Automatic Hair Extraction from 2D Images

Chuan-Kai Yang and Chia-Ning Kuo
Department of Information Management
National Taiwan University of Science and Technology
No. 43, Sec. 4, Keelung Road
Taipei, 106, Taiwan, ROC
ckyang@cs.ntust.edu.tw,M9709121@mail.ntust.edu.tw

ABSTRACT

Automatic hair extraction from a given 2D image has been a challenging problem for a long time, especially when complex backgrounds and a wide variety of hairstyles are involved. This paper has made its contribution in the following three aspects. First, it proposes a novel framework that successfully combines the techniques of *face detection*, *outlier-aware initial stroke placement* and *matting* to extract the desired hair region from an input image. Second, it introduces an *alpha space* to facilitate the choice of matting parameters. Third, it defines a new comparison metric that is well suited for the *alpha matte* comparison. Our results show that, compared with the manually drawn *trimaps* for hair extraction, the proposed automatic algorithm can achieve about 86.2% extraction accuracy.

Categories and Subject Descriptors

I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Object recognition*

Keywords

Hair Extraction, Matting, Face-Detection, Dynamic Programming

1. INTRODUCTION

It has been shown to be possible that, given a person's photo, he/she can "try on" different hairstyles by selecting the previously extracted hairs from a database, as proposed by Liu et al. [9] and Chou et al. [1]. However, extracting the hair from an input image has always been a challenging problem due to numerous intervening factors such as lighting, shadows, skin colors, backgrounds, clothing, and so on, especially when many of these factors can present large variety or complexity. For example, it is possible that both the environment background and the subject's clothing share similar color or textures to the hair, and thus a precise separation of the hair from the rest may be very difficult.

In this paper, an extension of our previous paper [17], we have proposed a novel framework that combines *face detection*, *outlier-aware initial stroke placement*, and *matting* to make automatic hair extraction possible. First, we apply the existing *active shape model* technique for identifying the facial contour and facial features from the input image. Second, by making use of the detected positions of the facial features, our system could place an initial stroke so that most of it would lie within the target hair region. A *dynamic programming* approach is applied to further adjust the position of the stroke to increase the degree of alignment of the stroke with the desired hair region. Finally, given the previously generated stroke, a well known *closed-form matting* is applied to derive a segmentation of the image so that the hair region is roughly separated from the background. Such a segmentation is used to generate a *trimap* so that *closed-form matting* is applied again to more accurately extract the hair as the foreground.

In addition to proposing a new framework to automatically extracting the desired hair from a given image, we have also made two more contributions as follows. First, it is widely known that eyebrows, with similar color and texture, are very easy to be tangled with hair, thus further complicating the task of extracting exclusively the hair. We propose a way that oftentimes could remove eyebrows from the hair that otherwise would look very weird. Second, we propose a *similarity metric* that could efficiently and intuitively compare two *alpha matting* results, and we believe such a mechanism can also be generalized to other comparisons as well. To validate the effectiveness of our system, we manually draw the *trimaps* for a set of images, and treat their corresponding matting results as the *ground truth*. In terms of comparisons against the ground truth, our system can achieve 86.2% extraction accuracy on an average. Comparisons of our results with those from others have also been conducted when applicable.

The rest of the paper is organized as follows. Section 2 briefs existing works which are related to this work. Section 3 details our core algorithm on extracting the desired hair from an input image. Section 4 compares our results with the ground truth or those from others when possible. Section 5 concludes this work and hints for several potential future directions.

2. RELATED WORK

There are at least four techniques involved in this study. The first one is *face detection*, which is to determine if a given image contains a human face or not, and if so, locate where it is. In general, one could apply traditional *object detection* approaches on human face for this purpose. Viola et al. [14] adopted a *machine learning* approach, where the involved features are calculated through *Haar wavelet*. A huge number of features can be found, and the technique of *AdaBoost* is utilized to choose good features among them. The final output is a rectangular window to mark the detected face.

Active shape model or *ASM* for short, proposed by Cootes et al. [4] is also a method based on machine learning; however, it represents the detected object through a collection of feature points instead of a rectangle. Via a *principle component analysis* or *PCA* for short, a *deformable model* is derived and used to locate the object in an image. Cootes et al. [3] later on proposed another approach called *active appearance model* or *AAM* for short, which is also based on machine learning, and uses training data to obtain a statistical appearance model. In addition to the shape information, the texture information of the detected object is kept as well.

The second one is *outlier detection* [10], which is to find the *outliers* from given data. In terms of data type, this could be a *univariate* or *multivariate* problem, and in terms of algorithm, it could be *parametric* or *non-parametric*, where the latter one could be further classified into *distance-based*, *clustering-based* and *spatial-based* [6]. In particular, the definition of a distance-based outlier is as follows: if β percentage of the data are at least r distance away from o, then o is an outlier. Sometimes the definition can be changed from β to be the nth closest neighbor, and it can be shown that these two definitions are equivalent.

The third related technique is *matting*. The problem of a matting is to separate the foreground from the background of a given image. In general, such a problem can be formulated as the following:

$$I_p = \alpha_p F_p + (1 - \alpha_p) B_p \quad (1)$$

where p is a pixel (coordinate), I the input image, F the foreground in I, B the background in I, and α_p the *opacity* of the foreground for pixel p, respectively. As there are too many unknowns in this equation, in general it cannot be solved directly without the assistance from users. Normally users need to provide *hints* through the form of a *trimap* to distinguish the *foreground*, *background* and *unknown* parts of an image. According to the survey [15], basically matting algorithms can be classified into three categories: *sampling-based*, *propagation-based* and *combined*. *Sampling-based* approaches derive/interpolate the α *values* for the unknown pixels based on the color distributions of the known foreground and background pixels. For example, Chuang et al.'s *Baysian matting* [2] solves this problem by maximizing the *log maximum a posterior* or *Log MAP* for short. In general this type of approaches may suffer if the user inputs are too sparse or there exist overlappings between the color distributions of the foreground and background. Through the definition of *affinity* between adjacent pixels, *propagation-based* approaches estimate the α *values* of neighboring pixels based on the known ones, and normally they convert matting problems into corresponding *linear systems* which are solved by matrix operations. For example, *Poisson matting* [13] assumes that *foreground* and *background* colors are *locally smooth* and therefore obtains the *Poisson equation* after differentiating Equation 1, that is, $\triangle = div(\nabla I/(F - B))$. And it can be solved aforementionedly to derive the desired α values for the unknown pixels. Similarly, *closed-form matting* [7] further assumes that the values of $1/(F - B)$ and $B/(F - B)$ are constants within a small 3×3 window to simplify and accelerate the derivation process. In general, due to the assumption, the results are relatively *blur*, compared with *sampling-based* approaches. Some approaches are combinations of previous ones. For example, robust matting [16] combines the merits of *sampling-based* and *propagation-based*, extends the concept of *closed-form* to treat foreground and background as neighbors for all the pixels, and proposes a new color sampling strategy to define the similarity among the pixels in the foreground and background. Evaluating the quality of a matting result is an important issue. Normally a *ground truth alpha map* should be obtained, and the resulting alpha map generated by each matting algorithm is compared against the ground truth. According to a popular website [11], four metrics are proposed to measure the quality, including *sum of absolute difference* or *SAD* for short, *mean square error* or *MSE* for short, *gradient* and *connectivity*, where the latter two are to compare the difference between two alpha maps in terms of gradient and connectivity, respectively. In this paper, we resort to *closed-form* matting for the following two reasons. First, our inputs are relatively sparse, and so far the best implementation of *propagation-based* matting approaches is *closed-form* matting. Second, the source code of *close-form* matting is publicly available.

The final related technique is *automatic hair segmentation*. Chou et al. [1] proposed to make use of ASM to identify the average colors for skin, hair, background and clothes to be used as four seed colors, and applied *K-means* to segment the input image into multiple regions. A trimap a formed by performing the *morphological* operations on the hair segment, and a *closed-form* matting is used to identify the hair region. Rousset et al. [12] made use of the work from Viola et al. [14] to define the head region, proposed *frequential mask* and *color mask*, and integrated them together to define foreground and background regions before applying the closed-form matting. Lipowezky et al. [8] also applied Viola et al.s' work for face detection, performed a clustering to find eyes and mouth, adopted a machine learning method for finding the seed in the hair region, and finally performed a *region-growing* from the seed region to derive segments to turn into a trimap, followed by closed-form matting. To sum up, it is interesting to see that, in addition to the initial segmentation/classification/clustering to identify the hair region, each of these approaches applies a *matting* process at the end, and this is simply because of the nature of a hair region, which is often very fuzzy and furry, and therefore is hard to capture without a matting process.

