
© Copyright 2010 [please consult the authors]
ACCURATE SILHOUETTE SEGMENTATION USING MOTION DETECTION AND GRAPH CUTS

Daniel Chen, Simon Denman, Clinton Fookes
Image and Video Research Laboratory
Queensland University of Technology
GPO Box 2434, Brisbane, Queensland 4001

ABSTRACT

Acquiring accurate silhouettes has many applications in computer vision. This is usually done through motion detection, or a simple background subtraction under highly controlled environments (i.e. chroma-key backgrounds). Lighting and contrast issues in typical outdoor or office environments make accurate segmentation very difficult in these scenes. In this paper, gradients are used in conjunction with intensity and colour to provide a robust segmentation of motion, after which graph cuts are utilised to refine the segmentation. The results presented using the ETISEO database demonstrate that an improved segmentation is achieved through the combined use of motion detection and graph cuts, particularly in complex scenes.

1. INTRODUCTION

Computer vision applications such as marker-less motion capture [1] require highly accurate segmentation of a person from the background. Small errors in the extracted silhouette can significantly degrade performance.

Foreground segmentation techniques are commonly used to separate the foreground objects from a known or learned background. Techniques such as the Mixture of Gaussian’s (MoG’s) technique proposed Stauffer and Grimson [2] model each pixel as a multi-modal distribution to allow for multiple modes of background and improve segmentation performance.

Various extensions to this technique include the ability to impose feedback on the background model [3], and the addition of shadow removal [4]. However, modelling each pixel with a GMM is very processor intensive and not ideal when foreground segmentation is only the first step in a multi-step process (i.e. surveillance). Similar techniques have also been proposed by Butler et al. [5], which modelled each pixel as a group of clusters (a cluster consists of a centroid, describing the pixels colour; and a weight, denoting the frequency of its occurrence); and Kim et al. [6], who proposed using a codebook to model individual pixels.

However whilst these approaches are all capable of a high level of accuracy and are suitable for tasks such as object tracking, they are still prone to segmentation errors caused by lighting fluctuations and low levels of contrast between the foreground and background. Whilst these errors can, to an extent, be corrected by morphological operations these operations can also distort the shape of the underlying object, rendering them unsuitable for applications such as marker-less motion capture.

Graph cuts, using the max-flow min-cut algorithm [7] have been successfully applied to produce clean image segmentation. It allows taking into consideration differences in neighbouring pixels when segmenting, grouping similar pixels together when they would have otherwise been wrongly segmented. The algorithm produces a globally optimal solution given the input weightings.

Butler and Jolly [8] applied it to image volumes, such as medical scans, segmenting out foreground regions seeded from manually selected foreground and background points. The GrabCut [9] algorithm takes an iterative approach and is able to accurately segment a region from the background based on an initial user selection box through the use of foreground and background colour models.

In this paper, motion segmentation and graph cuts are combined to give highly accurate segmentation in an automated manner. Motion segmentation is performed using a combination of colour, intensity and gradient information to extract all possible regions of motion. Graph cuts are applied to the output of the motion image, to remove spurious motion and fill in motion that could not be detected, resulting in accurate motion segmentation results suitable for pose estimation or similar tasks.

Results are presented using a portion of the ETISEO [10] database and it can be seen that combination of motion segmentation and graph cuts leads to improved accuracy, particularly in challenging conditions. Section 2 describes the motion segmentation process used in this paper, Section 3 details how the graph cuts algorithm is applied to the resultant motion segmentation output. Section 4 presents the results and Section 5 concludes the paper and outlines directions for future work.

2. MOTION DETECTION

The motion segmentation algorithm proposed by Butler et al. [5] is used within this work, and is outlined in Section 2.1. This algorithm is modified to allow an arbitrary number of input channels to be used. A combination of intensity, colour and gradient information is used within the background model, as described in Section 2.2.

2.1. Algorithm

An efficient method of foreground segmentation that is robust and adapts to lighting and background changes was proposed by Butler [5]. This approach is similar in design to the MoG’s approach proposed by Stauffer and Grimson [2], in that each pixel is modelled by a group of weighted modes that describe the likely appearance of the pixel.

