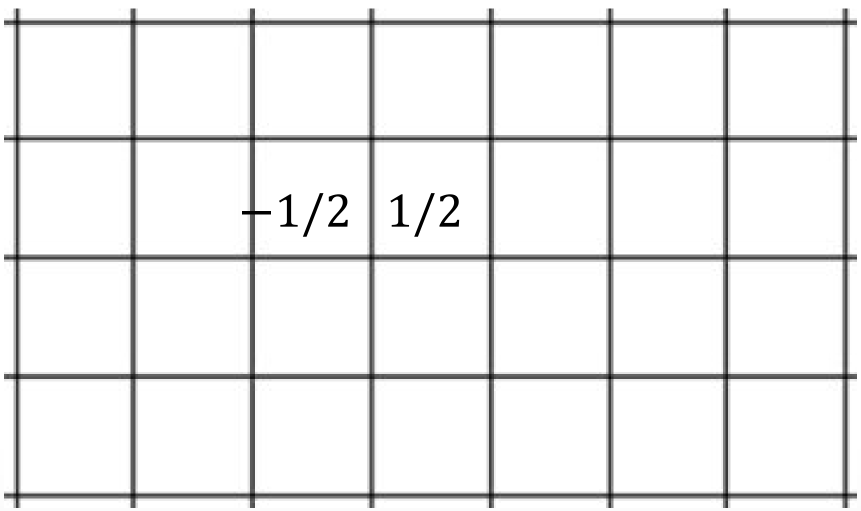
Remark: Laplace Matrix

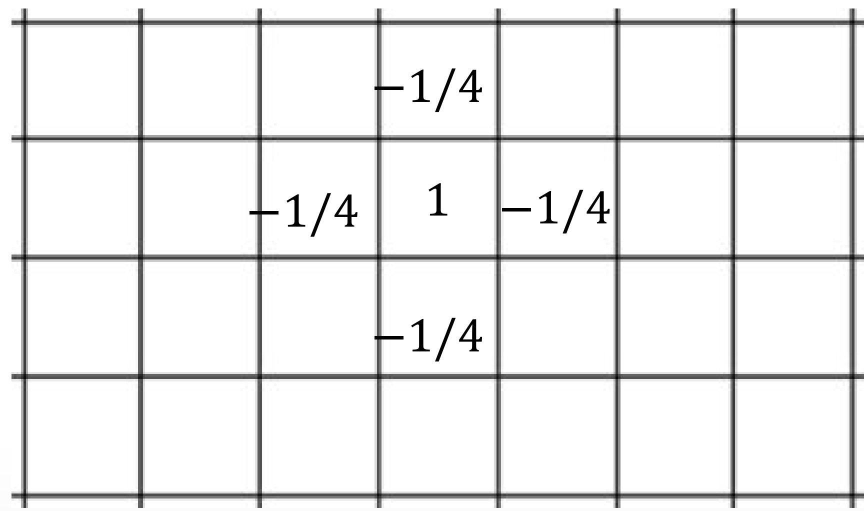
To get this pattern use also the “diagonal” edges in G.



The Laplace matrix

- $L = G^T G$
- A matrix of this type was used for colorization
- Difference of pixel value to the average of its neighbors





