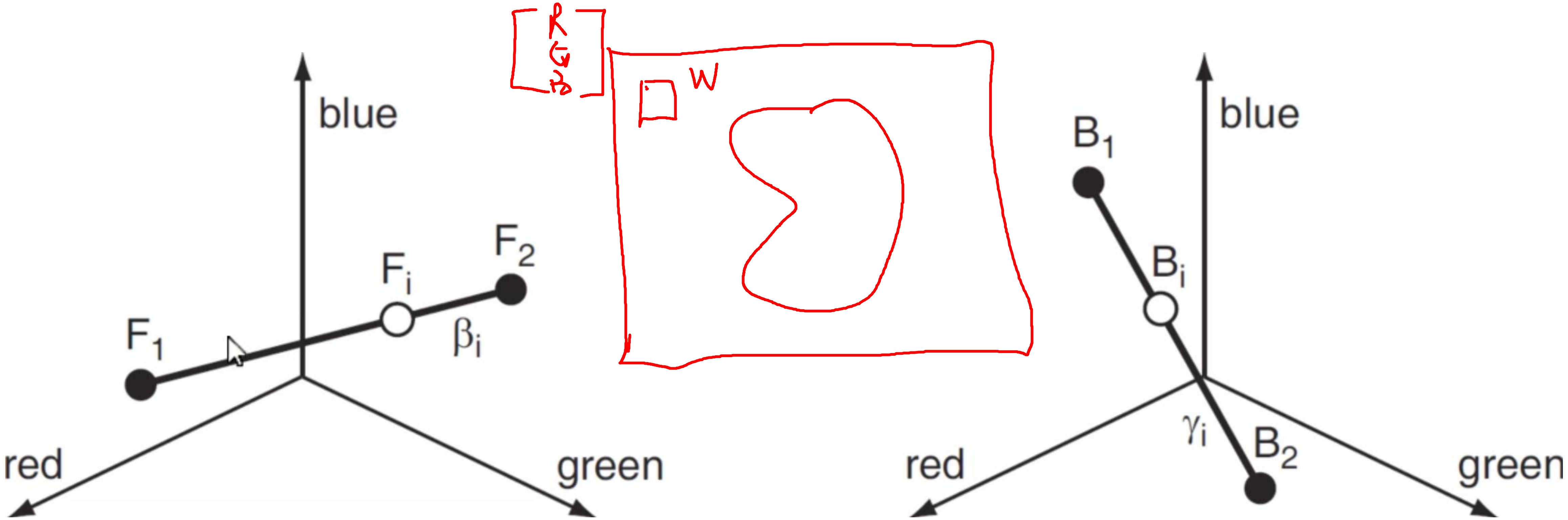
Lecture 20-2 Closed-form matting

Yuyao Zhang, Xiran Cai PhD
zhangyy8@shanghaitech.edu.cn caixr@shanghaitech.edu.cn
SIST Building 2 302-F/302-C
Course piazza link：
piazza.com/shanghaitech.edu.cn/spring2021/cs270spring2021

