Vision, Video and Graphics (2005)
E. Trucco, M. Chantler (Editors)

Natural Image Matting

Peter M Hillman[†] and John M Hannah[‡]

Institute of Digital Communications, School of Engineering and Electronics, University of Edinburgh, Kings Buildings, Edinburgh EH9 3JL

Abstract

Matte pulling — generating greyscale images which indicate segmentation of images into elements with subpixel accuracy and where blur causes pixels to be a mixture of elements — has received attention in recent years. Many of the algorithms are too slow or too unpredictable to be of practical use in motion picture Post-production. Assessing the performance of different algorithms is also a complex task.
This paper presents an optimisation which can be applied to many algorithms in order to allow them to run at interactive speeds, introduces a new algorithm based on Colour Lines, and presents a technique which can be used as a formal test-bench to measure the performance of matte algorithms.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Image Processing and Computer Vision]: Segmentation

1. Introduction

Many visual digital special effects and post production techniques require an initial foreground/background segmentation of the scene, called a *matte* or *alpha channel*. It might be necessary to insert a real actor into a computer generated scene (as is the case with Virtual Set Technology) or the effect may be more subtle, for example selectively adding a fog effect or a colour correction to the background portion of the image only. To eliminate pixel aliasing artifacts and to handle areas of the image which are blurred or otherwise appear transparent, the matte is greyscale, where values between zero and one indicate that the background and foreground are blended together. The film industry refer to the process of generating this segmentation as *matte pulling*.

Currently the most reliable way of matte pulling is to force the background to be a blue screen [Pet77]. This is not a problem for virtual sets, where an entirely new background is to be used, but if the matte is only required to process two parts of the image differently, using a bluescreen may require two separate passes. The first pass uses a bluescreen and is used to capture the foreground. A second pass is then filmed to capture the background and the two passes are composited together. This is an unnatural and time–consuming process. If such a matte-pass has not been acquired, the most reliable matte pulling technique is to generate the matte by hand, by "rotoscoping" the image – drawing a blurred edge around the foreground in key frames, and using geometrical interpolation to compute the matte for intermediate frames.

There has recently been interest in the generation of mattes from images without the need for a special matte pass — so called *Natural Image Matting*.

This paper is organised as follows: In section 2 we review published natural image matting systems. In section 3 we propose an optimisation which can be applied to many natural image matting techniques. In section 5 an adaptation to our previous work — based on the Colour Lines algorithm proposed by Omer and Werman [OW04] — is proposed. The lack of quantitive performance measures for natural image matting techniques has inspired the design of a formal matte algorithm test-bench, which is described in section 6.

2. Natural Image Matting Systems

These techniques have much in common - they work with an input image, called by some authors a Hint Image and others a Trimap, which indicates areas of known background and foreground, and has a band of pixels between the two

[†] Now with Square Eyes Software
[‡] This work was supported by EPSRC Platform Grant GR/S06578/01

— the unknown area — which must be processed in order to generate the final segmentation. For sequences with significant motion, a separate hint image is required for each frame. Chuang *et al* [CAC*02] use optical flow to update the images, Hillman *et al* use a colour difference approach [HHR01] to update the hint image. Techniques classify each pixel in the unknown area by sampling one or more pixels from each of the known areas, and by comparing these sampled clusters to the pixel under classification. For each pixel a separate estimate of alpha is required which indicates the amount of blending between foreground and background. Where alpha is between zero and one, it is also necessary to estimate, at the very least, the *clean foreground colour*. If the element is to be placed over a new background, all traces of the original background must be removed from the pixel otherwise it will appear as a soft fringe around the outside of the element. If the original background is to be used in the final sequence, the *clean background colour* must also be computed. If the clean foreground colour and alpha are both known, it is trivial to compute the clean background colour using the blended pixel sampled from the source image by inverting the *compositing equation*

$$P = \alpha F + (1-\alpha)B \quad (1)$$

$$B = \frac{P-\alpha F}{1-\alpha} \quad (2)$$

where P is the observed (source) image, F is the clean foreground image, B is the clean background image and α is the matte image.

The biggest difference between algorithms is in the alpha estimation step. Ruzon and Tomasi [RT00] split the background and foreground clusters into subclusters, and test every alpha value between zero and one, searching for an optimal value using a measure based on the probability distribution interpolated between background and foreground. Chuang *et al* [CBSS01] use a similar approach, but sample every foreground/background subcluster pair in turn and use a Maximum Likelihood approach to find the best alpha value. This is a less computationally intensive approach. Even less computationally intensive is the Principal Axis algorithm [HHR01] which is based on the observation that, in RGB space, the clusters form long cigar-shaped prolates, and uses the axis along which the cluster is formed — the principal axis. A more detailed explanation of this algorithm is given in Section 4.