3. AUTOMATIC HAIR EXTRACTION

3.1 System Overview

Our current system assumes that the face and hair of an input photo are strictly within the image without touching the boundary of the input photo. More recent matting algorithms normally start with *strokes* to find an initial *segmentation*, from which a *narrow band* is added along the boundary of the desired segmentation to derive a *trimap* that is used as the input for the final *matting* process. By adopting a similar framework, our *hair extraction* algorithm consists of the following three steps. First, we have to define the *foreground* and *background* through the help of *ASM* so that automatic *strokes* can be generated. Second, we apply *matting* techniques to identify the initial *image segmentations*. Along the hair segment, we extend its boundary into a *narrow band* to be used as a *trimap* for being the input for the next stage. Finally, given the trimap the final *matting* process is applied to extract the desired hair region from the input image. Figure 1 shows a system overview of our approach and the details regarding each step is to be given in the ensuing sub-sections.

3.2 Stroke Generation

Stroke generation includes three parts, the *face stroke*, the *hair stroke*, and the *boundary stroke*, where the first two are related to face detection, while the last one requires only the dimension information of the input image. We now describe the generation of these strokes in detail.

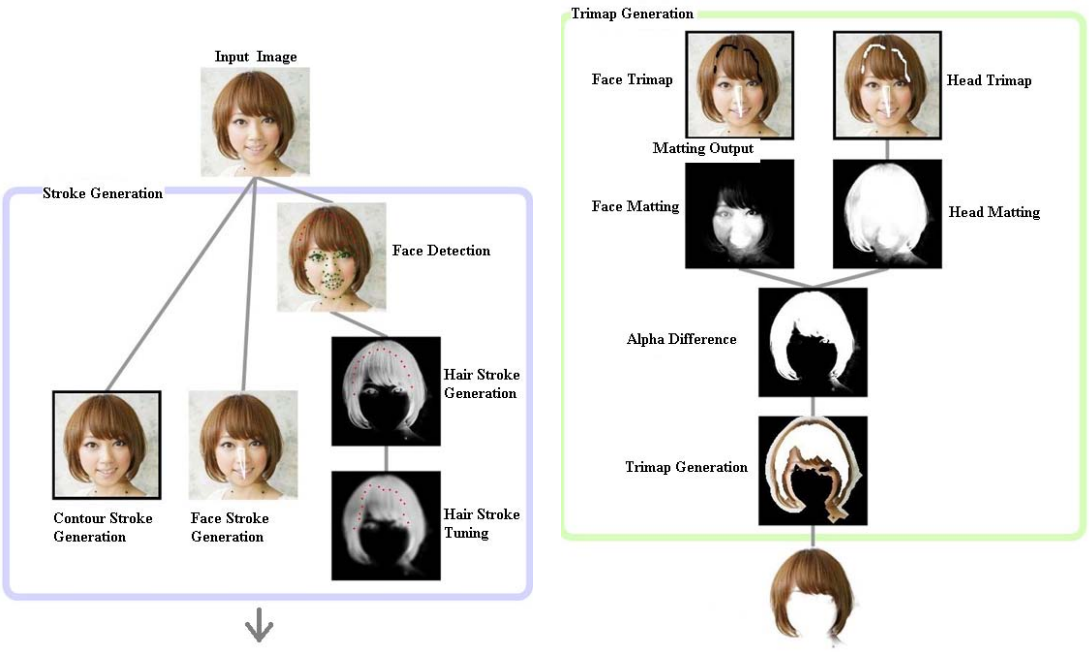


Figure 1: System overview.

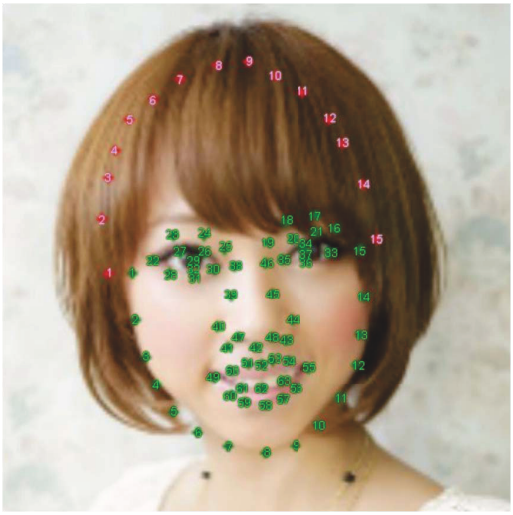


Figure 2: ASM for face detection, where the green dots are detected by ASM, and the red dots, generated based on the green dots, are for placing the hair strokes.

First, from the face detected by ASM, 63 feature points are defined, as shown in green in Figure 2. Let us denote these feature points as ASM_1 to ASM_{63} hereafter. To place the face stroke, the trick is to avoid touching the neck, eyes, and hair portion of a photo. Currently we set the *face stroke* to be of a triangular shape, consisting of ASM_{38}, ASM_{46} and ASM_{58}.

Second, we could set up the initial positions of the points in the hair, denoted as H_1 to H_{15} and are shown in red in this figure. More specifically, according to the well-known fact that human eyes often locate near the vertical mid point between the top of the head and bottom of the chin, we therefore treat the line connecting both eyes (ASM_{32} and ASM_{37}) as a horizontal mirroring axis, to flip up the facial contour points from ASM_1 to ASM_{15} to become the points from $Mirror_1$ to $Mirror_{15}$. That is, $Mirror_i$ is the *reflection point* of ASM_i by treating the line segment formed by ASM_{32} and ASM_{37} as the reflection axis. By setting the mid point of ASM_1 and ASM_{15} to be the central position C for a face, that is:

$$C = \frac{ASM_1 + ASM_{15}}{2} \tag{2}$$

and then the initial point for H_i is:

$$H_i = [1 + 0.2(\frac{8-i}{7})^2](Mirror_i - C) + C \tag{3}$$

And then we connect two consecutive H_i and H_{i+1} to form hair stroke L_i, as shown in Figure 3(a), with i being marked at the lower right corner. Later on we show how to tune the positions of these initial points so that they could be good candidates for representing the hair. We then connect these points to form a set of *piecewise linear* line segments, and from which we select a *reliable* subset of these line segments for being the *hair strokes*.

Finally, the boundary stroke is composed of the rectangular boundary of the input image, and based on the assumption that the input photo is fully contained within the image, the boundary stroke would not intersect with the hair portion.

3.3 Hair Stroke Tuning

It is easy to see that the previous setting of hair strokes may not be accurate as the hair configuration could vary from person to person. The goal for this hair stroke tuning is to avoid hair strokes from

(a) hair strokes

(b) corresponding alpha maps from (a)

Figure 3: Hair stroke generation

intersecting with the skin or background area of the input photo. For a better tuning, we first calculate the *similarity* among L_is, and then based on the *proximity matrix* to remove *outliers* of the initial hair strokes, followed by adjustments on the positions of the remaining hair strokes.

By treating hair stroke L_i as the foreground stroke, and the boundary stroke as the background stroke, we could obtain the resulting *alpha map* A_i, and the similarity among L_is is defined upon the similarity among A_is. Figure 3(b) shows the corresponding alpha map for each L_i. In order to perform comparisons among all these alpha maps, we need to define a proper *similarity metric*. According to *Jaccard Index*, the similarity between two sets can be defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4}$$

that is, the ratio of the size of two sets' intersection versus the size of two sets' union. However, as the pixel value in an alpha map is a floating point, also to ease the computation, we propose to define the *alpha similarity* between two alpha maps A and B as:

$$alpha_{similarity}(A, B) = \frac{\sum_p min(A_p, B_p)}{\sum_p max(A_p, B_p)} \tag{5}$$

Therefore the *alpha similarity matrix*, where each entry represents the similarity between two corresponding alpha maps, is calculated and shown in Table 1. It is evident that the more similar two image are to each other, the higher this value is. For outlier detection, we adopt a *distance-based* approach, and the parameter setting is, $r = 0.7$ and $n = 6$. And in case more than half (7 hair strokes) are determined to be outliers, we only remove the 7 hair strokes that are the most far from others.

We now describe how to tune the remaining hair strokes. Assume that there are q hair strokes left, and S be the average map of these q alpha maps, as shown in Figure 4(a). To avoid the interference of potential noise, we further apply a 5×5 *uniform blur filter* on S, as shown in Figure 4(b). As a higher value in S represents a higher probability to be the hair, we try to adjust the positions of hair point H_i so that the resulting hair strokes can pass through the area with higher values. Note that a hair point H_i cannot move too drastically so that the hair point itself or the connecting hair stroke L_i may fall out of the hair area. For this reason, let $M_i = H_i - C$, then $B_i = C + 0.7M_i$ and $E_i = C + 1.3M_i$ denote the lower (closer to the center) bound and upper bound for the hair point movement, respectively, as shown in Figure 4(c). Let HM_i to be the set of grid points on $\overline{B_iE_i}$, and further let $HM_{i,j}$ be a point j in this set, where j ranges from 1 to $|HM_i|$, and it represents the jth point in this set. And $LM_{i,j,k}$ denotes the line connecting $HM_{i,j}$ and $HM_{i-1,k}$. Figure 4(c) shows 15 green lines, from HM_1 to HM_{15}, where each line has three red points, denoting the corresponding H_i, B_i and E_i mentioned previously.