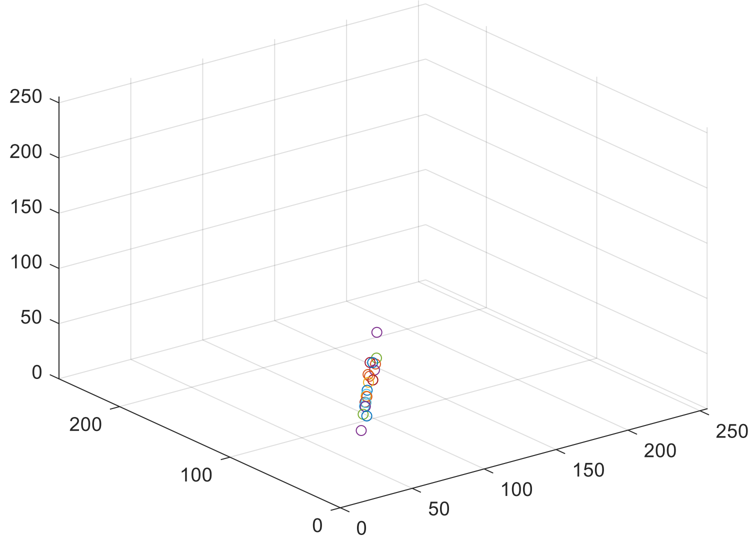
上海科技大学
ShanghaiTech University

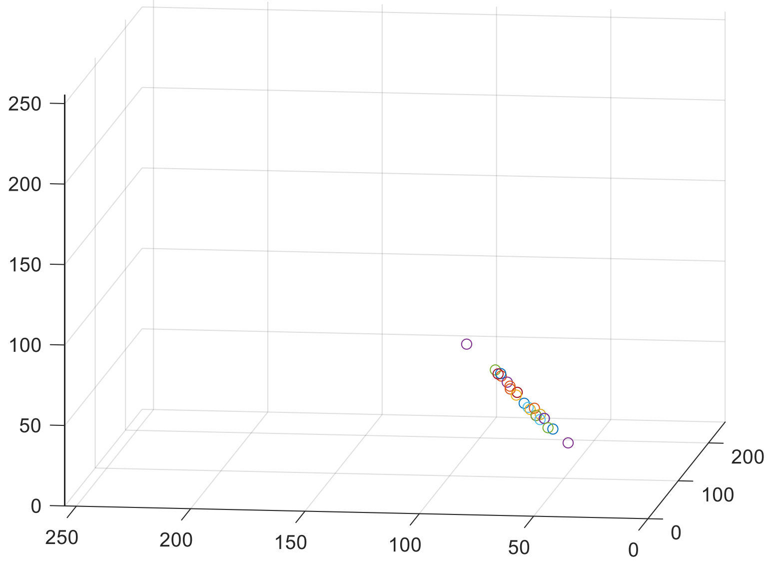
Closed-form matting

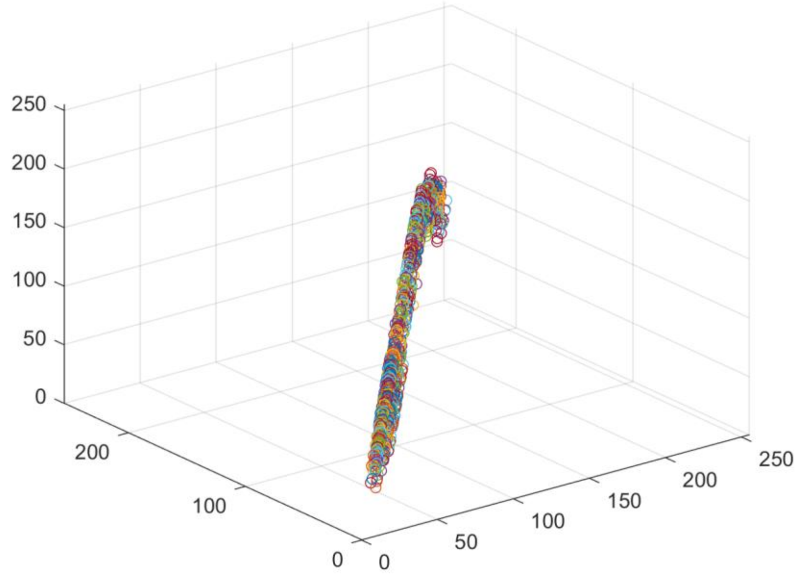
$$I = \alpha \cdot F + [1 - \alpha] \cdot B$$

