Published in Image Processing On Line on YYYY–MM–DD.
Submitted on YYYY–MM–DD, accepted on YYYY–MM–DD.
ISSN 2105–1232 © YYYY IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://www.ipol.im/

A Closed Form Solution to Natural Image Matting

Mahdi Ranjbar, Aissa Abdelaziz, Mohammad Ali Jauhar

École Normale Supérieure Paris-Saclay, France
{mahdi.ranjbar, aissa.abdelaziz, mohammad.jauhar}@ens-paris-saclay.fr

Abstract

Natural image matting refers to the process of estimating the foreground opacity, also known as the alpha matte, of an input image and extracting the foreground layer. It is a fundamental and challenging task in computer vision and finds applications in various fields related to image processing such as film and image editing, advertising, and medical diagnosis. Numerous approaches have been proposed to address this challenge incorporating both traditional and deep learning techniques. In this paper, we reproduce the paper "*A Closed Form Solution to Natural Image Matting*", also referred to as "*Closed Form Matting*". The proposed algorithm is based on local smoothness assumptions on foreground and background colors, the *color-line model*, and provides a simple closed-form solution that requires sparse scribble constraints instead of more cumbersome and hard-to-develop trimap constraints. We present extended derivations, a Python-based implementation, an online demo, along with comparison on contemporary methods and ablation study on the effects of hyper-parameters.

Source Code

The source codes and documentation for the algorithms presented in this paper are available here[1]. Usage instructions are included in the `README.md` file.

Keywords: alpha matte, closed form solution, image matting, interactive image editing

1 Introduction

Separating a foreground element of an image from its background for later compositing into a new scene is one of the most basic and common tasks in visual effects production. This problem is typically called matting. This technique is particularly suitable for categories that have very fine details, such as humans, animals, and plants. Image matting serves as an essential procedure for various downstream tasks, including promotional advertising in practical e-commerce platforms, image editing for daily life entertainment, background replacing for online video conferences, and metaverse applications such as virtual reality and game industries.

[1] https://github.com/reproductible-research/image-matting-with-a-closed-form-solution.git

Image matting is probably the oldest visual effects problem in filmmaking, and the search for a reliable automatic matting system has been ongoing since the early 1900s [21]. A major research milestone was a family of effective techniques for matting against a blue background developed in the Hollywood effects industry throughout the 1960s and 1970s. Such techniques have matured to the point that blue- and green-screen matting is involved in almost every mass-market TV show or movie.

On the other hand, putting an actor in front of a green screen to achieve an effect isn't always practical, and situations abound in which the foreground must be separated from the background in a natural image. The computer vision and computer graphics communities have proposed methods for semi-automatic matting with complex foregrounds and real-world backgrounds.

In this work, we reproduce the paper "*A Closed Form Solution to Natural Image Matting*" [20], also referred to as "*Closed Form Matting*". We begin by introducing matting terminology and the basic mathematical problem (Section 2.1). Then we give a complete derivation of closed-form natural image matting (Section 3) along with the algorithm (Section 3.5). Finally, we discuss ablation study, comparisons with current methods, and aspects of complexity and reproducibility (Section 4).

2 Background

2.1 Problem Statement

We assume that a color image I is represented by a 3D discrete array of pixels, where $I(x, y)$ is a vector of (red, green, blue) values, usually in the range [0,1] or [0,255]. The matting problem is to separate a given color image I into a foreground image F and a background image B. Our fundamental assumption is that the three images are related by the matting equation:

$$I(x, y) = \alpha(x, y)F(x, y) + (1 - \alpha(x, y))B(x, y), \tag{1}$$

where $\alpha(x, y)$ is a number in $[0, 1]$. That is, the color at (x, y) in I is a mix between the colors at the same position in F and B, where $\alpha(x, y)$ specifies the relative proportion of foreground versus background. If $\alpha(x, y)$ is close to 0 , the pixel gets almost all of its color from the background, while if $\alpha(x, y)$ is close to 1 , the pixel gets almost all of its color from the foreground. Figure 1 illustrates the idea. We abbreviate Equation (1) as,

$$I = \alpha F + (1 - \alpha)B, \tag{2}$$

with the understanding that all the variables depend on the pixel location (x, y). Since α is a function of (x, y), we can think of it as a grey-scale image, which is often called a matte or alpha matte. Therefore, in the matting problem, we are given the image I and want to obtain the images F , B, and α.

Fractional Value of Alpha At first, it may seem like $\alpha(x, y)$ should always be either 0 (that is, the pixel is entirely background) or 1 (that is, the pixel is entirely foreground). However, this isn't the case for real images, especially around the edges of foreground objects. The main reason is that the color of a pixel in a digital image comes from the total light intensity falling on a finite area of a sensor; that is, each pixel contains contributions from many real- world optical rays. In lower resolution images, it's likely that some scene elements project to regions smaller than a pixel on the image sensor. Therefore, the sensor area receives some light rays from the foreground object and some from the background. Even high resolution digital images (i.e., ones in which a pixel corresponds to a very small sensor area) contain fractional combinations of foreground and background in regions

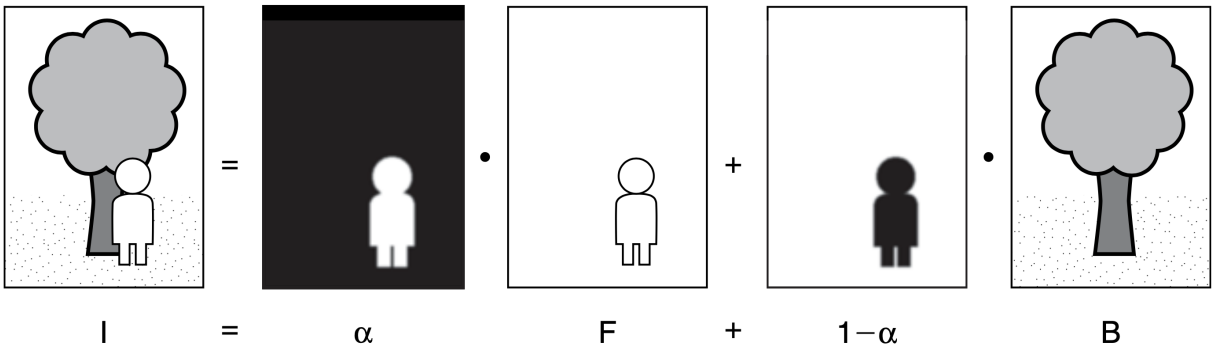


Figure 1: An illustration of the matting equation $I = \alpha F + (1 - \alpha)B$. When α is 0, the image pixel color comes from the background, and when α is 1, the image pixel color comes from the foreground.

like wisps of hair. Fractional values of α are also generated by motion of the camera or foreground object, focal blur induced by the camera aperture, or transparency or translucency in parts of the foreground. Thus, every image, no matter how high resolution, has fractional alpha values.

