# INTERACTIVE SEGMENTATION OF IMAGE SEQUENCES WITH NEAR-STATIC BACKGROUNDS

**Peter Hillman, John Hannah**

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

**Keywords:** Segmentation, background removal, colour classification

**Abstract**

Segmentation of image sequences is a major and time consuming part of digital special effects production. Where the sequence is too complex to segment using fully automatic techniques it is often segmented entirely by hand. This paper presents a fast semi–automatic technique which allows interactive segmentation of image sequences with near–static backgrounds. An operator initially creates a very approximate segmentation of the scenes. A foreground probability map is automatically generated by using mixture models which represent known foreground and background colour distributions. The operator then iteratively refines the initial segmentation by thresholding the generated probability maps until a suitably accurate segmentation has been achieved. Straightforward sequences can be segmented with very little human interaction: significant effort is required only on the most challenging sequences. Results are presented which demonstrate the use of our system on a variety of image sequences.

## 1 Introduction

Visual media post-production frequently requires segmentation of image sequences to separate an object from a static background. Often, the camera is also static (or else the sequence can be stabilised in software to create the same effect). However, the observed background may not be completely static. Trees blowing in the wind or even sensor noise will cause the pixel values in the background to change over time.

A number of different approaches exist for segmentation with a truly static background. If there is a reference background image then thresholding the absolute difference image is often used. This is often described as the "background difference" approach. Where the true reference background is not known, it can be estimated. A median, low pass [6] or Kalman [1] filter can be used on the original sequence to estimate the background (the latter two filters can also allow the background to evolve over time to permit lighting changes). Other segmentation techniques can be considered to be special cases of the background difference approach. The lumakey method simply takes the intensity of the foreground image as the threshold, which is equivalent to the background difference approach with a black background. Chromakey (bluescreen) algorithms [7] assume the background is uniformly coloured with a known colour.

Where the background is subject to flicker, techniques which assume the background is unchanging will tend to identify large areas of background as changing and therefore classify these areas as foreground. Stauffer and Grimson [8] model near-static backgrounds using an on-line K-means algorithm to build and update a mixture model for each pixel. This model represents the value that each pixel takes over time when it is background. If in a new frame the pixel value is sufficiently unlike any previous background value, it is considered to be foreground. Raja *et al* [5] take a similar approach, but model the foreground instead of the background, and present an adaptation to allow tracking of the foreground object.

If the background is static or nearly static, there will only be a small motion in the background areas compared to the foreground. Many approaches (*e.g.* [2]) use motion to help with segmentation. However, we have found that motion estimation is too slow and/or too inaccurate, where there is significant inter-frame movement.

Tsai *et al* [10] present a technique for interactive segmentation by using motion estimation to locate manually corrected pixels in neighbouring frames of a sequence. Interactive algorithms are well suited for use in the post-production environment where the restrictions of operating in real time on live data need not apply. Therefore frames can be processed using information from later frames and human interaction with the system during processing may be acceptable. Currently in post-production the speediest and most reliable approach to segmentation is usually for the compositor to build a reference background image by hand from a selection of frames, difference each frame with this image using a threshold tuned for each frame, and then hand edit the result to produce the required segmentation. In the worst case, it may be necessary to use an accurate *rotoshape* to segment manually the entire sequence. This is a closed spline curve defined by control points which are set in reference or keyframes. The shape of the curve for intermediate frames is obtained by interpolating the control points between these keyframes. Drawing an accurate rotoshape can be a laborious process: a sequence of 100 frames can take half a day or more of work to produce a suitable result.

In this paper, we present a semi-automatic solution by allowing some human interaction with our algorithm. We begin by drawing two rotoshapes which only mark the approximate position of the foreground and background in each frame of the

sequence. These are similar to *garbage mattes* which are drawn to eliminate unwanted objects from automatically generated segmentation results. We anticipate that an experienced compositor would be able to generate a suitable pair of rotoshapes in a few minutes for a typical 100 frame sequence. In our technique, areas marked as foreground and background are used to form mixture models of probable foreground and background colours. The probability that each pixel belongs to the background or foreground is estimated using these models. The operator can then interactively select thresholds, either to provide a segmentation of the scene directly, or else to generate refined rotoshapes for subsequent iterations of the process in order to converge toward an acceptable segmentation of the scene. The final probability map is used as an alpha or matte channel for future compositing.

