

Image Processing 2

Optimization-based Image Processing

TI3135TU Visual Data Processing

Lecture 3



Klaus Hildebrandt



Overview

- › Lecture 2: Image Processing 1 - Basics
 - › Raster, Pixels, Colors
 - › Histograms
 - › Point operations
 - › Image filters
 - › Gradients and edges
- › Lecture 3: Image Processing 2 - Optimization-based image processing
 - › Colorization by optimization
 - › Gradient matrix
 - › Gradient-based image sharpening
 - › Gradient-based image blending

Literature – Lecture 3

- › Gradient-based image sharpening and blending
 - › *GradientShop: A Gradient-Domain Optimization Framework for Image and Video Filtering*
Pravin Bhat, C. Lawrence Zitnick, Michael Cohen, Brian Curless
SIGGRAPH 2010
<http://grail.cs.washington.edu/projects/gradientshop/>
 - › *Poisson Image Editing*
Patrick Pérez, Michel Gangnet, Andrew Blake
SIGGRAPH 2003
 - › *Colorization using optimization*
Anat Levin, Dani Lischinski, and Yair Weiss
ACM SIGGRAPH 2004
- › 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>

Colorization by Optimization

Image Colorization/Recolorization

- › Problem

- › Add color to monochrome images or movies

Original photo



After colorization in Recolored

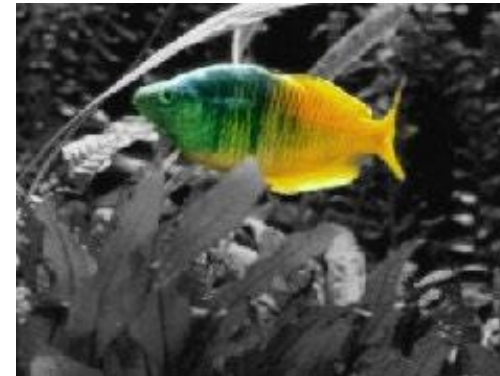


Recolored

Image Recolorization

- › Problem

- › Change the colors in images or videos



Weiss et al.

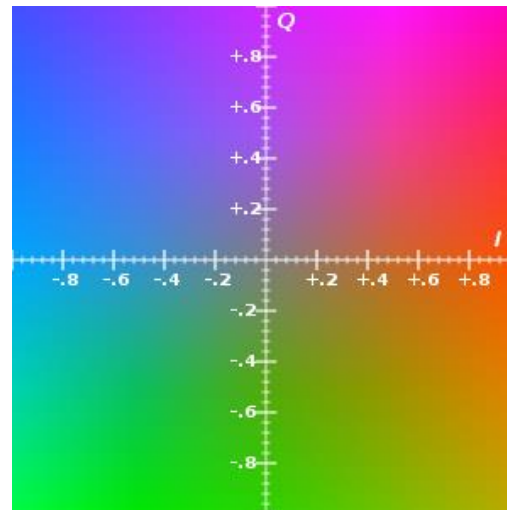


Cedeño et al.

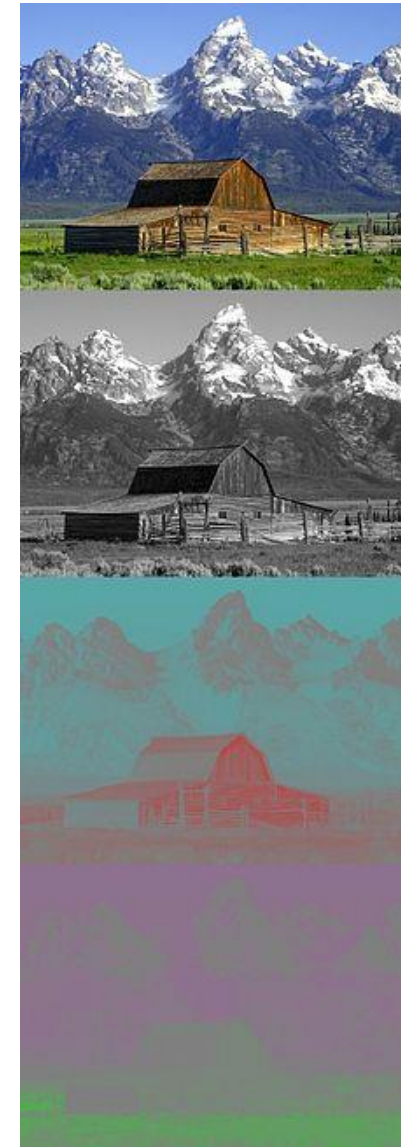
Reminder: YIQ Color Space

- › Color space YIQ
 - › 3 color values are stored
 - › Y for luminance
 - › I,Q for chrominance (color)
 - › YIQ is the color space used by the (analog) NTSC color TV system

Remark concerning notation:
We use I to denote the intensity for grayscale images. The YIQ also uses I for one channel but this different from the intensity of a grayscale image.



IQ for Y=0.5



YIQ

Y

I

Q

Colorization using Optimization

› Approach

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



Levin et al.

Propagation of Colors

- › Propagation

- › Neighboring pixels with similar intensities should get similar colors

$$Y \rightarrow I, Q$$

Luminance
channel

Color
channels



Levin et al.

Propagation of Colors

› Propagation

› Neighboring pixels

Remark: Only the color channels, I and Q, are taken from the scribbles. The luminance channel, Y, of the input image is preserved.