Understanding Matting with Image Segmentation Matting is closely related to a computer vision problem called image segmentation, but they are not the same. In segmentation, the goal is to separate an image into hard-edged pieces that snap together to reconstitute the whole, which is the same as creating a binary (0′s and 1′s) alpha matte for each object. On the other hand, for matting we expect that the foreground "piece" should have soft edges, so that pixels at the boundary contain a combination of foreground and background colors. As illustrated in Figure 2, clipping a hard-segmented object out of one image and placing it in another generally results in visual artifacts that would be unacceptable in a production environment, while a continuous-valued alpha matte produces a much better result.

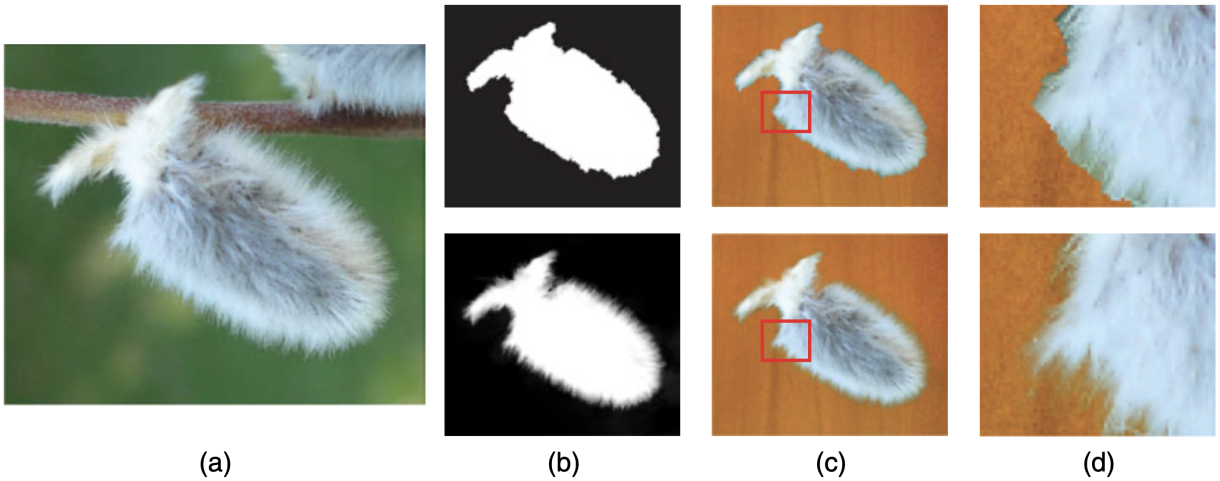


Figure 2: Image segmentation is not the same as image matting. (a) An original image, in which the foreground object has fuzzy boundaries. (b) (top) binary and (bottom) continuous alpha mattes for the foreground object. (c) Composites of the foreground onto a different background using the mattes. The hard-segmented result looks bad due to incorrect pixel mixing at the soft edges of the object, while using the continuous alpha matte results in an image with fewer visual artifacts. (d) Details of the composites in (c).

Foreground-Background Ambiguity in Matting Unfortunately, the matting problem for a given image can't be uniquely solved, since there are many possible foreground/background explanations for the observed colors. We can see this from Equation (2) directly, since it represents three equations in seven unknowns at each pixel (the RGB values of F and B as well as the mixing proportion α). This is a severely underconstrained problem, and we need to supply a matting algorithm with additional assumptions or guides in order to recover mattes that agree with human perception about how a scene should be separated. For example, the assumption that the background is known (e.g., it is a constant blue or green), removes some of the ambiguity. However, we consider a situation in which the background is complex and unknown and there is little external information other than a few guides specified by the user.

In modern matting algorithms, these additional guides frequently take one of two forms. The first is a trimap, defined as a coarse segmentation of the input image into regions that are definitely foreground (F), definitely background (B), or unknown (U). This segmentation can be visualized as an image with white foreground, black background, and gray unknown regions (Figure 3b). An alternative is a set of scribbles, which can be quickly sketched by a user to specify pixels that are definitely foreground and definitely background (Figure 3c). Scribbles are generally easier for a user to create, since every pixel of the original image doesn't need to be explicitly labeled. On the other hand, the matting algorithm must determine α for a much larger number of pixels.

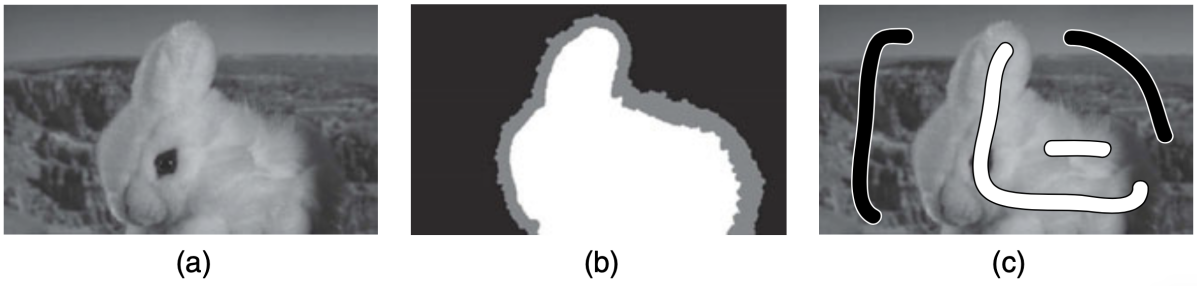


Figure 3: An example of natural image, its user-drawn trimap, and user-drawn scribbles. (a) The original image. (b) Trimap, in which black pixels represent certain background, white pixels represent certain foreground, and gray pixels represent the unknown region for which fractional α values need to be estimated. (c) Scribbles, in which black scribbles denote background pixels, and white scribbles denote foreground regions. α must be estimated for the rest of the image pixels.