To guide the movement of H_i, we define *objective functions* to determine the hair strokes in S. More specifically, we use a vector J to represent the choices we collectively make on each green line, and further assume that on HM_i we choose J_i, then the objective function can be formulated as the following:

Table 1: The alpha similarity matrix for the hair strokes.

	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12	H13	H14
H1	100%	72%	69%	92%	68%	56%	52%	46%	43%	44%	41%	41%	41%	42%
H2	72%	100%	96%	72%	92%	74%	70%	63%	59%	60%	58%	58%	57%	58%
H3	69%	96%	100%	70%	93%	75%	71%	64%	60%	61%	59%	59%	58%	59%
H4	92%	72%	70%	100%	69%	56%	53%	46%	43%	44%	42%	41%	41%	42%
H5	68%	92%	93%	69%	100%	80%	75%	68%	64%	65%	62%	62%	62%	62%
H6	56%	74%	75%	56%	80%	100%	94%	78%	80%	81%	76%	76%	75%	72%
H7	52%	70%	71%	53%	75%	94%	100%	79%	83%	84%	77%	77%	76%	71%
H8	46%	63%	64%	46%	68%	78%	79%	100%	92%	93%	90%	90%	89%	86%
H9	43%	59%	60%	43%	64%	80%	83%	92%	100%	98%	93%	92%	91%	86%
H10	44%	60%	61%	44%	65%	81%	84%	93%	98%	100%	91%	91%	90%	84%
H11	41%	58%	59%	42%	62%	76%	77%	90%	93%	91%	100%	99%	97%	92%
H12	41%	58%	59%	41%	62%	76%	77%	90%	92%	91%	99%	100%	98%	93%
H13	41%	57%	58%	41%	62%	75%	76%	89%	91%	90%	97%	98%	100%	94%
H14	42%	58%	59%	42%	62%	72%	71%	86%	86%	84%	92%	93%	94%	100%

$$J^* = arg\max_J(\sum_{i=1}^{14} cost(LM_{i,J_{i+1},J_i})) \tag{6}$$

and $cost(L)$ can be defined in three ways:

$$costSum(L) = \sum_{p \in L} S_p \tag{7}$$

$$costAverage(L) = \frac{1}{|L|} \sum_{p \in L} S_p \tag{8}$$

$$costMin(L) = \min_{p \in L} S_p \tag{9}$$

where S_p is a pixel p in the average alpha map S. Assume that each HM_i contains about 30 points, then to find the optimal hair stroke in a brute force way requires a combinatorial time complexity of 30^{15}, which is evidently intractable. Instead, let $L_{i,j}$ be the maximum cost that a hair stroke ends at $HM_{i,j}$, then such an optimization problem can be solved through a *dynamic programming* method by defining the following recursive relationship:

$$L_{i,j} = \begin{cases} 0 & \text{if } i = 1 \\ \max_k(cost(LM_{i,j,k}) + L_{i-1,k}) & \text{if } i > 1 \end{cases} \tag{10}$$

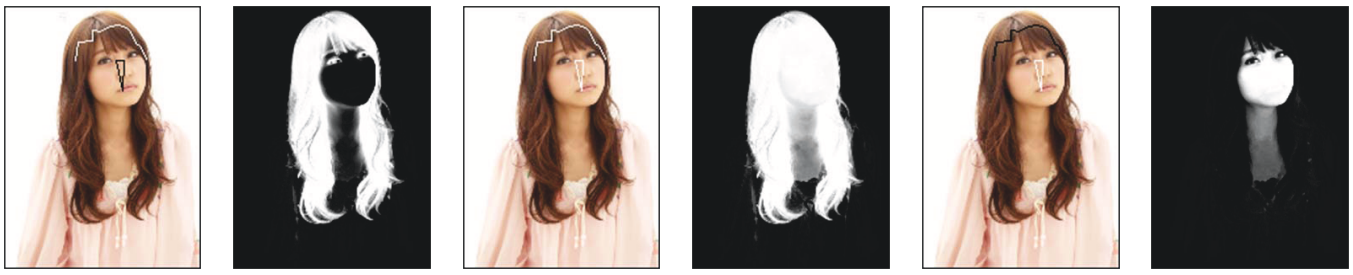
Once the dynamic programming is finished, we could find the best hair strokes. To better refine the results, we perform another run of *outlier detection* to remove possible mistakes made at this stage.

3.4 Trimap Generation

To derive the desired hair portion through matting, intuitively we could treat the boundary stroke and face stroke as the background strokes and the hair strokes as the foreground strokes, as shown in Figure 5(a). However, the shadow on the neck, as well as the eye balls could potentially be mistaken as parts of the hair, as shown in Figure 5(b). To deal with this problem, we separately perform *head matting* and *face matting* and subtract the alpha map of the latter one from that of the former one to derive the desired hair matting result.

The head matting is performed by setting the hair strokes and face stroke as the foreground strokes, and the boundary stroke as the background stroke, as shown in Figure 5(c), and the resulting alpha map, called *head alpha* hereafter, is shown in Figure 5(d). As for the face matting, it treats the face stroke as the foreground stroke, and hair strokes together with the boundary stroke as the background strokes, as shown in Figure 5(e), and the resulting alpha map, called *face alpha* hereafter, is shown in Figure 5(f). We observe that head alpha and face alpha overlap significantly, and their difference roughly corresponds to the *hair alpha* that we want to obtain.

Figure 6 shows various results, referred to as the *alpha space* hereafter, of using different weighting associated with the head alpha and face alpha before taking their difference. For example, the result at (m, n) corresponds to $m \times head_{alpha} - n \times face_{alpha}$. After numerous experiments and observations, we found that the combination of $(2, 4)$ in the alpha space normally gives us the best result, that is, we could obtain a clean and clear alpha that excludes the neck. We will call this linear combination result the *alpha difference*, or denoted as $alpha_{difference}$ hereafter. However, as can be seen from Figure 6, the resulting *alpha difference* still includes undesired eye regions, therefore a further processing is required. For this, we first *binarize* the resulting alpha, and perform a *morphological erosion* to get rid of the eye regions, and according to our observation, the *erosion radius* is set to be *one tenth of the face size* determined by the ASM process, with the *erosion centers* to be the centers of both eyes. More specifically, assume pixel values are normalized to be ranging from 0 to 1, we treat the pixels with value larger than 0.5 as the foreground stroke, as shown in white in Figure 7(a), and the pixels with value smaller than 0.5 as the background stroke, as shown in black in Figure 7(b). In addition, we also set the eye regions to be the background so that they will not appear in the final result. This is done by dilating the eye regions detected through ASM by 2 pixels. In order to avoid affecting the foreground setting, when the dilated eye regions intersect with the foreground, any intersected pixel still maintains as a foreground pixel. As a result, the *final trimap*, as shown in Figure 7(c), can be generated. The *final alpha map*, denoted as $final_{alpha}$ hereafter, as shown in Figure 7(d), as well as the *final extracted hair*, as



(a) (b) (c) (d) (e) (f)

Figure 5: Trimap generation. (a): the foreground (white) and background (black) strokes. (b): the alpha for (a). (c): the foreground and background strokes. (d): the alpha for (c). (e): the foreground and background strokes. (f): the alpha for (e).

(a) (b) (c) (d) (e)

Figure 7: Matting. (a): the foreground for the final trimap. (b): the background for the final trimap. (c): the final trimap. (d): the final alpha. (e): the final result.

shown in Figure 7(e), can be obtained.

4. RESULTS

We have conducted our experiments using a machine with an Intel Core 2 Quad CPU @ 2.83GHz and 4GB memory, running on Windows XP and the programming language is C++. Although the open source of closed-form matting is written in Matlab, it can be interfaced from C++ through the Matlab engine.

To test our proposed automatic hair extraction algorithm, we have randomly collected 70 photos from a Japanese hairstyle website http://www.beauty-box.jp, manually drawn their trimaps, executed the closed-form matting, and used the resulting alpha maps as the ground truth to evaluate the results. We adopt the evaluation metrics MSE and SAD, together with our proposed *alpha similarity* metric (Section 3.3). We also study how different cost functions influence the hair strokes, how alpha difference relates to the final alpha map, and how various parameters affect the outlier detection results. Note that in terms of MSE and SAD metrics, they are the smaller the better, our proposed alpha similarity, on the other hand, is originally the larger the better. As a result, in the following comparisons, we modify the definition of *alpha similarity* to become $1 - alpha_{similarity}$. And to ease the computation, all the images are *uniformly scaled* to the size of $width \times 200$ or $200 \times height$ before the experiments, where the $width$ and $height$ are in the range from 100 to 200.