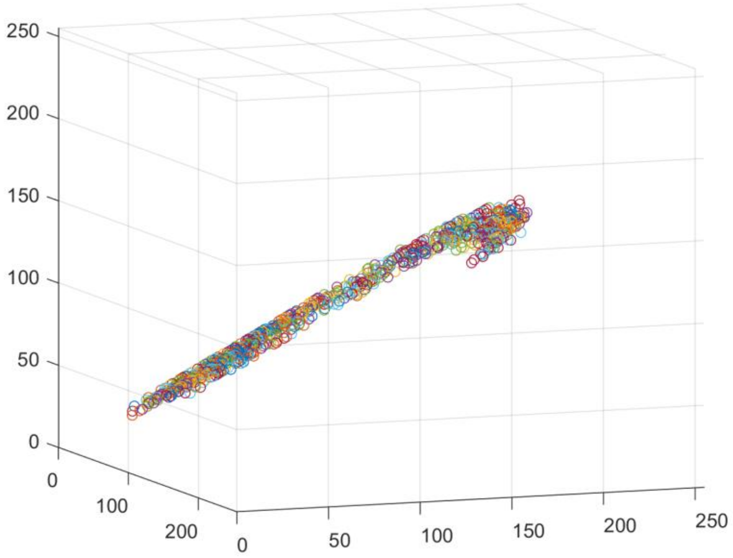


Color line assumption


250
200
150
100
50
0
200
100
0
0
50
100
150
200
250


250
200
150
100
50
0
250
200
150
100
50
0
200
100
0

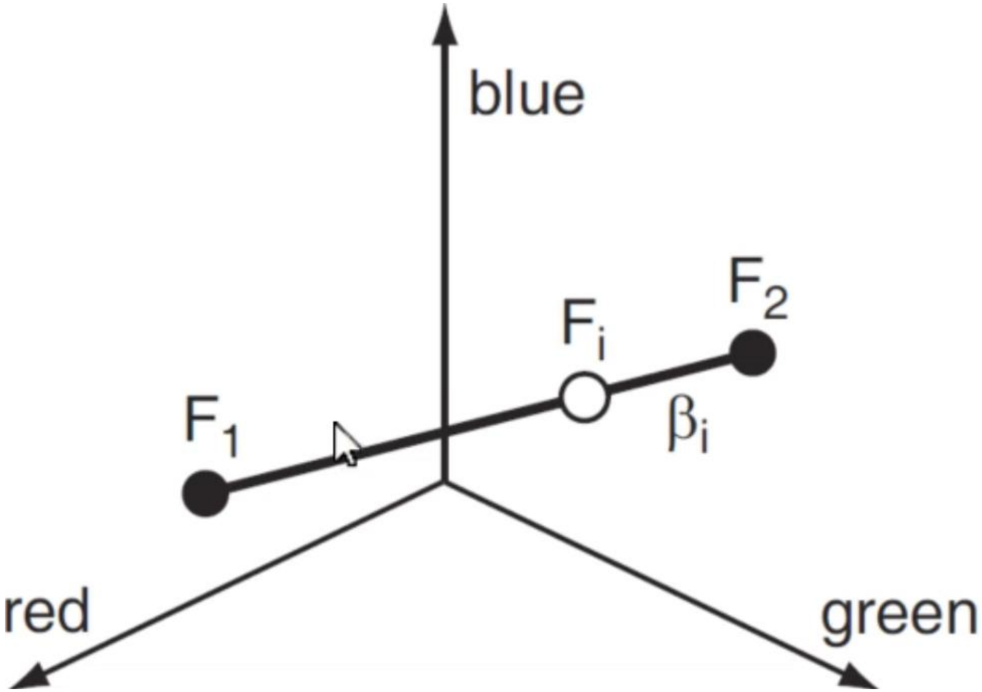

250
200
150
100
50
0
200
100
0
0
50
100
150
200
250


250
200
150
100
50
0
0
100
200
0
50
100
150
200
250

Closed-form matting

- Color line assumption:
- FG and BG colors in a small window lie on a straight line in RGB space.
- Line depends on which window we chose.

Closed-form matting



$$F_i = \beta_i F_1 + (1-\beta_i)\, F_2$$

$$B_i = \gamma_i B_1 + (1-\gamma_i)\, B_2$$

If color line assumption holds,

then the true matte (α) satisfies

$$\alpha_i = a^T I_i + \mathrm{b}$$

for all pixels in the window.

Prove for the affine transformation

We combine the matting equation and the color line assumption and get:

Cost function

$$J(\alpha_i, a_i\,, b_i) = \sum_{j=1}^{N} \sum_{i \in window j} (\alpha_i - (a_j^T I_i + b_j))^2$$

- i is the index of every pixel
- j is the index of every window
- For every pixel, we need to determine α_i
- For every window, we need to determine a_j& b_j

上海科技大学 ShanghaiTech University

Cost function

There exists a tight constrain between what happen on each pixel.

$$\arg\min J(\alpha_i, a_i\,, b_i) = \sum_{j=1}^{N} \sum_{i \in window\,j} (\alpha_i - (a_j^T I_i + b_j))^2$$

A relaxation for optimization

$$\sum_{j=1}^{N} \left\| G_j \begin{bmatrix} a_j \\ b_j \end{bmatrix} - \alpha_j \right\|^2$$

Minimizing each of these equations is a linear least square problem.

Suppose we knew the $\alpha's$ and the $a\&b$ that make $\left\| G_j \begin{bmatrix} a_j \\ b_j \end{bmatrix} - \alpha_j \right\|^2$ as small as possible are:

$$\begin{bmatrix} a_j \\ b_j \end{bmatrix}^* = (G_j^T G_j)^{-1} G_j^T \alpha_j$$

Then we have $a\&b$ are functions of α!