This means the pixel values of the result at the scribbles are a blending of the luminance of the input and the color information of the scribbles.

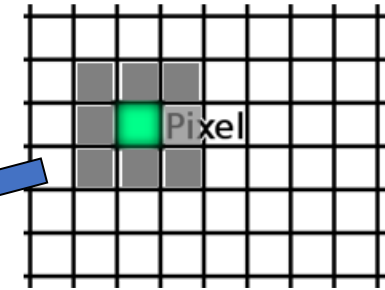


Levin et al.

Formulating the Optimization Problem

> Notation

- > Every pixel gets an index $1, 2, \dots, n$
- > The Y, I, Q values of the i^{th} pixel are Y_i, I_i, Q_i
- > We denote the set of neighbors of pixel i by $N(i)$
- > We denote the set of pixels for which colors are given by C and by \bar{I}_i and \bar{Q}_i the given color values



> Minimization problem (only for I)

Find the color values I_i such that

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

Analog for Q

is minimal and $I_i = \bar{I}_i$ for all $i \in C$



What Happens?

- › Optimization problem

- › 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 $i \in C$

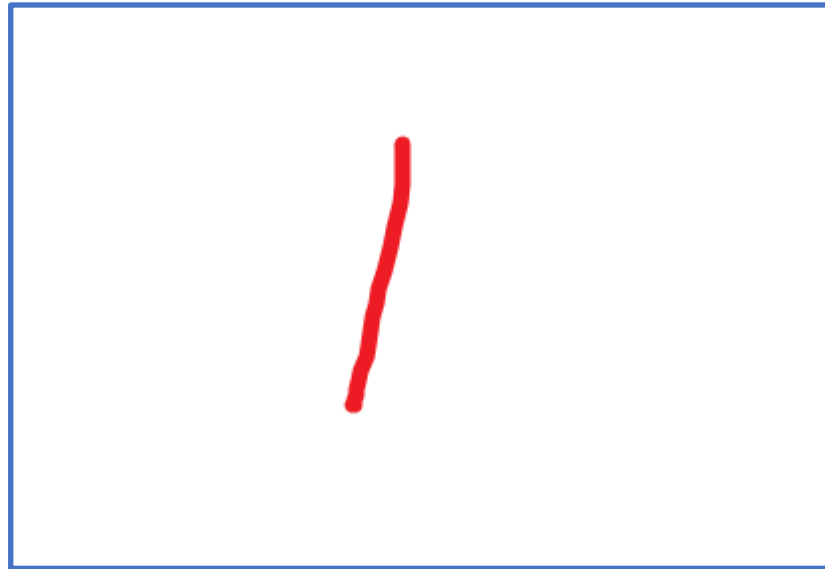
Minimize difference between color at a pixel and a weighted average of the neighbors!



Simple Examples

› What is the minimizer?

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



The Weights

> The weights

> $\tilde{w}_{ij} = e^{-(Y_i - Y_j)^2 / 2\sigma_i^2}$

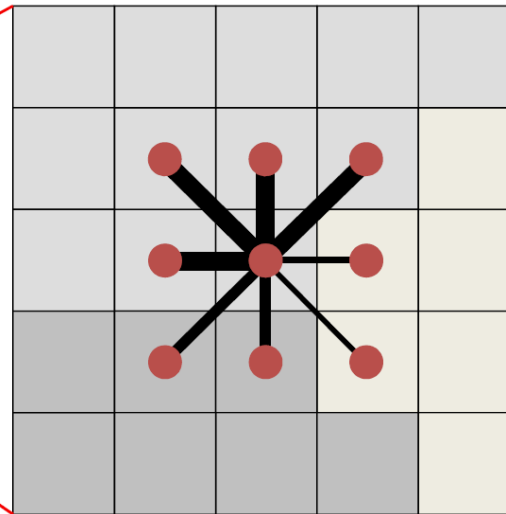
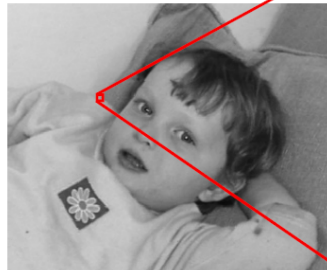
Variance of
luminance around i

> $w_{ij} = \tilde{w}_{ij} / (\sum_{j \in N(i)} \tilde{w}_{ij})$

Normalization of weights

> Weights are positive and $\sum_{j \in N(i)} w_{ij} = 1$

> Effect: pixels with similar luminance value have stronger influence on each others color



Simple Example

› What is the minimizer?

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



Solve the Problem!

- 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?
 - › Let us take a step back

