Colorization

CVFX @ NTHU

3 March 2015

Outline

Colorization

The paper

Colorization using optimization

- Levin, Lischinski, and Weiss
- SIGGRAPH 2004
- Useful skills of linear algebra

Example

Advantages

Solving a sparse linear system

- Many numerical methods can be used

No segmentation required

- Users only need to draw some color scribbles

Intensity contrast is maintained

BlackMagic/TimeBrush Real-Life-Color

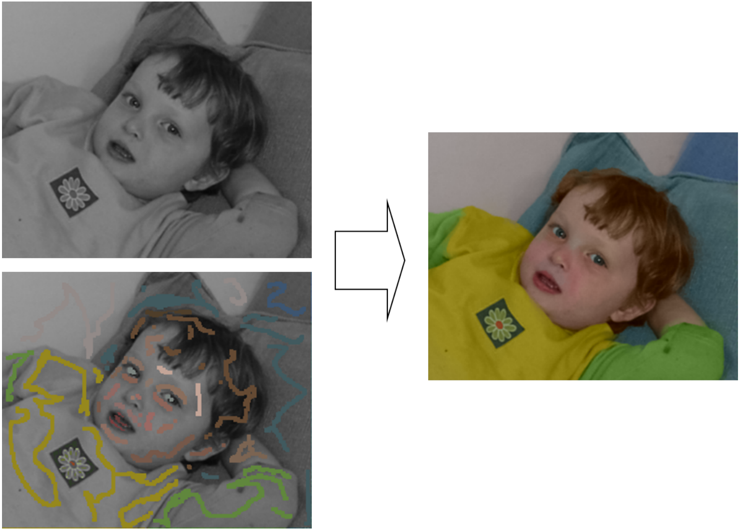
http://www.blackmagic-color.com
http://www.timebrush.com/

http://www.blackmagic-color.com/ http://www.timebrush.com/


http://www.blackmagic-color.com/alt/bm_samples2.html


http://www.blackmagic-color.com/alt/bm_samples3.html

Colorization



Recoloring

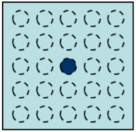
Modification for recoloring

```
sgI = rgb2ntsc(gI);
scI = rgb2ntsc(cI);
ntscIm(:,:,1) = sgI(:,:,1);
ntscIm(:,:,2) = scI(:,:,2);
ntscIm(:,:,3) = scI(:,:,3);

idx = find(sum(cI,3)>=3); % For white scribbles,
tmp1 = ntscIm(:,:,2);     % we have to copy from
tmp2 = sgI(:,:,2);        % the original image
tmp1(idx) = tmp2(idx);    % those colors of the
ntscIm(:,:,2) = tmp1;     % corresponding pixels.
tmp1 = ntscIm(:,:,3);
tmp2 = sgI(:,:,3);
tmp1(idx) = tmp2(idx);
ntscIm(:,:,3) = tmp1;
```

Local linearity

Assume that the color (U or V) at a pixel is a linear function of the intensity, and the linear coefficients are the same for all pixels in a small neighborhood



$$U_i \simeq a_k Y_i + b_k$$

Ω_k

$$(a_k, b_k)$$

Cost function

The squared error of linear approximation for the whole image

$$J(U) = \sum_k \left(\min_{a_k, b_k} \sum_{i \in \Omega_k} (U_i - a_k Y_i - b_k)^2 \right)$$

Scirbbles?

Consraints

$$U_i = u_i$$

Optimization

Find U that minimizes

$$J(U) = \sum_k \left(\min_{a_k, b_k} \sum_{i \in \Omega_k} (U_i - a_k Y_i - b_k)^2 \right)$$

subject to the constraints

$$U_i = u_i$$

Rewrite the cost function

$$J(U) = \sum_k \left(\min_{a_k, b_k} \sum_{i \in \Omega_k} (U_i - a_k Y_i - b_k)^2 \right)$$

$$J(U) = \sum_k \min_{a_k, b_k} \left\| \begin{pmatrix} Y_1 & 1 \\ Y_2 & 1 \\ \vdots & \\ Y_{|\Omega_k|} & 1 \end{pmatrix} \begin{pmatrix} a_k \\ b_k \end{pmatrix} - \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{|\Omega_k|} \end{pmatrix} \right\|^2$$

Let $P_k = \begin{pmatrix} Y_1 & 1 \\ Y_2 & 1 \\ \vdots & \\ Y_{|\Omega_k|} & 1 \end{pmatrix}$ and $\widetilde{U_k} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{|\Omega_k|} \end{pmatrix}$