Matting Laplacian

- So the whole optimization is a function of only α.

$$\arg\min J(\alpha_i, a_i, b_i) = \sum_{j=1}^{N} \sum_{i \in window\, j} (\alpha_i - (a_j^T I_i + b_j))^2$$

$$\arg\min J(\alpha_j) = \sum_{j=1}^{N} \left\| G_j (G_j^T G_j) G_j^T \alpha_j - \alpha_j \right\|^2$$

$$= [\alpha_1 \quad \cdots \quad \alpha_N] L \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}$$

$$= \alpha^T L \alpha$$

Solution

$$\arg\min J(\alpha) = \alpha^T L \alpha$$

$$\frac{d}{d\alpha} = 2L\alpha = 0$$

$$L\alpha = 0$$

- Null vectors of L solves matting equation.
- Bad news: many 0-eigenvectors
- To constrain the matte, we need user input (scribbles).
- i.e. some pixels are forced to have $\alpha = 0$ for BG and $\alpha = 1$ for FG.

Solution

- So we actually solve:

$$\arg\min \alpha^T L\alpha + \lambda\left(\sum_{i\epsilon FG}(1-\alpha_i) + \sum_{i\epsilon BG}\alpha_i\ \right)^2$$

- We seek $\alpha's$ eigenvectors of L with eigenvalue 0 (null vectors).
- There are many such eigenvectors. And the matte we look for is a linear combination of these eigenvectors since:

if $Lv = 0$, and $w = \beta_1 v_1 + \beta_2 v_2 + \cdots + \beta_k v_k$

Then $Lw = 0$

Solution

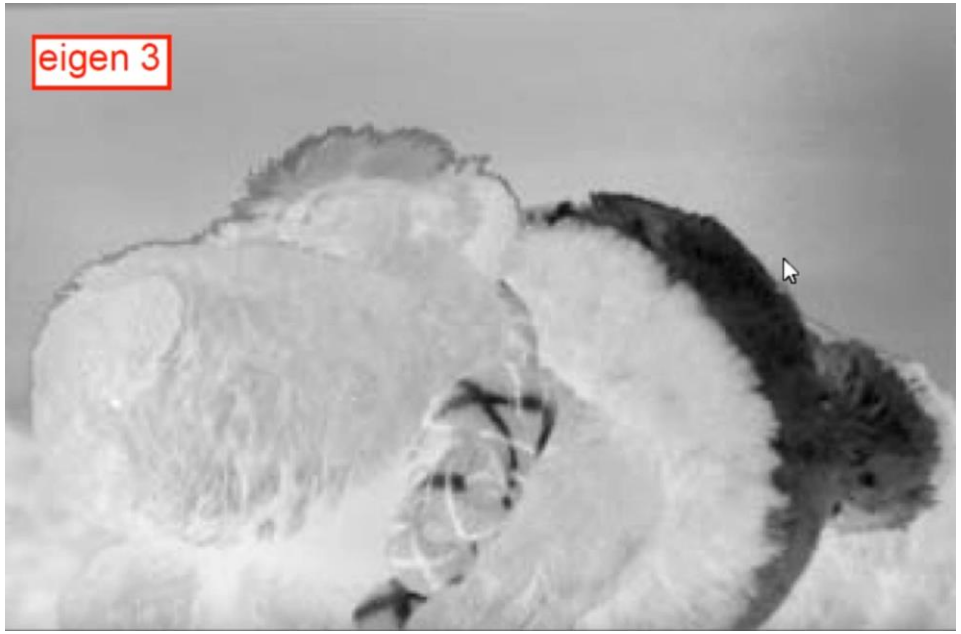
- Idea: we have 100 grayscale eigenvectors, we can combine them to build the binary matte.
- This is called "spectral matting".
- Cost function:

$$\min J(\alpha) = \min \sum_{i,k} \left|1 - \alpha_i^k\right|^\gamma + \left|\alpha_i^k\right|^\gamma$$

$$\alpha^k = E\beta^k$$

上海科技大学 ShanghaiTech University


eigen 1


eigen 3


3

Other application

$$I = Jt + (1 - t)A$$

上海科技大学 ShanghaiTech University

Take home message

- Bayesian image matting
- Closed-form image matting
- http://people.csail.mit.edu/alevin/papers/Matting-Levin-Lischinski-Weiss-PAMI-o.pdf
- https://arxiv.org/pdf/2004.00626v2.pdf
- https://grail.cs.washington.edu/projects/background-matting/