In Table 2 and Table 3, we evaluate the *alpha difference* and *final alpha* using different cost functions and different evaluation functions, where Max means the image with the maximal value among the 70 input images, while Min the minimal value, and Avg the average value, respectively. All these values are computed based on the comparisons with the ground truth.

Table 2: The evaluation of the alpha difference. The best result is highlighted with a gray background.

Eval. Func.	Stat.	CostAverage	CostMin	CostSum
MSE	Max	86314	86264	121467
	Min	10654	10655	19610
	Avg	44899	45596	56876
SAD	Max	233	320	470
	Min	29	29	65
	Avg	118	121	176
Alpha Sim.	Max	0.805	0.88	0.96
	Min	0.071	0.07	0.111
	Avg	0.22	0.233	0.421

It can be seen from these two tables that the performances of CostAverage and CostMin are close, while CostAverage still wins. On the other hand, CostSum tries to select more pixels, and as a result it pays less attention to the content, thus leading to the worst performance.

Furthermore, in the following we also demonstrate the resulting images using different evaluation functions and different cost func-

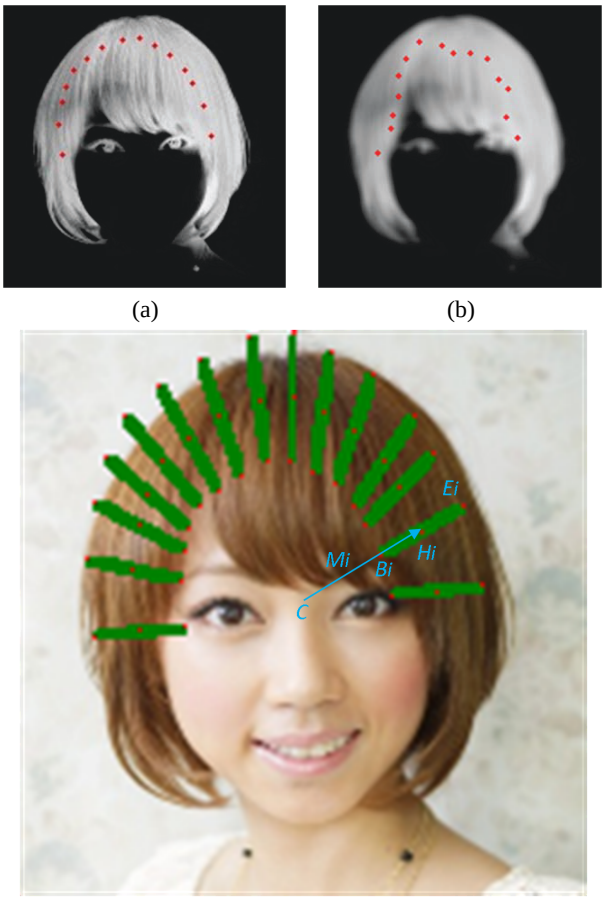


Figure 4: Stoke tuning. (a): the average alpha. (b): the uniformly blurred and adjusted hair points. (c): the range of each HM where points can move.

tions. The first row shows the hair trimaps, the second row the final alpha maps, while the third row the ground truth alpha maps as mentioned previously. Figure 8 shows the 10 best results based on MSE, SAD, and our proposed alpha similarity using CostAverage cost function. Note that the leftmost column shows the best among these 10 images, and the rightmost one the worst.

On the contrary, Figure 9 shows the 10 worst results based on the same three metrics with the same cost function of CostAverage. Note that the leftmost column shows the worst among these 10 images, and the rightmost one the best.

Figure 10 shows the 10 best results based on MSE, SAD, and our proposed alpha similarity using CostMin cost function. Figure 11 shows the 10 worst results based on the same three metrics with the cost function of CostMin.

Figure 12 shows the 10 best results based on MSE, SAD, and our proposed alpha similarity using CostSum cost function. Figure 13 shows the 10 worst results based on the same three metrics with the cost function of CostSum.

It can be observed that when the hair strokes intersect with the background or the face area, the results become evidently worse,

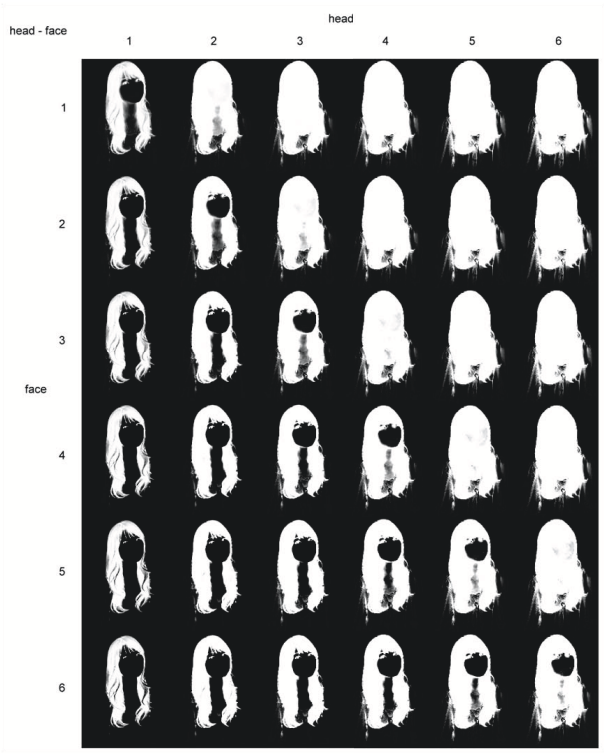


Figure 6: Alpha space

and the example for the former can be seen from many images, especially the one with a green background in Figure 13, while the example for the latter can be seen from the $7th$ image (counting from the left) based on the *Alpha Similarity* in Figure 11. In addition, using MSE and SAD as the similarity metrics tend to favor the cases where the foreground area, i.e., the hair region, is smaller, or corresponds to a shorter hairstyle, as can be observed in Figure 8, Figure 10 and Figure 12. On the other hand, when these two metrics produce the worst results, as can be observed in Figure 9, Figure 11 and Figure 13, it is when the hair region becomes larger, or corresponds to a longer hairstyle. However, our proposed metric, i.e., the *Alpha Similarity*, can be shown from the corresponding images, to be less sensitive to this factor, and thus is better. Using such a metric to evaluate the resulting alpha maps, our proposed method can achieve about 86.2% accuracy on an average. More specifically, this accuracy is obtained by taking the average accu-

Table 3: The evaluation of the final alpha. The best result is highlighted with a gray background.

Eval. Func.	Stat.	CostAverage	CostMin	CostSum
MSE	Max	86392	86842	119088
	Min	8843	8844	18053
	Avg	42807	43482	54997
SAD	Max	219	305	478
	Min	21	21	56
	Avg	104	108	155
Alpha Sim.	Max	0.804	0.841	0.957
	Min	0.024	0.024	0.057
	Avg	0.138	0.149	0.349

Table 4: The relationship between the best-10 alpha difference and best-10 final alpha, in terms of the CostAverage cost function.

Picture Type	Evaluation Method	Result Type	1	2	3	4	5	6	7	8	9	10
Alpha difference	Alpha Similarity	best 10	66	24	63	45	31	62	4	26	55	6
Final alpha			30	66	38	12	54	69	49	31	57	28
Alpha difference	Alpha Similarity	worst 10	53	10	9	13	23	39	41	27	16	68
Final alpha			53	9	23	10	13	41	68	5	34	40
Alpha difference	SAD	best 10	44	54	42	39	62	38	25	12	49	14
Final alpha			39	44	54	42	38	25	62	12	49	14
Alpha difference	SAD	worst 10	11	52	7	47	4	17	41	8	64	45
Final alpha			11	52	7	4	47	17	41	64	45	9
Alpha difference	MSE	best 10	39	44	54	42	25	62	38	12	14	23
Final alpha			39	44	54	42	25	62	38	12	49	14
Alpha difference	MSE	worst 10	11	52	47	4	7	45	64	37	58	51
Final alpha			11	52	4	47	7	45	64	37	17	58

racy from all testing images, where each testing image's alpha map is compared against the corresponding *ground truth alpha map* derived from a manually marked trimap using the proposed similarity metric.

The next experiment is to study if a better *alpha difference* would lead to a better *final alpha*. For this purpose, we compare the best 10 alpha differences with the best 10 final alphas, and if a best-10 alpha difference also appears as one of the best-10 final alphas, the cell is highlighted with a gray background, as shown in Table 4, Table 5, and Table 6, where the number in each cell represents the input image index, ranging from 1 to 70.

It is evident that alpha difference is *positively related to* final alpha. However, this relationship is less obvious in terms of best-10 in CostAverage and CostMin cost functions. A possible reason is that the alpha difference still includes the eye regions, and also because final alpha is very similar to the ground truth, therefore the sorting results are affected by the size of the eye regions.