Gradient-based Image Blending

Gradient-based Blending

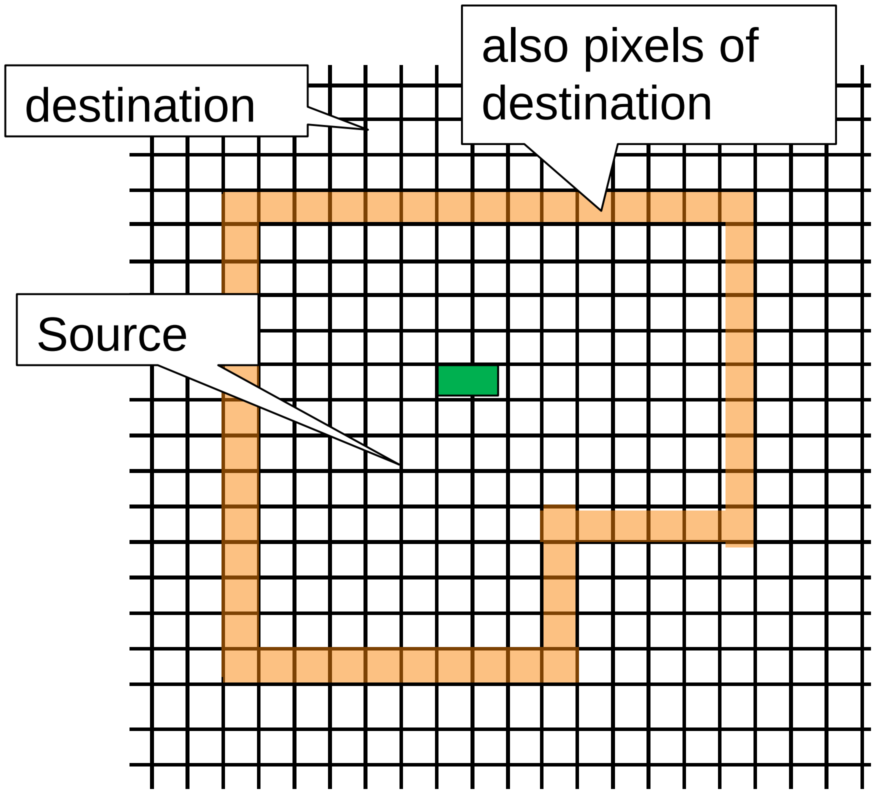
The problem

sources destinations cloning seamless cloning

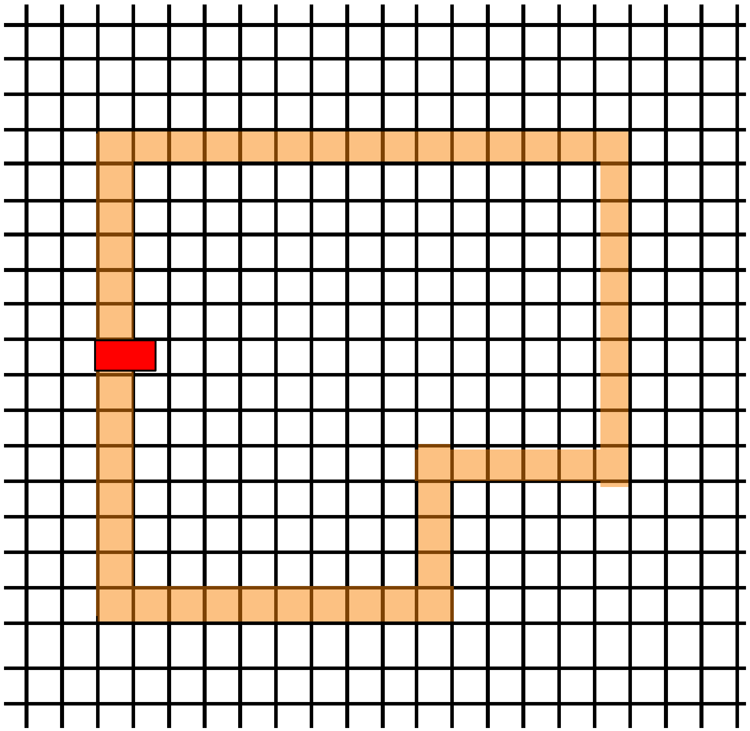
Closer Look

Zoom-in

Types of gradients



Inner gradients in source image: Difference of the values of two pixels in the interior of the source