Optimization (rewritten)

Find U that minimizes

$$J(U) = \sum_k \left(\min_{a_k, b_k} \left\| P_k \begin{pmatrix} a_k \\ b_k \end{pmatrix} - \widetilde{U_k} \right\|^2 \right)$$

subject to the constraints

$$U_i = u_i$$

Subproblem

to minimize

$$Q_k(a_k, b_k) = \left\| P_k \begin{pmatrix} a_k \\ b_k \end{pmatrix} - \widetilde{U_k} \right\|^2$$

is equivalent to find the solution of

$$P_k^T P_k \begin{pmatrix} a_k \\ b_k \end{pmatrix} = P_k^T \widetilde{U_k}$$

the least squares solution

$$\begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix} = \left(P_k^T P_k\right)^{-1} P_k^T \widetilde{U_k}$$

Substitute the solution into the sub-cost

$(AB)^T = B^T A^T$

$$\begin{pmatrix} a_k^* \\ b_k^* \end{pmatrix} = \left(P_k^T P_k\right)^{-1} P_k^T \widetilde{U}_k$$

$\|g\|^2 = g^T g$

$$Q_k^*(a_k^*, b_k^*) = \left\| P_k \left(P_k^T P_k\right)^{-1} P_k^T \widetilde{U}_k - \widetilde{U}_k \right\|^2$$

$$Q_k^*(a_k^*, b_k^*) = \underbrace{\left\| \left(P_k \left(P_k^T P_k\right)^{-1} P_k^T - I \right) \widetilde{U}_k \right\|^2}_{g} \qquad \|g\|^2 = g^T g$$

$$Q_k^*(a_k^*, b_k^*) = \underbrace{\widetilde{U}_k^T \left(P_k \left(P_k^T P_k\right)^{-1} P_k^T - I \right)^T}_{g^T} \underbrace{\left(P_k \left(P_k^T P_k\right)^{-1} P_k^T - I \right) \widetilde{U}_k}_{g}$$

$$P_k \left(P_k^T P_k\right)^{-1} P_k^T P_k \left(P_k^T P_k\right)^{-1} P_k^T - 2 P_k \left(P_k^T P_k\right)^{-1} P_k^T + I$$
$$= I - P_k \left(P_k^T P_k\right)^{-1} P_k^T$$

Rewrite the sub-cost

$$
\begin{aligned}
Q_k^*(a_k^*, b_k^*) &= \widetilde{U_k}^T \left(P_k \left(P_k^T P_k \right)^{-1} P_k^T - I \right)^T \left(P_k \left(P_k^T P_k \right)^{-1} P_k^T - I \right) \widetilde{U_k} \\
&= \widetilde{U_k}^T \left(I - P_k \left(P_k^T P_k \right)^{-1} P_k^T \right) \widetilde{U_k} \\
&= \widetilde{U_k}^T L_k \widetilde{U_k}
\end{aligned}
$$

$L_k = \left(I - P_k \left(P_k^T P_k \right)^{-1} P_k^T \right)$ is a matrix of size $|\Omega|_k$ by $|\Omega|_k$

Details

The (i,j) element of L_k is

$$\left(I - P_k\left(P_k^T P_k\right)^{-1} P_k^T\right)_{ij}$$

$$= \delta_{ij} - (Y_i \; 1)\begin{pmatrix} \sum_l^{|\Omega_k|} Y_l^2 & \sum_l^{|\Omega_k|} Y_l \\ \sum_l^{|\Omega_k|} Y_l & |\Omega_k| \end{pmatrix}^{-1} \begin{pmatrix} Y_j \\ 1 \end{pmatrix}$$

$$\begin{array}{cc} & j \\ i & \begin{bmatrix} 0 & & \\ & & \\ & & \end{bmatrix} \end{array}$$

$$\delta_{ij} = \begin{cases} 1, & i=j \\ 0, & i\neq j \end{cases}$$

$$I = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}$$

$$j$$

$$i \quad \begin{pmatrix} Y_1 & 1 \\ Y_2 & 1 \\ \vdots & \\ Y_{|\Omega_k|} & 1 \end{pmatrix} \left\{ \begin{pmatrix} Y_1 & Y_2 & \dots & Y_{|\Omega_k|} \\ 1 & 1 & & 1 \end{pmatrix} \begin{pmatrix} Y_1 & 1 \\ Y_2 & 1 \\ \vdots & \\ Y_{|\Omega_k|} & 1 \end{pmatrix} \right\}^{-1} \begin{pmatrix} Y_1 & Y_2 & \dots & Y_{|\Omega_k|} \\ 1 & 1 & & 1 \end{pmatrix}$$