In addition, we also want to observe if alpha difference is definitely worse than the final alpha. This is done by evaluating: $alpha_{difference} - final_{alpha}$ to see if they are all greater than 0. As a result, out of the 70 images, 68 of them are greater than 0, while 2 are less than 0. However, the 2 rare cases are all among the worst-10 side. That is, in general, most of the images satisfy the relationship that $alpha_{difference} - final_{alpha} > 0$.

Table 7: The final alpha statistics for DB outlier with parameters r=0.7 and n=6.

Eval. Func.	Stat.	CostAverage	CostMin	CostSum
MSE	Max	86392	86842	119088
	Min	8843	8844	18053
	Avg	42807	43482	54997
SAD	Max	219	305	478
	Min	21	21	56
	Avg	104	108	155
Alpha Sim.	Max	0.804	0.841	0.957
	Min	0.024	0.024	0.057
	Avg	0.138	0.149	0.349

We also conduct experiments on the parameters of *outlier detection*, in terms of the final alpha statistics using different evaluation methods, as shown in Table 7 and Table 8, and their difference is shown in Table 9.

Table 8: The final alpha statistics for DB outlier with parameters r=0.5 and n=6.

Eval. Func.	Stat.	CostAverage	CostMin	CostSum
MSE	Max	87982	88168	119088
	Min	9480	11137	23121
	Avg	43390	44293	58085
SAD	Max	220	305	478
	Min	23	22	56
	Avg	106	111	168
Alpha Sim.	Max	0.804	0.841	0.957
	Min	0.024	0.025	0.057
	Avg	0.151	0.169	0.388

Through the experiments we found that the result for $r = 0.5$ is worse than that for $r = 0.7$, although a more detailed inspection on every image reveals that the case for $r = 0.7$ is not always better. One interesting finding is that, the case where $r = 0.5$ is better does not show evident visual difference; however, the case where $r = 0.7$ is better is more noticeable.

Table 9: The difference result for Table 7-Table 8.

Eval. Func.	Stat.	CostAverage	CostMin	CostSum
MSE	Max	1590	1326	0
	Min	637	2293	5068
	Avg	583	811	3088
SAD	Max	1	0	0
	Min	2	4	0
	Avg	2	3	13
Alpha Sim.	Max	0	0	0
	Min	0	0.001	0
	Avg	0.013	0.02	0.039

For example, the image with index number 49 is such a case, as shown in Figure 14. The reason we set $n = 6$ is because there are 14 pictures in Figure 3. Assume that from H_1 to H_{14} more than half (7) of the stokes will fall on the hair region, and when some strokes don't fall on the hair region, they will also show some

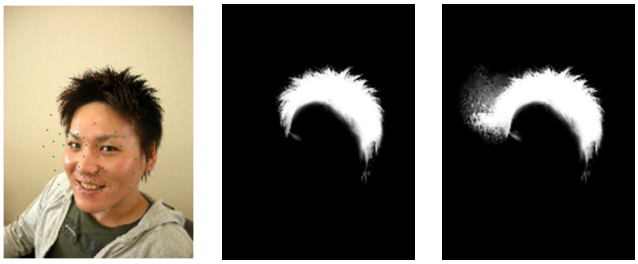
Table 5: The relationship between the best-10 alpha difference and best-10 final alpha, in terms of the CostMin cost function.

Picture Type	Evaluation Method	Result Type	1	2	3	4	5	6	7	8	9	10
Alpha difference	Alpha Similarity	best 10	66	24	63	31	45	62	4	55	26	6
Final alpha			30	66	38	12	54	69	49	31	67	57
Alpha difference	Alpha Similarity	worst 10	42	43	23	9	13	10	39	18	41	27
Final alpha			43	42	9	23	10	13	41	18	5	68
Alpha difference	SAD	best 10	44	54	39	62	38	12	25	26	14	49
Final alpha			39	44	54	38	25	62	12	14	49	36
Alpha difference	SAD	worst 10	43	11	52	7	42	47	4	17	41	64
Final alpha			43	11	52	7	4	42	47	17	41	64
Alpha difference	MSE	best 10	39	44	54	25	62	38	12	14	49	13
Final alpha			39	44	54	25	62	38	12	49	14	23
Alpha difference	MSE	worst 10	11	52	43	4	47	7	45	64	37	58
Final alpha			11	52	4	47	43	7	45	64	37	17

Table 6: The relationship between the best-10 alpha difference and best-10 final alpha, in terms of the CostSum cost function.

Picture Type	Evaluation Method	Result Type	1	2	3	4	5	6	7	8	9	10
Alpha difference	Alpha Similarity	best 10	45	63	37	24	64	66	4	47	65	29
Final alpha			63	45	66	24	4	37	65	69	46	64
Alpha difference	Alpha Similarity	worst 10	42	54	49	39	9	13	38	28	26	23
Final alpha			42	9	54	49	28	26	13	23	39	62
Alpha difference	SAD	best 10	65	44	23	25	39	66	69	12	21	24
Final alpha			65	44	39	25	23	12	69	66	21	22
Alpha difference	SAD	worst 10	42	11	6	49	28	3	7	8	52	51
Final alpha			42	11	6	28	49	8	52	3	7	51
Alpha difference	MSE	best 10	44	39	23	65	25	12	14	13	21	62
Final alpha			44	39	23	65	25	12	13	10	21	69
Alpha difference	MSE	worst 10	42	11	52	6	7	47	4	51	8	28
Final alpha			42	11	6	52	7	47	4	28	51	8

clustering effect, therefore if n is too small then we cannot correctly tell the hair region.



(a) (b) (c)

Figure 14: (a): the face detection result for image 49. (b): the final alpha for $r = 0.7$. (c): the final alpha for $r = 0.5$.

To compare our work with others, we first extract images from the PDF files from others, resize those images uniformly so that the larger dimension of the width and height scales to 200 to be used as input images to our system. The first work we compare with is the one by Rousset et al. [12], and the results are shown in Figure 15.

Compared with [12], it is evident that our results, from Figure 15(a) to Figure 15(e) are worse, and it is due to the mis-detection on the right part of the input image where the dark background bears a similar color to the hair. As our algorithm is composed of multiple closed-form mattings, which is with a degree of fuzziness, the confusion of similar background could lead to erroneous results. To fix this, according to the definition of head in [12], we clip a portion related to the head region from the input image and re-run our system to derive a better result, as shown in Figure 15(f) to Figure 15(i). Note that, although it is admittedly that our result may not be better than the one in [12], their alpha map indeed erroneously includes the eye regions and part of the lower right facial contour of the lady in the image.

The comparison with the second similar work from Lipowezky et al. [8] is shown in Figure 16. Note that as there is no alpha map from the paper, we could only compare with the *hair mask* shown in Figure 16(e). As can be seen from this figure, our system generates a similar and comparable result. Again, our result may not be better than the one in [8], we must point out the testing image here is quite challenging since the involved hair is very *fluffy*, and therefore the extracted hair region is worse than expected. However, for most of the 70 testing images we processed, as can be seen from Figure 8, Figure 10 and Figure 12, where the hairs are not fluffy, our proposed approach performs reasonably well. Also notice that, in Figure 15 and Figure 16, we get to compare with the works from [12] and [8], while each with only one testing result available, and presumably one of their best results, respectively, we instead, provide a more extensive testing results and numerous statistics to show that our proposed method works with an acceptable accuracy. Moreover, we believe our method could be advanta-

geous when dealing with long hairstyles, which, unfortunately, are not presented in both [12] and [8].

The third similar work we compare about is from Chou et al. [1], and the results are shown in Figure 18, where the corresponding testing input images are in Figure 17. As can be observed from this figure, our approach clearly outperforms the one in [1], which applies *K-means* to segment the input image, together with some heuristics to locate the hair region. On the other hand, we propose a more robust way to properly probe the hair region and seek for a better trimap generation, thus leading to a better hair extraction result.

In terms of timing, it normally takes around 8 minutes in terms of performing the closed-form matting (in Matlab) for 31 times, while the rest of the computation can be finished within 5 seconds. However, we must point out that the Matlab code is not optimized, so the performance of our system can be improved either by optimizing the Matlab code or replacing the closed-form matting with other more efficient matting implementation.

Finally, our system is not without limitations. As mentioned before, we assume that the hair portion of an input image cannot touch the boundary of the image, and at least half of the initial hair strokes should fall on the hair region. In general, there are several reasons for the failures or poor results. First, it is because of face detection, as shown in Figure 19(a). Second, it is due to the fragmented alpha maps generated from the closed-form matting algorithm, as shown in Figure 19(b)$\sim$(e). Third, if the background colors are similar to the hair or skin colors, or the hair colors and skin colors are very close to each other, as shown in Figure 19(f)$\sim$(g).

5. CONCLUSIONS

We have proposed an automatic hair extraction system. The contribution of this paper is three fold. First, given a photo, our algorithm can *automatically generate a desired trimap*, and in particular, by removing the unwanted portion of the neck and eyes from the matting process. Second, we propose an *alpha similarity* metric that can better match human vision system in terms of evaluating the quality of the derived alpha map. Finally, we introduce an alpha space that allows us to systematically probe for better parameters used to produce the best matting result. In general, our hair extraction can achieve 86.2% accuracy on an average.

