Image Matting – Review of Anat Levin et al.'s Algorithms

Jan Kotera, 2013
Source

- A Closed-Form Solution to Natural Image Matting (Levin et al., PAMI 2008)
- Spectral Matting (Levin et al., PAMI 2008)
Image matting – model and objective

= soft background-foreground separation

\[ I_i = \alpha_i F_i + (1 - \alpha_i) B_i, \quad \alpha_i \in [0, 1], \quad \forall i \]
Image matting – model and objective

User input

- [Image of user input 1]
- [Image of user input 2]

or

- [Image of user input 3]
- [Image of user input 4]
Algorithm 1 – model (grayscale)

Compositing equation: \[ I_i = \alpha_i F_i + (1 - \alpha_i) B_i, \]

Constant color assumption:
\[ \alpha_i \approx a I_i + b, \quad \forall i \in w, \]
\[ (a = 1/(F - B), \quad b = -B/(F - B)) \]

Cost function (model term):
\[ J(\alpha, a, b) = \sum_{j \in I} \left( \sum_{i \in w_j} (\alpha_i - a_j I_i - b_j)^2 + c a_j^2 \right), \]
Algorithm 1 – model (grayscale)

Simplification:

**Theorem 1.** Define $J(\alpha)$ as

$$J(\alpha) = \min_{a,b} J(\alpha, a, b).$$

Then,

$$J(\alpha) = \alpha^T L \alpha,$$

where $L$ is an $N \times N$ matrix, whose $(i, j)$th entry is

$$\sum_{k \mid (i,j) \in w_k} \left( \delta_{ij} - \frac{1}{|w_k|} \left( 1 + \frac{1}{|w_k| + \sigma_k^2} (I_i - \mu_k)(I_j - \mu_k) \right) \right).$$
Algorithm 1 – model (grayscale)

Proof:

\[
J(\alpha, a, b) = \sum_k \left\| G_k \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \bar{\alpha}_k \right\|^2,
\]

\[
(a_k^*, b_k^*) = (G_k^T G_k)^{-1} G_k^T \bar{\alpha}_k.
\]

\[
J(\alpha) = \sum_k \bar{\alpha}_k^T \bar{G}_k^T \bar{G}_k \bar{\alpha}_k
\]

\[
\bar{G}_k = I - G_k (G_k^T \bar{G}_k)^{-1} G_k^T.
\]
Algorithm 1 – model (color)

“Color line model”:

\[ F_i = \beta_i^F F_1 + (1 - \beta_i^F) F_2 \]

\[ B_i = \beta_i^B B_1 + (1 - \beta_i^B) B_2. \]

In each neighborhood \( w \).

Then

\[ \alpha_i = \sum_c a^c I^c_i + b, \quad \forall i \in w. \]
Algorithm 1 – model (color)

Model term:

\[ J(\alpha, a, b) = \sum_{j \in I} \left( \sum_{i \in w_j} \left( \alpha_i - \sum_c a_j^c I_i^c - b_j \right)^2 + \epsilon \sum_c a_j^c \right) \]

Simplification:

\[ J(\alpha) = \alpha^T L \alpha. \]
Algorithm 1 – full cost function

Including user input:

$$\alpha = \text{argmin} \; \alpha^T L \alpha + \lambda (\alpha^T - b_S^T) D_S (\alpha - b_S),$$

Solution:

$$(L + \lambda D_S) \alpha = \lambda b_S.$$
Algorithm 1 – further user input

Two other “brush types”:

• Color-key brush
• Constant-color brush
Algorithm 1 – optimization

- Direct linear solver (small images)
- Multiscale (downsample, upsample $\alpha$, solve only for “undecided” pixels)
- Multigrid solver (fast, worse quality)
Algorithm 1 – image separation

F and B are recovered by solving

\[
\min \sum_{i \in I} \sum_c (\alpha_i F_i^c + (1 - \alpha_i) B_i^c - I_i^c)^2 \\
+ |\alpha_{ix}| \left( (F_{ix}^c)^2 + (B_{ix}^c)^2 \right) + |\alpha_{iy}| \left( (F_{iy}^c)^2 + (B_{iy}^c)^2 \right),
\]
Algorithm 1 – results
Algorithm 1 – results
Algorithm 1 – results
Algorithm 2 - introduction