Some techniques require a separate pass, but of the background only, and use in effect an advanced form of frame differencing to pull the matte. This approach was originally proposed by Wexler *et al* [WFZ02] and expanded by Apostollof *et al* [AF04].

Sun *et al* [JJCH04] take an entirely different approach. They use the nearest known background and foreground points as the initial estimates for the clean colours and refine these estimates along with the estimate alpha channel by solving Poisson equations. Their technique, however, does not generate clean foreground colours and it is not-trivial to adapt it to do so, which makes it of limited practical use for motion picture applications.

3. Fast cluster collection

Motion Picture post-production requires software to work in one of two ways. Either the software must work perfectly all the time without any user intervention or interaction, in which case run-times can be arbitrarily long. Hundreds or thousands of high performance processors are available in *render farms*, so run-times of several hours per frame for a frame-parallel algorithm are manageable as the process can be scheduled to run overnight. If the software requires interactive input and/or does not necessarily produce acceptable results, then the software must operate quickly. Image processing software that requires almost as much user interaction as a manual approach and sometimes produces results which are not much better is of no interest — it is better to "play it safe" and use a manual approach at all times. Unfortunately, many natural image matting algorithms are too slow, too unreliable and require tweaking of parameters or adjustment of hint images to be of practical interest to Motion Picture Post-production companies.

This section proposes a solution to this problem in the form of a hybrid solution. Cluster collection is one of the most time-consuming part of algorithms such as Chuang *et al*'s Bayesian matting approach, and the Principal Axis and Colour Lines algorithms (see sections 4 and 5). The solution presented here speeds up cluster processing significantly, but is likely to introduce small artifacts. Thus, operators can use the fast version to pull a rough matte at interactive speeds. They can then test whether the results will be suitable by compositing them into the final shot. Once they are satisfied that the result will be suitable, the speed optimisation can be removed to achieve more accurate results in a longer run. In many cases, the fast result may be deemed suitable.

We have experimented with different selection algorithms and have concluded that sampling pixels from previously classified results and from pixels within a certain distance of the edge of the known area is the most effective technique. Allowing the system to sample pixels some distance away from the border of the unknown region makes it harder to analyse the cause of poor results.

Collecting the pixels involves searching in a square-spiral pattern outwards, checking whether pixels are suitable for collection and adding them to clusters accordingly, until enough pixels have been sampled.

Our speed acceleration is similar to that proposed for sequence segmentation in [HHR03]. It uses a courser grained spiral and relies on a structure called an *ImageBlockArray*. For an image of dimensions $(W \times H)$ we create a $(W/S \times H/S)$ array. Each element contains lists of the known foreground colours and colours for the corresponding $S \times S$

block of the corresponding frame of the sequence. Once the ImageBlockArray has been generated, gathering pixels to form the sets S_F and S_B only requires a series of `memcpy` operations from the appropriate element in the ImageBlockArray. Blocks are visited in a square-spiral fashion, and the copying operation continues until both foreground and background clusters are full.

Clean colours computed from previous results can be appended to the appropriate lists in the array after classification of each pixel, so that future pixels can be estimated using values from previously computed results, which makes the classification more resilient to backgrounds and foregrounds which change gradually within the unknown area.

The penalty of using an ImageBlockArray is the quantisation of clusters: a group $S \times S$ pixels will use almost identical blocks, but the next group will use a different set of blocks. Pixels on the edge of two blocks will therefore have significantly different blocks, which may result in a visible edge.

It would be possible to accelerate the processing further by computing the clusters only once and using the cluster for each pixel in the $S \times S$ block. However, this does not allow the use of clean colours estimated from some pixels in the block to be used in others and we have found that the actual speed-up is not significant. The results presented here do not use this additional optimisation.