2.2 Related Works

2.2.1 Traditional Methods

Traditional methods use RGB image as inputs along with additional inputs such as trimaps. Based on their usage of the additional inputs, they could be divided into two categories, *sampling-based* methods and *affinity-based* methods. Sampling-based methods assume that the true α could be inferred by analyzing nearby known pixels, i.e., via a model built using the colour features and additional low-level pixel features. On the other hand, affinity-based methods, such as *closed form matting*, infer the alpha of the ambiguous regions by propagating from the alpha values of the known regions by calculating similarities in spatial and colour features[2].

Bayesian Matting[5] employ sets of spatially varying Gaussians to model the foreground and background and then implements a maximum-likelihood criterion to find the optimal alpha. He et al.[9] propose a global sampling method and models it as a correspondence problem to counter the increase in computational complexity over local sampling methods. Among more recent method, Feng et al.[7] employ K-means clustering to cluster foreground and background pixels in a spatially

motivated bi-level method and solve for α using sparse coding techniques. *KNN Matting*[3] employs k-nearest neighbors to incorporate non-locality into the method and derives a closed-form solution. Grady et al.[8] proposed a random walk method that estimates the alpha values in the unknown region as the probability that a random walker leaving those pixels first reaches a known foreground pixel before striking a background pixel. Among more recent works, Aksoy et al.[1] designs a color-mixture pixel-to-pixel information model to propagate opacity information from known to unknown regions.

2.2.2 Deep Learning Methods

Deep learning has dramatically improved the state-of-the-arts in various learning problems in vision, speech, and language. It have also been successfully applied to natural image matting and currently represents the state-of-the-art. In terms of the architecture employed, earliest attempts were primarily convolutional neural networks (CNN). Attentions soon became popular for capturing global context. Recent methods have also used GANs, transformers and diffusion networks. Earliest approaches employed deep learning as an intermediary step in the matting pipeline. Cho et al. [4] employed a CNN to refine the α obtained from *closed form matting* and *KNN matting*. More recent methods could be categorized as a) *trimap-based*[26][13][14][17][15], and b) *trimap-free/end-to-end method*[12][4][25]. More recently, however, interactive methods have also been proposed[25].

Deep Image Matting[26] uses two separate convolutional neural networks. The first one takes the input image and the associated trimap to estimate the alpha. The second network is a smaller network intended for refining the alpha produced by the first network. *GCA*[13] employs a convolutional neural network in the form of UNet[22] along with guided cross attention (GCA) modules to better capture the high level global opacity information based on the learned low level affinity derived from the UNet feature maps and thus produce improved alpha matte. *LFPNet*[14] is a CNN based encoder-decoder network with *propagating* and *matting* modules. The *propagating module* learns the context alpha matte and associated feature map which is then added to the *matting module* to provide long range context. *AIMNet*[12] is an end-to-end method that uses a modified ResNet34 backbone as encoder and two decoder networks, one to capture high-level semantic context (in lieu of traditional trimap inputs), and other to guide matte production. *TIMI-Net*[15] fully mines and integrates the complementary global information from RGB image and trimap. The method utilised two encoder networks, ResNet18 and ResNet34, for encoding the trimap and RGB image, respectively, and a small upsamlpling CNN decoder. *MatteFormer*[17] is the first transformer-based method which tokenizes the trimap to create global priors. The network itself is an encoder-decoder network with skip connections between different parts of the encoder and decoder network. The encoder consists of blocks derived from swin transformers and termed as *Prior-Attentive Swin Transformer*. Wei et al. [25] employs a UNet based network that does away with trimaps and only requires sparse user input to produce results comparable with the state-of-the-art. The network contains uncertainty estimation modules that is helped by the user interaction inputs and guides the matte refinement.

3 Derivation

Computer Vision for Visual Effects (CVVE) [19] provides extensive analysis of matting as well as compositing theory, which makes this book one of the most detailed reference materials in matting field. This derivation is founded upon the principles outlined in the aforementioned reference.

3.1 The Color Line Assumption

It has been observed that in many natural images, the foreground and background distributions look similar to lines or skinny cigar shapes [16]. Hence, it is assumed that within a small window w_j around each pixel j, the sets of foreground and background intensities each lie on a straight line in RGB space. That is, for each pixel i in w_j,

$$\begin{aligned} F_i &= \beta_i F_1 + (1-\beta_i)\, F_2 \\ B_i &= \gamma_i B_1 + (1-\gamma_i)\, B_2 \end{aligned} \tag{3}$$

Here, F_1 and F_2 are two points on the line of foreground colors, and β_i represents the fraction of the way a given foreground color F_i is between these two points. The same idea applies to the background colors.

Under the color line assumption, the α value for every pixel in the window is simply related to the intensity by

$$\alpha_i = a^\top I_i + b \tag{4}$$

where a is a 3×1 vector, b is a scalar, and the same a and b apply to every pixel in the window. That is, we can compute α for each pixel in the window as a linear combination of the RGB values at that pixel, plus an offset.

