12/20/2016

CIS602-01 Computational Reproducibility

Project Report

On

Spectral Matting

A Levin, A. Rav Acha and
D. Lischinski

Ankita Anil
MS in Computers & Information Science
University of Massachusetts Dartmouth
Student Id- 01592435

Paper Information

Title: Spectral Matting
Author: A Levin, A. Rav Acha and D. Lischinski
Journal: IEEE Transaction on Pattern Analysis and Machine Intelligence
Year: 2008

Paper Summary

In the paper Spectral Matting, a new approach towards natural image matting has been presented, called Spectral Matting. While the term Matting or Digital matting refers to the process of estimating accurate foreground in images and videos, and accurately extracting the foreground object from the background along with an opacity estimate for each pixel covered by the object. Further, these operation enables compositing the extracted foreground object over a new background and supports tools used for image editing, video production and adding special effects in motion pictures.

The process of Spectral Matting explained in the paper automatically computes a set of fundamental fuzzy matting components from the smallest eigenvectors from a well defined Laplacian Matrix. In image processing, a Laplacian matrix is simply a matrix representation of a graph, where each node in a graph represents a matting components and the edge weights indicates the matting penalty.

The researchers, in the paper explains that all the existing methods to extract the foreground of an image requires the user to provide some constraints in the form of user-inputs like trimaps or simply a set of brush strokes, but the process of Spectral Matting can automate the matting process. Spectral matting approach is considerably based on Spectral Segmentation Techniques whose goal is to extract the hard segments. Spectral Matting extends this approach to generate soft matting components and explores the relationship between eigenvectors and matting components. It performs unsupervised image segmentation by examining the smallest eigenvectors of the image's graph's Laplacian Matrix. A linear transformation of these eigenvectors leads to the generation of a real-valued matting components. Finally, these matting components obtained, forms the building blocks of the complete foreground. It attempts to recover fractional foreground coverage at each pixel. This unsupervised matting algorithm is extended to constructing a particular matte in two different ways, first, user choosing from several matting components; second, user guided matting where user specifies her intention by a few mouse clicks.

According to a matting algorithm, each pixel I in an image is a combination of a foreground color and the background color.

$$I_i = \langle_i F_i + (1 - \langle_i) B_i$$

The above is called the ***Compositing equation***, which says each pixel is a combination of K image layers from F^1 to F^K, while the K vectors $\boldsymbol{\alpha}^K$ is defined as the *matting component*. These matting components are non-negative and should sum to 1 at every pixel and each component should preferably be completely transparent or completely opaque over as many pixels as possible. But, practically most images do not have matting laplacian whose multiple eigenvectors values to 0.

The paper also extends the work to derive an analogy between hard segmentation and matting to show that fuzzy matting components may be extracted from the smallest eigenvectors of matting Laplacian matrix. A laplacian matrix, L can also be defined as L = D-A , where A is an n*n affinity matrix and D is the diagonal matrix $D(i, i) = \sum_j A(i, j)$. The smallest eigenvectors store information about fuzzy cluster assignments of the pixels in the image. When these smallest eigenvectors are computed, the result maybe any linear combination of different matting components and recovering each components is like linearly transforming the eigenvectors. Also, matting components are not binary vectors unlike hard segments.

Once, the matting components are obtained from the Laplacian matting matrix, Unsupervised Matting or User-guided Matting can be used to obtain the foreground components. In unsupervised matting, foreground components are combined together to form the required foreground objects. But, in case of images where foreground and background objects consists of many different components, unsupervised matting fails. In User-Guided Matting, the user guides the matting process by deciding the foreground/ background assignments of the components. This generally done by implementing min-cut graph formulation. Further, user-guided matting leads to matte extraction using ***Scribbles***, where foreground matte is extracted from an image based on a small number of white markings for foreground and black markings for the background. By marking the components, a component is being constrained to belong to foreground whenever its area contains white scribbles or belong to background whenever its area contains black scribbles. The drawback of this approach lies in between when an images contains distinct connected components without any markings on them, these components are assigned with an average non-opaque values; and matte extraction by ***component labeling***, where pre computed matting components are simply labelled as background or foreground. The advantage of this approach lies in fuzzy areas, like hair strands in an image, do not offer themselves to hard constraints.