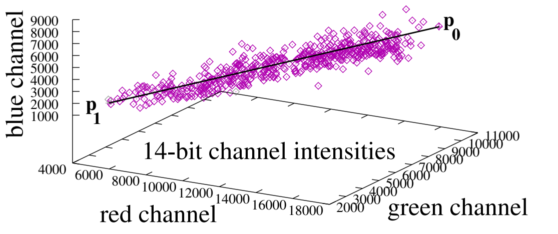

Figure 1: *Cluster of points in RGB space with the line* $\overline{p_0p_1}$

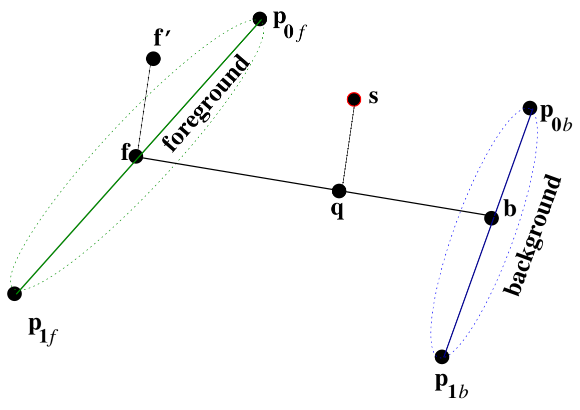

Figure 2: *Position of points used to classify in colour space*

4. Principal Axis matting

In this section, we briefly summarise the Principal Axis algorithm presented in [HHR01]. Let s be the pixel under classification. The algorithm assumes that the clusters tend to form **prolates** (cigar shapes) when plotted in RGB colourspace. Using Principal Components Analysis, we find the principal axis of each cluster. That is, we find a line $\overline{p_0p_1}$ through the centre of each cluster. The length of the line is limited by the range of the cluster. The foreground f and background b colours are the closest points on each of these lines to the point s, as shown in Fig. 2. Point q is the nearest point on the line $\overline{bf}$ to s. Alpha is then given by the ratio of the length $\overline{bq}$ to the length $\overline{bf}$. Final clean foreground and background colours f' and b' (not shown in Fig. 2) are computed by adding the vector $\overline{qs}$ to points f and b. This ensures that compositing the clean foreground and background points together using α exactly reproduces s. The input must be exactly recreated when compositing the element over the computed clean background.

5. Colour Lines

Colour Lines were originally proposed by Omer and Werman [OW04] as a distortion free compact colour representation system. Here, we adapt the technique and apply it to cluster representation.

In RGB space, Colour Lines operate as follows: all the colour samples in the source (*e.g.* the input image) are divided into bands based on their luminosity. Within each band, the normalised colour for each pixel $(\frac{r}{r+g+b}, \frac{g}{r+g+b})$ is computed, and these values form a 2D histogram. This histogram image is smoothed, and maxima found within it. These maxima are then transformed back to the original colourspace, and joined together (if they are close enough) to form the colour lines. Thus, colour lines connect separate regions of dense colour within each band.

In our system, we are using colour lines to approximate small clusters, and wish to have an algorithm that operates extremely quickly. The Principal Axis algorithm is a poor cluster model where the clusters are bent or bimodal. Colour lines can reduce this problem. In most cases, the clusters are small enough that there will be only one line passing through each band. Therefore, we can reduce the computational complexity of the colour line generation algorithm. Our algorithm is defined below

To form a colour line with B bands from input set S where $|S| = n$:

```
1. Compute Means:
   a. for $x$ $(0..B)$ Total$(x) = 0$
   b. for $x$ $(0..B)$ Counts$(x) = 0$
   c. for $x$ $(0..n)$ Lum$(x) = \|S(n)\|$
   d. for $x$ $(0..n)$
```

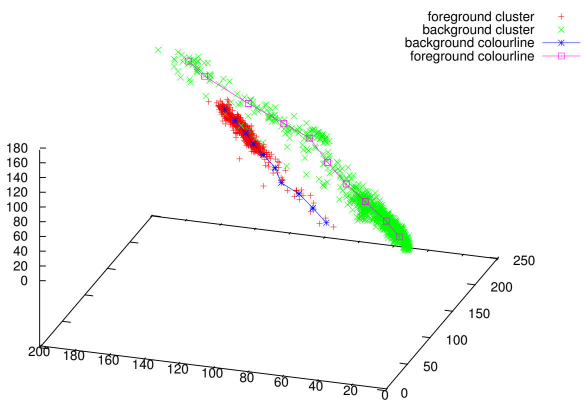

Figure 3: *Foreground and background clusters approximated by colour lines. The axes are the ordinates of RGB colourspace*

```
            i. Bucket=$\frac{Lum(x)-min(Lum)}{max(Lum)-min(Lum)}$
           ii. Total(Bucket) += $S(x)$
          iii. Counts(Bucket) +=1
      e. for $x$ (0..$B$) Means($x$)=Total($x$)/Counts($x$)
2. Connect Lines:
      a. for $x$ (1..$B$)
            i. if $\|\text{Means}(x)-\text{Means}(x-1)\| <$ threshold
                  A. Connect Means($x$) and Means($x-1$)
```

Fig. 3 shows an example of foreground and background clusters and the colour lines used to approximate. In this case the lines are unbroken; this is not always the case.