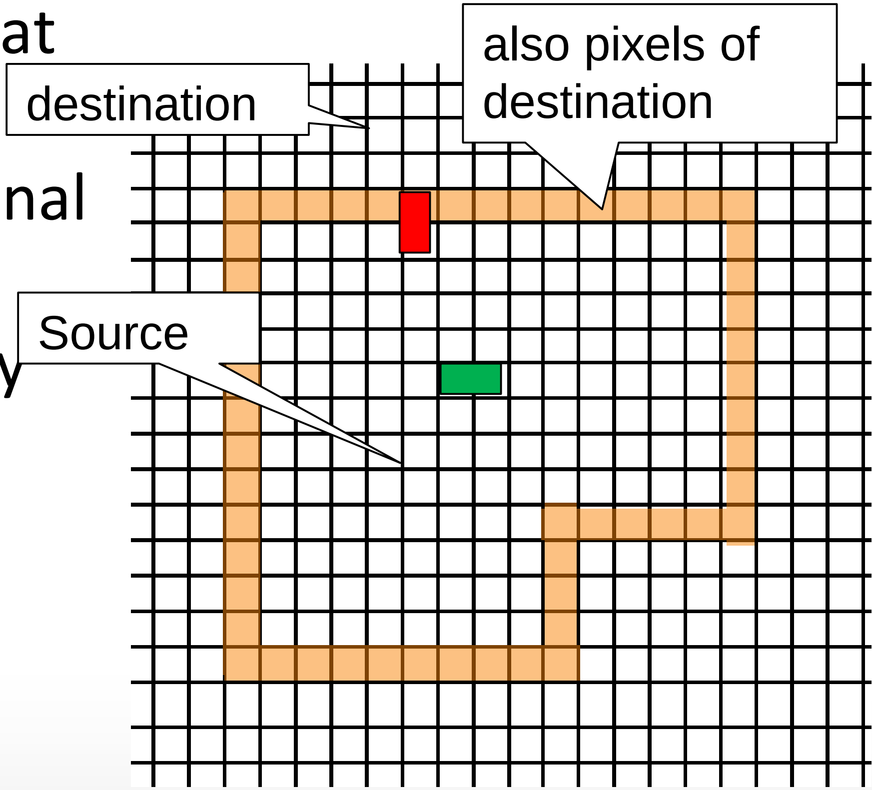


Boundary gradients: Difference of the values of two pixels one from the source and one from the destination (orange)

Gradient-based Image Blending

Idea

- Compute inner gradients $\tilde{g}$ of source image
- Compute “blended source” that is a new source image whose gradients best match the original source while the boundary of the destination is preserved by fixing the pixel values of the destination image at the boundary and minimizing the boundary gradients



Gradient-based Image Blending

Explicit:

- Let I and B denote the sets of indices of the inner and the boundary gradients
- Let $\tilde{g}_i$ and g_i denote the gradients of the source image and the image that is constructed
- Consider the objective function

Difference to source image

Difference to destination image at the boundary

$$f(U) = \sum_{i \in I} |g_i - \tilde{g}_i|^2 + \sum_{i \in B} |g_i|^2$$

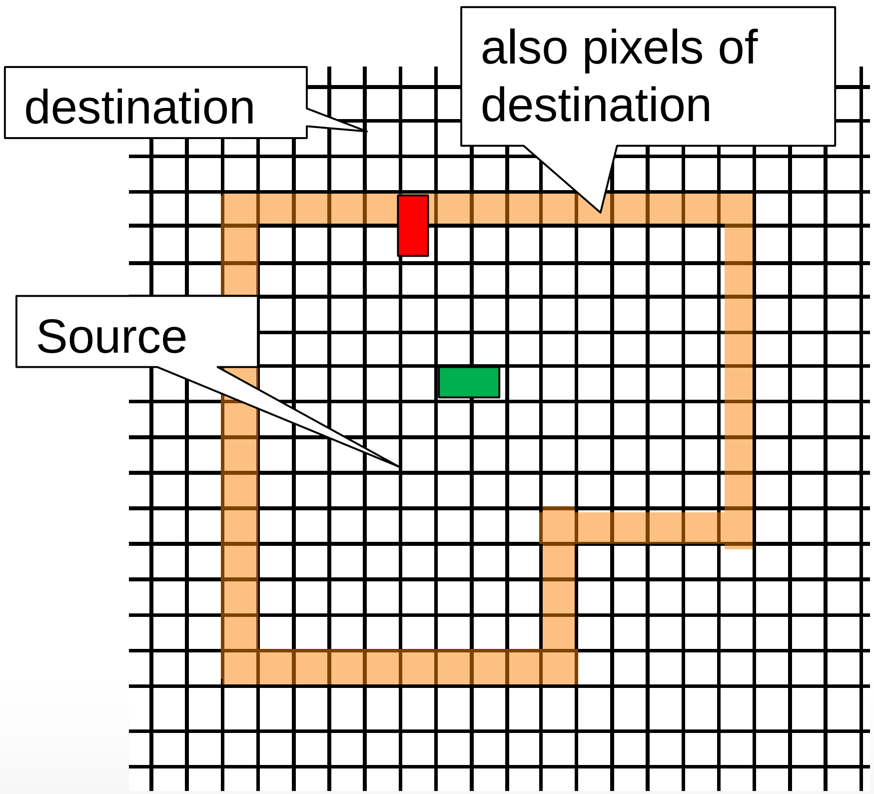
- Minimize f subject to the constraint destination pixels that are not overlaid are preserved

