Image matting using comprehensive sample sets

Henri Rebecq

March 25th, 2014

Abstract

This is a scientific project report for the Master of Science class "Advanced mathematical models for computer vision" by Nikos Paragios, given at École Normale Supérieure de Cachan between January and March 2014. It describes our implementation of a novel image matting algorithm [1] by Shahrian & al.

1 What is image matting ?

A very complete review of image matting can be found in [2]. This brief review (Section 1) is strongly inspired by the latter.

Image matting refers to the problem of accurate foreground estimation in images and video. Extracting foreground objects from still images or video sequences plays an important role in many image and video editing applications, thus it has been extensively studied for more than twenty years. Accurately separating a foreground object from the background involves determining both full and partial pixel coverage, also known as pulling a matte, or digital matting. Porter and Duff in 1984 introduced the alpha channel as the means to control the linear interpolation of foreground and background colors for anti-aliasing purposes when rendering a foreground over an arbitrary background.

1.1 Mathematical model for image matting

1.1.1 The compositing equation

Mathematically, the observed image I_z is modelled as a convex combination of foreground image F_z and background image B_z by using the alpha matte α_z:

$$I_z = \alpha_z F_z + (1 - \alpha_z) B_z$$

where $\alpha_z \in [0, 1]$. If $\alpha_z = 1$ or 0, we call pixel z definite foreground or definite background, respectively. Otherwise we call pixel z mixed. In most natural images, although the majority of pixels are either definite foreground or definite background, accurately estimating alpha values for mixed pixels is essential for fully separating the foreground from the background. An example of alpha matte is given in Figure 1.

1.1.2 An underconstrained problem

Given only a single input image (Figure 1), all three values α, F and B are unknown and need to be determined at every pixel location. The known information we have for a pixel are the three dimensional color vector I_z (assuming it is represented in some 3D color space), and the unknown variables are the three dimensional color vectors F_z and B_z, and the scalar alpha value α_z. Matting is thus inherently an under-constrained problem, since 7 unknown variables need to be solved from 3 known values. Most matting approaches rely on user guidance and prior assumptions on image statistics to constrain the problem to obtain good estimates of the unknown variables. Once estimated correctly, the foreground can be seamlessly composed onto a new background, by simply replacing the original background B with a new background image B' in the first equation.

1.1.3 The trimap

Without any additional constraints, it is obvious that the total number of valid solutions to the first equation is infinite. To properly extract semantically meaningful foreground objects, almost all matting approaches start by having the user segment the input image into three regions: definitely foreground R_f, definitely background R_b and unknown R_u. This three-level pixel map is often referred to as a trimap. The matting problem is thus reduced to estimating F, B and α for pixels in the unknown region based on known foreground and background pixels. An example of a trimap is shown in Figure 1.

Figure 1 – Input and output of the algorithm

(a) Input image (b) Trimap (c) Ground-truth alpha matte

1.2 Color sampling methods

Although the matting problem is ill-posed, the strong correlation between nearby image pixels can be used to reduce the complexity of the problem. Statistically, neighboring pixels that have similar colors often have similar matting parameters (i.e., alpha values). A straightforward way to use the local correlation is to sample nearby known foreground and background colors for each unknown pixel I_z. According to the local smoothness assumption on the image statistics, it can be assumed that the colors of these samples are "close" to the true foreground and background colors (F_z and B_z) of I_z, thus these color samples can be further processed to get a good estimation of F_z and B_z. Once F_z and B_z are determined, α_z can be easily calculated from the compositing equation :

$$\alpha_z = \frac{(I_z - B_z)\,(F_z - B_z)}{||F_z - B_z||^2}$$

Implementing such an algorithm that works well for general images is difficult. There are a number of questions that need to be answered, for instance, how to define the "neighborhood" of pixels. In other words, within what distance can the foreground and background samples be trusted ? How many samples should be collected ? How can we reliably estimate F_z and B_z from these samples ? Shahrian & al. proposed a novel algorithm [1] to solve this problem, which will be described in detail in the next section.

2 Image matting using a comprehensive sample set

Color sampling based methods collect a set of known foreground and background samples to estimate alpha values of unknown pixels. Most existing algorithms use different combinations of spatial, photometric and probabilistic characteristics of images to find the known samples that best represent the true foreground and background colors of unknown pixels (which are then used to extract the alpha matte). The quality of the extracted matte is highly dependent on the selected samples. It degrades when the true foreground and background colors of unknown pixels are not in the sample sets. This is called the missing true samples problem. Hence, the challenge is to select a comprehensive set of known samples that encompass the different F and B colors in the image.