5.1. Processing using colour lines

Once colour lines have been formed, processing proceeds exactly as before. The initial clean foreground and background colour estimates f and b are the nearest points on the foreground and background lines. These are found by taking the nearest point on each section of the line to s, the point under classification, and finding the closest of these.

5.2. Results

Figure 4 shows the results of processing the *Gema* image with the Colour Lines algorithm. This image is 1612×1673 pixels, typical of motion picture resolution data. There is little contrast around the top of the head, which causes difficulty extracting individual hairs. Some of the highlights on the left hand side are too close to the background colour and appear as holes. Details of alpha channels produced by the

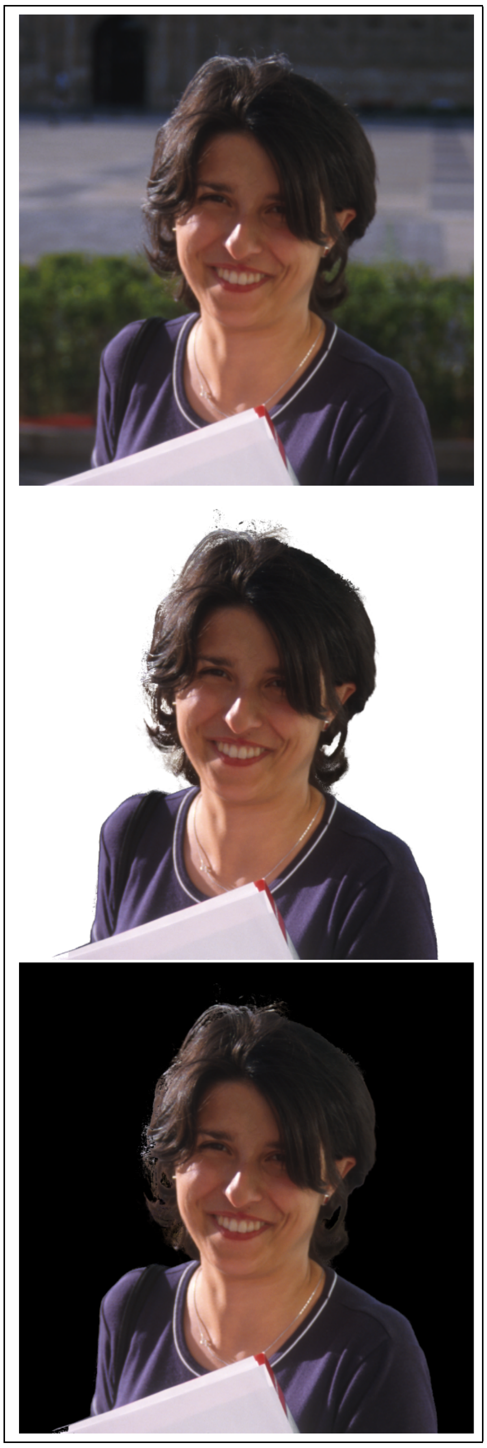

Figure 4: *Results with of applying the* ***Colour Lines*** *algorithm to the* Gema *image (top), shown over white and black backgrounds*

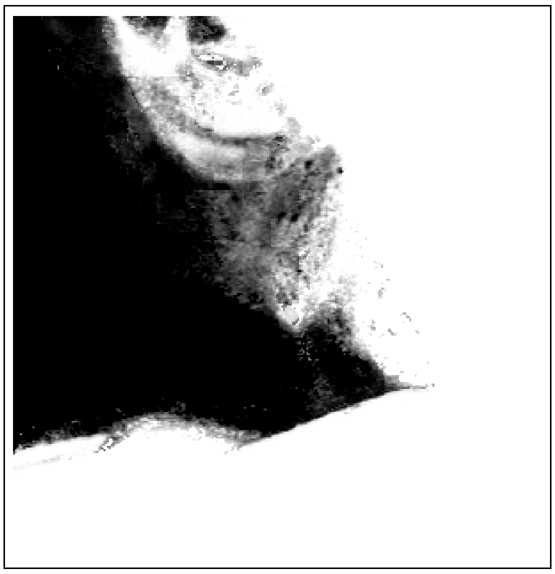

Figure 5: *Detail of alpha channel produced for the* Gema *image using the* ***Colour Lines*** *algorithm*