This technique is particularly attractive for interactive segmentation because it allows scalability of effort: for a very simple sequence (such as a uniformly coloured object on a uniformly coloured background) only a very small amount of human effort is required. For more complex sequences, more effort can be employed and a better result achieved. Fully automatic techniques tend either to work well or to fail completely, whereas manual techniques always require effort even in simple cases.

We have previously used a similar technique for segmentation of single images. This uses a hand edited *hint image* to indicate areas which are definitely foreground and background. In [3] we presented an algorithm for generating these images automatically for a sequence given one or two hand-segmented frames. That algorithm also used colour distributions sampled from known foreground and background areas to classify pixels, but only took pixels from single frames. The work presented here improves upon these results and operates faster by assuming a near-static background and requiring a marking of known foreground and background for the entire sequence rather than just one frame. By analogy with the term hint images we have used previously [3], we shall refer to the pair of rotoshapes used as input for the system presented in this paper as a *Hint Sequence*.

## 2 Hint Sequence generation

The first step required by our algorithm is the generation of rough rotoshapes to form the input Hint Sequence. The first of these — the outer shape — marks the approximate position of the foreground object, and it is important that every part of the foreground is covered by this shape. The second of these — the inner shape — is entirely inside the foreground, and it is important that no part of the background is covered by this shape. Thus, the edge of the foreground object will always lie between the two, covered by the outer shape but not the inner shape. We call this area the *unknown area*, and it appears grey in the Hint Sequence. Since areas outside the outer shape must be background and areas inside the inner shape must be foreground, they do not require processing. It is not required that the inner and outer shapes are drawn for each frame — it is possible, for example, to mark the foreground in just one or two frames and leave the entire object in other frames marked entirely as unknown. This will, however, require more pixels to be processed automatically and therefore take longer to process.

For the results presented in this paper, we use Apple's *Shake* package to generate the Hint Sequence. Figs. 3(c) and 3(d) show part of the Hint Sequence used for the *Tom* sequence. We mix together the inner and outer rotoshape to produce the required three level image: areas inside the inner and outer rotoshape appear white, areas outside the outer rotoshape shape appear black. The unknown area between the inner and outer rotoshapes appears grey as required.

## 3 Processing pixels in the unknown area

We represent the set of colours that a pixel is likely to be when it is foreground and when it is background using a separate mixture model for each state. These two models are derived from pixels sampled from the sequence which are definitely background or foreground as indicated in the Hint Sequence. The following subsections describe which pixels are used to form the foreground and background models, how the models are formed, and finally how each pixel is classified.

### 3.1 Sampling pixels

To process a single pixel $\boldsymbol{p}$ at location $ij$ in frame $f$ within the unknown area, foreground and background pixels are sampled from the sequence and used to model the likely values for $\boldsymbol{p}$ given that $\boldsymbol{p}$ is foreground or background respectively.

We assume *spatio-temporal consistency*: if the pixel is foreground, its colour is likely to be close to that of another nearby foreground pixel in the same frame (*spatial consistency*), or to a pixel at the same location in a neighbouring frame (*temporal consistency*). Since we are assuming that the background is nearly static, we allow the temporal consistency to dominate over the spatial and prefer to sample pixels at a similar location to the current pixel in a distant frame over a pixel further away in the current frame. As a simplification, we apply the same assumption to the collection of both background and foreground pixels. In some circumstances it might be preferable to adopt a strategy that tends to select foreground pixels from the current frame from a small area around $ij$ and background pixels from location $ij$ in different frames.

To form the sets of background and foreground pixels, we search forward and backward through the sequence, collecting all pixels within a box of size $n$ centred on $ij$ (the location of the current pixel). Initially $n = 1$, so that all pixels marked as foreground or background at location $ij$ in any frame of the sequence are sampled first. $n$ subsequently increases, causing an increasingly larger area of the image sequence to be sampled. Pixels are extracted from every frame at each stage.