Compute the inverse

$$\begin{pmatrix} \sum_l^{|\Omega_k|} Y_l^2 & \sum_l^{|\Omega_k|} Y_l \\ \sum_l^{|\Omega_k|} Y_l & |\Omega_k| \end{pmatrix}^{-1} = \frac{\begin{pmatrix} |\Omega_k| & -\sum_l^{|\Omega_k|} Y_l \\ -\sum_l^{|\Omega_k|} Y_l & \sum_l^{|\Omega_k|} Y_l^2 \end{pmatrix}}{|\Omega_k| \sum_l^{|\Omega_k|} Y_l^2 - (\sum_l^{|\Omega_k|} Y_l)^2}$$

$$= \frac{|\Omega_k| \begin{pmatrix} 1 & -\mu_k \\ -\mu_k & \sum_l^{|\Omega_k|} Y_l^2/|\Omega_k| \end{pmatrix}}{|\Omega_k|^2 \sigma_k^2}$$

$$= \frac{1}{|\Omega_k| \sigma_k^2} \begin{pmatrix} 1 & -\mu_k \\ -\mu_k & \sum_l^{|\Omega_k|} Y_l^2/|\Omega_k| \end{pmatrix}$$

Some efforts

The (i,j) element of L_k

$$
\begin{aligned}
(L_k)_{ij} &= \left(I - P_k\left(P_k^T P_k\right)^{-1} P_k^T\right)_{ij} \\
&= \delta_{ij} - (Y_i \;\, 1)\begin{pmatrix} \sum_l^{|\Omega_k|} Y_l^2 & \sum_l^{|\Omega_k|} Y_l \\ \sum_l^{|\Omega_k|} Y_l & |\Omega_k| \end{pmatrix}^{-1}\begin{pmatrix} Y_j \\ 1 \end{pmatrix} \\
&= \delta_{ij} - (Y_i \;\, 1)\frac{1}{|\Omega_k|\sigma_k^2}\begin{pmatrix} 1 & -\mu_k \\ -\mu_k & \sum_l^{|\Omega_k|} Y_l^2/|\Omega_k| \end{pmatrix}\begin{pmatrix} Y_j \\ 1 \end{pmatrix} \\
&= \delta_{ij} - \frac{1}{|\Omega_k|\sigma_k^2}\left(Y_iY_j - Y_i\mu_k - Y_j\mu_k + \frac{\sum_l^{|\Omega_k|} Y_l^2}{|\Omega_k|}\right) \\
&= \delta_{ij} - \frac{1}{|\Omega_k|\sigma_k^2}\left(Y_iY_j - Y_i\mu_k - Y_j\mu_k + \mu_k^2 + \frac{\sum_l^{|\Omega_k|} Y_l^2}{|\Omega_k|} - \mu_k^2\right) \\
&= \delta_{ij} - \frac{1}{|\Omega_k|\sigma_k^2}\left((Y_i - \mu_k)(Y_j - \mu_k) + \sigma_k^2\right) \\
&= \boxed{\delta_{ij} - \frac{1}{|\Omega_k|}\left(1 + \frac{1}{\sigma_k^2}(Y_i - \mu_k)(Y_j - \mu_k)\right)}
\end{aligned}
$$

$\boxed{\tilde{U}_k^T L_k \tilde{U}_k}$

Optimization

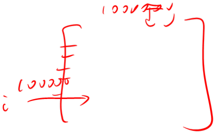
Find U that minimizes

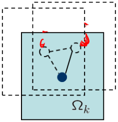
$$J(U) = \sum_k \widetilde{U_k}^T L_k \widetilde{U_k} = U^T L U$$

subject to the constraints

Solve a sparse linear system

$$\widetilde{U_k} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{|\Omega_k|} \end{pmatrix}$$





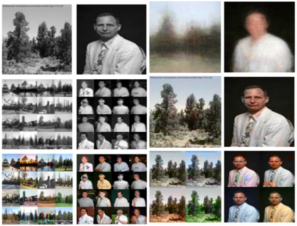
L is a large sparse N-by-N matrix whose (i, j) element is

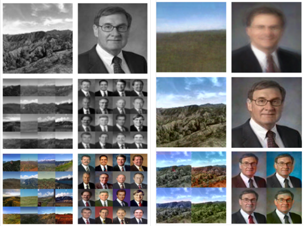
$$\sum_{k|(i,j)\in\Omega_k} \left(\delta_{ij} - \frac{1}{|\Omega_k|} \left(1 + \frac{1}{\sigma_k^2} (Y_i - \mu_k)(Y_j - \mu_k) \right) \right)$$

N is the number of pixels in the image

Learning from examples

- 80 million tiny images (Torralba et al.)