What happens?

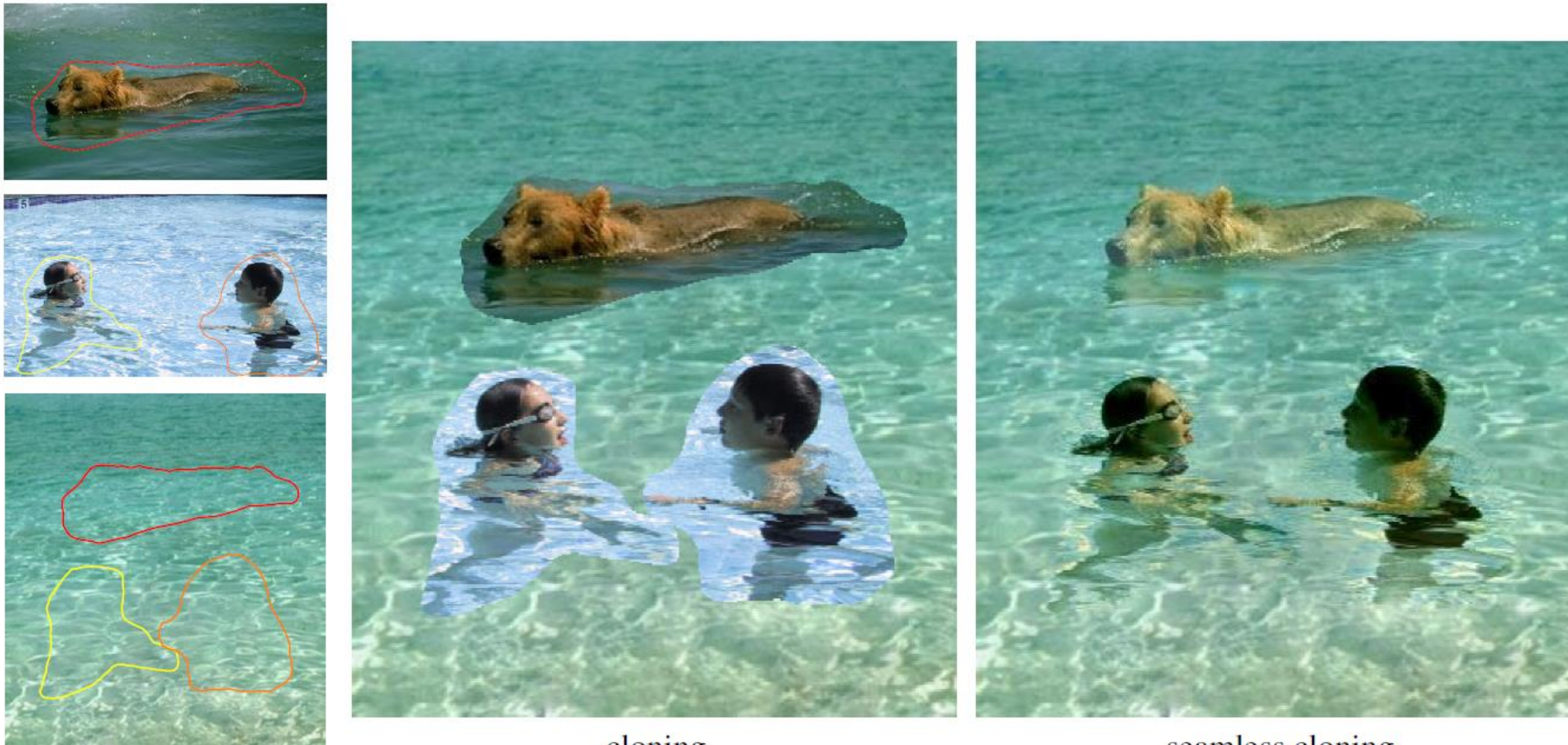
$$f(U) = \sum_{i \in I} |g_i - \tilde{g}_i| + \sum_{i \in B} |g_i|$$