The process continues until a suitable number of foreground and background pixels have been collected.

### 3.2 Forming models

At this stage, we have two sets $S_B$ and $S_F$ of background and foreground pixels. We wish to use these sets to obtain a probability estimate to classify $\boldsymbol{p}$. We use a Mixture Model to model the distributions of background and foreground. For speed, we use a clustering method rather than an approach such as Expectation Maximisation or K–means to compute the mixture model parameters. We use Orchard Bouman Colour Quantisation [4] as our clustering method. This sorts the set into $K$ subsets with $n_i$ pixels in each. We then calculate the mean $\mu$ and covariance estimate $\Sigma$ of each subcluster of $S_B$ and of $S_F$ to form two mixture models $\{(\mu_1, \Sigma_1, n_1), (\mu_2, \Sigma_2, n_2), \cdots, (\mu_K, \Sigma_K, n_K)\}$ $B$ and $F$ respectively.

### 3.3 Pixel classification

The probability of $\boldsymbol{p}$ being foreground is estimated from the mixture models for the foreground and background. We base our probability estimate on the Mahalanobis distance [9] rather than the standard Gaussian mixture model probability. We have found that our model produces results with fewer erroneously classified pixels. Which pixels are selected to form the subclusters, and therefore which pixels influence the estimated covariance matrix $\Sigma$, is dependent on the position of the unknown area. As a relatively small number of pixels are used to compute these estimates, the covariance matrices are susceptible to errors. Under the Gaussian assumption, the pdfs $p(f|\boldsymbol{p})$ and $p(b|\boldsymbol{p})$ conditional on these estimated covariance matrices are extremely sensitive to these errors, due to the exponential nature of the Gaussian. This error sensitivity makes the computed probabilities dependent on the exact position of the unknown area, which is undesirable. The measure we use is designed to be less sensitive to errors in the covariance and therefore to the position of the unknown area.

The probability that $\boldsymbol{p}$ is part of the background or the foreground is given by

$$P(f|\boldsymbol{p}) \propto \frac{1}{|F|} \sum_{i=1}^{K} \frac{n_i^{(F)}}{m(\boldsymbol{p}, \mu_i^{(F)}, \Sigma_i^{(F)}) \sqrt{\left|\Sigma_i^{(F)}\right|}} \quad (1)$$

$$P(b|\boldsymbol{p}) \propto \frac{1}{|B|} \sum_{i=1}^{K} \frac{n_i^{(B)}}{m(\boldsymbol{p}, \mu_i^{(B)}, \Sigma_i^{(B)}) \sqrt{\left|\Sigma_i^{(B)}\right|}} \quad (2)$$

where $(F)$ and $(B)$ imply indexing of parameters in the foreground and background mixture model respectively

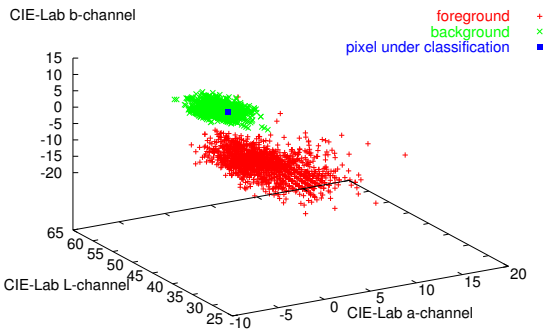



Figure 1: Clusters sampled to classify a single pixel

and $|F|$ and $|B|$ are the number of pixels in $S_F$ and $S_B$ respectively. Function $m(\boldsymbol{p}, \mu, \sigma)$ is the Mahalanobis distance

$$m(\boldsymbol{p}, \mu, \Sigma) = (\boldsymbol{p} - \mu)^T \Sigma^{-1} (\boldsymbol{p} - \mu) \quad (3)$$

The final classification (probability of foreground) is given by

$$C(\boldsymbol{p}) = \frac{P(f|\boldsymbol{p})}{P(b|\boldsymbol{p}) + P(f|\boldsymbol{p})} \quad (4)$$

This formulation is preferred to the more standard ratio of probabilities to simplify thresholding: all values of $C$ lie between 0 and 1.