There are still limitations in our work. If the hair/skin color of an input photo is similar to the color of the background, the *ASM face detection* may fail. In addition, the *head matting* and *face matting* mentioned in Section 3.4 could also fail. One possible solution is to combine the color and texture information as in [12], although racial differences should be taken into consideration. Another possibility is to adopt a machine learning approach as in [8], where more work should be involved. In fact, as most effort of our proposed method is spent on generating the hair strokes, the idea of *upper hair shape model*, proposed by Julian et al. [5], may be another possible reference, although how much time needed for the *model fitting* was not mentioned.

In the future, in addition to addressing the aforementioned limitations, extending our work to objects other than human heads is also an interesting direction. For example, clothing or human body are some possible alternatives. Together with this work, we could easily construct a database that contains various clothing and hair styles that suit our needs.

6. REFERENCES

[1] J.-K. Chou and C.-K. Yang. Simulation of face/hairstyle swapping in photographs with skin texture synthesis. *Multimedia Tools and Applications*, 63(3):729–756, April 2013.

[2] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *Computer Vision and Pattern Recognition 2001*, volume 2, pages 264–271, 2001.

[3] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 23(6):681–685, June 2001.

[4] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, Jan 1995.

[5] P. Julian, C. Dehais, F. Lauze, V. Charvillat, A. Bartoli, and A. Choukroun. Automatic hair detection in the wild. In *International Conference on Pattern Recognition 2010*, pages 4617–4620, 2010.

[6] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal - The International Journal on Very Large Data Bases*, 8:237–253, February 2000.

[7] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):228–242, 2008.

[8] U. Lipowezky, O. Mamo, and A. Cohen. Using integrated color and texture features for automatic hair detection. In *Convention of Electrical and Electronics Engineers in Israel 2008*, pages 51–55, 2008.

[9] L. Liu, H. Xu, J. Xing, S. Liu, X. Zhou, and S. Yan. "wow! you are so beautiful today!". In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 3–12, New York, NY, USA, 2013. ACM.

[10] O. Maimon and L. Rokach, editors. *Data Mining and Knowledge Discovery Handbook*, chapter Outlier Detection. Springer, 2005.

[11] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. A perceptually motivated online benchmark for image matting. In *Computer Vision and Pattern Recognition 2009*, pages 1826–1833, 2009.

[12] C. Rousset and P. Y. Coulon. Frequential and color analysis for hair mask segmentation. In *International Conference on Image Processing 2008*, pages 2276–2279, 2008.

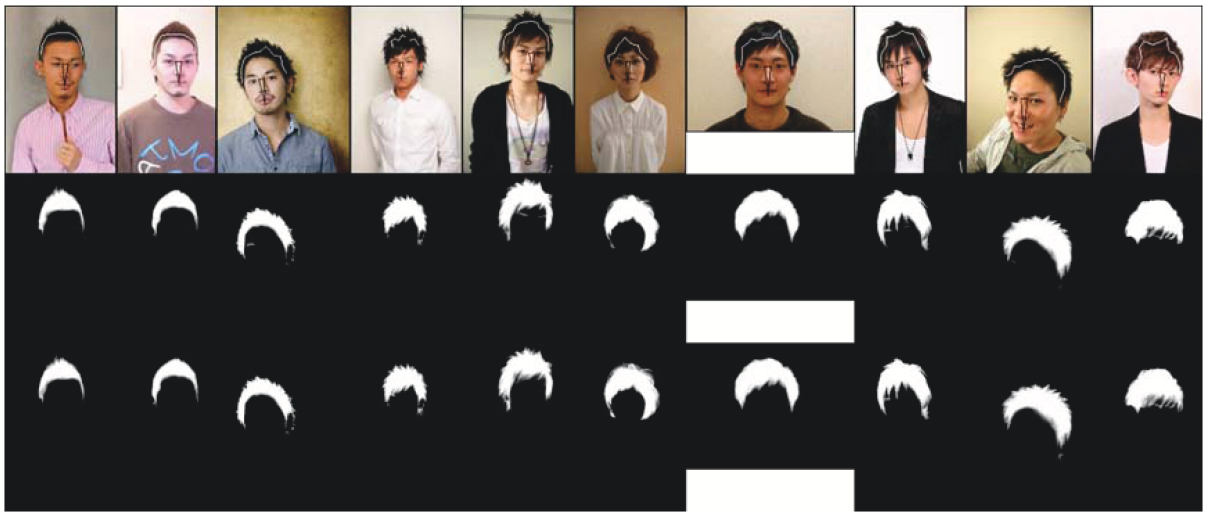
[13] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. Poisson matting. In *ACM SIGGRAPH 2004*, pages 315–321, 2004.

[14] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision And Pattern Recognition 2001*, pages 511–518, 2001.

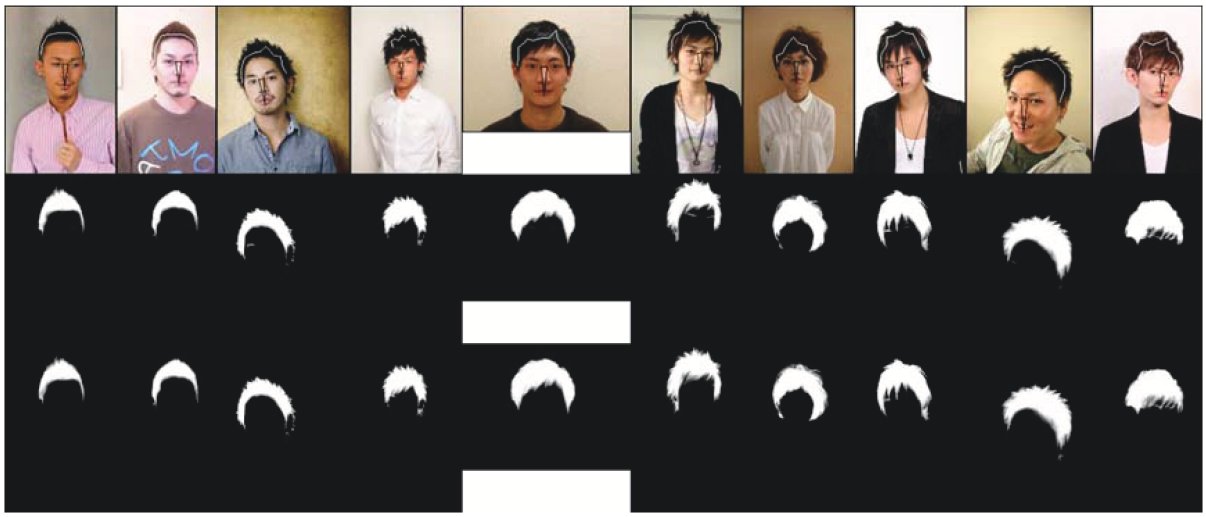
[15] J. Wang and M. F. Cohen. Image and video matting: a survey. *Foundations and Trends in Computer Graphics and Vision*, 3(2):97–175, January 2007.

[16] J. Wang and M. F. Cohen. Optimized color sampling for robust matting. In *Computer Vision and Pattern Recognition 2007*, pages 1–8, 2007.

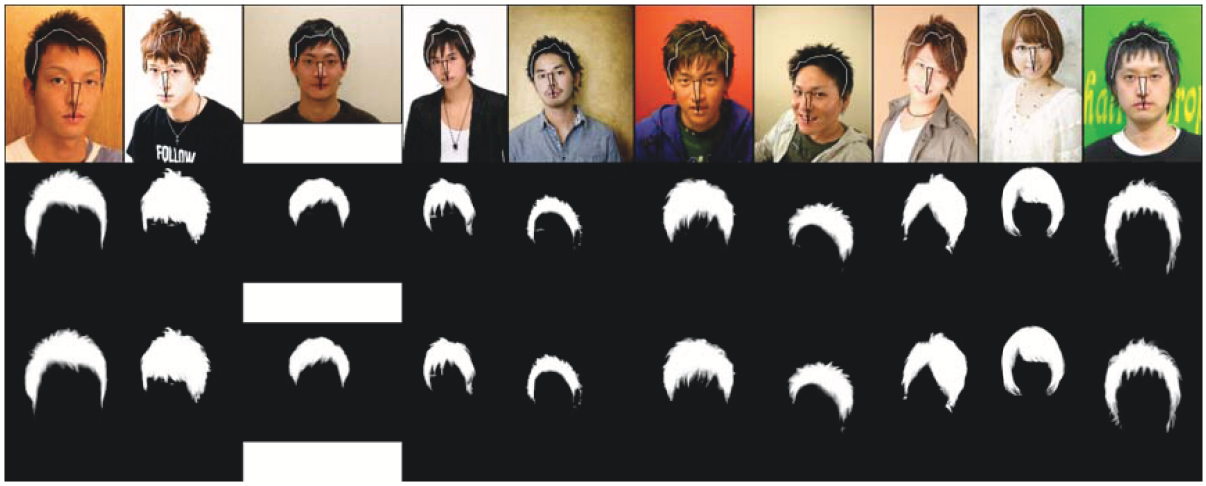
[17] C.-K. Yang and C.-N. Kuo. Automatically extracting hairstyles from 2d images. *Lecture Notes in Computer Science*, 8034:406–415, 2013.