[1] propose a novel strategy for generating such a comprehensive sample set, which guarantees that all color distributions are represented. Also, an efficient objective function over the pairs of candidate samples is proposed, which forces the algorithm to select the best pair that can represent the true foreground and background colors.

2.1 Generating the sample set

2.1.1 Splitting each region in subregions

First, the range over which samples are gathered is varied according to the distance of a given pixel to the known foreground and background. The motivation for this is that the closer an unknown sample is to known regions, the higher is the likelihood of a high correlation with known samples and thus known samples can estimate true samples robustly.

The trimap thus is divided into regions to obtain a set of known F and B samples which form foreground-background pairs for an unknown pixel. The width of the regions increases as each region subsumes the previous regions. The last subregion is simply the entire region (foreground or background). Figure 2 shows what a 4-subregions partitioning looks like.

Figure 2 – Subregion partitioning

(a) Input (b) Trimap (c) Subregions

The widths of the regions follow an incremental sequence starting from the region closest to the boundary. This is because for an unknown pixel that is close to the boundary, it is usually true that the correlation is likely to be highest with pixels in a narrow region close to the boundary.

Our implementation of subregion partitioning Although the original paper does not give further details regarding the choice of the scheme for region partitioning, we chose to fix the number N of subregions and to set the width of consecutive regions to be growing quadratically with respect to the index of the region (the width being measured using the L^2-distance transform to the unknown region R_u). Algorithm 1 gives the pseudo-code for the exact procedure that we used to perform subregion partitioning. In practice we found that using $N = 4$ subregions yields good results but experiments still need to be made in deeper detail.

Algorithm 1 Subregion partitioning

Input: region to partition R, unknown region R_u
Output: N subregions R_k with $k \in [1..N]$

1. $dt \leftarrow$distance transform from R to R_u
2. $d_{max} \leftarrow max\,(dt)$
3. **for** k from 1 to N
   (a) $R_k \leftarrow \{\}$
   (b) **for** each pixel $z \in R$
      i. **if** $dt\,(z) < \left(\frac{k}{N}\right)^2 d_{max}$
         - $R_k \leftarrow R_k \cup \{z\}$

2.1.2 Two-level hierarchical color and spatial clustering

For each subregion, a two-level hierarchical clustering is applied. In the first level, the samples are clustered with respect to color through Gaussian mixture models (GMM). In the second level, the

same clustering process is applied on samples of each cluster but with respect to spatial index of pixels. The mean value of the color in each cluster at the second level constitutes the set of candidate samples in each region. Thus, we obtain a comprehensive sample set that includes samples from all color distributions thereby handling the missing samples problem. Figure 3 shows some typical clustering and sample set obtained.

Figure 3 – Two-level hierarchical clustering

(a) Input (b) Clusters (false colors) (c) Clusters (mean colors) + sample set

Our implementation of subregion clustering Though the paper suggests using the number of peaks in the histogram as the number of components for the GMM, we think that in practice this definition would need to be further explicited as there are numerous ways to detect peaks in a histogram. In practice (time being limited), we chose to fix the number of color clusters N_C and spatial subclusters N_S for each subregion. Naturally these numbers should grow with the size of the subregion and thus these numbers depend on the index of the subregion. We set $N_C^{(k)} = N_C \times k^\lambda$ and $N_S^{(k)} = N_S \times k^\lambda$ where $\lambda = \frac{1}{3}$.

GMM clustering is done using the Expectation-Maximization algorithm (EM) initialized by K-means. Note that for efficiency reasons we constrained the covariance matrices to be scaled identity matrices such that there is only one parameter to be estimated for each matrix. We observed that the results obtained were similar to those obtained using diagonally-constrained matrices.

2.2 Choosing the candidate samples and selecting the best (F,B) pair

Each pixel in the unknown region collects a set of candidate samples that are in the form of a foreground-background pair. Pixels close to the boundary should sample candidate pixels which come from the region closest to the boundary of the foreground (resp. background) because the color correlation of the unknown pixel is likely to be the highest with pixels in this region. However the exact scheme for the choice of candidate samples for each pixel is not given in the paper. We chose to associate each pixel $z \in R_u$ to a given subregion by following the procedure detailed in Algorithm 2 to perform this operation.