Proof. First we plug Equation (3) into the matting equation (2) to obtain:

$$I_i = \alpha_i \left(\beta_i F_1 + (1-\beta_i)\, F_2\right) + (1-\alpha_i)\left(\gamma_i B_1 + (1-\gamma_i)\, B_2\right) \tag{5}$$

Expanding it, we get:

$$I_i = \alpha_i\beta_i F_1 + \alpha_i F_2 - \alpha_i\beta_i F_2 + \gamma_i B_1 - \alpha_i\gamma_i B_1 + B_2 - \alpha_i B_2 - \gamma_i B_2 + \alpha_i\gamma_i B_2 \tag{6}$$

Rearranging the terms, we get a 3×3 system of linear equations as follows:

$$\begin{bmatrix} F_2 - B_2 & F_1 - F_2 & B_1 - B_2 \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_i\beta_i \\ (1-\alpha_i)\,\gamma_i \end{bmatrix} = I_i - B_2 \tag{7}$$

Note that the 3×3 matrix on the left-hand side only depends on $F1$, $F2$, $B1$, and $B2$, which we assumed were constant in the window. We multiply by the inverse of this matrix on both sides, and denote the rows of this inverse by r's:

$$\begin{aligned} \begin{bmatrix} \alpha_i \\ \alpha_i\beta_i \\ (1-\alpha_i)\,\gamma_i \end{bmatrix} &= \begin{bmatrix} F_2 - B_2 & F_1 - F_2 & B_1 - B_2 \end{bmatrix}^{-1} (I_i - B_2) \\ &= \begin{bmatrix} r_1^\top \\ r_2^\top \\ r_3^\top \end{bmatrix} (I_i - B_2) \end{aligned} \tag{8}$$

Taking just the first element of the vector on both sides, we see that,

$$\alpha_i = r_1^\top I_i - r_1^\top B_2 \tag{9}$$

which corresponds to Equation (4) with $a = r_1$ and $b = -r_1^\top B_2$.

3.2 The Matting Laplacian

The assumption that the α values and colors inside a window are related by Equation (4) leads to a natural cost function for the matting problem:

$$J\left(\{\alpha_i, a_i, b_i, i = 1, \ldots, N\}\right) = \sum_{j=1}^{N} \sum_{i \in w_j} \left(\alpha_i - \left(a_j^\top I_i + b_j\right)\right)^2 \tag{10}$$

This cost function expresses the total error of the linearity assumption in Equation (4) over each window. We want to minimize J to find α_i at each pixel as well as the coefficients a_i and b_i for every window w_i around pixel i. For brevity, we'll write the left-hand side as $J(\alpha, a, b)$, where α is an $N \times 1$ vector that collects all the α values in the image, and a and b represent the collections of affine coefficients for each window. Since the windows between adjacent pixels overlap, the α estimates at each pixel are not independent. We also add a regularization term to Equation (10):

$$J(\alpha, a, b) = \sum_{j=1}^{N} \left(\sum_{i \in w_j} \left(\alpha_i - \left(a_j^\top I_i + b_j\right)\right)^2 \right) + \varepsilon \left\| a_j \right\|_2^2 \tag{11}$$

This term prevents a_j vector to become too large and acts to bias the mattes toward being constant, since if $a = 0, \alpha_i = b_j$ within the whole window w_j.

This formulation, however, still doesn't help us solve the matting problem, since we still have many more equations than unknowns (i.e., the five values of α, a, and b at each pixel). Hence, we now try reducing the number of unknowns to exactly the number of pixels.

First, we rearrange Equation (11) as a matrix equation:

$$J(\alpha, a, b) = \sum_{j=1}^{N} \left\| \begin{bmatrix} I_1^j & 1 \\ \vdots & \vdots \\ I_W^{j\top} & 1 \\ \sqrt{\varepsilon} I_{3\times 3} & 0 \end{bmatrix} \begin{bmatrix} a_j \\ b_j \end{bmatrix} - \begin{bmatrix} \alpha_1^j \\ \vdots \\ \alpha_W^j \\ 0_{3\times 1} \end{bmatrix} \right\|^2 \tag{12}$$

where W is the number of pixels in the window and $\left\{I_1^j, \ldots, I_W^j\right\}$ and $\left\{\alpha_1^j, \ldots, \alpha_W^j\right\}$ represent the ordered list of image colors and α values inside window j. More compactly, we can write Equation (12) as

$$J(\alpha, a, b) = \sum_{j=1}^{N} \left\| G_j \begin{bmatrix} a_j \\ b_j \end{bmatrix} - \bar{\alpha}_j \right\|^2 \tag{13}$$

where $\bar{\alpha}_j$ is a $(W + 3) \times 1$ vector containing the α's in window j followed by three 0's. If we suppose that the matte is known, then this vector is constant and we can minimize Equation (12) for the individual $\{a_j, b_j\}$ as a standard linear system:

$$\begin{bmatrix} a_j^* \\ b_j^* \end{bmatrix} = \left(G_j^\top G_j\right)^{-1} G_j^\top \bar{\alpha}_j \tag{14}$$

That is, the optimal a and b in each window for a given matte α are linear functions of the α values. This means we can substitute Equation (14) into Equation (12) to get

$$J(\alpha) = \min_{a,b} J(\alpha, a, b) \tag{15}$$

$$= \min_{\mathbf{a},\mathbf{b}} \sum_{j=1}^{N} \left\| G_j \begin{bmatrix} a_j \\ b_j \end{bmatrix} - \bar{\alpha}_j \right\|^2 \tag{16}$$

$$= \sum_{j=1}^{N} \left\| G_j \begin{bmatrix} a_j^* \\ b_j^* \end{bmatrix} - \bar{\alpha}_j \right\|^2 \tag{17}$$

$$= \sum_{j=1}^{N} \left\| G_j \left(G_j^\top G_j\right)^{-1} G_j^\top \bar{\alpha}_j - \bar{\alpha}_j \right\|^2 \tag{18}$$

$$= \sum_{j=1}^{N} \bar{\alpha}_j^\top \left[I_{(W+3)\times(W+3)} - G_j \left(G_j^\top G_j\right)^{-1} G_j^\top \right] \bar{\alpha}_j \tag{19}$$

$$= \alpha^\top L \alpha \tag{20}$$

In the last equation, we've collected all of the equations for the windows into a single matrix equation for the $N \times 1$ vector α. The $N \times N$ matrix L is called the matting Laplacian.

Simplifying Equation (19), we get the elements of the matting Laplacian as:

$$L(i,j) = \sum_{k|(i,j)\in w_k} \left[\delta_{ij} - \frac{1}{W} \left(1 + (I_i - \mu_k)^\top \left(\Sigma_k + \frac{\varepsilon}{W} I_{3\times 3} \right)^{-1} (I_j - \mu_k) \right) \right] \tag{21}$$

where μ_k and Σ_k are the mean and covariance matrix of the colors in window k and δ_{ij} is the Kronecker delta. Frequently, the windows are taken to be 3×3, so $W = 9$. The notation $k \mid (i,j) \in w_k$ in Equation (21) means that we only sum over the windows k that contain both pixels i and j; depending on the configuration of the pixels, there could be from 0 to 6 windows in the sum.