The best 10 photos based on MSE



The best 10 photos based on SAD



The best 10 photos based on Alpha Similarity

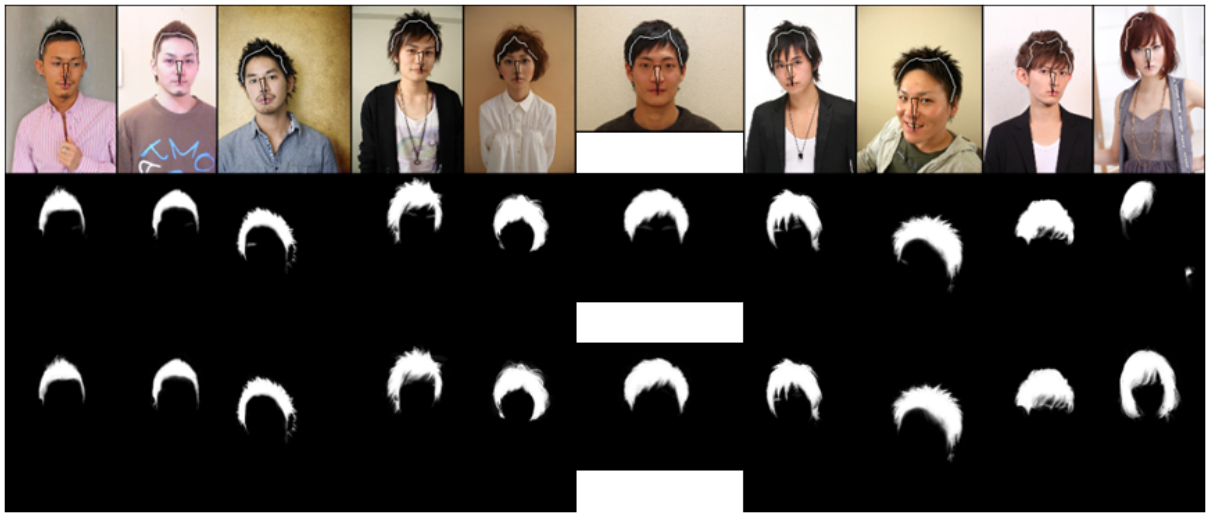
Figure 8: The best 10 photos using CostAverage cost function.

The worst 10 photos based on MSE

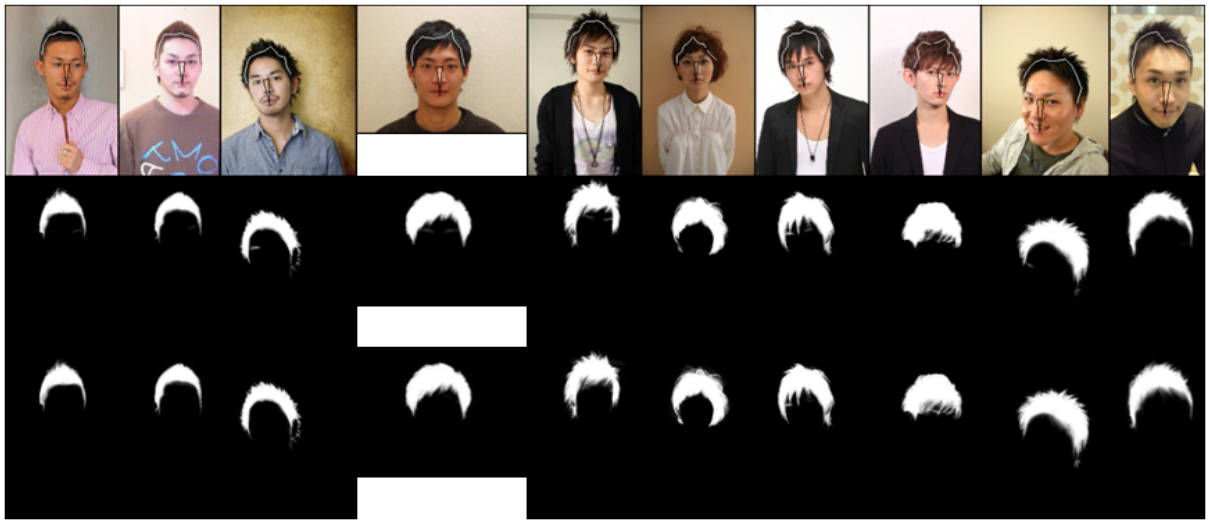
The worst 10 photos based on SAD

The worst 10 photos based on Alpha Similarity

Figure 9: The worst 10 photos using CostAverage cost function.



The best 10 photos based on MSE



The best 10 photos based on SAD

The best 10 photos based on Alpha Similarity

Figure 10: The best 10 photos using CostMin cost function.

The worst 10 photos based on MSE

The worst 10 photos based on SAD

The worst 10 photos based on Alpha Similarity

Figure 11: The worst 10 photos using CostMin cost function.

The best 10 photos based on MSE

The best 10 photos based on SAD

The best 10 photos based on Alpha Similarity

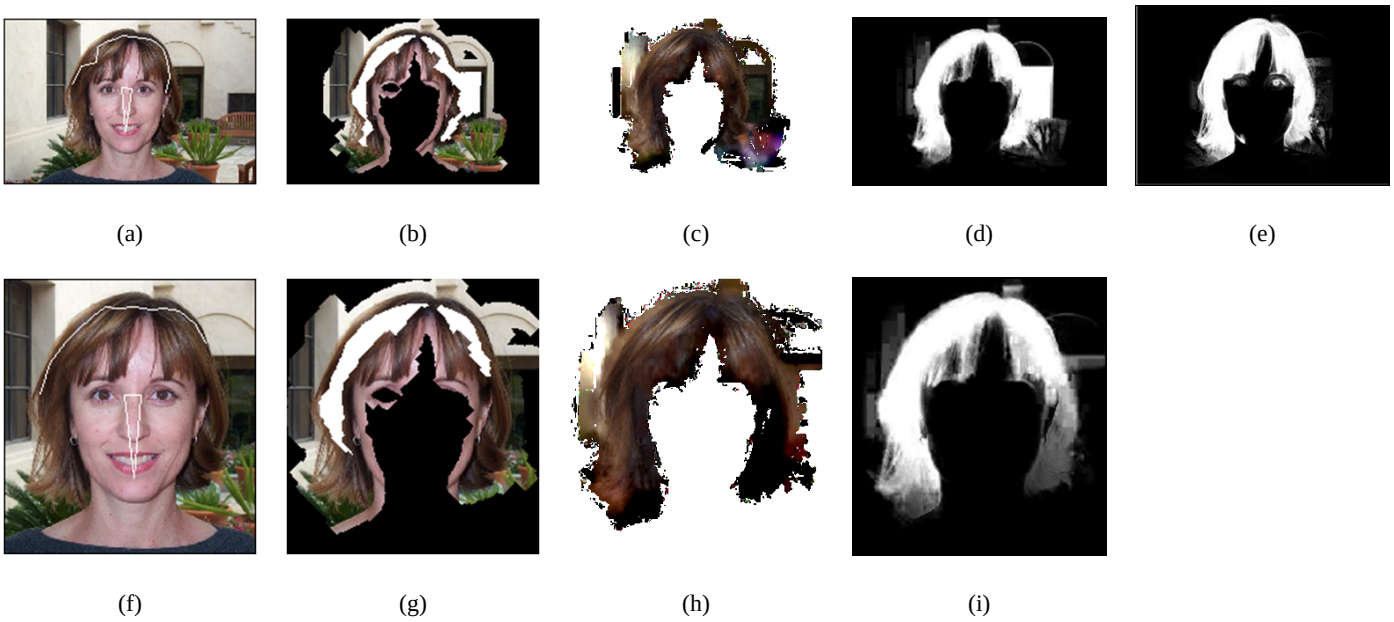
Figure 12: The best 10 photos using CostSum cost function.