(Normalized) graph cuts for segmentation:

\[ A_{i,j} = \text{similarity}(i, j) \]

\[ D_{i,i} = \sum_j A_{i,j} \]

\[ (D - A)y = \lambda y \]

Simple case – one connected component:

\[ m_i^C = \begin{cases} 1 & i \in C' \\ 0 & i \notin C' \end{cases} \]

= 0-eigenvector of \( L = (D - A) \)
Algorithm 2 – introduction

Generalized compositing equation:

\[ I_i = \sum_{k=1}^{K} \alpha_i^k F_i^k. \quad \sum_k \alpha_i^k = 1 \quad \forall k \]

Matting Laplacian:

\[ J(\alpha) = \alpha^T L \alpha. \]

where

\[ L = \sum_q A_q, \]

\[ A_q(i, j) = \begin{cases} 
\delta_{ij} - \frac{1}{|w_q|} \\
\left(1 + (I_i - \mu_q)^T \left(\Sigma_q + \frac{\sigma_q}{|w_q|} I_{3 \times 3}\right)^{-1} (I_j - \mu_q)\right) \\
0 
\end{cases} \quad (i, j) \in w_q 
\text{otherwise.} \]
Algorithm 2 – Laplacian nullspace

General idea: matting components are 0-eigenvectors of $L$, if every image window $w$ satisfies the following model (one of the conditions)

1. A single component $\alpha^k$ is active within $w$.
2. Two components $\alpha^{k_1}, \alpha^{k_2}$ are active within $w$ and the colors of the corresponding layers $F^{k_1}, F^{k_2}$ within $w$ lie on two different lines in RGB space.
3. Three components $\alpha^{k_1}, \alpha^{k_2}$ and $\alpha^{k_3}$ are active within $w$, each layer $F^{k_1}, F^{k_2}, F^{k_3}$ has a constant color within $w$, and the three colors are linearly independent.
Algorithm 2 – computation

Matting components are linear combination of the smallest eigenvectors:

\[ E = [e^1, \ldots, e^K] \]

\[ \alpha^k = E y^k \]

For some \( y^k \)

“Final” model:

\[
\min \sum_{i,k} |\alpha_i^k|^\gamma + |1 - \alpha_i^k|^\gamma, \quad \text{where } \alpha^k = E y^k
\]

subject to \( \sum_k \alpha_i^k = 1. \)

- Solved using Newton’s method (series of second-order approximations)

- Initialized by \( \alpha^k = EE^T m^{C^k} \), where \( m^{C^k} \) are results of pixel k-means clustering
Algorithm 2 – smallest eigenvectors
Algorithm 2 – matting components
Algorithm 2 – combining components

Final matting = combination of “foreground” components:

$$\alpha = \sum_{k} b^k \alpha^k.$$  

Matting quality score:

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

Precomputed as:

$$J(\alpha) = b^T \Phi b, \quad \Phi(k, l) = \alpha^k L \alpha^l.$$
Algorithm 2 – unsupervised matting
Algorithm 2 – user input

Matting energy expressed in pairwise terms:

\[ J(\alpha) = E(b) = \sum_k E_k(b^k) + \sum_{k,l} E_{k,l}(b^k - b^l)^2, \]

where

\[ E_{k,l} = \max(0, -\phi_{k,l}). \]

and

\[ E_k(0) = \infty \quad \text{If k-th component is marked is foreground} \]
\[ E_k(1) = \infty \quad \text{If k-th component is marked is background} \]
\[ E_k(\cdot) = 0 \quad \text{Otherwise} \]

- Solved using graph min-cut method
- Constraints are specified as scribbles or component labeling
Algorithm 2 – results

Input   Our result   Levin et al. [14]
Implementations

• Algorithm 1
  http://people.csail.mit.edu/alevin/matting.tar.gz

• Algorithm 2
  http://www.vision.huji.ac.il/SpectralMatting/code.tar.gz