Alternately, we can write

$$L(i,j) = \begin{cases} \sum_k A(i,k) & \text{if } i = j \\ -A(i,j) & \text{if } i \neq j \end{cases} \tag{22}$$

where

$$A(i,j) = \sum_{k|(i,j)\in w_k} \frac{1}{W} \left[1 + (I_i - \mu_k)^\top \left(\Sigma_k + \frac{\varepsilon}{W} \mathbf{I}_{3\times 3} \right)^{-1} (I_j - \mu_k) \right] \tag{23}$$

The matrix A specified by Equation (23) is sometimes also referred as the matting affinity.

To minimize Equation (20), we set the derivative to 0:

$$\begin{aligned} \min J(\alpha) &\implies \frac{\partial}{\partial \alpha} J(\alpha) = 0 \\ &\implies \frac{d}{d\alpha} = 2L\alpha = 0 \\ &\implies L_{(N\times N)} \alpha_{(N\times 1)} = 0_{(N\times 1)} \end{aligned} \tag{24}$$

That is, we simply need to find a vector in the nullspace of L.

3.3 Constraining the Matte

Thus far we haven't taken into account any user-supplied knowledge of where the matte values are known; without this knowledge, the solution is ambiguous. For example, it turns out that any constant α matte is in the nullspace of L. In fact, the dimension of the nullspace is large (e.g., each of the matrices in the sum of Equation (19) has nullspace of dimension four [23]). Therefore, we rely on user scribbles to denote known foreground and background pixels and constrain the solution. That is, the problem becomes:

$$\begin{array}{lll} \min & \alpha^\top L \alpha & \\ \text{s.t.} & \alpha_i = 1 & i \in \mathcal{F} \\ & \alpha_i = 0 & i \in \mathcal{B} \end{array} \tag{25}$$

Another way to phrase this is:

$$\min \alpha^\top L \alpha + \lambda \left(\alpha - \alpha_K\right)^\top D \left(\alpha - \alpha_K\right) \tag{26}$$

where α_K is an $N \times 1$ vector equal to 1 at known foreground pixels and 0 everywhere else, and D is a diagonal matrix whose diagonal elements are equal to 1 when a user has specified a $\mathcal{F}$ or $\mathcal{B}$ scribble at that pixel and 0 elsewhere. λ is set to be very large number (e.g., 100) so that the solution is forced to agree closely with the user's scribbles. Setting the derivative of Equation (26) to 0 results in the sparse linear system:

$$(L + \lambda D)\alpha = \lambda \alpha_K \tag{27}$$

3.4 Recovering F and B from α

After solving the linear system in Equation (27) we obtain α values but not estimates of F and B. One way to get these estimates is to treat α and I as constant in the matting equation and solve it for F and B. Since this problem is still underconstrained, literature [11] suggest incorporating the expectation that F and B vary smoothly (i.e., have small derivatives), especially in places where the matte has edges. The corresponding problem is:

$$\begin{aligned} \min_{F_i, B_i} \sum_{i=1}^{N} & \left\| I_i - \left(\alpha_i F_i + \left(1 - \alpha_i\right) B_i\right) \right\|^2 \\ & + \left|\nabla_x \alpha_i\right| \left(\left\|\nabla_x F_i\right\|^2 + \left\|\nabla_x B_i\right\|^2\right) + \left|\nabla_y \alpha_i\right| \left(\left\|\nabla_y F_i\right\|^2 + \left\|\nabla_y B_i\right\|^2\right) \end{aligned} \tag{28}$$

where the notation $\nabla_x I$ represents the gradient of image I in the x direction, which is a scalar for a grayscale image and a vector for a color image. Solving Equation (28) results in a sparse linear system instead of a problem solved independently at every pixel, since the gradients force interdependence between the pixels.

3.5 Pseudocode

Algorithm 1 Computing Alpha (α)

Input: Image $I_{W\times H\times 3}$, Image with Scribbles $S_{W\times H\times 3}$, Regularization factor ϵ, Window Radius k
Output: Alpha Matting $\alpha_{W\times H}$

Initialization:
Normalize I and S
$S^{'} \leftarrow I - S$
PRIOR $\leftarrow$ 1 for foreground, 0 for background, 0.5 for unknown ▷ From $S^{'}$
CONST_MAP $\leftarrow$ 1 for known pixels and 0 for unknowns
$\lambda \leftarrow 100$ ▷ Confidence matrix: [0,100] for trimap
$\alpha_k \leftarrow \lambda \times$ CONST_MAP ▷ Confidence conditioned on scribbles
$D \leftarrow$ CONST_MAP.diagonal ▷ Diagonal Matrix from CONST_MAP
Computation:
$L \leftarrow$ COMPUTE_LAPLACIAN$(I, \text{CONST_MAP}, \epsilon, k)$
$\alpha \leftarrow (L + \lambda D)\alpha = \lambda\alpha_K$ as in Eq. (27)

Algorithm 2 Computing Matting Laplacian

Input: Image I, Constant Map, ϵ, window size k
Output: Matting Laplacian $L_{W\times H}$

procedure COMPUTE_LAPLACIAN$(I, \text{CONST_MAP}, \epsilon, k)$
  Initialization:
  $neigh\_size \leftarrow (k \times 2 + 1)^2$ ▷ Neighborhood size
  $h, w, c \leftarrow I.shape$ ▷ height, width, channels
  Calculate pair relations between unknown pixels as $Pair\_Rel$
  Initialize row_inds of length Pair_Rel.length
  Initialize col_inds of length Pair_Rel.length
  Initialize value matrix Val_Mat to save values
  Computation:
  for $j = k$ **to** $(w - k)$ **do**
    for $i = k$ **to** $(h - k)$ **do**
      if $I[i, j]$ is known **then**
        Continue ▷ Has a value 1 in a constant map
      end if
      Set window W of size $(k \times 2 + 1)$ around pixel(i, j)
      $\mu \leftarrow$ mean of W
      $\sigma \leftarrow$ variance of W
      $Val\_Mat[i, j] \leftarrow \left[1 + (I_i - \mu)^\top \left(\sigma^2 + \frac{\epsilon}{\text{neigh\_size}}\mathbf{I}_{3\times 3}\right)^{-1} (I_j - \mu)\right]$ as given in Eq. (21)
      Append i to row_inds, j to col_inds
    end for
  end for
  Construct a sparse matrix A from values, row_inds, and col_inds, with shape $w \times h$
  Compute the sum of each row of matrix A as $A^{'}$
  $L \leftarrow A - A^{'}$
  Return L

4 Discussions

4.1 Implementation & Reproducibility

Implementation We implement the algorithm using Python in lieu of the original implementation in MATLAB because Python is open source and hence accessible and reproducible.