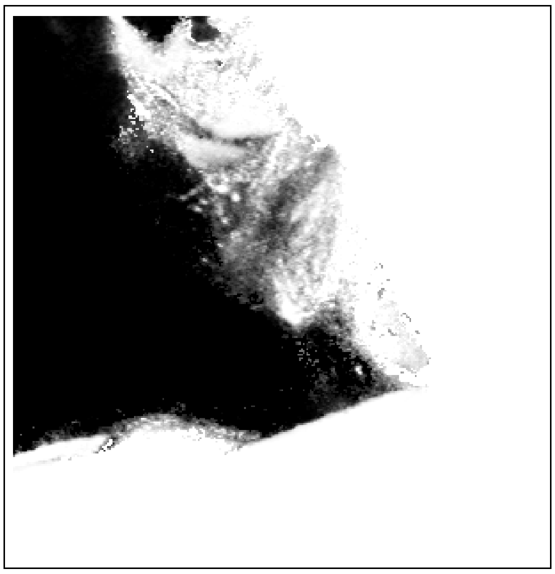

Figure 6: *Detail of alpha channel produced for the* Gema *image using the* ***Principal Axis*** *algorithm*

colour lines algorithm and the Principal Axis algorithm are shown in Figs. 5 and 6 respectively. Whilst both algorithms appear to have handled this complex area relatively well, the Principal Axis technique produces an alpha channel that appears more blotchy. A line is visible at the very top of the Colour Lines result, which is caused by the blocking problem described in section 3.

6. A Quantitive performance analysis test-bench for matting systems

Judging the quality of a matte is normally highly subjective. Artists working in post-production will simply use the matte to composite a scene and judge the matte to be successful if it "looks right" [SB96]. Often, a fairly poor quality matte may be sufficient (for example if the element is to be re-composited over the background after some processing).

In this section, we propose a quantitive measure of the success of matte pulling. We use images which are available as standard with a "ground truth" alpha channel to produce quality analysis. A similar technique was published by Hillman *et al* [HHR04] which measures the accuracy of the alpha channel alone, compared to the ground truth. However, it is also important to measure the quality of the clean foreground, and to do so in conjunction with the quality of the alpha channel. We can achieve this by generating a composite image from both the ground truth and the proposed automatic image and comparing these. However, the result is then dependent on the background: this technique can hide correlated errors in the alpha channel and the clean foreground. Therefore, we measure errors using two different backgrounds, black and white as follows:

1. Form composited images with black and white backgrounds for both the estimated results and the ground truth results.
2. Let RMSE_w and RMSE_b be the RMSE difference in CIE-Lab colourspace between the two white background images and the two black background images respectively.
3. $\mathrm{PSNR} = 20\log_{10}\frac{255}{2(\mathrm{RMSE}_w+\mathrm{RMSE}_b)}$

We have used a standard image sequence to test our algorithms. We take the NTSC Akiyo_foreground sequence available from, for example, the CityU-VIL Image Database [Cit] (see http://tinyurl.com/4dxfb) and composite it over the bottom left hand corner of the $512x512$ Lena image, scaling the image by $\frac{45}{32}$ to ensure the width is the same. We use the first 31 frames of the sequence. Fig. 7 shows an example frame. To eliminate problems with generation of multiple hint images, the same hint image (Fig. 8) is used for all 31 frames (there is sufficiently small movement in these frames for a single hint image to suit all the initial frames). The top portion of the image is lit from behind. This backlighting is extremely common, especially in virtual set scenarios, and many algorithms perform poorly. We have applied the backlighting optimisation described in [HHR03] to