The algorithm uses YCbCr 4:2:2 images as input, and adjacent horizontal pixels are paired to form clusters, such that pixel pair contains two luminance values and two chrominance values (in YCbCr 4:2:2, each pixel has two values, a luminance value and a chrominance value that alternates between blue and red). This pairing results in motion detection being performed at half the horizontal resolution of the original image, with the benefit being increased speed.

Let \( p(x, y, t) \) be a pixel in the incoming YCbCr 4:2:2 image, \( I(x, y, t) \) where \( [x, y] \) is in \([0..X - 1, 0..Y - 1]\) and \( t \) is in \([0..T]\). A pixel pair, \( P(x, y, t) \) (where \([x, y]\) is in \([0..\frac{X}{2} - 1, 0..Y - 1]\)) is formed from \( p(x, y, t) = [g_y, cb] \)
and \( p(x_i + 1, y_j, t) = [y_2, cr] \) to obtain four colour values, 
\[
P(x, y, t) = [y_1, cb, y_2, cr] \quad \text{where } x_1 = x \times 2, \text{ and } y_1 = y.
\]
These four values are treated as two centroids \((y_1, y_2)\) and \((cb, cr)\). Each image pixel, \( p(x_i, y_j, t) \), is only used once when forming pixel pairs \( P(x, y, t) \).

Let \( f(x, y, t) \) be a frame sequence, and \( P(x, y, t') \) be a pixel pair in the frame at time \( t' \). Pixel colour history is reordered
\[
C(x, y, t, 0, K - 1) = [y_1, y_2, Cb, Cr, w],
\]
which represents a multi-modal PDF. Each cluster contains two luminance values \((y_1, y_2)\), a blue chrominance value \((Cb)\), and red chrominance value \((Cr)\) to describe the colour; and a weight, \( w \). The weight describes the likelihood of the colour described by that cluster being observed at that position in the image. Clusters are stored in order of highest to lowest weight.

For each \( P(x, y, t) \) the algorithm makes a decision assigning it to background or foreground by matching \( P(x, y, t) \) to \( C(x, y, t, k) \), where \( k \) is an index in the range 0 to \( K - 1 \). Clusters are matched to incoming pixels by finding the highest weighted cluster which satisfies,
\[
|P(y_1) - C(k)(y_1)| + |P(y_2) - C(k)(y_2)| < T_{Lum}, \quad (2)
\]
\[
|P(Cb) - C(k)(Cb)| + |P(Cr) - C(k)(Cr)| < T_{Chr}, \quad (3)
\]
where \( P = P(x, y, t) \) and \( C(k) = C(x, y, t, k) \); and \( T_{Lum} \) and \( T_{Chr} \) are fixed thresholds for evaluating matches. The centroid of the matching cluster is adjusted to reflect the current pixel colour,
\[
C(x, y, t, m) = C(x, y, t, m) + \frac{1}{L} \sum_{i} P(x, y, t') - C(x, y, t, m), \quad (4)
\]
where \( m \) is the index of the matching cluster; and the weights of all clusters in the pixels group are adjusted to reflect the new state,
\[
w_k' = w_k + \frac{1}{L} (M_k - w_k), \quad (5)
\]
where \( w_k \) is the weight of the being adjusted; \( L \) is the inverse of the traditional learning rate; \( \alpha \); and \( M_k \) is 1 for the matching cluster and 0 for all others.

If \( P(x, y, t) \) does not match any \( C(x, y, t, k) \), then the lowest weighted cluster, \( C(x, y, t, K - 1) \), is replaced with a new cluster representing the incoming pixels. Clusters are gradually adjusted and removed as required, allowing the system to adapt to changes in the background.

After the updating of weights and clusters, the cluster weights are normalised to ensure they sum to one using
\[
w_k = \frac{w_k}{\sum_{k=0}^{M} w_k}, \quad (6)
\]

Based on the accumulated pixel information, the probability of a pixel being foreground becomes,
\[
P_{fgnd}(x, y) = \sum_{i=0}^{m} C(x, y, t, i)(w), \quad (7)
\]
where \( P_{fgnd}(x, y) \) is the probability that a pixel is foreground and \( m \) is the matching cluster. A threshold can be applied to these probability to classify pixels as motion.