Polynomials

› Quadratic polynomial on \mathbb{R}

› General form: $f(x) = \frac{1}{2} a x^2 + b x + c$

Quadratic term

constant term

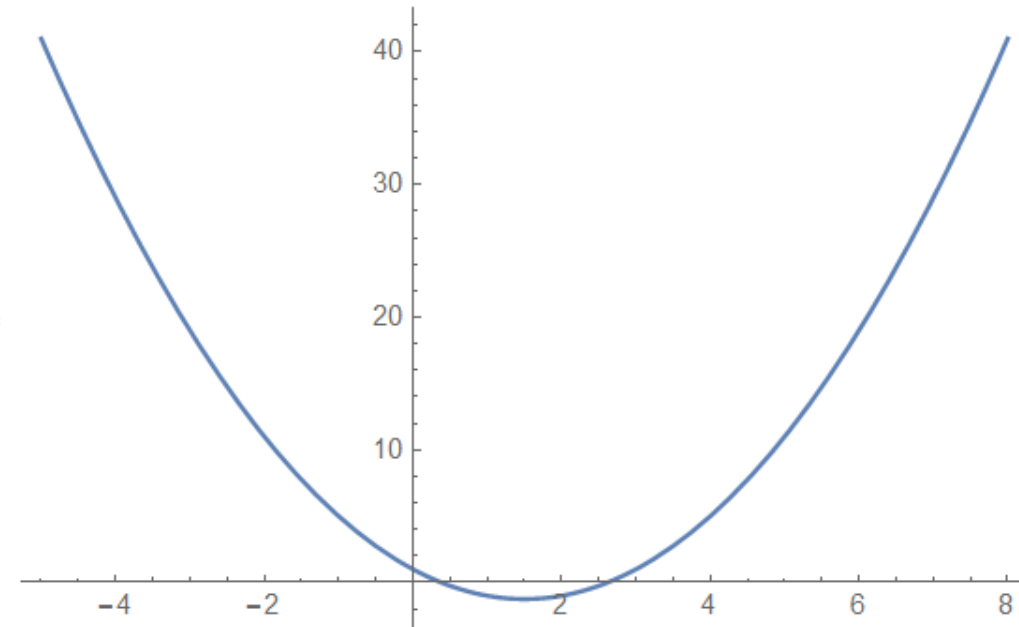
Linear term

› $a, b, c \in \mathbb{R}$

› Example:

```
In[4]:= Plot[x^2 - 3 x + 1, {x, -5, 8}]
```

Out[4]=



Euler Lagrange Equation

- › Critical points

- › The derivative vanishes

- › Example in \mathbb{R}

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

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

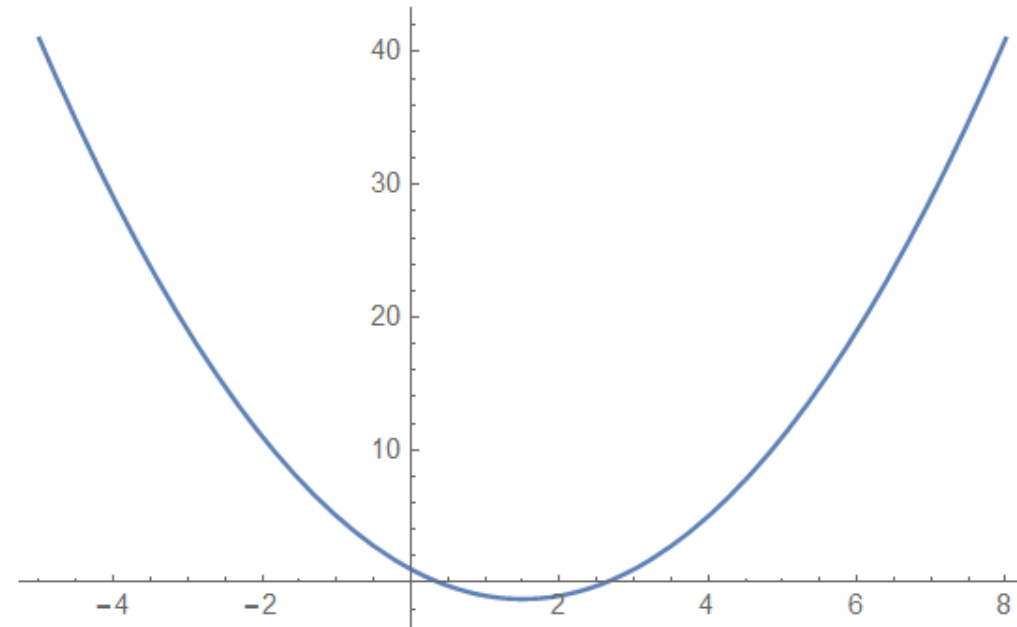
- › Critical point satisfies the linear equation

$$ax + b = 0$$

- › Solution x is a minimum if $a > 0$
and a maximum if $a < 0$

```
In[4]:= Plot[x^2 - 3 x + 1, {x, -5, 8}]
```

Out[4]=



Quadratic Polynomial

› In \mathbb{R}^2 :

› General form:

$$f(x_1, x_2) = a_1x_1^2 + a_2x_1x_2 + a_3x_2^2 + b_1x_1 + b_2x_2 + c$$

Quadratic Polynomial

› In \mathbb{R}^2 :

› General form:

$$f(x_1, x_2) = a_1 x_1^2 + a_2 x_1 x_2 + a_3 x_2^2 + b_1 x_1 + b_2 x_2 + c$$

$$= \frac{1}{2} (x_1 \quad x_2) \begin{pmatrix} & \\ & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + c$$

Quadratic Polynomial

› In \mathbb{R}^2 :