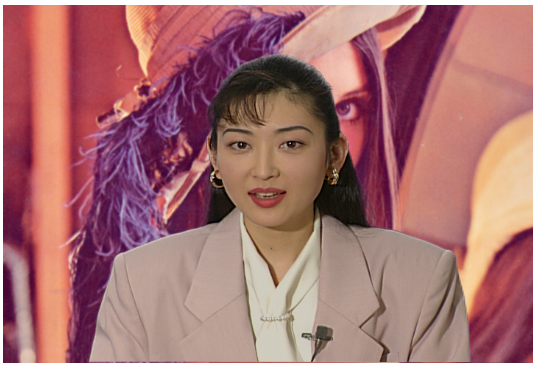

Figure 7: *Example frame (frame 23) from* Akiyo/Lena *sequence*

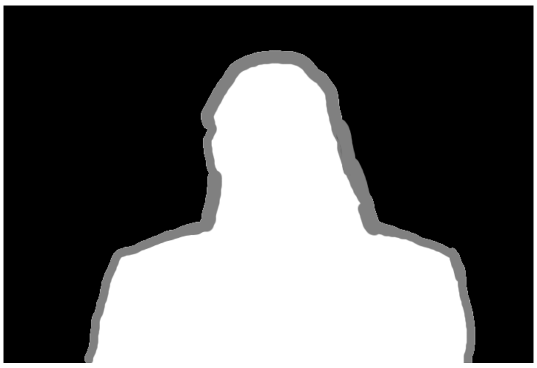

Figure 8: *Hint image used to process the Akiyo/Lena sequence*

handle this situation. Since backlighting only effects the top 175 rows, we give figures for the whole image and for the non-backlit bottom 311 rows separately.

Figs 13 and 14 graph the PSNR results over the whole image sequence. Results over the entire image sequnce are shown in Table 1. "Baseline" results are those obtained by using the input hint image as the output alpha channel and the input image as the output clean foreground. Thus, they show the performance of a simple algorithm which does no processing. Unlike the other results, the baseline performance depends on the drawing of the hint image.

The graphs show an important characteristic — the frame–to–frame difference in PSNR. Techniques whose PSNR values tend to fluctuate between frames tend to give a more flickery results, which is highly undesirable. With this in mind, the best overall performance — both by qualitive observation and by analysis of the graphs — appears to be the Colour Lines algorithm using backlight estimation. Using an ImageBlockArray also means that results are available fast enough to make the technique practical: on the Akiyo/Lena sequence a non-optimal implementation us-

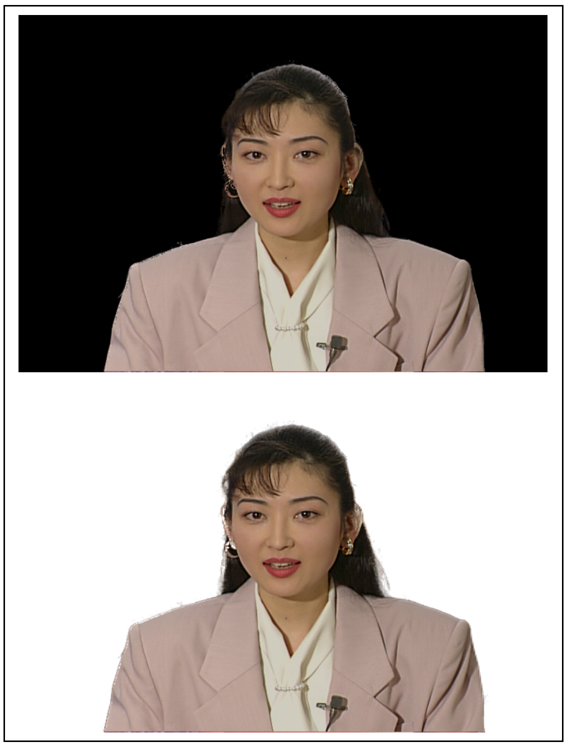

Figure 9: *Result of processing frame 23 of the Akiyo/Lena sequence with the* **Colour Lines** *algorithm using backlight computation*

	PSNR (dB)	
Algorithm	Non-backlit	whole image
Principal Axis w/Backlighting	37.45	38.58
Colour Lines w/Backlighting	37.68	38.82
Principal Axis	37.80	38.48
Colour Lines	38.17	38.54
Chuang *et al*	35.93	36.30
Baseline	29.95	30.29

Table 1: *Results for Akiyo/Lena sequence: performance averaged over sequence*

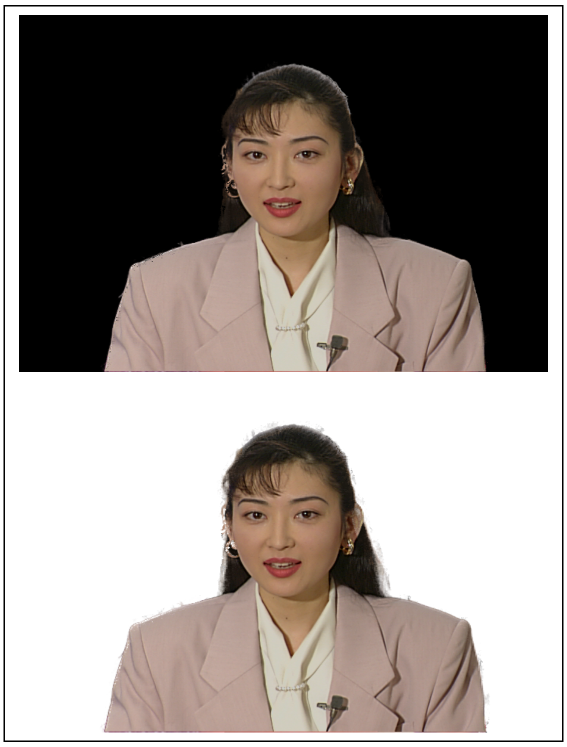
Figure 10: *Result of processing frame 23 of the Akiyo/Lena sequence with the* ***Principal Axis*** *algorithm using backlight computation*