cloning

seamless cloning



Examples

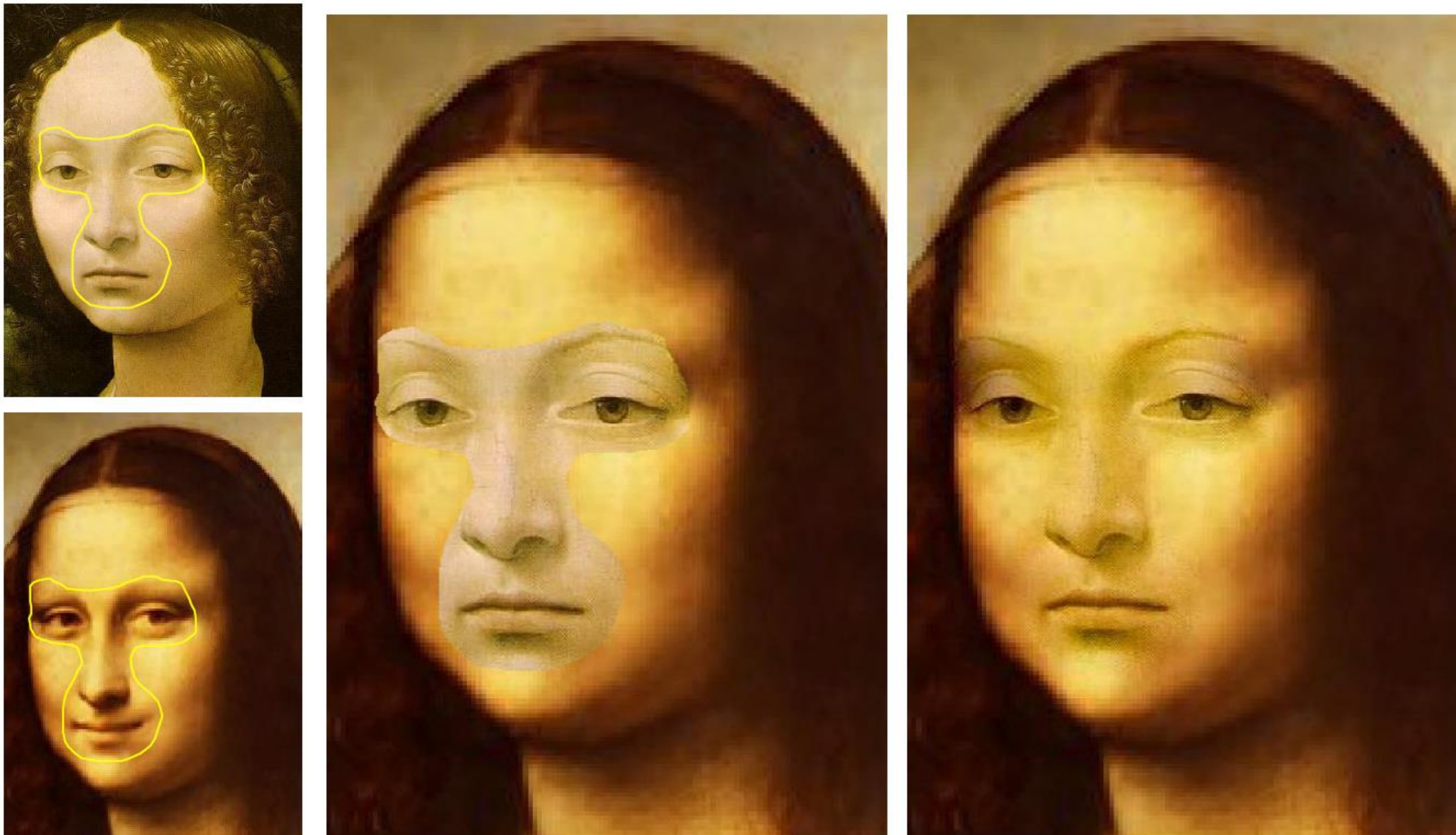


sources/destinations

cloning

seamless cloning

Examples



source/destination cloning seamless cloning

Implementation

A "Simple" Implementation (part 1)

- Create two images: both of the same width and height
  - image 1 contains source, remaining pixels are white
  - image 2 is the destination
- Compute a gradient matrix G that contains only the inner gradients and the boundary gradients (but not the gradients between the white pixels in image 1)
- Repeat the following steps for all color channels
- Compute the source gradients

$$\tilde{g} = GU^{im1}$$