Conclusion

Scribbles

- Cannot fully rely on computers
- To get good results on hard problems, we must make some efforts too
- Human-computer interaction, distributed human computing, Amazon MT

Optimization

- Formulate your problem based on reasonable assumptions
- You are lucky if the cost function is quadratic and the constraints are linear

Another approach to scribble-based colorization

Fast Image and Video Colorization Using Chrominance Blending

- Yatziv and Sapiro
- IEEE Transactions on Image Processing, vol. 15, no. 5, 2006

YCbCr color spaces

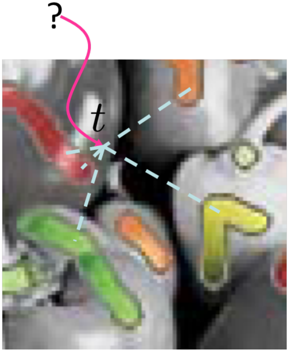
- Y channel: luminance
- Cb and Cr channels: chrominance

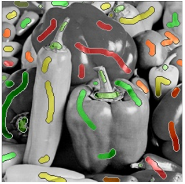
```
JPEG-YCbCr from 8-bit RGB
===============================================
Y =       + 0.299*R     +    0.587*G +    0.114*B
Cb = 128 - 0.168736*R - 0.331264*G +      0.5*B
Cr = 128 +      0.5*R - 0.418688*G - 0.081312*B

R, G, B in {0, 1, 2, ..., 255}
```

Idea

chrominance unknown





Ω: all pixels

Ω_c: scribbled pixels

Problem

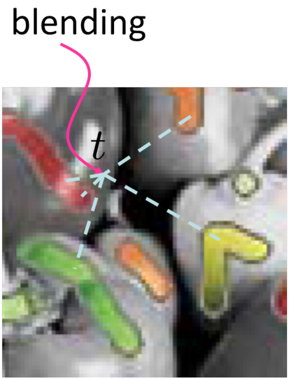
Given

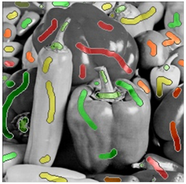
- the chrominance channels in Ω_c
- the luminance channel in Ω

To estimate

- the chrominance channels in $\Omega \setminus \Omega_c$

Chrominance blending





Ω: all pixels

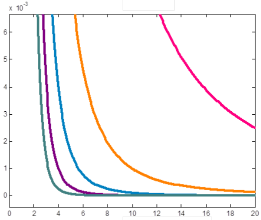
Ω_c: scribbled pixels

How to blend

$$\text{chrominance}(t) \leftarrow \frac{\sum_{\forall c \in \text{chrominances}(\Omega_c)} W(d_c(t))c}{\sum_{\forall c \in \text{chrominances}(\Omega_c)} W(d_c(t))}$$

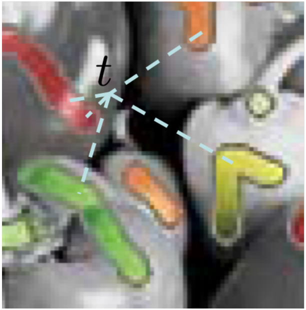
$d_c(t)$: the distance from pixel t to the 'nearest' pixel with chrominance c

$W(d) = d^{-b}$, $1 \leq b \leq 6$

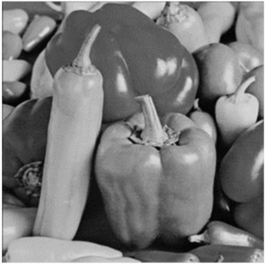


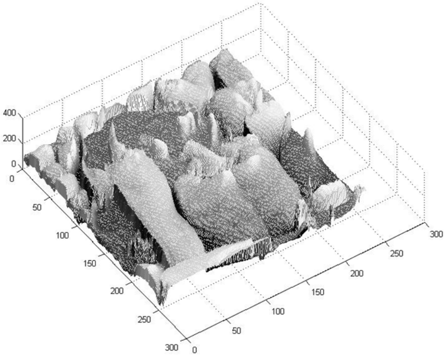
How toe define nearness?