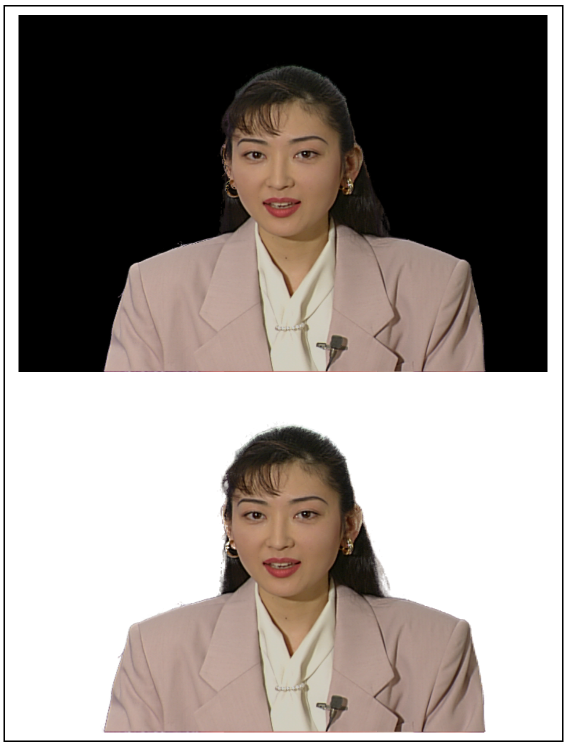
Figure 11: *Result of processing frame 23 of the Akiyo/Lena sequence with the* ***Colour Lines*** *algorithm* without *backlight computation*

ing an ImageBlockArray with $S = 16$ takes approx 22 seconds per frame overall including file loading and saving on a 2GHz Xeon.

7. Conclusions and future work

In this paper we have presented an optimisation which can be applied to many natural image matting techniques. By allowing operators to see the results of processing images more quickly, it makes natural image matting more appealing to motion picture post-production. We have also developed a high speed algorithm which is an alternative to the Principal Axis algorithm and in some cases outperforms it. We have also attempted to provide a methodology for testing matting algorithms using publically available image sequences. This technique allows comparison of different algorithms and assesment of their performance. Using this technique we have measured the performance of different algorithms and shown that the Colour Lines approach seems to outperform other published algorithms.

Our testbench does not measure perceived error. For example a hair which is continuously missed from the alpha channel is better than one which "flicks" in and out of the matte in subsequent frames. Future work will attempt to address this problem, and extend the testbench to test different approaches to updating the hint images between frames.

References

[AF04] APOSTOLOFF N., FITZGIBBON A.: Bayesian video matting using learnt image priors. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2004)* (2004), pp. 407–414.

[CAC*02] CHUANG Y.-Y., AGARWALA A., CURLESS B., SALESIN D. H., SZELISKI R.: Video matting of complex scenes. *SIGGRAPH 2002 (ACM Transactions on Graphics) 21*, 3 (July 2002), 243–248.

[CBSS01] CHUANG Y.-Y., BRIAN C., SALESIN D. H., SZELSIKI R.: A bayesian approach to digital matting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR* (9th-14th December 2001), vol. 2, pp. 264–71.

[Cit] CITY UNIVERSITY OF HONG KONG: Cityu-vil image database. `http:// abacus.ee.cityu.edu.hk/ %7Ebenjiman/imagedb/`.

