INFORMATIONAL RETRIEVAL ASSISTED OBJECT SEGMENTATION IN VIDEO

Dan Ring, Anil Kokaram

Electronic & Electrical Engineering Dept., Trinity College Dublin. {dan@unworkabletor, anil.kokaram@tcd.ie}

Keywords: Semi-Automatic Information-Retrieval Video Segmentation Feature-Points

Abstract

Accurate object segmentation in video is difficult. The dynamic nature of the medium causes drifts in the feature spaces traditionally used in segmentation of objects in still images. For example, colour distributions, shape models and motion tracks of objects typically vary and / or deteriorate over time, resulting in the need to explicitly correct the object by hand in every few frames. The presented work exploits recent feature-based object detection work from information retrieval (IR) literature to propagate information from frames that are far apart to greatly reduce the amount of time required to manually correct segmentations.

Figure 1: We present a method using feature-point based object detection to automatically segment an object throughout a video. The object is initially selected by the user, seen in Fig. 1(a). Feature-points belonging to the object are identified, Fig. 1(b), and used as a query to detect objects in subsequent frames, Fig. 1(c). The detection is refined, and used as input to a modified “Grab-Cut” algorithm [18] to achieve an accurate matte, Fig. 1(d).

1 Introduction

Video compositing workflows rely on the ability to accurately select and extract objects from shots of video. For complex scenes, most production environments do this by hand. Semi-automatic or interactive segmentation systems aim to reduce the time spent manually cutting out object masks. This makes sense, as objects do not usually undergo dramatic transformations between frames, and an object mask from one frame often closely resembles the mask in adjacent frames. The idea is to intelligently use the information supplied by the user in one frame to automatically segment other frames. However, to date no technique is robust enough to propagate that information over more than a few frames. The novelty of this paper lies in exploiting feature-point based object detection research from information retrieval literature to propagate user-supplied segmentation information through video.

A short review is given to put this idea in context. Previous research [1] [25] [18] [11] [20] has resulted in many successful frameworks for the semi-automatic segmentation of objects in still images. The Grab-Cut [18] algorithm employs a strong probabilistic framework, allowing additional feature spaces such as shape and motion to be sensibly incorporated. Research such as “Interactive video cutout” [24], [25], [2] demonstrate how still image segmentation solutions can be extended to label objects in video. Typically, this means the user marks regions of object and not-object in an initial frame, and the algorithm uses this information to segment subsequent frames. If the algorithm fails in later frames of the video, the user corrects the segmentation with additional markings, and the algorithm proceeds using the updated corrections. The accuracy of the automatic portion of the algorithm and the frequency of the manual corrections depend on how much the chosen segmentation feature space (i.e. colour, shape, contour, geodesic distance etc.) varies over time and how intelligently the user-supplied information is incorporated. Generally, the user has no control over the video feature space, and so the overall success of the object segmentation system is determined by how well the user-supplied information is utilised.

Given a user created object mask in one frame, it is logical to use the apparent motion to propagate the mask to neighbouring frames, as shown by [12] and [5]. A large strand of video segmentation research uses motion to remove the user from the workflow, automating the entire process. In this way, motion vector fields are used as an additional feature space in a probabilistic framework, as shown in [10], [9], [3], [23].
and [6]. The success of these algorithms relies heavily on the accuracy and reliability of the motion vectors. If the motion estimation process fails, the automated segmentation process fails too, and the user is back to manual correction. This paper presents a method of intelligently exploiting user-supplied information without the dependency on temporal coherency of the object.

1.1 Related Work in Information Retrieval in Segmentation

An initial step of detecting previously selected objects in a video frame is a natural way to incorporate spatial and shape information into a segmentation system. The use of object detection to aid segmentation has been demonstrated in the “Obj-Cut” system, [11]. Configurations of known shapes representative of object parts are used to detect the object, and bootstrap the segmentation process. This technique relies on extensive prior training of object parts, parts which must first be segmented themselves. Once trained for a particular object, the system is an unsupervised segmentation process, and does not explicitly incorporate user correction of the segmentation. By applying the robust text-based information retrieval (IR) techniques to feature-points, arbitrary objects can be selected and reliably detected.

The work of [22, 21] introduced the idea of using feature-points such as Harris-Corners, MSER regions or SIFT points as “visual words” in a traditional text-based IR system. This resulted in a fast and practical example of a large-scale object detection system for video. The work in this paper makes use of these IR techniques in pre-processing feature-points to improve the feature matching stage. This is discussed later in Section 2.2. Following the detection stage, a rough area enclosing the object is used to boot-strap the segmentation, shown in Figure 1(c). For the purposes of accurate segmentation, the regions returned by the system in [22] are not reliable enough. The works of [20] and [4] use a “star-graph” probabilistic framework to improve feature matches and generate a better rough mask around the initially detected object.

1.2 Novelty

To improve the use of interactive segmentation techniques like Grab-Cut in video, the main issue to be resolved is propagating information across large numbers of frames. Obj-Cut, which potentially provides a useable framework, requires too much training and is too object specific for use in post-production. Zisserman et al. [22] showed that a collection of low level features, the “bag of words” model, is particularly good at object recognition in video.