For this application, a generalised form of this algorithm is implemented, operating at pixel resolution and allowing an arbitrary number of channels. The matching of incoming pixels to the stored model (Equations 2 and 3) becomes
\[
|P(\alpha_x) - C(k)(\alpha_x)| < T_x, \quad (8)
\]
where \( \alpha_x \) is the values for channel \( x \), and \( T_x \) is the threshold for this channel. Updating of clusters and weights is otherwise unchanged.

### 2.2. Motion Segmentation Inputs

In the proposed algorithm, image gradients are to be used as a feature for use motion detection in addition to intensity and colour. For simplicity only the image intensity is used to calculate gradients. Four gradient channels are used, one for each direction (horizontal, vertical, and the two diagonals). The values are calculated by finding the difference in intensity in the two adjacent pixels along a given direction. It is desirable to ensure that as much foreground as possible is flagged within the foreground image, even at the cost of some false positives.

The addition of extra channels such as gradient (instead of simply lowering the thresholds) is more desirable as objects with similar colouring to the background will still produce changes in gradient, whilst the movement of shadows should leave the underlying texture of the background the same. Simply lowering thresholds on intensity and colour channels often results in low contrast regions still going undetected, and greatly increases false motion caused by shadows or lighting fluctuations.

Due to edges of objects generally being areas of strong gradient, and that gradient is calculated between pixels, pixels that lie just outside the object boundary can be erroneously detected as motion. A simple method to remove this is to apply a simple binary morphological thinning. However, not all edges are as prominent, for example when the object moves in front of a similar coloured background, and as such thinning may do more harm than good when trying to preserve the original shape of the object.

Comparing to using colour and intensity (YCbCr) as the input, gradient appears to be more tolerant towards the effects of shadows, allowing more aggressive thresholding. It is also, in some cases, able to detect motion where the original could not (without lowering the threshold so much that excessive false detection of motion occurs) due to similarity of colours. However, when textureless objects are observed over textureless backgrounds (i.e. a car on a road), gradient alone is unable to detect the motion.

The two techniques can be used to cover each other’s weaknesses, and therefore a fusion of the feature sets can provide improved performance. Thresholds can be raised to hide false detections, relying on the combination of their strengths to provide good motion detection. This can be implemented simply by amalgamating the two feature sets into one, resulting in seven channels of information passed onto the motion detector.

A comparison of the motion detection can be seen in Figure 1. Thresholds were selected to keep the detection of shadows to a minimum. Using only YCbCr, regions of the car and the pedestrian in white was not able to be detected due to colour similarities with the background. The gradient only approach however fails with relatively smooth, featureless surfaces, illustrated by the cars. Including both feature sets allow a more complete segmentation of motion. Note the extra thickness around objects due to edges mentioned previously.

The result of the fusion is not perfect. On more challenging sequences, regions of motion can still be missed (see Section 4).

### 3. IMAGE SEGMENTATION USING GRAPH CUTS

Using the max-flow min-cut theorem, graph cuts are employed to clean up the motion detection results, using the probability outputs (see Equation 7) to drive the image segmentation. The segmentation problem is comparable to that in GrabCut and thus the graph is set up in a similar fashion.

A graph is constructed such that every pixel in the image corresponds to a vertex, with edges connecting it to its 8 neighbours. Costs are associated with these edges relative to the gradient between pixels. Every pixel also forms edges to each of the two special vertices known as the source \((s)\) and sink \((t)\), which
in this case represents the background and foreground motion. Costs are also linked to these edges, weighted by how likely the pixel belongs to either the foreground or background.

Using the max-flow min-cut algorithm, the graph is cut such that \( s \) and \( t \) are separated, while minimising the total costs incurred while breaking the edges. Pixels are labelled as either background or foreground depending on which of the two resulting graphs they end up on.

The background costs are given as the motion detection output probabilities, while the foreground edges are all given a constant weighting. If the edge costs to pixel neighbours are set to zero, the graph cut simplifies down to simply thresholding the motion probabilities by the constant. For the implementation in this paper, the motion output was arbitrarily linearly scaled between 0.25 and 0.75, with the threshold constant set half way in between at 0.5.