- Set the boundary gradients in $\tilde{g}$ to zero

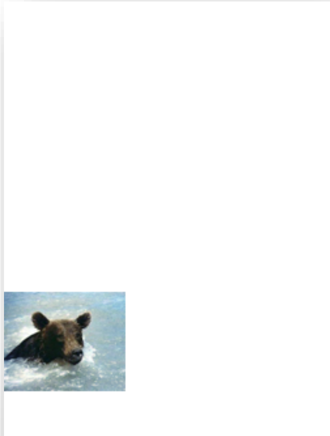


image 1

image 2

Implementation

A "Simple" Implementation (part 2)

- Compute selector matrix S for the pixels in the destination that are not overlaid (same as the white pixels in image 1)
- Solve the linear system

$$(G^T G + aS^T S)U = G^T \tilde{g}$$

I am using soft enforcement of constraints here. Hard constraint were discussed in the colorization lecture

The value of a should be large enough a to ensure that the destination image is preserved

- U are the pixels of the blended image!

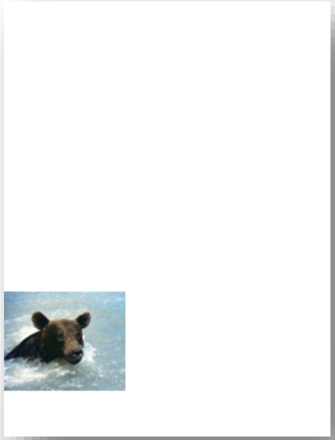
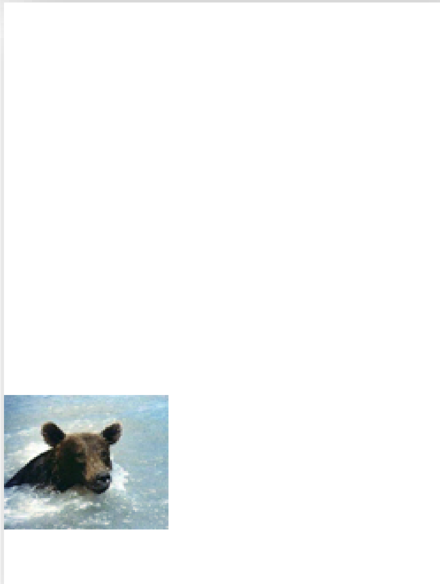