The novel aspect in this paper is to propose that collections of features can be used to locate and shape an object roughly in any video. The garbage matte so generated can be refined by Grab-Cut to yield a useful post-production segmentation. In addition, we present a new framework for enforcing spatial coherency of the feature detected object by drawing on ideas from the “star-graph” probabilistic framework of [20] and [4]. Lastly, we also use the collections of features to present diagnostic information to the user, showing how reliably the object can be segmented with the current data, and indicating the optimum frames to correct to improve the overall segmentation.

Section 2 details how IR techniques are applied to the problem of segmentation, Section 3 describes the object detection and refinement operations, Section 4 gives video segmentation results from this system, with a discussion given in Section 5.

2 Application of IR techniques in Object Detection

2.1 Choice of Feature-Point Detectors

Many studies have examined the merits and pitfalls of various feature point detectors [14, 15, 26]. The overall success of this segmentation algorithm not highly dependent on the choice of detector and descriptor, the reader may use a preferred feature detection scheme. However, finding feature-points in video is a lengthy process, and as such, care must be taken to select a detector scheme that balances the tradeoff between accuracy and speed. For this reason, the feature-points used in this paper are Hessian-Laplace points as outlined in [14]. The Hessian-Laplace detector finds maxima of the Hessian matrix \( H \) across multiple scales \( \sigma \), as shown in follows.

\[
H(x, \sigma) = \sigma^2 \begin{bmatrix}
L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\
L_{xy}(x, \sigma) & L_{yy}(x, \sigma)
\end{bmatrix}
\]

where \( L_{xx}(x, \sigma) \) is the second horizontal-derivative of an image at pixel site \( x \) and scale \( \sigma \). This results in the detection of isotropic blobs. To describe the image patches of the features, the popular SIFT descriptors are used, as detailed in [13]. Our SIFT descriptors are the default 128 dimension vectors.

2.2 Feature Pre-Processing

Feature-based object detection requires the comparison of many feature descriptors in the candidate image frame against a set of known object descriptors. Comparing many raw, high-dimensional, feature descriptors is slow. A crucial goal in video segmentation is to keep the processing time added by the object detection stage to a minimum. Most interesting examples of large-scale information retrieval (IR) frameworks achieve faster descriptor comparison by means of some form of clustering. For example, the works of [22] and [21] use simple k-means clustering. To speed up online cluster centroid matching, additional search strategies can be used, such as k-d trees in the case of [13]. The simple clustering idea is extended in [17] by the use of a hierarchical k-means clustering scheme, allowing much faster indexing and lookup in larger databases. In this paper, we use k-means clustering, \( N_c \) number of clusters are trained offline on a large number (\( > 5000 \)) of descriptors taken at random from the video to be segmented. After clustering, feature vectors are quantized, and represented now by code-book values \( i \) corresponding to the numbers of
Figure 2: Use of Star-Graph probabilistic model to improve feature matches. Query feature points \( P \) (red) are found in the user selection, and the polar translation \((\rho, \theta)\) from each point to the centre \( c \) (green) is calculated, shown in Fig. 2(a). Features in another video frame \( Q \) (blue) are calculated and matched against descriptors points \( P \), Fig. 2(b). Note the two incorrect matches. Features \( P \) are projected about every point in \( Q \), Fig. 2(c). The probable locations of \( c \) relative to \( Q \), given by \( c' \) (yellow), is shown by large clusters of points in Fig. 2(e). At the site \( c' \), features \( P \) are projected back out as \( P' \) using the reverse of \((\rho, \theta)\), Fig. 2(d). Finally, the distances between \( P' \) and \( Q \) can be used as a spatial probability to help identify correct (green) and incorrect (red) feature matches, Fig. 2(e).

the nearest clusters.

An offline clustering system is presented here as we assume we are going to process the entire video at once. However, a practical application would generally prefer an “on-line” system with less pre-processing. This can be achieved by doing a one-time a-priori clustering on a large random set of descriptors from representative video footage\(^1\). For each unseen video frame encountered, the feature points are calculated, and a nearest-neighbour search assigns the feature descriptors (“on-line”) to their nearest cluster and associated code-book entry. The rest of the system proceeds as usual.

Using quantized features, we can get an overview of the feature population. This is useful in identifying how descriptive a given feature is for the object relative to the rest of the video. For example, if both the object and video contain many small, right-angled corners, then a small, right angled corner is not much use in describing the object. Similarly, if the video contains a single instance of a very unfamiliar type of corner, it does not make sense to describe the object using this unfamiliar corner, as there is a good chance it will never be seen in the video. The works of [22, 21] demonstrate the ability to apply the text-based IR technique of term frequency-inverse document frequency or tf-idf to re-weight descriptors based on their occurrence frequency within the descriptor database. Given \( N_f \) frames of video, the weight \( t_{if} \) for a given code-book entry \( i \) in frame \( f \), is calculated as follows.

\[
t_{if} = \frac{n_{if}}{n_f} \log \frac{N_f}{n_i}
\]  

(2)

where \( n_{if} \) is the number of occurrences of the code-book entry \( i \) in frame \( f \), \( n_f \) is the number of clusters in \( f \), and \( n_i \) is the number of frames in which code-book entry \( i \) occurs throughout the video. The cluster weight \( t_{if} \) is used to compensate for very common and very uncommon descriptors. Very common descriptors have little descriptive power as they are likely to occur in an image regardless of the content. On the other hand, very uncommon descriptors do not have sufficient reliability. The success of feature-point object detection greatly improves when the matched descriptors lie in the middle of the two extremes. However, we find that as in [17], the rejection of clusters with frequencies in the top and bottom 5% percentile (a “stop-list”) to have no significant impact on the performance of the algorithm.