As with Rother et al. [9], the edge costs between pixels are weighted by the function \( y = \gamma e^{-\beta x^2} \) where \( x \) is the gradient value (0-255) and \( y \) is the output. \( \gamma \) and \( \beta \) control the scaling of the function. For this application, the automatically determined value for \( \beta \) did not seem appropriate. For the \( s \) and \( t \) weightings listed previously, it was experimentally determined that a \( \gamma \) value of 2 and a \( \beta \) value of 0.05 gave reasonable results in the test sequences used.

By segmenting based on regional similarity, this post processing step is able to fill in small areas of missing motion, as well as remove falsely detected motion. It does however require a significant amount of motion detected in the first pace and will actually remove detected motion should the missed detection outweigh it; too much false detections, and the graph cut process will increase it. Hence the need for the gradient component in the motion detection to provide a more robust output.

4. RESULTS AND ANALYSIS

The algorithm proposed in this paper is tested using sequences from the ETISEO database [10]. Two datasets are used, the VS2-RD6 dataset (see Figure 2) which shows a roadway containing people and vehicles, and the BC-16 dataset (see Figures 3 and 4), which shows a building corridor with people moving about. For the Figures 2 to 4, the first row shows the input image, the second row shows the segmentation result of [5], the third row depicts the motion detection output that is used as input to the graph cut, the results of which shown below on the bottom row.

Intensity and colour thresholds for the original and combined motion detection are kept the same. Note that for [5], thresholds are actually doubled as each comparison involves the difference in two pixel values.

Figure 2 shows the output for the RD6 sequence. It can be seen that a clean segmentation of the moving objects is achieved, with a small improvement over the [5]. There are some small issues with the shadows of the cars, and an error in frame 900 due to the reflective patch on the road, however the accuracy of the extracted silhouettes (particularly for the people) is improved.

Figure 3 shows the output for the BC-16 sequence. This sequence is significantly more challenging due to the poor contrast between the floor, walls and people, and the reflections from the floor. Lower thresholds are used in accordance with these conditions.

For some frames (1500, 2100, 2400) clear improvement in segmentation can be seen. However in many other frames little to no improvement is achieved. Some problems can be partially attributed to the motion blur present in the people walking close to the camera, such as in frame 1800 where the person on the right was not segmented cleanly. The blurring smoothes out the edges, lowering the effectiveness of the graph cut process. Most of the errors, however, are due to the missed motion detection. This results in the graph cut being initialised with the majority of an object or texture region that should be foreground, classified as background. As a result, the graph cut process identifies the rest of this region and assigns it all to background.

In order to identify more motion and improve results, thresholds are lowered further. This increases the false detection of shadows and reflections and introduces noise into the motion detection, therefore relying on the graph cut process to clean it up. Results are shown in Figure 4. It can be seen that the additional noise and false motion detected from the reflections can be removed by the graph cut process. In the low contrast regions (i.e. the person’s shirt in frame 1800) there is now sufficient motion detected to enable the graph cut process to classify the rest of the region as foreground. The original motion segmentation algorithm, even with the low thresholds, is unable to extract all motion in the scene, and detects a large amount of false motion due to the thresholds.

5. CONCLUSIONS AND FUTURE WORK

This paper has demonstrated an automated approach to extracting accurate silhouettes in complex environments. By initializing a graph cut process with motion detection output, foreground
regions can be accurately extracted and errors within the motion segmentation can be removed. The inclusion of gradient also helps provide a more robust motion detection. The use of graph cut allows low thresholds to be used in the motion detection to ensure that no motion is lost, with it being able to clean up any false motion due to image noise, shadows or reflections.

Future work will focus on using the graph cut segmentation results to provide feedback to the motion image, to correct errors in the background model, tune motion segmentation thresholds, and improve the output of the motion segmentation. Scheduling schemes that allow the process to run in real time (i.e. graph cut) helps provide a more robust motion detection. The use of graph cut allows low thresholds to be used in the motion detection to provide feedback to the motion image, to correct errors in the background model, tune motion segmentation thresholds, and improve the output of the motion segmentation. Scheduling schemes that allow the process to run in real time (i.e. graph cut) will also be investigated.

6. REFERENCES