The CIE-Lab colourspace is used for all colour measurements. In this colourspace, colour differences tend to be small for two pixels within the same object and large for pixels within different objects. Since the colour space also models human perception, it should help to give intuitive results to the operator: if an object is being incorrectly classified, it is likely to be apparent what similarly coloured object in the scene is the cause of the problem, allowing the user to modify the rotoshapes accordingly.

Figure 1 shows a typical cluster of foreground and background pixels plotted in CIE-Lab space with a pixel under classification. In this case the pixel under classification is 0.9% likely to be foreground as the pixel is very close to the background cluster.

## 4 Interactive thresholding

In some sequences the result of the pixel classification $C$ (Equation 4) acceptable. However, it is likely that further refinement will be required. In this case, the operator interactively selects thresholds to provide a more precise Hint Sequence which will be used as input to a subsequent iteration of the program. Five thresholds are applied to mark each pixel $\boldsymbol{p}$ in the sequence as follows:

- if $P(b|\boldsymbol{p})+P(f|\boldsymbol{p}) <$ **warn_thresh** then mark pixel green
- else if $P(b|\boldsymbol{p}) \geq$ **back_thresh** and $P(f|\boldsymbol{p}) \geq$ **fore_thresh** then mark pixel red
- else if $P(b|\boldsymbol{p}) \geq$ **back_thresh** or $C_{\boldsymbol{p}} \leq$ **low_thresh** then mark pixel background (black)
- else if $P(f|\boldsymbol{p}) \geq$ **fore_thresh** or $C_{\boldsymbol{p}} \geq$ **high_thresh** then mark pixel foreground (white)
- else mark pixel unknown (grey)

Pixels which are marked foreground or background will not be reprocessed in future iterations but will be sampled during processing of other pixels. Pixels which do not pass any of the threshold tests are marked as unknown (grey), in order that they will be reprocessed at the next iteration. In order to aid the operator in selecting these thresholds, we mark pixels which pass both **background_thresh** and **foreground_thresh** in red to indicate that the colour clusters were overlapping in colourspace — the background and foreground colours were too close to provide a reliable segmentation. The remedy would be either to raise the background or foreground thresholds until the pixel resolved to foreground or background respectively, or else to mark pixels by hand.

Pixels are marked green using **warn_thresh** to indicate that the colour of this pixel was too dissimilar to both foreground and background clusters — no objects which have been marked as foreground or background are similar to the colour of this object. It is probable that the pixels have been classified correctly, and raising **low_thresh** or lowering **high_thresh** will provide the correct result. If this is not the case (or if changing the thresholds causes other pixels to be misclassified), some of the pixels in the green area will need to be manually marked as background or foreground. This will provide extra information to the classifier during the next iteration, and similarly coloured pixels in this area should now be classified correctly. If either the background or foreground is very uniform, then all areas marked in green will be foreground or background respectively and can be manually assigned as such. See Fig. 2(c) for an example output from the thresholding step, showing red and green areas.

Any red and green areas remaining after thresholds have been chosen are automatically converted to grey levels, which forces such pixels to be reprocessed during a future iteration.

Note that during the interactive thresholding step the only processing required is the thresholding of pre-computed probability values. This makes the thresholding step fast enough for the effect of changing a threshold to be visible immediately.

### 4.1 Hint Sequence Noise Reduction

The thresholding step often leaves small groups of pixels marked as foreground, background or unknown. These small groups are likely to be erroneous and may degrade overall performance. Therefore, a small region removal step can be applied.

Each frame of the Hint Sequence is processed as a separate image and each region in the frame is extracted using a region growing technique. While growing the region, the pixels outside the region's perimeter are examined in order to determine whether the area surrounding this region is predominantly foreground, background or unknown area. If the region is smaller than a user selectable threshold size, it is removed by merging it with the predominant region surrounding it. The unknown area can also be dilated by a pixel or two to ensure that no pixel is incorrectly marked as background or foreground along edges.