image 1

image 2

Result



Source: A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, M. Cohen, “Interactive Digital Photomontage”, SIGGRAPH 2004

set of originals

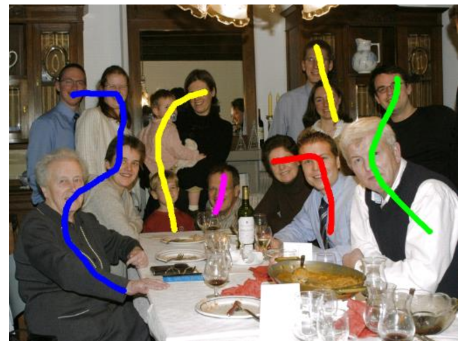
actual

photomontage

perceived

Source images

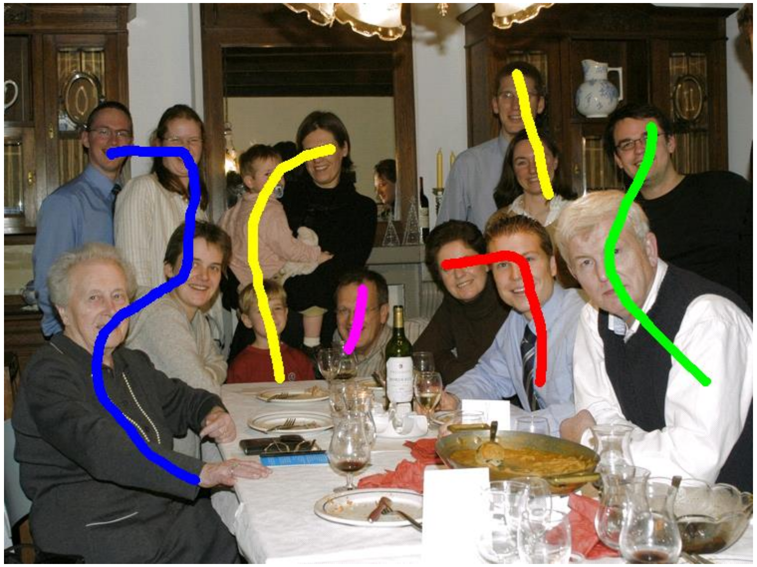
Brush strokes



Computed labeling

Composite

Brush strokes



Computed labeling

Sparse Matrices

Sparse Matrices

Compressed Column Form

An $m \times n$ matrix with up to $nzmax$ entries is represented by

- an integer array p of length $n + 1$
- an integer array i of length $nzmax$ and
- a real array a of length $nzmax$

Example

Example

$$A = \begin{bmatrix} 4.5 & 0 & 3.2 & 0 \\ 3.1 & 2.9 & 0 & 0.9 \\ 0 & 1.7 & 3.0 & 0 \\ 3.5 & 0.4 & 0 & 1.0 \end{bmatrix}$$

```
int p [ ]     = { 0,                  3,                  6,            8,           10 } ;
int i [ ]     = { 0,    1,    3,    1,    2,    3,    0,    2,    1,    3    } ;
double a [ ] = { 4.5, 3.1, 3.5, 2.9, 1.7, 0.4, 3.2, 3.0, 0.9, 1.0 } ;
```

- Row indices of entries in column j are stored in $i[p[j]]$ through $i[p[j+1]-1]$, and the numerical values are stored in the same locations in a
- The entry $p[n]$ gives the number of entries in the matrix

Sparse Matrices

Construction

- Sparse matrices are typically constructed from lists of triplets specifying for every entry its row and column index and its value.
    - Sparse matrix in compressed column form is constructed using a bucket sort algorithm

Examples of operations

- Matrix-vector multiplication, matrix-matrix multiplication
- Matrix addition, matrix transposition
- Row and column permutations
- **Solving linear systems**

Assignment

Assignment -- Informal

Implementations using Matlab

- Gradient-based image sharpening
- Gradient-based image blending
- (Bonus) Colorization

Report

- Report on experiences
- Examples of results
- Division of labor (work was divided in the group)