We used a sparse representation to reduce the size of matrices containing many zeros, enabling efficient memory usage and speeding up the calculation of alpha. Various libraries, in particular, SciPy[24] and packages were employed to compute the sparse matrices and solve the sparse linear system (Equation 24). The algorithm is detailed below in Algorithm 1.

Additionally, we integrated our Python implementation with the demonstration provided by the IPOL journal. This integration ensures accessibility and usability for a wider audience, enabling users to interact with the algorithm and explore its capabilities, such as distinguishing between the background and foreground by incorporating minimal information about the two regions through scribble input. Through this implementation process, our goal is to enhance our understanding of the algorithm and facilitate its usage in practical applications.

Computational Complexity The time and space complexity of *Closed Form Matting* is dependent on the type of solver employed. For a standard LU solver, as employed by SciPy's default UMFPACK[6], In the worst case, for a very dense matrix, for an input of size $N = (w \times h)$, where w and h are the width and height of the input image, respectively, the time complexity for solving such system would be of the order $O(N^{\frac{3}{2}})$. Hence, the worst case complexity of the algorithm would be $O(k^2N + N^{\frac{3}{2}})$, where k is the window size. Similarly, the space complexity for the an standard LU solver, the UMFPACK, would be $O(N^3)$. Hence, for *Closed Form Matting*, the worst case space complexity would be $O(cN + N^3)$ where $c = 3$ is the number of color channels. The worst case space complexity, thus resolves to $O(N^3)$.

Reproducibility Translating the method as mentioned in the paper is challenging and ambiguous on many details. The paper mentions window size as one of the hyperparameters. However, it does not specify the relation between neighborhoods and window size and therefore, forces using ad hoc formulae when computing matting Laplacian and thus makes it difficult for the implementation to work on all kinds of images. Furthermore, the authors do not mention how to handle the boundary pixels and whether to pad the image in some ways according to the window size or not. In spite of all these shortcomings, since the method solves simple sparse linear system, the availability of community vetted open source tools to do it makes the job easier and reproducible. However, there are trade-offs among different sparse solver with respect to time and space complexity. Hence, an standard UMFPACK solver is not suitable for using in the case of high-resolution image matting where the system quickly runs out of memory. Hence, providing implementations that work in wider types of inputs is challenging.

4.2 Experiments

We test the algorithm using different images in various scenarios and hyperparameters. We evaluate our implementation with images used in the original paper to compare the results with the original implementation(Table 1), as well as with other images to further assess the effectiveness and robustness of the algorithm. The results are presented in Table 2.

Table 1: Matting Examples of Images from the Original Paper

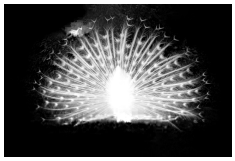
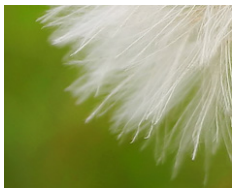
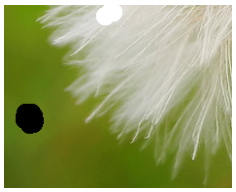
Original image	Scribbled image	Output alpha
		
		
		

Table 2: Matting Examples on New Images

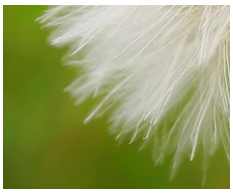
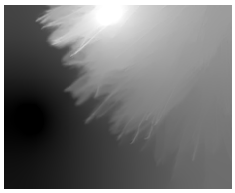
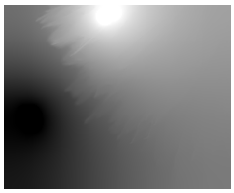
Original image	Scribbled image	Output alpha
		
		
		

4.2.1 Impact of Different Epsilon Values

As described in the original paper [20], ε is introduced as a form of regularization. We analyse the impact of changing values of ε ($\varepsilon = 10^{-2}$, $\varepsilon = 10^{-3}$, $\varepsilon = 10^{-4}$, $\varepsilon = 10^{-5}$) and show its impact on the accuracy of the matting Laplacian and the resulting alpha matte.

The reason behind the observed effect, Table 3, is that choosing higher values of ε affects the influence of the covariance matrix, which captures the relationship between the color values of neighboring pixels. Consequently, higher values of ε lead to greater smoothing, effectively reducing the influence of local color variations on the computation of pixel similarities.

Table 3: Image Matting with different ϵ values

Original image	ϵ=0.00001	ϵ=0.0001	ϵ=0.001	ϵ=0.01
				
				

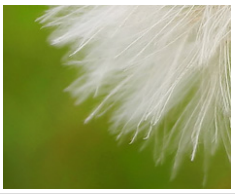
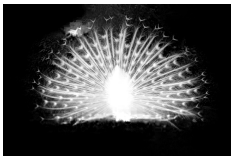
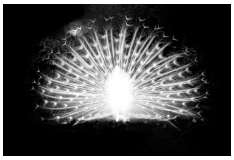
4.2.2 Impact of Different Window Sizes

Choosing the right window size for closed-form matting can be a tricky problem, depending on the resolution of the image and the fuzziness of the foreground object (which may not be the same in all parts of the image).

The advantage of using large windows is that many distant pixels are related to each other; however, space and time complexities increase significantly, making it less suitable for real-world implementation.

We test the algorithm with two window sizes: 3×3, where we take only one neighbor on all sides, and 9×9, where we take two neighbors from all sides around the pixel(See results in table 4). We noticed that, in all the images we tested, a window size of 3×3 was sufficient to solve the system and return a high-quality alpha matte.