The output of thresholding, possible hand editing and noise reduction is a new Hint Sequence which is more precise than that used for the previous sequence. This can now be used to reclassify the sequence during a further iteration step.

## 5 Alpha channel output

Once a satisfactory segmentation has been obtained, the probability of foreground image $C$ can be used to generate an alpha channel for the sequence. Individual pixels in the alpha channel of an image sequence should be 1 to indicate foreground, 0 to indicate background and intermediate values to indicate that the pixel is a mixture of background and foreground. In the probability of foreground sequence ($C$), background areas will have small, non-zero values and foreground areas will have values slightly less than 1. The final step is therefore to apply a soft threshold, to force all pixels with a probability less than **low_thresh** to be background and all with a probability of more than **high_thresh** to be foreground, intermediate values being scaled accordingly.

## 6 Optimisation considerations

Processing each pixel independently proves to be slow. We have obtained a significant speed increase with very little loss of quality by processing pixels in $S \times S$ blocks. We assume that the models for the probable background and foreground colours for all pixels within the $S \times S$ block are almost identical. This means that the foreground and background mixture models need be computed once only for all $S^2$ pixels within the block. (For the results presented later, $S = 6$.)

Given this assumption, an additional speed increase (with no further degradation of quality) can be achieved by the use of an *ImageBlockList*. For a sequence of $F$ frames of dimensions $(W \times H)$ we create a $F$ element list of $(W/S \times H/S)$ arrays. Each element in the arrays contains a list of the known foreground colours and a list of the known background colours for the corresponding $S \times S$ block of the corresponding frame of the sequence. Once the ImageBlockList 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 ImageBlockList.

It is also possible to run the algorithm in frame parallel mode on multiple processors. A single ImageBlockList would be maintained for all processors. Each processor is independently assigned a single frame (or a small contiguous group of frames if there are more frames than processors). Before each subsequent iteration of the algorithm, the processors would update the central ImageBlockList and process the same frame(s) in the next iteration.

## 7 Results

We have implemented a single processor version of the classifier, and prototyped the interactive threshold step as a node in the *Shake* image compositing package. This allows immediate visualisation of the effect of modifying any of the thresholds and also allows the thresholds to change for different frames in the sequence by interpolating the threshold levels between keyframes. We were also able to apply small modifications to the resultant hint sequence using the built-in painting functions. Final thresholding to produce an alpha channel was achieved using the Lumakey node.

Fig. 2 shows the *Patrice* sequence. Much of this sequence is simple to segment as it is akin to a bluescreen sequence. However, the desk in the background is close in colour both to the subject's skin and the sweater, causing areas of confusion. This problem is exacerbated by large amounts of sensor noise due to low lighting levels. Creating a reference background image would be difficult because the subject moves only slightly during the sequence. The sequence here is 50 frames long and $720 \times 576$ (except for Fig. 2(a) we show a $412 \times 329$ crop of the sequence for clarity). The initial rotoshape (Fig. 2(b)) was drawn crudely because it was supposed this sequence would be segmented well. Fig. 2(c) shows the result of thresholding the processed result. A red area on the right hand side of the image indicates where both the background and foreground have high probability due to the clusters overlapping in colourspace. A small area of this red section was marked by hand to eliminate this problem. The green area to the left shows where part of the background is too dissimilar to the background marked in the Hint Sequence due to the presence of a shadow in this area. After the second iteration the problem has disappeared. For many purposes, the result shown in Fig. 2(d) would be acceptable. However, three iterations provide an even more acceptable result. The unknown area in the final Hint Sequence (Fig. 2(e)) was dilated slightly to ensure a soft edge in the segmentation.