Algorithm 2 Choosing candidate samples

Input: pixel $z \in R_u$, region R (either foreground R_f or backround R_b)
Output: $indexR$, index of the region R_k where candidate samples for z will be taken

1. $dt \leftarrow$distance transform from R to R_u
2. $d_{max} \leftarrow max\,(dt)$
3. **for** k from 1 to N
   (a) **if** $dt\,(z) < \left(\frac{k}{N}\right)^2 d_{max}$
   - indexR $\leftarrow k$

Once the set of candidate (F, B) pairs is determined for unknown pixels, the task is to select the best pair that can represent the true foreground and background colors and estimate its α. The

selection is done through a brute-force optimization of an objective function based on photometric and spatial image statistics.

2.2.1 Expression of the objective function

It consists of three parts as follows:

$$O_z\left(F_i, B_i\right) = K_z\left(F_i, B_i\right) \times S_z\left(F_i, B_i\right) \times C_z\left(F_i, B_i\right)$$

where:

- $K_z\left(F_i, B_i\right) = exp\left(-||I_z - \left(\alpha F_i + \left(1-\alpha\right) B_i\right)||\right)$: the compositing equation must successfully explain the color of a pixel as a convex combination of (F_i, B_i)
- $S_z\left(F_i, B_i\right) \propto exp\left(-||z - F_i^s||\right) \times exp\left(-||z - B_i^s||\right)$: favors spatially close pairs (which is intuitively satisfying)
- $C_z\left(F_i, B_i\right) \propto d\left(F_i, B_i\right)$ where $d\left(F_i, B_i\right)$ is Cohen's d value for the color distributions were F_i and B_i were taken from. It is inversely proportional to the overlap between the two distributions and favors pairs that come from well-separated distributions.

A measure of overlap between distributions Cohen's d value for distributions is given for 1D distributions in the article only as:

$$d\left(F_i, B_i\right) = \frac{\mu_{F_i} - \mu_{B_i}}{\sqrt{\frac{\left(N_{F_i}-1\right)\sigma_{F_i}^2 + \left(N_{B_i}-1\right)\sigma_{B_i}^2}{N_{F_i}+N_{B_i}-2}}}$$

The distributions with which we actually work are 3D distributions (colors). We must therefore choose a way to map this 1D definition to a 3D space. We chose to define:

$$d^{color}\left(F_i, B_i\right) = \left\| \begin{array}{l} d\left(F_i, B_i\right)^R \\ d\left(F_i, B_i\right)^G \\ d\left(F_i, B_i\right)^B \end{array} \right\|_{L^2}$$

Efficiency of the overlapping term In practice we observed on several examples that C_z has a negative effect on the objective function. It indeed often forces the algorithm too much to select pairs from very-well separated distributions. Its relative importance to K_z and S_z is too big. In our implementation the default objective function drops this term (though the choice of the objective function is up to the user). One way of to overcome this problem would be to assign a small weight to C_z in the objective function (experiments were made but didn't give satisfying results so far).

2.3 Pre-processing and post-processing

The obtained alpha matte is pre-processed and post-processed to refine the result. Pre-processing expands known regions to the unknown region according to certain distance and color conditions. Pre-processing is used to obtain a smooth matte by considering correlation between neighboring pixels. Details are left for the reader to consult in [1].

3 Our implementation

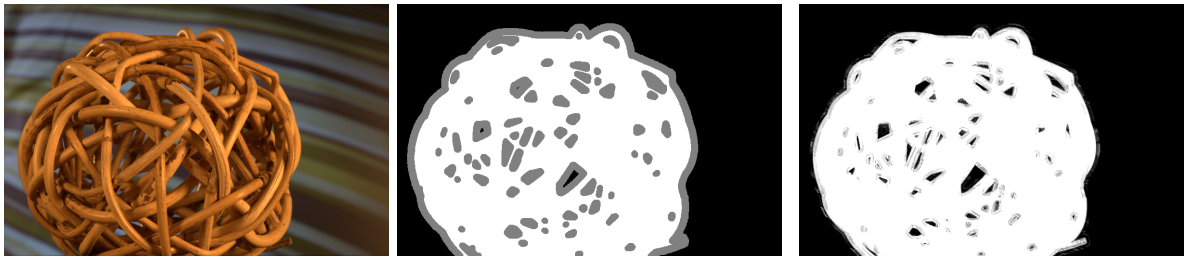
3.1 Results

In this section we give some sample results that were generated by our implementation of comprehensive sample set matting. Note that no quantitative evaluation of these results has been done because we didn't implement the pre/post-processing phases, therefore comparing our results with the original paper author's ones would be highly unfair. It's indeed clear that smoothing the matte would greatly improve its quality.