Matlab Tips

Some Matlab Functions

Read, write and display images

- image=imread('in.png')
- imwrite(image,'out.png')
- figure, imshow(image)

Get width, height and number of chanels

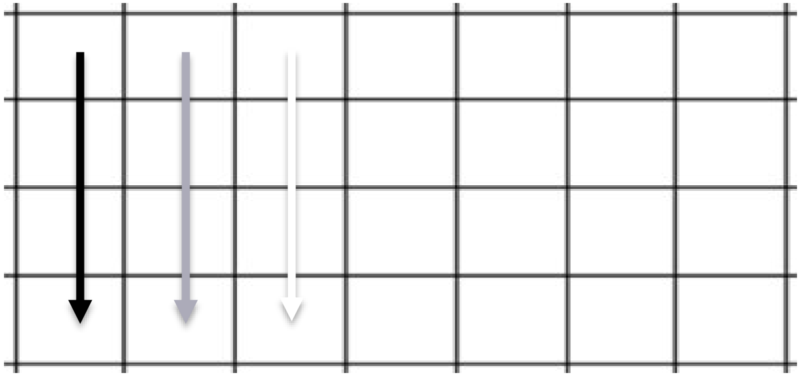
- [h w d]=size(image);

Remember: arrays, vectors, matrices etc. start with index 1

Some Matlab Functions

Create vectors of pixel values

- coords = double(reshape(image,w*h,3))/255;
  - Pixels are ordered column after column



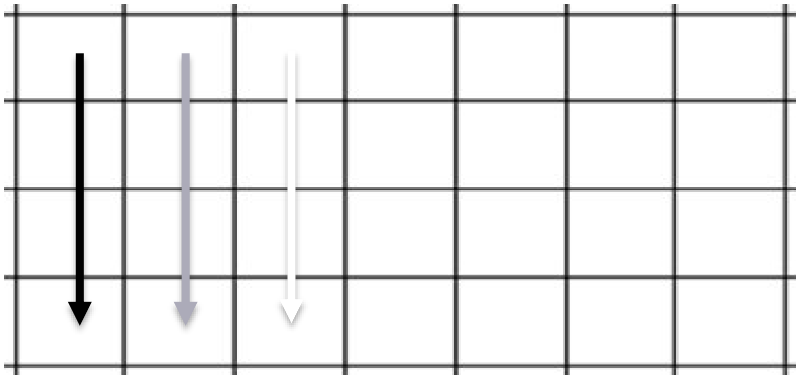
Read values of individual pixels

- val = coords(34,1);

Some Matlab Functions

Create image from vectors of pixel values

- image =uint8(reshape(coords,h,w,d)*255);
  - Same ordering is used



- Use imwrite(image,'out.png') or figure, imshow(image) to save and display the image

Matlab – Sparse Matrix Construction

Construct sparse matrices

- Collect list of triplets specifying column index, row index and value for every matrix entry
- Example:

  i = [1 2 2 3] ;

  j = [2 1 3 3];

  v = [2.3 1.2 -2.2 0.3];
- To construct sparse matrix

  smatrix = sparse(i, j, v);

  To get a matrix with the same entries n columns and m rows:

  smatrix = = sparse(i, j, v, n, m);
- What is the resulting matrix?

Matlab – Matrix Operations

Matrix addition/multiplication

- $M + N$
- $M * N$

Transposition

- G'
- Example: $L = G' * G$

Solve linear systems

- $x = A\backslash \mathrm{b};$
- The backslash operation analyzes the matrix and selects a solver. Of course solvers can also be called directly.

Further Reading

Sparse matrices and sparse direct solver

- A survey of direct methods for sparse linear systems
  T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar
  http://faculty.cse.tamu.edu/davis/publications.html

Gradient-based image sharpening and blending

- *GradientShop: A Gradient-Domain Optimization Framework for Image and Video Filtering*
  SIGGRAPH 2010
  http://grail.cs.washington.edu/projects/gradientshop/
- Poisson Image Editing, SIGGRAPH 2003