The *Tom* sequence was filmed with a hand-held camera and stabilised as a pre-processing step to provide a near-static background. Stabilisation was achieved by rotating, translating and scaling each frame to align four tracked points. The black triangle to the left of Fig. 3(b) is an artifact of this warping. There is still significant residual motion in the background (too much to apply a technique which requires a perfectly static background), and the motion blur caused by the camerashake has not been eliminated. Additionally, there is a breeze which causes significant movement of the plants. The initial rotoshapes for two frames are shown in Figs. 3(c) and 3(d). The segmentation after just two iterations (Figs. 3(g) and 3(h)) would be acceptable for many applications. There are still a few patches of background showing because the colour difference between the hair and the tree-bark is very small. The wrists also are missing from frame 622 because of the small colour difference between them and the background. A further iteration could rectify both of these problems. As expected, the trousers and the sweater are extracted well since there is significant colour difference. After processing, the stabilisation could be reversed to warp the alpha channel back to the correct location in the original sequence.

The *Treewalker* sequence (Fig. 4) is extremely challenging, since there is a lot of background movement. The trousers and arms are also virtually indistinguishable from the background. The subject (foreground) is also very small (frame size is $106 \times 244$). The continuous change from shade to light in the foreground also causes many problems. As with all sequences shown here, this sequence was captured with a Camcorder on DV tape. The DCT compression and chroma-decimation produces significant artifacts and causes edges to become indistinct. Although the segmentation result after four iterations is far from perfect (note the missing arms and foot, which were not marked as foreground in the initial rotoshape), this may be acceptable for certain applications. Not surprisingly the red shirt is extracted without difficulty. Given the small colour differences in this sequence, manually segmenting all 248 frames to this level of precision would be a rather laborious task.

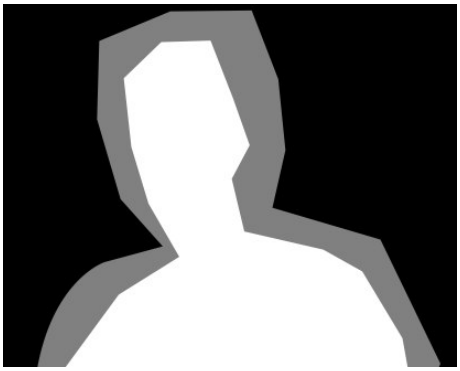
## 8 Conclusions

This paper has presented an algorithm for the semi- automated interactive segmentation of image sequences. Our results show that acceptable segmentations can be achieved with much less effort than manual segmentation, and with little more effort than a completely automatic technique. Adequate results can be achieved in a short time, even with challenging sequences. If the initial results are not sufficiently accurate, performing further iterations including some human interaction can achieve continual improvement of results until the segmentation is deemed acceptable.

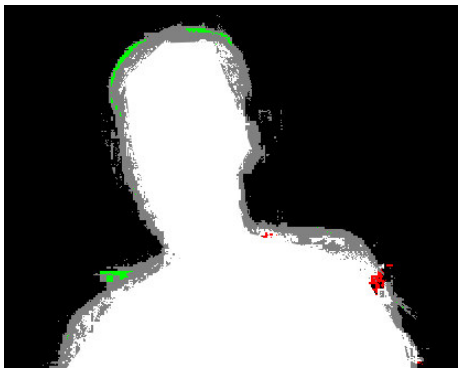
A non-optimised implementation of the algorithm took approximately 8 seconds per frame to segment the *Patrice* sequence using a 2.4GHz Intel Xeon processor. Given that the algorithm can be implemented in frame-parallel mode, and that dozens or even hundreds of separate workstations (normally used for image rendering) are typically available to post production facilities, the results of each iteration could be available in a very short time.