Table 4: Result with different window sizes

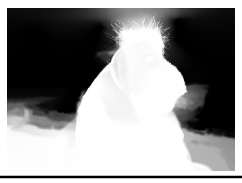
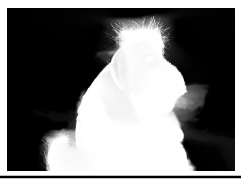
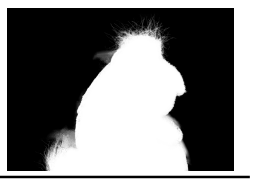
Original image	Window size = 3x3	Window size = 9x9
		
		
		

4.2.3 Impact of Varying Amount of User Input

The amount of user input should affect both the quality of the matte as well the processing time of the algorithm. This is because with an increase in useful constraints, the resulting matte is pushed closer towards the ground truth α. Furthermore, with an increase in the amount of scribbles, the Laplacian matrix gets more and more sparse, thus speeding up the computations.

This could be seen in Table (5), where we see that as we increase the amount of scribbles, the matte gets progressively better. And with trimap, it is the best.

Table 5: Image Matting with different amount of scribbles

Original image	Few scribbles	More scribbles	Most scribbles	trimap
				
				

4.3 Comparison With Current Methods

4.3.1 Benchmarks

- Alphamatting.com: One of the earliest benchmark [20] in image matting, it contains 8 test images of varying opacity. The ground truth alpha is hidden. Images and associated trimaps are available in both low and high-resolutions.

- Composition-1k: One of the standard benchmark [26], it is a large scale dataset created using composition. It contains images with objects on simple backgrounds extracted and composited onto new background images resulting into a dataset with 49300 training images and 1000 test images.

- Distinction-646: Another large scale dataset [18] with 59, 600 training images and 1000 test images, total 646 distinct foreground alpha mattes. Its primary contribution is to provide a larger scale benchmark.

- AIM-500: It is a benchmark [12] aimed at end-to-end methods that don't require trimap as part of their input. It contains 500 diverse natural images covering all types along with manually labeled alpha mattes, making it feasible to benchmark the generalization ability of AIM models.

4.3.2 Metrics

- Sum of Absolute Difference (SAD): This metric calculates the sum of the absolute difference of corresponding pixels in each pair of images. This serves as a similarity metric between two images and in the case of image matting, this serves as the ground truth matte and generated matte.

- Mean Squared Error (MSE): It is a similar metric to that of SAD, however, instead of being the sum of absolute difference, MSE is the average of the sum of squared difference values.

- Gradient (GRAD): Gradient is a metric that represents visual quality. It is the difference between the computed alpha α and ground truth alpha α^*. Mathematically, it is $\sum_i(\nabla\alpha - \nabla\alpha^*)^q$, where ∇ represents the gradient operator, and q is a integer hyperparameter (often 2). The gradient is obtained by convolving the alpha mattes with first-order Gaussian derivative filters with a variance.

- Connectivity (CONN): It is another perceptive metric. It defines the degree of connectedness based on the connectivity in binary threshold images computed from the grayscale alpha matte. Mathematicaly, connectivity error of an alpha matte α with its corresponding ground truth α^* is defined as $\sum_i(\phi(\alpha_i, \Omega) - (\phi(\alpha_i^*, \Omega))^p$ where ϕ measures the degree of connectivity for pixel i with transparency α_i to a source region Ω, and p is an integer hyperparameter.

4.3.3 Results

We compare the results of *Closed Form Matting* with current deep learning based methods. In Table (6), we compare the results of our method on the *Composition-1K* dataset and in Table (7), we compare the results on the *Alphametting.com* dataset. We see that the quality of matting has increased significantly across all metrics. We also observe that trimap-based methods have an edge over end-to-end methods, even in the deep learning setting.

Methods	SAD	MSE (10^{-3})	GRAD	CONN
Closed Form Matting	168.1	91	126.9	167.9
Deep Image Matting	50.9	14	31.0	50.8
GCA	35.2	9.1	16.9	32.5
TIMI-Net	29.1	6.0	11.5	25.4
MatteFormer	23.8	4.0	8.7	18.4
LPFNet	22.4	3.6	7.6	17.1
Wei et al.	7.9	3.5	4.6	6.7

Table 6: Comparison on the Composition-1K dataset. The lower the number for each metric, the better the method.

Methods	SAD	MSE (10^{-3})	GRAD	CONN
Closed Form Matting	50	49.5	48.9	21.5
Deep Image Matting	18.5	21.5	26.3	22.3
GCA	16.5	17.4	16.1	24.6
TIMI-Net	5.4	6.2	7.3	9.7
LPFNet	4.5	4.5	3.3	12.6

Table 7: Comparison on the Alphamatting.com dataset. The lower the number for each metric, the better the method.

4.4 Limitations

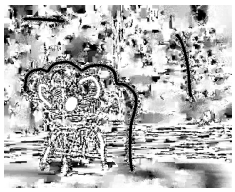
Singraju et al.[23], show that *Closed Form Matting* [10] does not always recover the ground truth alpha mattes due to several reasons, a few of which are outlined below. Several of the method's limitations arise from it being a local method which results into the method not capturing structural details that are more global in nature.

Violation of the color-line model Natural images don't often satisfy the color line model. For example, in the case of complex intensity variations, it is obvious that the color line model is too simple to explain the image intensities. In such scenarios, one would have to resort to data driven schemes for generating candidate foreground and background colors. Another violation of color lie model occurs in images with fine-grained textures, as shown in Figure (8). The issue arises due to the method being local and thus it fails to capture the global complexity of images. Furthermore, in the cases when the intensity variations are much simpler than the color line, such as being locally constant, the color line model provides an overfit.

Insufficient user interaction Recall that the system constructed in Equation (20) has a 4-dimensional null space for each local image patch considered by the algorithm. Hence, high level user interaction is required to resolve any ambiguities. One could potentially incorporate prior knowledge in order to bias the system to- wards certain family of solutions. [6] biases the mattes to be locally constant, which can prove to be unsatisfactory in practice. Furthermore, as seen in Figure (5), in the absence of enough user scribble, the quality of matte is sub-par. Hence, a method that iteratively takes user inputs to refine matte serves better.