› General form:

$$f(x_1, x_2) = a_1 x_1^2 + a_2 x_1 x_2 + a_3 x_2^2 + b_1 x_1 + b_2 x_2 + c$$

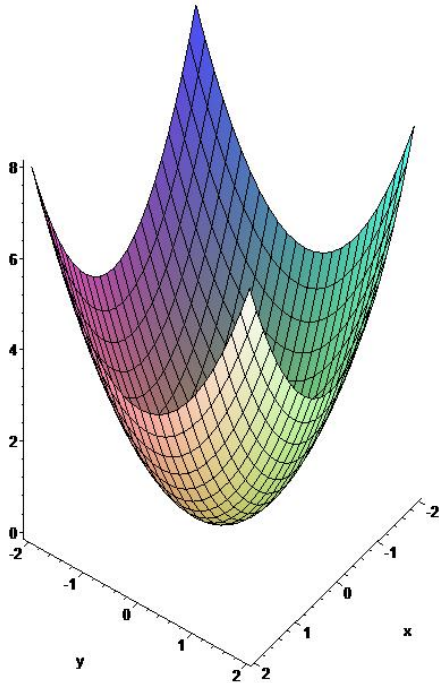
$$= \frac{1}{2} \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 2a_1 & a_2 \\ a_2 & 2a_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 & b_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + c$$

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

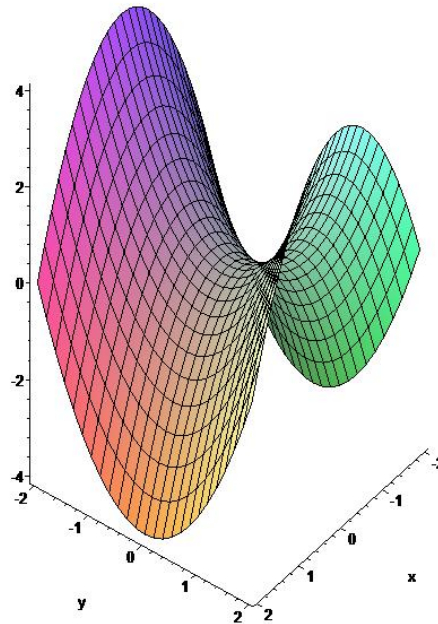
Optimization

- › Quadratic Polynomials

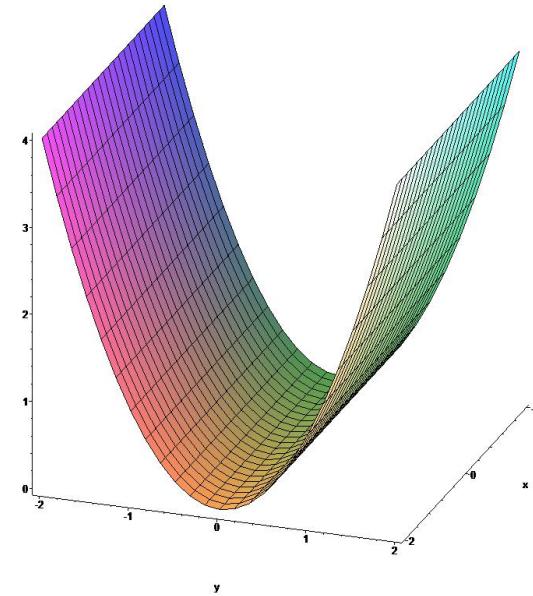
- › Example: in \mathbb{R}^2



$$\lambda_1 = 1, \lambda_2 = 1$$



$$\lambda_1 = 1, \lambda_2 = -1$$



$$\lambda_1 = 1, \lambda_2 = 0$$

Quadratic Polynomial

› In \mathbb{R}^n :

› General form:

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

› $A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n, c \in \mathbb{R}$

› A is symmetric---- Remark: this is no restriction. If A is not symmetric, the symmetric matrix $\tilde{A} = \frac{1}{2}(A + A^T)$ yields the same function.

Euler Lagrange Equation

- › A critical points
 - › The derivative vanishes
 - › Example in \mathbb{R}^n

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

- › The critical point satisfies the linear equation
$$A x + b = 0$$
- › 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:

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

Is it quadratic?

How can we find out?

Back to Colorization

› Objective:

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

All terms are quadratic: $I_i I_j$ or I_i^2 .

Hence $E(I)$ is quadratic

$$E(I) = \frac{1}{2} \sum_{i,j=1}^n a_{ij} I_i I_j = \frac{1}{2} I^T A I$$

But what are the matrix coefficients a_{ij} ?

I^T denote the transpose of
the vector I

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

$$\sum_{i=1}^n x_i^2 = \|x\|^2 = x^T x$$

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

$$E(I) = \frac{1}{2} \sum_{i=1}^n ((L I)_i)^2 = \frac{1}{2} \|L I\|^2 = \frac{1}{2} I^T L^T L I$$

So the matrix we look for is:
 $A = L^T L$

Back to Colorization

› Objective:

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

Hence, we have to solve $L^T L I = 0$

Are we missing something?

Yes, the constraints!



Constraints

- › Two variants:

- › Hard constraints: constraints are exactly satisfied

$$I_i = \bar{I}_i \text{ for all } i \in \mathcal{C}$$

- › Soft constraints: constraints are not exactly satisfied, but only in least squares sense.

- › Add the term $\frac{a}{2} \sum_{i \in \mathcal{C}} (I_i - \bar{I}_i)^2$ to the objective, where $a > 0$ is a weight. A larger value of a means that more penalty is paid for deviating from the constraints.