The worst 10 photos based on MSE

The worst 10 photos based on SAD

The worst 10 photos based on Alpha Similarity

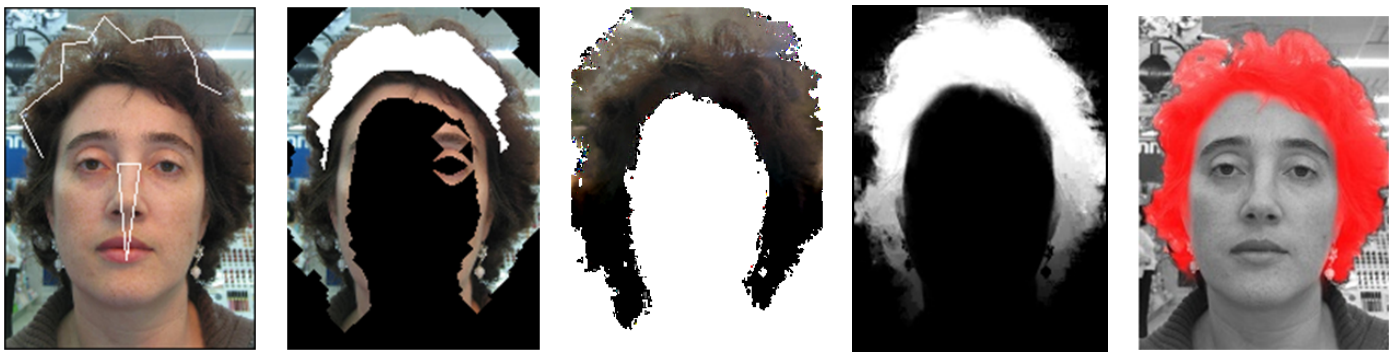
Figure 13: The worst 10 photos using CostSum cost function.



(a) (b) (c) (d) (e)

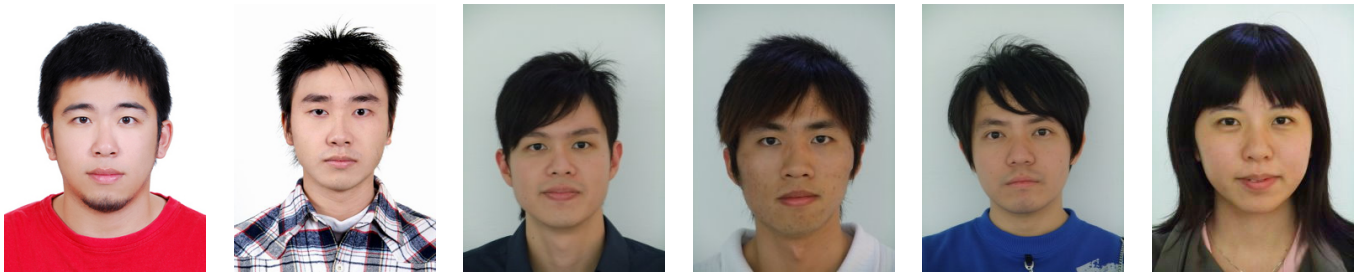
(f) (g) (h) (i)

Figure 15: Comparisons with the work from Rousset et al. [12]. (a):our foreground strokes. (b):our final trimap. (c):our final result. (d):our final alpha. (e):the final alpha from [12]. (f):our foreground strokes on the clipped portion. (g):our final trimap on the clipped portion. (h):our final result on the clipped portion. (i):our final alpha on the clipped portion.



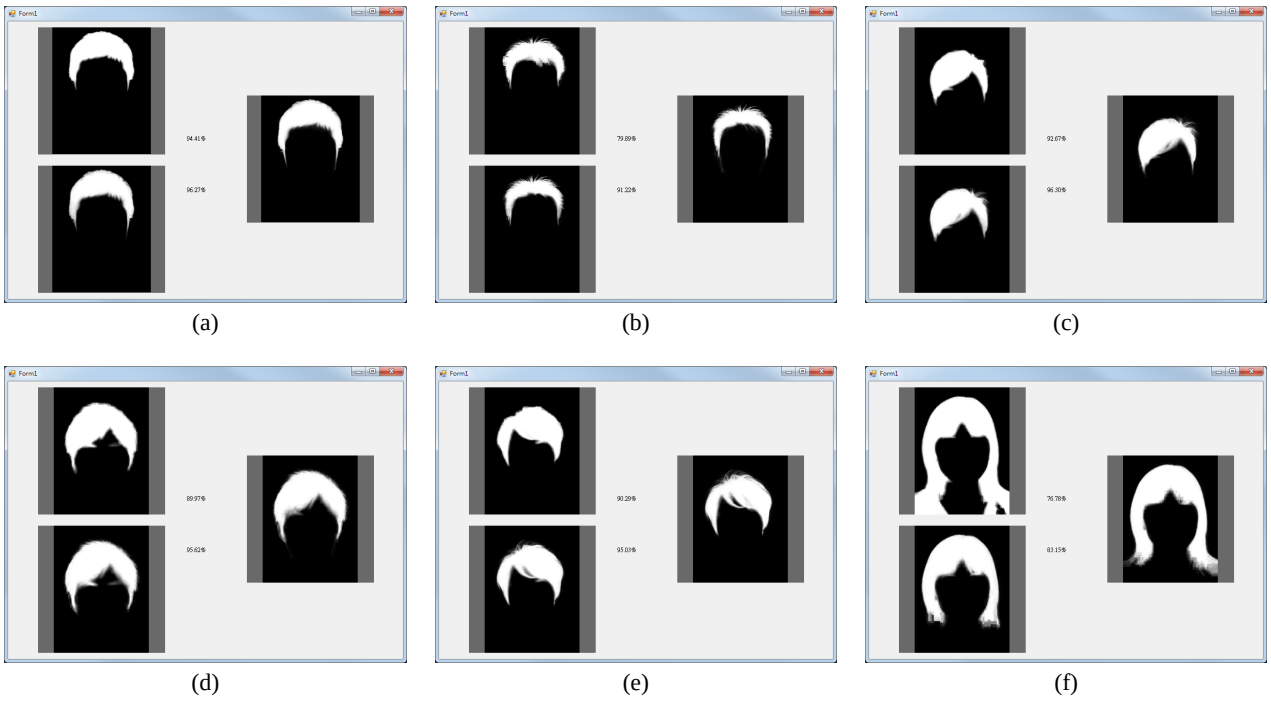
(a) (b) (c) (d) (e)

Figure 16: Comparisons with the work from Lipowezky et al. [8].(a): our face stroke. (b): our final trimap. (c): our final result. (d): our final alpha. (e) hair mask from [8].



(a) (b) (c) (d) (e) (f)

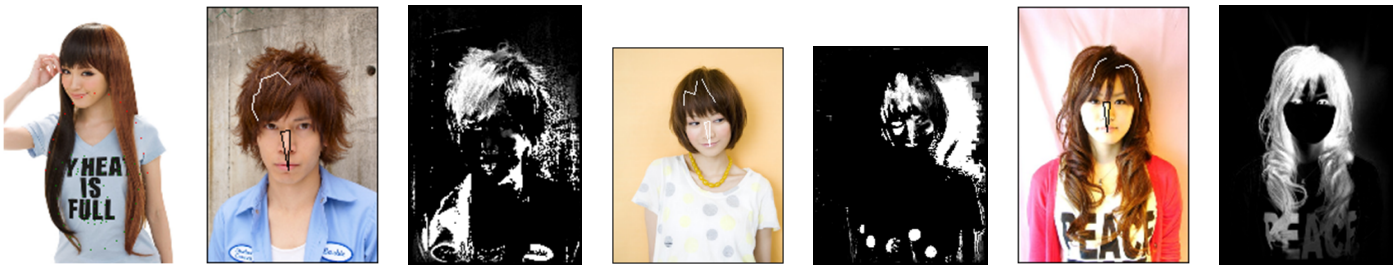
Figure 17: The testing input images used in Figure 18.



(a) (b) (c)

(d) (e) (f)

Figure 18: Comparisons with the work from Chou et al. [1], where in each comparison, i.e, from (a) (f), the one on the upper left is the result from [1] (with the corresponding accuracy shown on its right) , the one on the lower left the result from our approach (with accuracy shown on its right), while the one on the right the ground truth, respectively.



(a) (b) (c) (d) (e) (f) (g)

Figure 19: Limitations. (a):reason 1, face detection failure. (b):reason 2, foreground (white) and background (black boundary) strokes. (c):reason 2, fragmented alpha of (b). (d):reason 2, foreground and background strokes. (e):reason 2, fragmented alpha of (d). (f):reason 3, foreground and background strokes, and note that the word *PEACE* is with similar colors to the hair. (g):reason 3, the final alpha of (f), note that the word has been erroneously included.

Chuan-Kai Yang received his Ph.D. degree in computer science from Stony Brook University, USA, in 2002, and his M.S. and B.S. degree in computer science and in mathematics from National Taiwan University in 1993 and 1991, respectively. He is currently a Processor of the information management department at National Taiwan University of Science and Technology. His research interests include computer graphics, scientific visualization, multimedia systems, and computational geometry.

Chia-Ning Kuo received his Bachelor degree and his Master degree in information management from National Taiwan University of Science and Technology, Taiwan, in 2008 and 2010, respectively. He is interested in computer graphics and multimedia systems, with a particular focus on image matting.