Future work will investigate the use of a Bayesian formula-

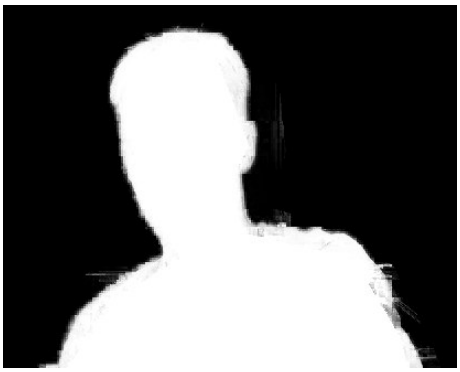
(a) Input Frame



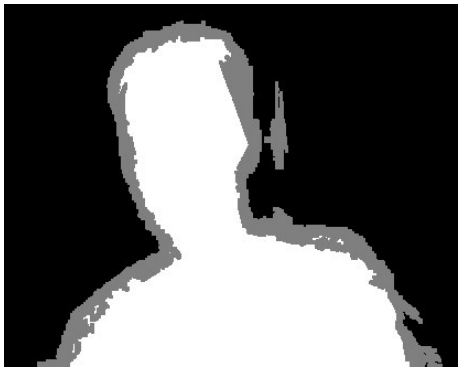
(b) Initial Rotoshape



(c) Thresholds selected for iteration 2 showing green and red areas



(d) Foreground probability image after iteration 2



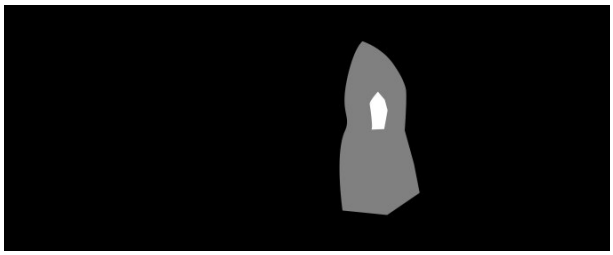
(e) Hint Sequence for iteration 3 after noise removal

(f) Final result after 3 iterations

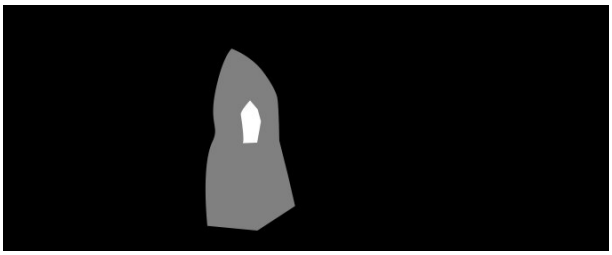
Figure 2: Results the *Patrice* sequence: Frame 11

(a) Input Frame 622

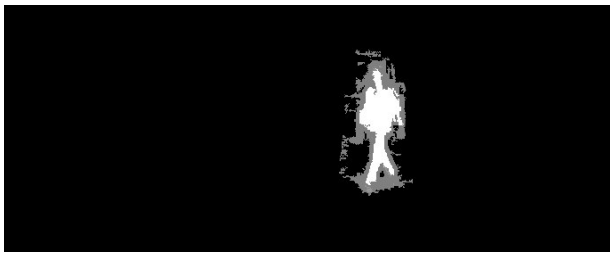
(b) Input Frame 647



(c) Rotoshape for frame 622



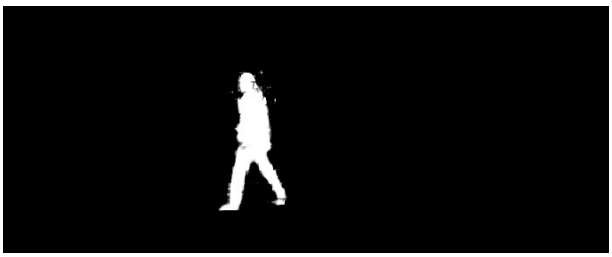
(d) Rotoshape for 647



(e) Hint Sequence for iteration 2: frame 622

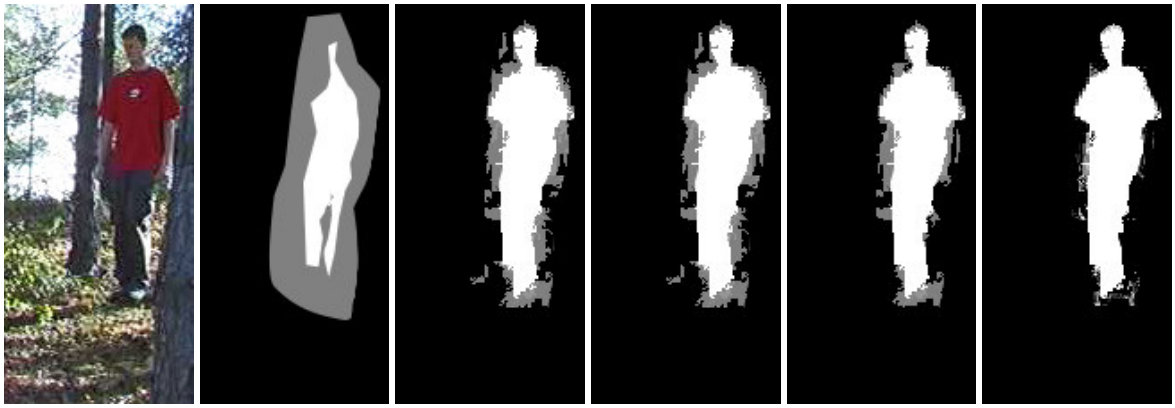
(f) Hint Sequence for iteration 2: frame 647