Soft Constraints

› Least squares term

$$\frac{a}{2} \sum_{i \in \mathcal{C}} (I_i - \bar{I}_i)^2$$

To compute the solution, set up the linear system

› Use a rectangular matrix $S \in \mathbb{R}^{m \times n}$, where m is the number of constraints

$$\|S(I - \bar{I})\|^2 = \sum_{i \in \mathcal{C}} (I_i - \bar{I}_i)^2$$

› $n > m$

$$\left\| \begin{pmatrix} \vdots \\ \vdots \end{pmatrix} \left(\begin{pmatrix} \vdots \\ \vdots \end{pmatrix} - \begin{pmatrix} \vdots \\ \vdots \end{pmatrix} \right) \right\|^2$$

What is S ?

Soft Constraints

› Selector Matrix

S is a rectangular matrix that selects the pixels that are constrained

› Let $\{c_1, c_2, \dots, c_m\}$ be the constrained pixels. Then the selector matrix S has the entries

$S_{ic_i} = 1$ and all other entries are zero

Example of a Selector Matrix

Setting: We have an image with four pixels and want to select the pixels with indices 2 and 4.

1	2
3	4

› The selector matrix in this setting is:

$$S = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

› Test: Applying S to a vector v with four pixel should result in the vector w with two entries, which are the values of the second and fourth entries of v .

$$Sv = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} v_2 \\ v_4 \end{pmatrix}$$

Soft Constraints

Since only the m selected pixels of $\bar{I} \in \mathbb{R}^n$ are relevant, it is more convenient to work with $\hat{I} \in \mathbb{R}^m$.

› Resulting linear system?

› $\hat{I} = S\bar{I} \in \mathbb{R}^m$ lists the color values of the constrained pixels

› The quadratic polynomial modeling the constraints is

$$\text{› } \frac{1}{2} \|S I - \hat{I}\|^2 = \frac{1}{2} I^T S^T S I - \hat{I}^T S I + \frac{1}{2} \hat{I}^T \hat{I}$$

› Putting all together: To solve the soft constraint problem, we minimize the objective

$$\frac{1}{2} \|L I\|^2 + \frac{a}{2} \|S I - \hat{I}\|^2$$

by solving the linear system

$$(L^T L + a S^T S) I = a S^T \hat{I}$$

Hard Constraints

- › Linear system for hard constraints
 - › Use Lagrange multipliers $\lambda_1, \lambda_2, \dots, \lambda_m$
 - › Artificial variables are discarded after solving
 - › The color values for I are the solution of the system

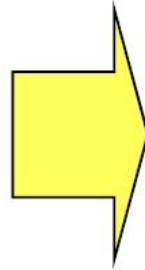
$$\begin{pmatrix} L^T L & S^T \\ S & 0 \end{pmatrix} \begin{pmatrix} I \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ \hat{I} \end{pmatrix}$$

- › We need to solve the system for I and λ . I describes the solution and λ can be discarded.

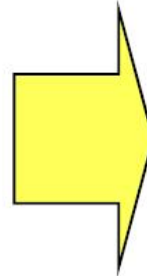
Results – Comparison to Original



Results



Results



Recoloring



Summary of Algorithm

› Colorization

- › Load image and scribbles

 - › Convert to YIQ color space

- › Construct matrices L , S and vectors \hat{I} and \hat{Q}

- › Construct linear systems for I and Q and solve them

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

- › Save resulting image

 - › Convert resulting image to RGB color space

Literature

- › Research Paper on Colorization

- › Colorization using optimization

- Anat Levin, Dani Lischinski, and Yair Weiss

- ACM SIGGRAPH 2004*

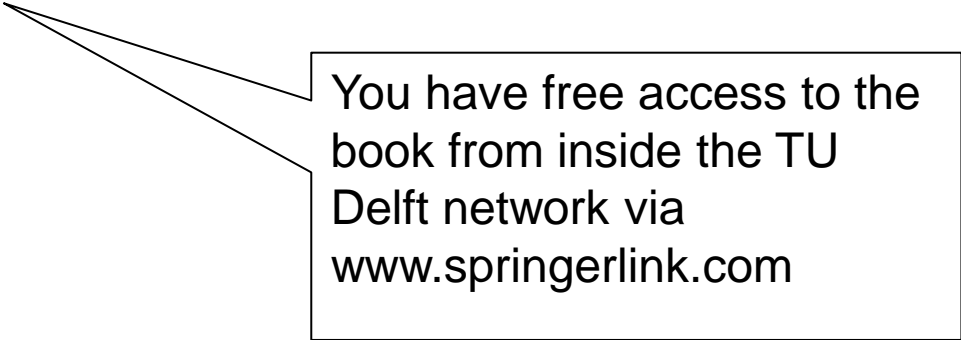
- › Book Chapter on Quadratic Programs with Constraints

- › Numerical Optimization

- Jorge Nocedal, Stephan J. Wright

- Springer 2006

- Chapter 16: Quadratic Programming



You have free access to the
book from inside the TU
Delft network via
www.springerlink.com