To quantitatively evaluate the Spectral Matting approach, ground truth data were captured and were compared with previous methods. The ground truth data consisted of three different dolls with seven different backgrounds each. For the approach mentioned in the paper, 60 matting components were extracted using 70 smallest eigenvectors of each cropped window, and the running of Matlab implementation for each image window was a few minutes. Later, the SSD errors between the mattes produced by different matting algorithms and ground truth data is plotted.

Conclusions inferred from the paper:

1. An analogy was derived between hard spectral image segmentation and image matting.
2. Paper showed how fundamental matting components are automatically extracted from the smallest eigenvectors of the matting Laplacian matrix.
3. It showed how matting components can help in automating the matte extraction process and reduce user effort.
4. It was also proposed how matting components enable user control and interaction over extracted matte, as generating matting components provide user with preview of optional outputs where user can directly control the outcome of the mattes.

Limitations of Spectral Matting:

1. In case of **highly cluttered images**, the process of component extraction becomes quite a challenging task, as cluttered images require a large number of matting components which in turn comes from large number of smallest eigenvectors as there is a linear transformation between them.
2. Another major challenge of spectral matting is **determining the appropriate number of matting components** for an image. If the matting components are too few, then it may not contain the desired matte; and if there are too many, then it complicates the computation and user interaction.

Future Works

1. The process of pre-computing the matting components of an image provides an option to compute color/ texture histograms and other statistics within each component. The **histogram similarity** may provide an important key for **component grouping** which would help matting algorithms using color models.

2. Designing a comparison between **different user interaction modes** and different matting algorithms. The ground truth data provided is a step towards this goal.
3. Designing a comparison between the **amount of user time required for extracting the mattes** in component picking interface and scribble based interface.

REPRODUCIBILITY OVERVIEW

1. **Code**

   The code is available for download at Spectral Matting web page on http://www.vision.huji.ac.il/SpectralMatting/ .
   The code has been compiled and tested on MATLAB 7.

   The spectralMattingCode folder contains MATLAB code in the form of functions and scripts. There are around 12 Matlab function files performing different operations on the input image and 3 Matlab scripts, out of which "***runAll***" script is used to reproducing the results mentioned in the research paper.

   runAll.m file re-creates all the examples from the paper on unsupervised matting and generates the following:

- alpha matte of the image
- all the matting components of an input image
- the results for unsupervised matting
- all candidates for alpha matte
- and the best alpha matte of the image which is same as the alpha matte.

   The spectralMattingCode folder also contains two already MATLAB formatted data files, ***scribbles.mat*** and ***my_images.mat***, which are loaded into runAll.m file for implementation.

   The folder also contains C++ programs to implement the concept of minimum cut and maximum flow algorithm on two types of graph representation, i.e. Adjacency List and Forward Star. Min-Cut graph formulation is used to speed up the search of optimal background or foreground mainly in the process of User-Guided Matting.

   There was no success in implementing Supervised Matting alone because, in order to execute supervised group matting, the Matlab function uses the C++ min-cut code when the number of matting components is large, but unfortunately, the C++ code has been showing clang errors. Further, site mentioned in the documentation to download the Min-Cut software http://www.cs.cornell.edu/People/vnk/software.html is no longer available.

2. **Data**

   The input data for the Matlab code is in the form of .tif images and are downloaded from the same above mentioned site. Following are the 5 .tif input images made available for performing unsupervised group matting to obtain the foreground mattes:

   1. face.tif
   2. kid.tif
   3. kim.tif
   4. wind.tif
   5. woman.tif

The site also provides the **ground truth data** which is used to quantitatively evaluate the Spectral Matting approach and *compare it with other approaches* mentioned in the paper. These ground truth data consist of three dolls (lion, monkey, monster) with seven different backgrounds each. The ground truth matte is extracted for each doll using the least square frameworks. Each image is down sampled to 560*820 pixels and test were performed on 200*200 windows cropped from the images.

1. **VMs/ Containers**
   No Virtual Machine or Containers were used in the research process.

2. **Workflows**
   There is no use of Workflow Management Software in the research.

3. **Provenance**
   No provenance information available.