$d_c(t)$: the distance from pixel t to the 'nearest' pixel with chrominance c



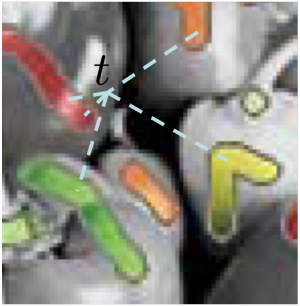
Incorporate luminance information

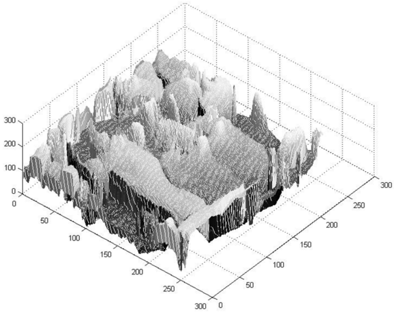




Incorporate luminance information

How far do we need to travel (uphill and downhill) from t to a scribbled pixel?

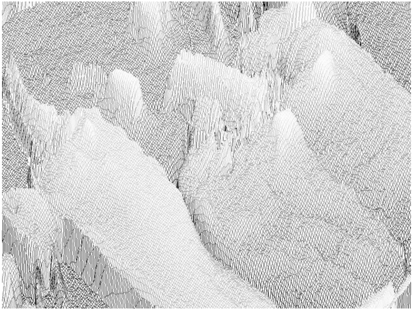




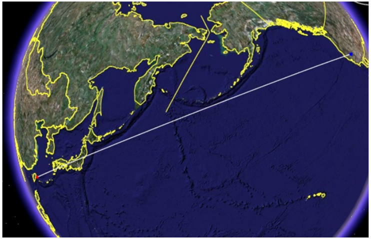
Reasonable?

Consider

- smooth areas
- rugged areas
- edges

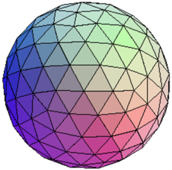


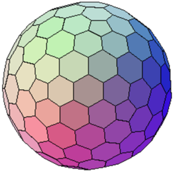
Geodesic distance



Geodesic distance

the shortest path between points on the space



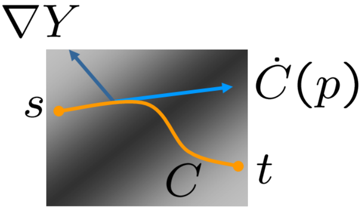


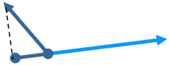
http://en.wikipedia.org/wiki/Geodesic_dome

Definition

Let s and t be two points in Ω and let $C(p) : [0, 1] \to \Omega$ be a curve in Ω. Also, let $C_{s,t}$ represent a curve connecting s and t such that $C(0) = s$ and $C(1) = t$. We define the geodesic distance between s and t by

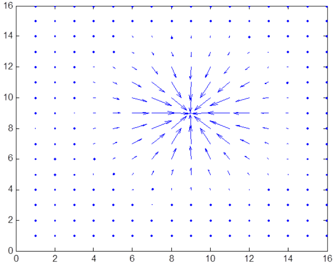
$$d(s, t) \equiv \min_{C_{s,t}} \int_0^1 |\nabla Y \cdot \dot{C}(p)| d\, p$$

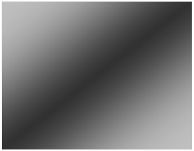


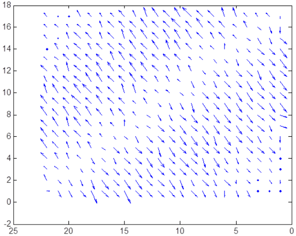


Visualization

$$d(s,t) = \min_{C_{s,t}} \int_0^1 |\nabla Y \cdot \dot{C}(p)|\,dp$$

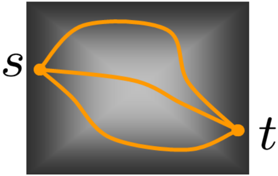






Find the shortest path

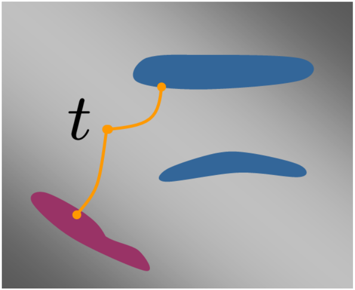
$$d(s,t) = \min_{C_{s,t}} \int_0^1 |\nabla Y \cdot \dot{C}(p)| d\,p$$



find the 'flatest' path connecting the two points

Distance to a certain chrominance

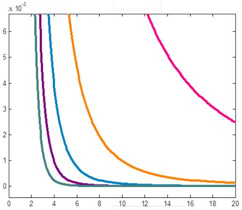
$$d_c(t) := \min_{\forall s \in \Omega_c : \text{chrominance}(s) = c} d(s,t)$$