Gradient Matrix

Image Gradients

› Image gradients

› Difference of neighboring pixel values

› right minus left

› up minus down

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

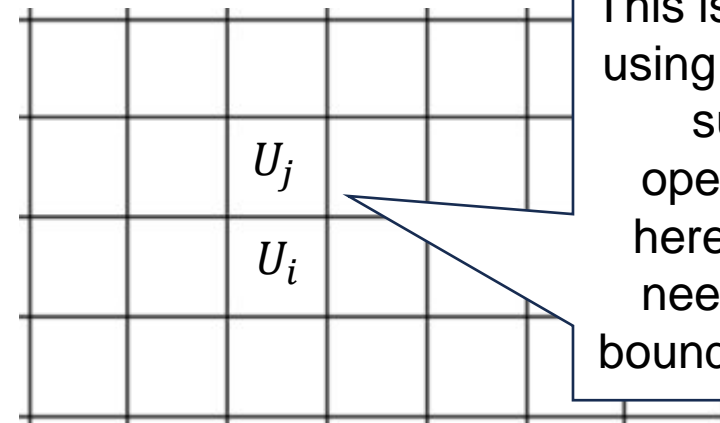
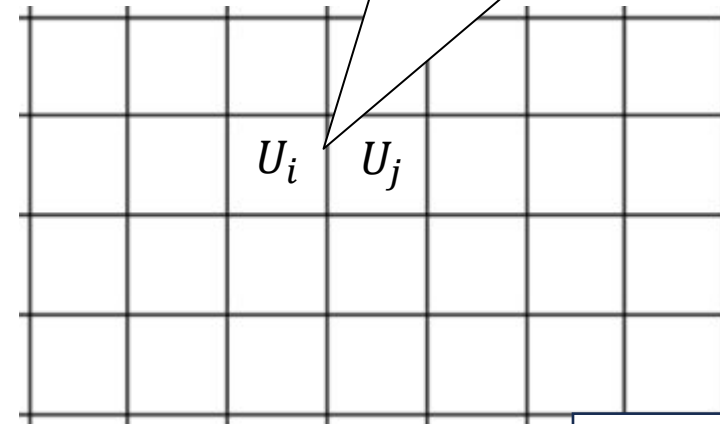
› In color images: three difference values for every pair of neighboring pixels

› Every such difference g_k gets an index $k \in \{1, 2, \dots, m\}$

› Question: For an image of width w and height h , what is m ?

The answer is:
 $w(h-1) + (w-1)h$

Pixels are indexed as discussed in the first lecture. U_i is the pixel value of the pixel with index i .



This is a bit different than using the Gradient filters such as a Sobel operator. The benefit here is that we do not need to concern with boundaries of the image.

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

$$G U = g$$

Remark: While the gradient matrix is not necessary to compute the gradients, it is required for computing an image whose gradients best match given gradients.

Example

› What is the gradient matrix G ?

For an image with only 3 pixels and 1 row?



Solution: Example 1

- › What is the gradient matrix G ?

For an image with only 3 pixels and 1 row?

- › Index the pixels

1	2	3
---	---	---

- › Index the gradients

$$g_1 = \frac{1}{2}(U_2 - U_1) \qquad g_2 = \frac{1}{2}(U_3 - U_2)$$

- › Then, the gradient matrix satisfying $GU = g$ is

$$G = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Remark

› Sanity check whether your result could be correct:

› $GU = g$

› Index the gradients

$$g_1 = \frac{1}{2}(U_2 - U_1)$$

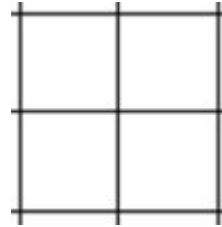
$$g_2 = \frac{1}{2}(U_3 - U_2)$$

$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}$$

Correct?

Example

- › What is the gradient matrix G ?
For an image with 4 pixels and 2 rows?



Solution: Example 2

- › What is the gradient matrix G ?

For an image with 4 pixels and 2 rows?

- › Index the pixels (you can choose any order)

1	2
3	4

- › Index the gradients

$$g_1 = \frac{1}{2}(U_2 - U_1)$$

$$g_2 = \frac{1}{2}(U_4 - U_3)$$

$$g_3 = \frac{1}{2}(U_1 - U_3)$$

$$g_4 = \frac{1}{2}(U_2 - U_4)$$

Solution: Example 2

› Then gradient matrix is

$$G = \frac{1}{2} \begin{pmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

› Test:

$$\frac{1}{2} \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} \stackrel{\text{Correct?}}{=} \begin{pmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{pmatrix}$$

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.

› Algorithm (Image sharpening)