4. **Other Reproducibility Information**
   A pdf file on technical report with some more details is also available on the site http://www.vision.huji.ac.il/SpectralMatting/ along with some supplementary images.

   Hardware Information
   A 3.2 GHz CPU is required to run the experiment, other than this there no other hardware specific information available on the execution environment.

   Software Information
   MATLAB 7 was used in the implementation. Also, it was mentioned in the ReadMe file that versions earlier to MATLAB 7 may show some problems. Though I was able to run the code on the current version of MATLAB under the student license (R2016b i.e. version 9.1).

EXPERIMENTS

In order to reproduce the results presented in the paper Spectral Matting on unsupervised grouped component matting, the key operation was to execute the runAll.m Matlab file, as provided in the documentation of the code. Some minor changes in the code of the runAll.m were done which generated the following 5 results for each input images.

- Final alpha matte of the input image.
- All the matting components of each input image.
- The result for unsupervised hard segmentation.
- All the candidates of alpha matte.
- The best alpha matte.

The input to this code were Matlab formatted data files scribbles.mat and my_images.mat, which consisted of 5 images, namely face.tif, kid.tif, kim.tif, woman.tif and wind.tif.

On execution of the runAll file, following operation were carried out on the images:

1. Computation of the Laplacian matrix.
2. Computation of the matting components of the image.
3. Computation of eigenvectors.
4. Computation of k-means algorithm for clustering the image pixels.
5. Grouping of matting components.

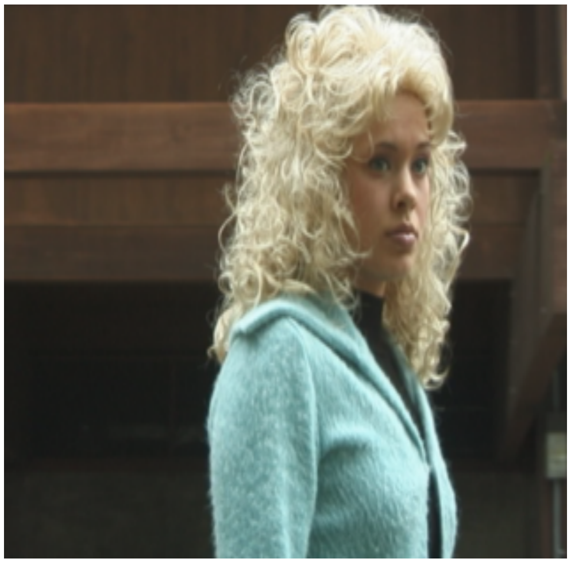
An execution video of the file runAll.m has been attached in the folder as Execution.mov .

Reproduced Results

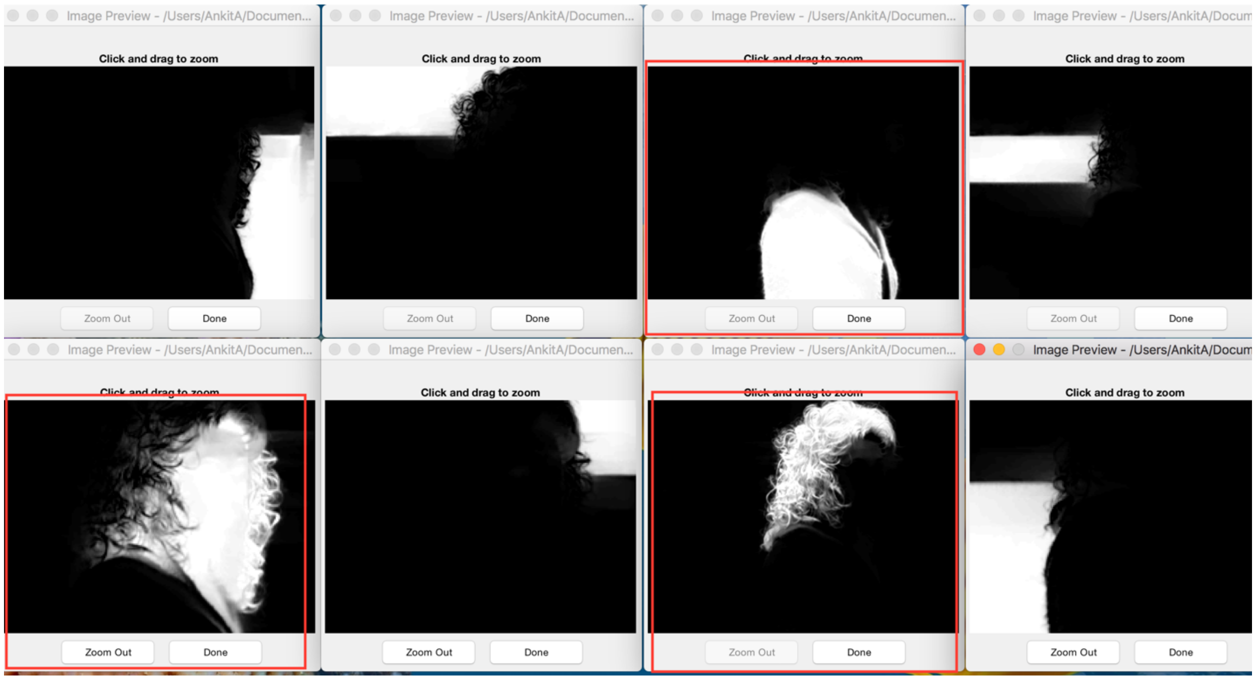
Following are some of the results that have been reproduced from the paper:

1. The code in runAll.m generates all the component images, alpha matte and unsupervised matte of the input image kim.tif, which is ***Figure 1*** in the paper Spectral Matting.

Input Image

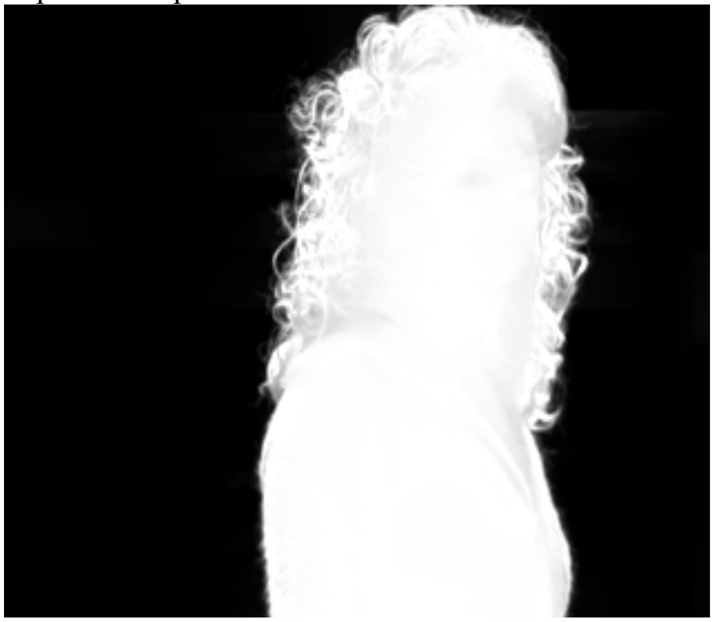


Following are the 8 matting components produced:

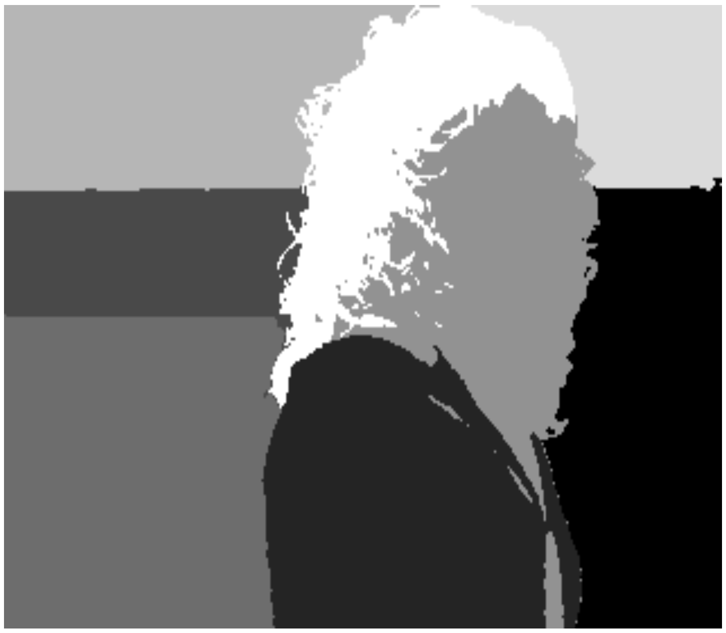


The components in red borders are combined together to form the Alpha matte of the foreground object.