(g) Final Result after two iterations for frame 622



(h) Final Result after two iterations for frame 647

Figure 3: Results with two frames of the *Tom* sequence



(a) Input frame (b) Initial shape (c) Hint for it. 2 (d) Hint for it. 3 (e) Hint for it. 4 (f) Final output

Figure 4: Results with the *Treewalker* sequence: Frame 80

tion to classify multiple pixels simultaneously, which should help prevent small areas of misclassified pixels and ensure that the edges of objects do not become too sharp. We will also investigate the incorporation of automatic alpha estimation algorithms [3] into this framework, and the use of sparse motion estimates to guide the sampling of foreground pixels. Applying an adapted connected components or shape analysis algorithm to the output alpha channel estimate may help to reduce the noise visible in Fig. 4(f) (most algorithms of this nature require a binary segmentation rather than an alpha channel)

The processes of generating the initial Hint Sequence and of selecting thresholds should be fast and intuitive for a skilled system operator as they are analogous to existing manual and semi-automatic segmentation techniques. The speed and interactive nature of the algorithm, combined with the possibility of implementing the system with an interface which has a similar "look and feel" to existing systems makes this technique well suited for use in Visual Media Post Production.

**Acknowledgements**

This work was conducted within the EPSRC Platform Grant GR/S06578/01. Special thanks to Tom Bishop and James Hopgood for discussions about probability, and to Peter Grecian of the Moving Picture Company for his assistance with *Shake*.

**References**

[1] M. Boninsegna, A. Bozzoli. "A tunable algorithm to update a reference image". *Signal Processing: Image Communication*, **16** pp. 353–365 (2000).

[2] P. E. Eren, Y. Altunbasak, A. M. Tekalp. "Region–Based Affine Motion Segmentation Using Color Information". In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP-97*, volume 4, pp. 3005–09 (1997).

[3] P. Hillman, J. Hannah, D. Renshaw. "An improved algorithm for segmentation of motion picture image sequences". In *Proceedings, 1st European Conference on Visual Media Production*, pp. 175–84 (2004).

[4] M. Orchard, C. Bouman. "Color Quantization of Images". *IEEE Transactions on Signal Processing*, **12** (39) pp. 2677–90 (1991).

[5] Y. Raja, S. J. McKenna, S. Gong. "Segmentation and Tracking Using Colour Mixture Models". In *Proceedings ACCV*, volume 1, pp. 607–614 (1998).

[6] Shen Yujian, He Xin, Hao Zhihang. "Real-time temporal recursive filtering system for detection of Small Moving Targets". In *Proceedings of the 5th International Conference on Signal Processing*, volume 2, pp. 1045–48 (2000).

[7] A. R. Smith, J. F. Blinn. "Blue Screen Matting". *Computer Graphics: Proceedings of the ACS*, pp. 259–268 (1996).

[8] C. Stauffer, W. Grimson. "Adaptive background mixture models for real–time tracking". In *Proceedings, CVPR*, volume 2, pp. 246–252 (1999).

[9] C. Therrien. *Decision estimation and classification*. John Wiley (1989).

[10] Yu-Pao Tsai, Y. P. Hung, Z. C. Shih, J.-J. Su, S.-R. Tsai. "Background Removal System for Object Movies". In *Proceedings, Intl Conf. on Pattern Recognition*, pp. 608–611 (2004).