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

Remark: For color image repeat the procedure for every channel

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

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

- › 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

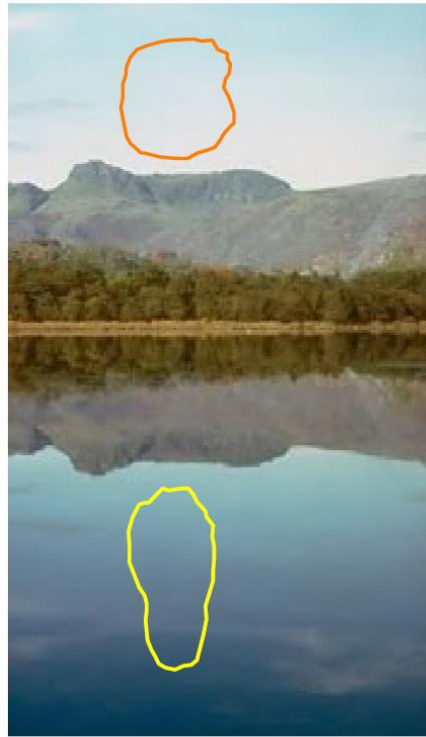
Gradient-based Image Blending

Gradient-based Blending

> The problem



sources



destinations



cloning



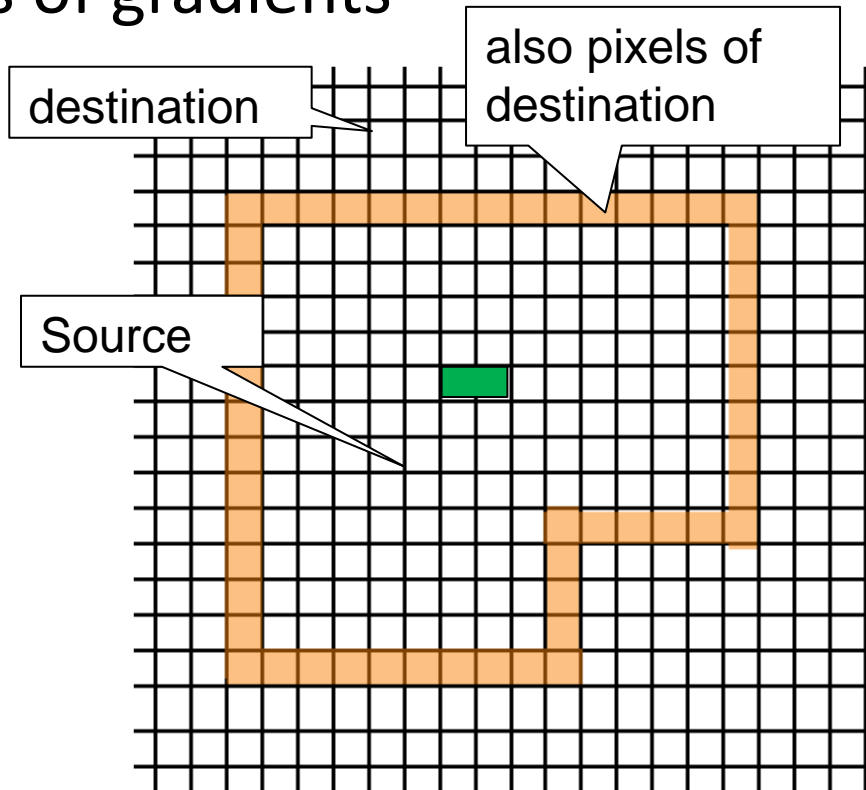
seamless cloning

Closer Look

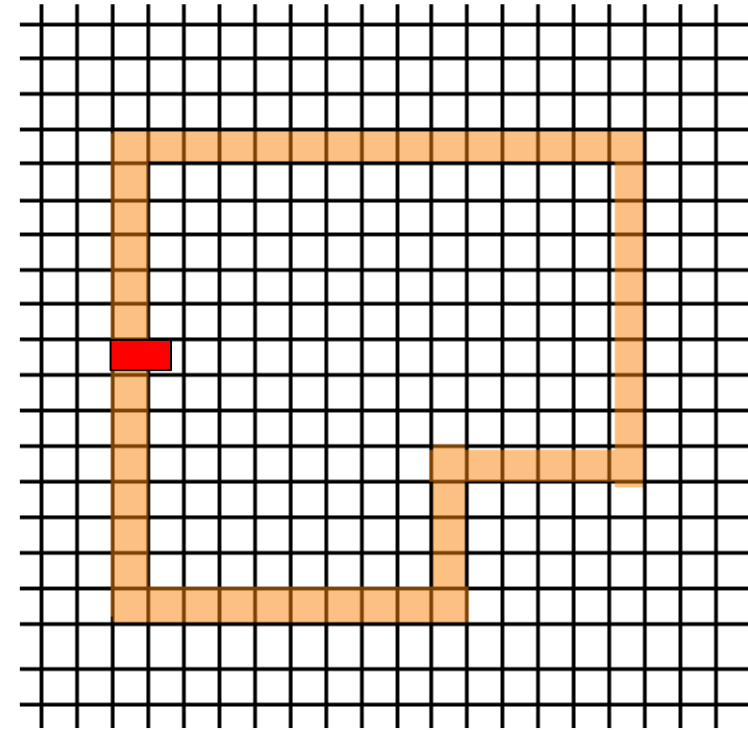


Zoom-in

> Types of gradients



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

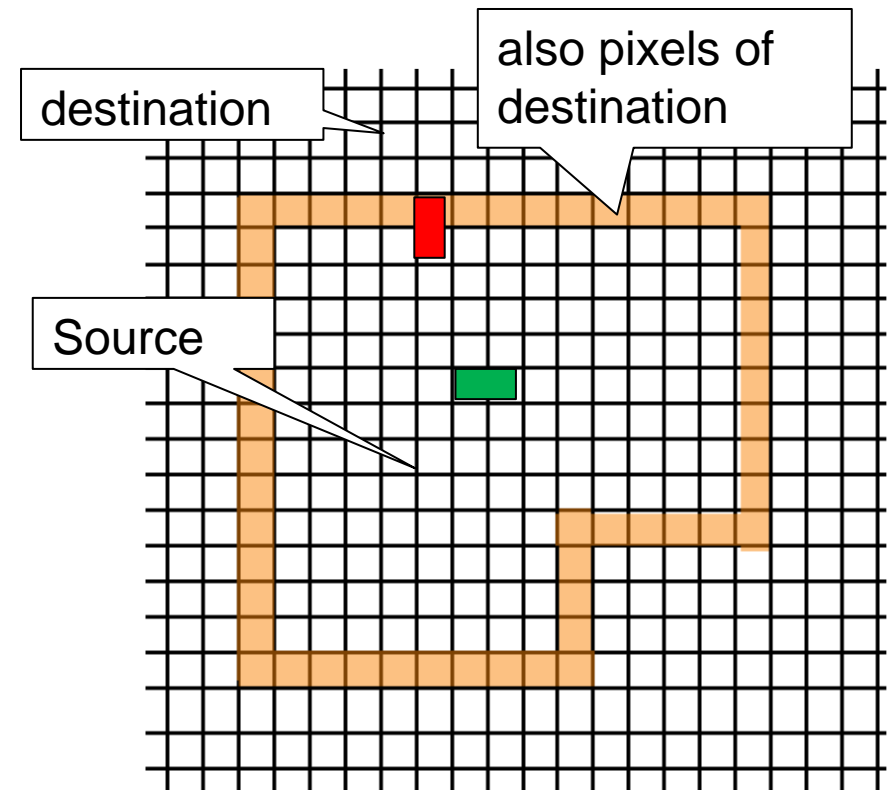


Boundary gradients:
Difference of the values of two
pixels one from the source and
one from the destination (orange)

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

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