Alpha Matte produced:

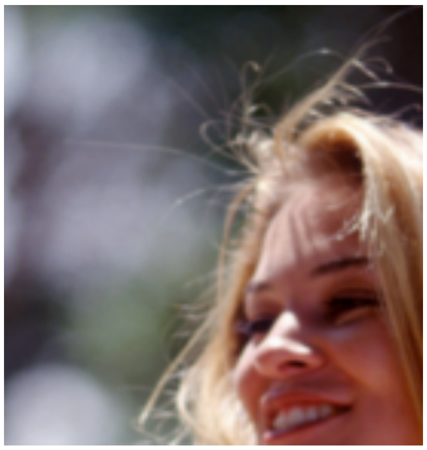


Unsupervised hard segmentation Output:

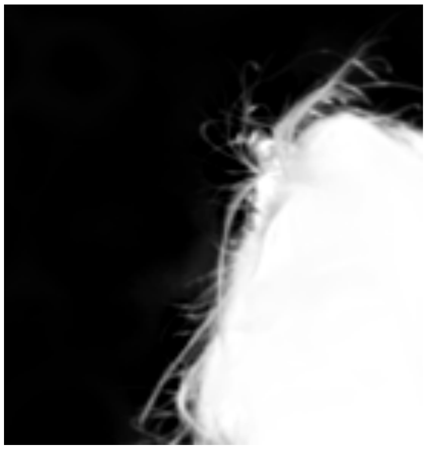


2. **Figure 3** from the paper Spectral Matting depicting unsupervised matting for two images i.e. **face.tif** and **woman.tif** have been reproduced along with all the candidates for alpha matte as shown in the figure.