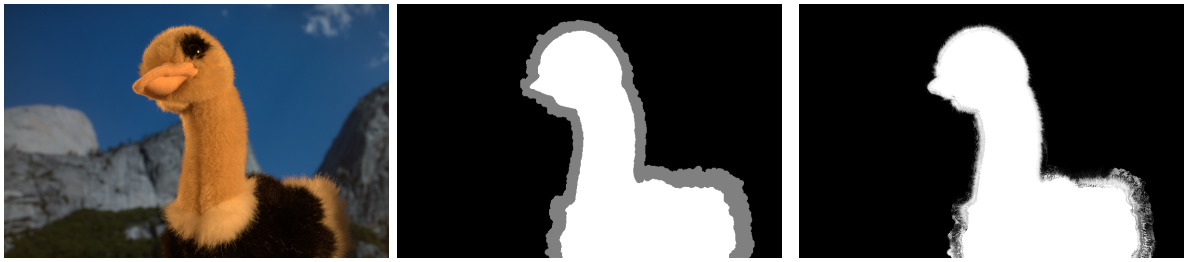
The results included here (figure 4) were generated using the default parameter values in our implementation (no values set manually), and should be easily reproducible.

Figure 4 – Sample results

(a) Giraffe (b) Trimap (c) Alpha matte



(d) Wood structure (e) Trimap (f) Alpha matte



(g) Ostrich (h) Trimap (i) Alpha matte

(j) Teddy bear (k) Trimap (l) Alpha matte

(m) Pencil holder (n) Trimap (o) Alpha matte

3.2 Where to find the code?

The code for our implementation of image matting using comprehensive sample sets can be found on Github. You can either use Git to clone the project or download the code as a compressed ZIP file.

3.3 How to make the program work?

3.3.1 Compilation

You need to have the library OpenCV installed on your computer (version 2.4.8 recommended, older versions not tested but should work properly). Details regarding the installation procedure for each platform won't be given here but are easily accessible online (here for example).

A Makefile is provided with the code so compilation shouldn't take more work than simply type make in the code directory. Note that you may have to change the lines **LIBS = -L/usr/local/lib** and **INC = -I/usr/local/include/opencv** to point to the directory where OpenCV is installed on your computer.

3.3.2 Usage

The program must be given as a command-line argument the name of the image (including the extension). Input images should be stored in directory **input**/ and trimaps in directory **trimap**/ with the exact same name. Usage example :

```
./cssmatting GT01.png
```

Note that a nice dataset of images can be found here.

3.4 How to use the graphical interface?

3.4.1 Displaying sample sets and best candidates

Once everything has been computed, the program will open three interactive windows that are synchronized together:

- "Input + (F,B)": Shows the input image. Any click on a pixel will show the best (F,B) pair associated with this point. The color of the line joining them gives an indication of the associated alpha value as a continuous variation from blue (0) to red (1).
- "Alpha Matte": Shows the computed alpha matte. Any click on a pixel will be passed on to the two other windows.
- "Sample set": When no pixel has been selected yet it shows the trimap. When a pixel is selected, this window shows its corresponding subregion, and the associated sample set.

Note that pressing any key will exit the program.

3.4.2 Changing the objective function

Move the slider in window "Input + (F,B)" to change the objective function for the selection of the best (F,B) pair. You can choose to use only the color constraint, the spatial constraint, the least-overlapping constraint or a combination of these. Note that the alpha matte will be updated (this can take some time depending on the size of the unknown zone).

3.5 Brief description of the data structures

Class Region The most important data structure used in this program is the class **Region**. It represents a subset of a given image by embedding a list of pixel positions (indexed over the main image 'input'). It provides facilities to get access to the barycenter, mean color and variance of the region, easy access to the equivalent binary map and a function to draw itself on an image. Foreground, Background, Unknown region, subregions, and all clusters are instances of this class.

Class CandidateSample This class is designed to represent a candidate sample. It contains the spatial position, color and a pointer to the region where it was extracted. Sample sets for each subregion are stored as lists of instances of CandidateSample.

3.6 Tweaking the parameters

You can tweak some parameters of the algorithm easily by changing values in the file **CSSMatting.cpp** (towards the beginning). Parameters that can be changed include the number of subregions, the number of clusters for the first subregion, the type of covariance matrix for the EM algorithm, the choice of the objective function that will be used.

References

[1] Ehsan Shahrian, Deepu Rajan, Brian Price, and Scott Cohen. Improving image matting using comprehensive sampling sets. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 636–643, Washington, DC, USA, 2013. IEEE Computer Society.

[2] Jue Wang and Michael F. Cohen. Image and video matting: A survey. *Found. Trends. Comput. Graph. Vis.*, 3(2):97–175, January 2007.