Difference to
source image

Difference to
destination
image at the
boundary

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

What happens?

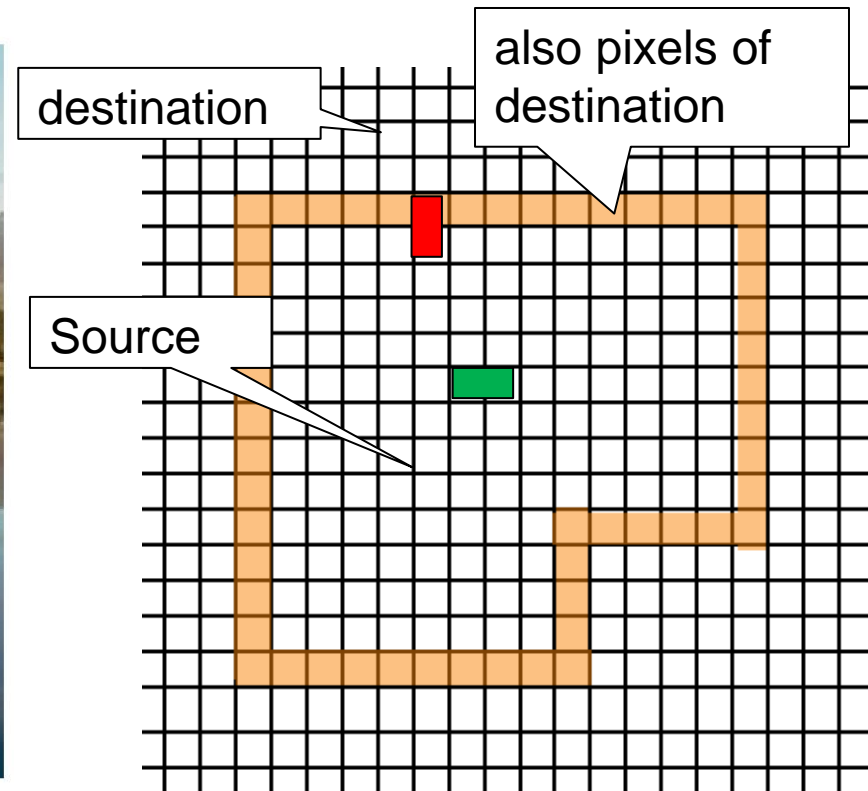
$$\triangleright f(U) = \sum_{i \in I} |g_i - \tilde{g}_i|^2 + \sum_{i \in B} |g_i|^2$$



cloning



seamless cloning



Examples



sources/destinations

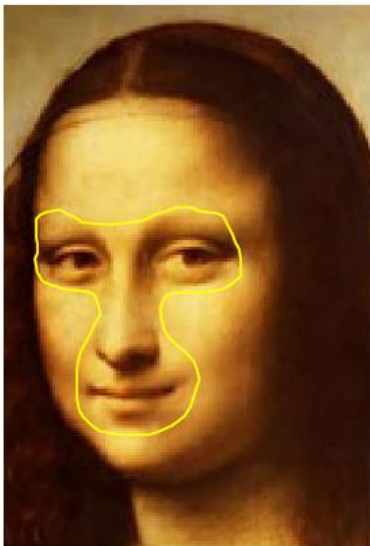
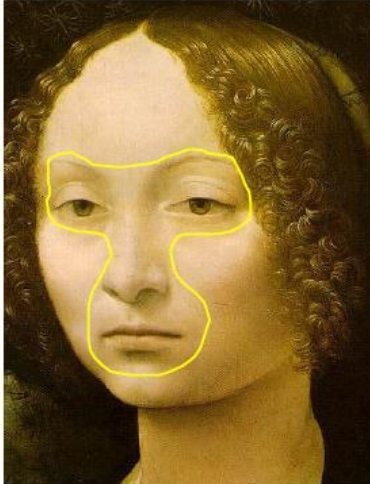


cloning



seamless cloning

Examples



source/destination



cloning



seamless cloning

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**

Explicit algorithms and more can be found in a survey paper, reference is given at the end of the lecture.

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>
- › More on iterative solvers: Conjugate Gradients
 - Numerical Optimization
Jorge Nocedal, Stephan J. Wright
Springer 2006
Chapter 5: Conjugate Gradients