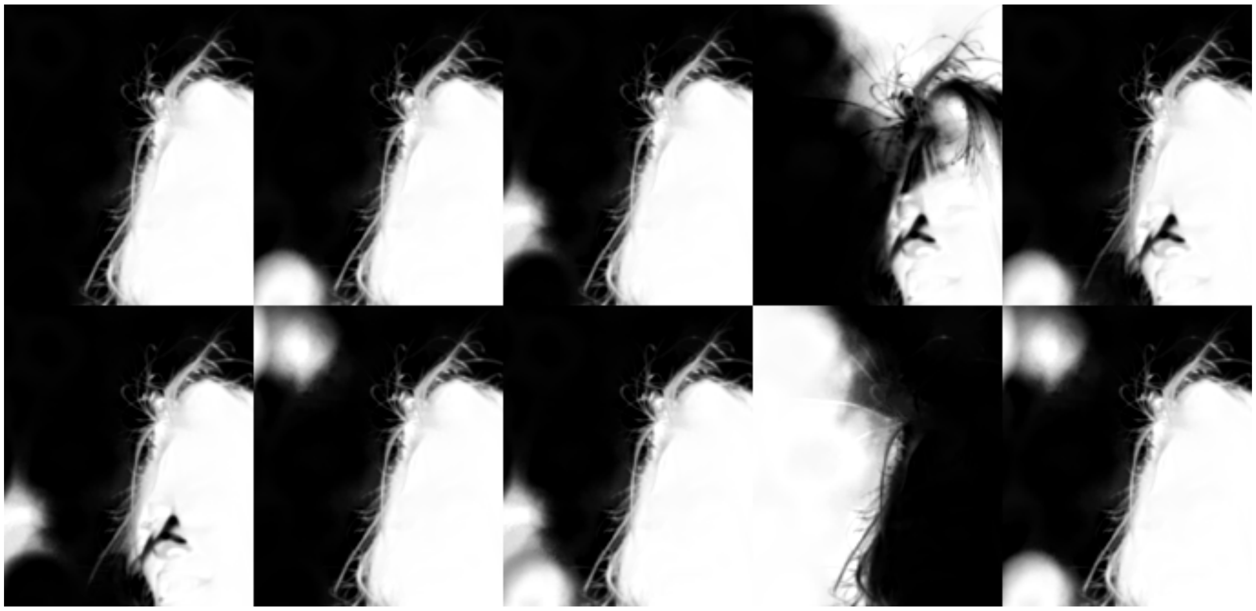
Input image face.tif



Below is the Alpha matte for face.tif



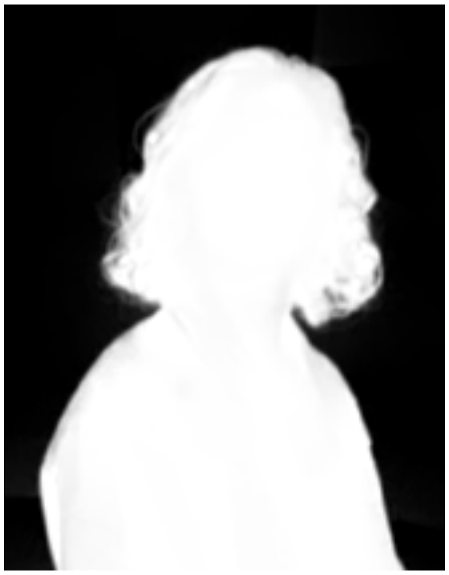
Below are all the candidates for Alpha Matte generated in the experiment:



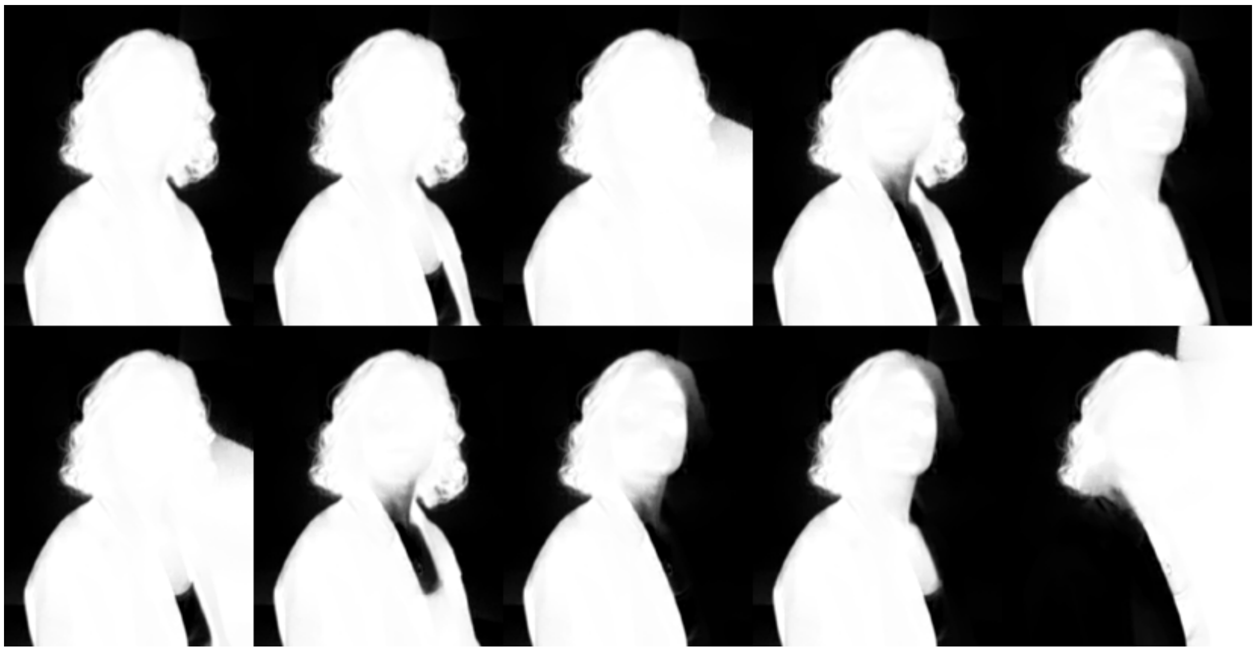
10 matting components were generated for face.tif image on implementation of the code runAll.m file along with the unsupervised hard segmentation image, which are all attached in the folder **outputImages**.