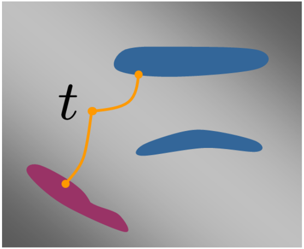


Chrominance blending

$$\text{chrominance}(t) \leftarrow \frac{\sum_{\forall c \in \text{chrominances}(\Omega_c)} W(d_c(t))c}{\sum_{\forall c \in \text{chrominances}(\Omega_c)} W(d_c(t))}$$

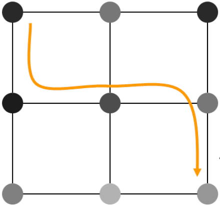
$W(d) = d^{-b}, \; 1 \le b \le 6$





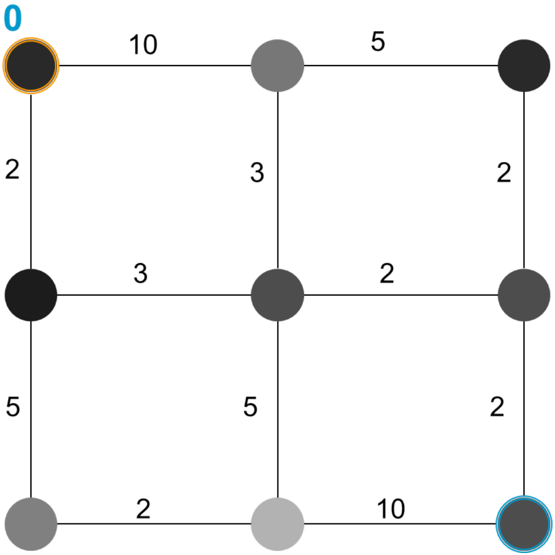
How to compute geodesic distance on an image?

- Create a graph, and find the shortest path
- Dijkstra's algorithm
- Using a priority queue
- $O((E + V) \log V)$

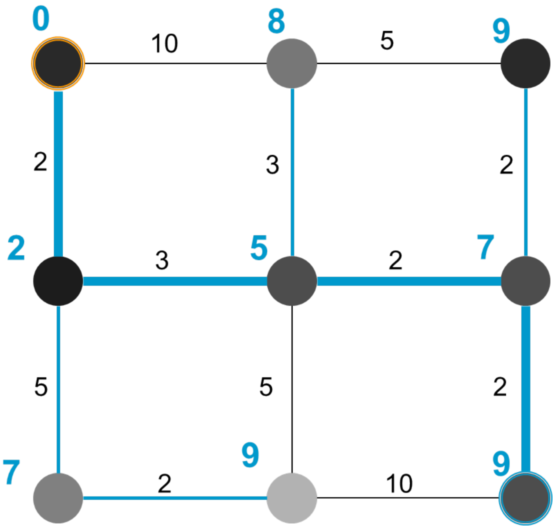


$$\Delta_{ab} = |Y(a) - Y(b)|$$

Dijkstra example



Result

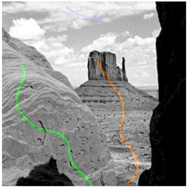


Comparison 1

[Yatziv & Sapiro]

[Levin *et al.*]

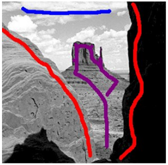
Comparison 2



[Yatziv & Sapiro]

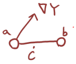
[Levin *et al.*]

Another application: segmentation



Numerical issues

- Dijkstra's algorithm has a bias when measuring distances on a grid



$$|\nabla Y \cdot \dot{c}| \cong |Y_a - Y_b|$$

- How to compute the gradients more accurately?
- How to achieve sub-grid accuracy?
- Level-set methods

Conclusion

Scribbles

- Interactive colorization tool

Fast blending

- Geodesic distance
- Dijkstra's algorithm