Time and Memory Inefficiency The method suffers heavily with an increase in the input image size, window size, as mentioned in the Section (4.1). This is also observed in Table (4) computation

Table 8: Violation of color-line model due to, (up) texture in the image, and (down) complex intensity variations

Original image	Scribbled image	Output alpha
		
		

where the increase in window size made the algorithm very slow. Finally, solving dense matrices is still very challenging and since this method requires solving dense matrices, it quickly runs out of memory on most consumer computing machines, due to its cubic space complexity with respect to the input image dimensions.

5 Conclusions

Image matting is an important task in image editing and is inherently challenging due to being severely constrained. Traditional methods relied on carefully hand-crafted techniques based on local and global assumptions and tried to manage the trade-off between matte quality and compute efficiency. More recent methods employ deep learning and significantly out-perform any traditional methods. However, a theme common to both types of methods is the requirement of additional constraints and context in the form of scribbles or trimaps along with the input image, which severely limits the usability in situations where trimaps are tedious and hard to develop and scribbles are either difficult to get or are insufficient. We still don't have a truly end-to-end method with state-of-the-art performance.

In this work, we took an extensive look on *Closed Form Matting*. We demonstrated its workings, subjected it to an ablation study, compared it with current deep learning methods, and analysed the aspects of computational complexity and reproducibility. We visualized the limitations of *color-line* assumption and the issues inherent with seeking user interactions for matting. We showed how it is affected by the significant time and space complexity associated with solving large linear systems and how it makes the method unusable in practice. We hope our work adds to the efforts in making research reproducible and accessible.

References

[1] Yagiz Aksoy, Tunç Ozan Aydin, and Marc Pollefeys, *Designing effective inter-pixel information flow for natural image matting*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 228–236.

[2] Guowei Chen, Yi Liu, Jian Wang, Juncai Peng, Yuying Hao, Lutao Chu, Shiyu Tang, Zewu Wu, Zeyu Chen, Zhiliang Yu, Yuning Du, Qingqing Dang, Xiaoguang Hu, and Dianhai Yu, *Pp-matting: High-accuracy natural image matting*, 2022.

[3] Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang, *Knn matting*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35 (2013), pp. 2175–2188.

[4] Donghyeon Cho, Yu-Wing Tai, and Inso Kweon, *Natural image matting using deep convolutional neural networks*, in Computer Vision – ECCV 2016, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, eds., Cham, 2016, Springer International Publishing, pp. 626–643.

[5] Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski, *A bayesian approach to digital matting*, in Proceedings of IEEE CVPR 2001, vol. 2, IEEE Computer Society, December 2001, pp. 264–271.

[6] Timothy A. Davis, *Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method*, ACM Trans. Math. Softw., 30 (2004), p. 196–199.

[7] Xiaoxue Feng, Xiaohui Liang, and Zili Zhang, *A cluster sampling method for image matting via sparse coding*, in Computer Vision – ECCV 2016, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, eds., Cham, 2016, Springer International Publishing, pp. 204–219.

[8] Leo J. Grady and Rüdiger Westermann, *Random walks for interactive alpha-matting*, 2005.

[9] Kaiming He, Christoph Rhemann, Carsten Rother, Xiaoou Tang, and Jian Sun, *A global sampling method for alpha matting*, in CVPR 2011, 2011, pp. 2049–2056.

[10] Anat Levin, Dani Lischinski, and Yair Weiss, *A closed-form solution to natural image matting*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 30 (2008), pp. 228–242.

[11] Anat Levin, Alex Rav-Acha, and Dani Lischinski, *Spectral matting*, IEEE transactions on pattern analysis and machine intelligence, 30 (2008), pp. 1699–712.

[12] Jizhizi Li, Jing Zhang, and Dacheng Tao, *Deep automatic natural image matting*, in Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, Zhi-Hua Zhou, ed., International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 800–806. Main Track.

[13] Yaoyi Li and Hongtao Lu, *Natural image matting via guided contextual attention*, Proceedings of the AAAI Conference on Artificial Intelligence, 34 (2020), pp. 11450–11457.

[14] Qinglin Liu, Haozhe Xie, Shengping Zhang, Bineng Zhong, and Rongrong Ji, *Long-range feature propagating for natural image matting*, in Proceedings of the 29th ACM International Conference on Multimedia, MM '21, ACM, Oct. 2021.

[15] Yuhao Liu, Jiake Xie, Xiao Shi, Yu Qiao, Yujie Huang, Yong Tang, and Xin Yang, *Tripartite information mining and integration for image matting*, in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2021, pp. 7555–7564.

[16] I. Omer and M. Werman, *Color lines: image specific color representation*, in Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., vol. 2, 2004, pp. II–II.

[17] GyuTae Park, SungJoon Son, JaeYoung Yoo, SeHo Kim, and Nojun Kwak, *Matteformer: Transformer-based image matting via prior-tokens*, 2022.

[18] Yu Qiao, Yuhao Liu, Xin Yang, Dongsheng Zhou, Mingliang Xu, Qiang Zhang, and Xiaopeng Wei, *Attention-guided hierarchical structure aggregation for image matting*, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[19] R. J. Radke, *Image matting*, in Computer Vision for Visual Effects, Cambridge University Press, Cambridge, U.K., 1st ed., 2012, ch. 2, pp. 9–54.

[20] Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott, *A perceptually motivated online benchmark for image matting*, in Proceddings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009. Posterpräsentation: IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR '09, Miami, Florida, USA; 2009-06-20 – 2009-06-25.

[21] Richard Rickitt, *Special Effects: The History and Technique*, Billboard Books, 2nd ed., 2007.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015.

[23] Dheeraj Singaraju, Carsten Rother, and Christoph Rhemann, *New appearance models for natural image matting*, in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 659–666.

[24] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272.

[25] Tianyi Wei, Dongdong Chen, Wenbo Zhou, Jing Liao, Hanqing Zhao, Weiming Zhang, Gang Hua, and Nenghai Yu, *Deep image matting with sparse user interactions*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 46 (2024), pp. 881–895.

[26] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang, *Deep image matting*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 311–320.