Input image- woman.tif

Below is the Alpha Matte for woman.tif



Below are all the candidates for alpha matte for the image woman.tif generated in the experiment:



10 matting components were generated for woman.tif image on implementation of the code runAll.m file along with the unsupervised hard segmentation image, which are all attached in the folder ***outputImages***.

3. **Figure 4** from the paper ***Spectral Matting*** depicting comparison of mattes produced by different matting methods from minimal user input, were reproduced only for the results of the approach presented in this paper.

   Scribbles is a form of user-guided matting, where foreground matte is extracted from an image based on a small number of ***white markings*** on foreground and ***black markings*** on the background. By marking the components, a component is being constrained to belong to foreground whenever its area contains white scribbles or belong to background whenever its area contains black scribbles.

   All other four column of the results were not reproduced as the respective codes were not available.

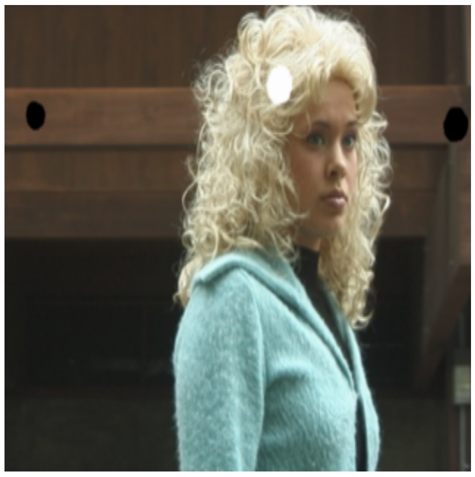
   The input images for this part of experiment were Matlab formatted images in the file ***scribbles.mat*** which were loaded in the code runAll.m along with the original images.

   Following are the images that were used in this experiment:

- kim.tif and kim_scribbles.tif
- wind.tif and wind_scribbles.tif
- kid.tif and kid_scribbles.tif

Image kim_scribbles.tif

Input original and scribble image:



Below is the matte generated for the image kim_scribbles.tif:

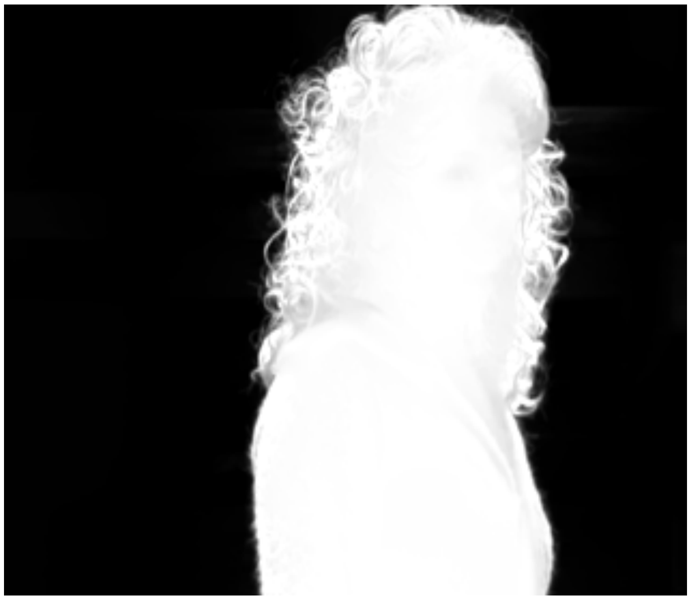
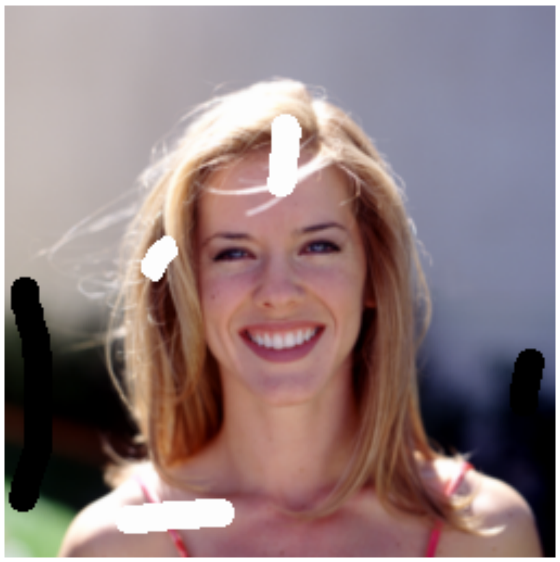