[HHR01] HILLMAN P., HANNAH J., RENSHAW D.: Al-

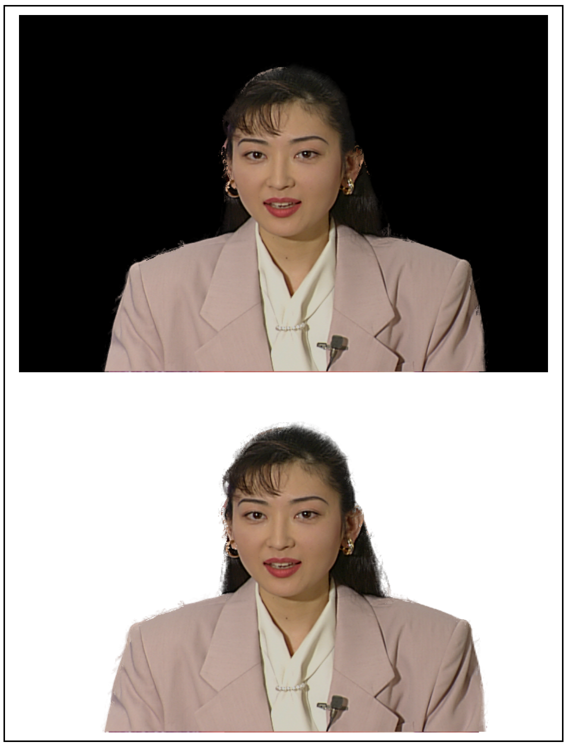

Figure 12: *Result of processing frame 23 of the Akiyo/Lena sequence using the algorithm by* ***Chuang* et al**

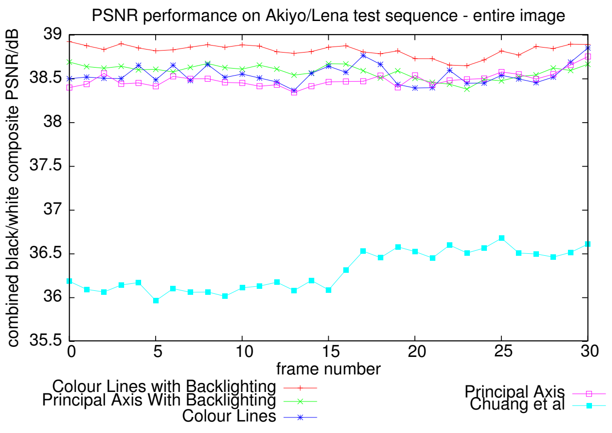

Figure 13: *Results for Akiyo/Lena sequence using the entire image*

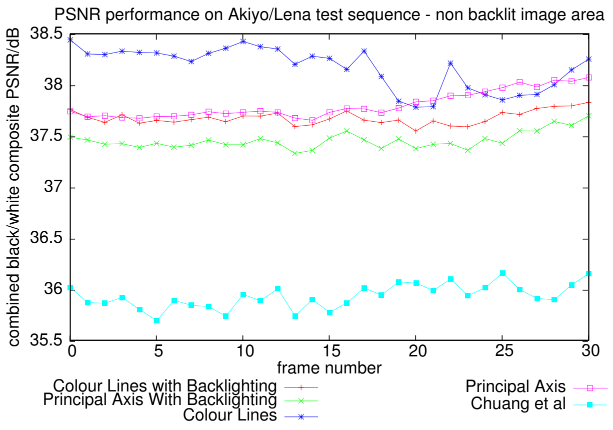

Figure 14: *Results for Akiyo/Lena sequence using the non-backlit image area*

pha estimation in high resolution images and image sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR* (9th-14th December 2001), vol. 1, pp. 1063–68.

[HHR03] HILLMAN P., HANNAH J., RENSHAW D.: Segmentation of motion picture images. In *IEE International Conference on Visual Information Engineering, VIE* (July 2003), pp. 97–101.

[HHR04] HILLMAN P., HANNAH J., RENSHAW D.: An improved algorithm for segmentation of motion picture image sequences. In *1st European Conference on Visual Media Production (CVMP)* (2004), pp. 175–84.

[JJCH04] JIAN SIN, JIAYA JIA, CHI-KEUNG TANG, HEUNG-YUENG SHUM: Poisson matting. *SIGGRAPH 2004 (ACM Transaction on Graphics) 23*, 3 (April 2004), 315–321.

[OW04] OMER I., WERMAN M.: Color lines: Image specific color representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 04)* (June 2004), vol. II, IEEE, pp. 946–953.

[Pet77] PETRO VLAHOS: Electronic composite photography with colour control. United States patent number 4,007,487, February 1977.

[RT00] RUZON M., TOMASI C.: Alpha estimation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (June 2000), vol. 1, pp. 18–25.

[SB96] SMITH A. R., BLINN J. F.: Blue screen matting. *Computer Graphics: Proceedings of the ACS* (1996), 259–268.

[WFZ02] WEXLER Y., FITZGIBBON A., ZISSERMAN A.: Bayesian estimation of layers from multiple images. In *Proceedings, ECCV* (2002).