Image wind_scribbles.tif

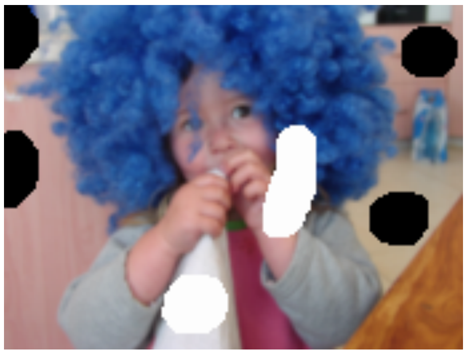
Input original and scribble image:



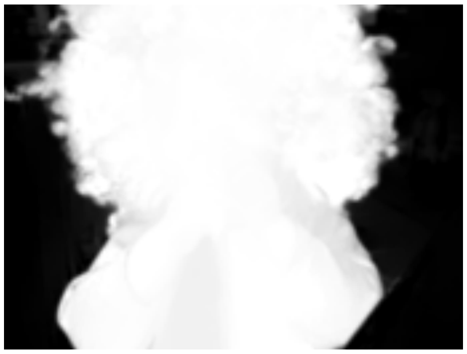
Below is the matte generated for the image wind_scribbles.tif:

Image kid_scribbles.tif

Input original and scribble image:



Below is the matte generated for the image kid_scribbles.tif:



Along with the mattes for the input scribble images, all the matting components for the extraction of the foreground objects (8 components for kim.tif, 29 components for wind.tif and 17 components for kid.tif) and unsupervised hard segmentation matting were also generated in the experiments which are attached in the ***outputImages*** folder.

Results that could not be reproduced

Figure 4 of the paper Spectral Matting, which illustrates different matte results produced by 5 different matting methods with minimum user input, I was able to reproduce only the first column of mattes (shown above) which corresponds to the approach discussed in this paper. But, unfortunately the last 4 columns of the matting results could not be reproduced because of the unavailability of the code.

Figure 7 from the paper could not be reproduced for the same reasons mentioned above.

In order to find the codes for the other four approaches, I individually looked for each author on Google and mailed them, if they can help me with the code.

I tried contacting Anat Levin twice, who is one of the authors of this paper as well as another paper, “A closed form solution to natural image matting.” whose results are used to generate the matte shown in column 3 of Figure 4. But, no response from the author was received.

Only for one of the matting algorithms, i.e. for the paper “Random walks for interactive alpha-matting.” Matlab code was found on author Leo Grady’s website, but the code in the folder “randome_walker_matlab” had a missing function file. I tried contacting the author for the same, but no response was received.

CONCLUSIONS

All the codes and the input images provided in the Spectral Matting code folder were quite useful in re-creating the examples from the paper.

However, due to the unavailability of the codes for implementing four other matting algorithms, and comparing the mattes produced by them to the approach presented in the paper, I was unable to plot the bar graph as shown in Figure 7. In Figure 7, the SSD errors between the mattes produced by different matting algorithms and ground truth data is plotted.

Also due to the above reason, last 4 columns of Figure 4, illustrating the different matte results of different algorithms, were also not reproduced.

There were no proper documentation available on how to use the ground truth data. In fact ground truth data was not used at all, only the images in the spectral matting code folder were used in the execution.

The data present under "Supplementary Materials" were of not much use, as it simply contained repeated figures from the Spectral Matting paper, but a few images under unsupervised matting and guided matting acted like a point of reference to verify and cross check the outputs produced on execution of the Matlab code.

It was strongly felt that if a research paper is mentioning and using the results of other researchers, it should explicitly provide the working code for those works as well to enable cent percent reproducibility and verification of the results.