---

\(^1\)This only needs to be performed once if the random set describes enough of the possible feature descriptor space using a sufficient number of clusters. The success of PCA-SIFT [8], and the strong relationship between PCA and k-means clustering [7] indicate that this is achievable.
3 Online Object Detection & Segmentation

Before the object can be localized and segmented, we need to first see if the object can be detected in the frame. With most video shots to be segmented, the object is assumed to be present for the entirety of the shot. In cases where the object becomes occluded, motion based segmentation methods will typically fail. By performing a quick detection reliability check, our system can diagnose occluded or deformed objects, and if necessary instruct the user for additional input.

3.1 Detection Reliability Test

The detection reliability measure is similar to the object detection procedure shown in [22] and [21]. A given frame to be searched \( f \) is represented by a vector \( V_f \) of length \( N_c \), where \( V_f = (t_0,f,t_1,f,\ldots,t_{N_c},f)^T \). The object query vector is similarly given by \( V_q = (t_0,q,t_1,q,\ldots,t_{N_c},q)^T \), where the concatenated user-selected features are considered to be a document \( q \). The object detection is performed by calculating the normalized scalar product \( f_d \) (cosine of angle) between the document vectors \( V_f \) and \( V_q \), as follows.

\[
f_d = \frac{V_q \cdot V_f}{||V_q|| ||V_f||}
\]  

(3)

How well the selected object can be detected in the current frame is given by the angle cosine \( f_d \). The value of \( f_d \) will vary depending on the number of features present, and the number of additional user feature selections. An example of this is shown in Figure 3. The values of \( f_d \) can be observed over the video, and will usually show \( f_d \) decreasing with time (Figure 3 (red)). As the object changes appearance, features selected in frame 1 disappear, making the object more difficult to detect in later frames. More manual re-selections of the object are needed to improve detection performance. For example, following the re-selection in frame 46, a sharp improvement in detection reliability is seen in Figure 3 (green, triangle). This is improved by the last re-selection in frame 28, shown in Figure 3 (blue, circles). Using the plots of Figure 3, it is possible to diagnose when the object is less likely to be reliably re-detected, and find the optimum frames in which to re-select the object to improve detection.

If the value of \( f_d \) for the current frame is sufficiently high (this threshold depends on the video), the object is considered detected. The feature points common between query and frame vectors \( V_q \) and \( V_f \) are assumed to be representative of the object, and are now used to refine the location and shape of the object within the video frame. The methods for pre-processing features and initial object detection have been described. We now present the online part of our system.

3.2 Overview of the Segmentation Process

1. The user selects an object in a frame of video. Features belonging to the object are collected, and added to the query document \( q \). The tf-idf scores \( t_{iq} \) and document vector \( V_q \) are updated.

2. For any other video frame \( f \), features of the image \( V_f \) are compared against \( V_q \) to try to detect the query object, as shown in Equation 3.

3. If the object is detected. The location and shape of the object is estimated, and a rough silhouette of the object is calculated, as shown later in Section 3.3.

4. If the object that is known to exist in the video frame was not detected, or if the rough silhouette does not entirely capture the object, the user can re-select the object in the frame, and add the newly selected features to the query document \( q \), update \( t_{iq} \) and \( V_q \) again, and the process returns to Step 2.

5. The candidate video frame region corresponding to the rough mask is used to generate the pixel statistics for use in an accurate, pixel-level object segmentation process, and a refined object mask is calculated.

6. If the object was still not satisfactorily segmented, the user can supply corrections directly to the object segmentation algorithm as would normally be the case.

3.3 Enforcing Object Spatial Coherency

Detecting objects using feature-points hinges on finding good matches between feature descriptors. A match between a
pair of raw descriptors is typically identified by thresholding some distance (euclidian, mahalanobis or chi-squared for example) measure between the descriptor vectors. In the clustered IR framework, a match is found if the pair of raw descriptors belong to the same cluster centroid or cluster branch. As features are not necessarily unique to a given object, a feature descriptor from the query object can easily match to a descriptor in the candidate image frame being searched. For example, given a query object of a chair and using corners as features, it is not difficult to imagine similar right-angular shapes being found on other objects. This means descriptor matches should be assessed to ensure they belong to the same, correctly identified object.

A number of ways exist for enforcing the spatial coherency of descriptor matches. The works of [16] and [19] fit a model such as an homography or epi-polar camera transform to best describe the apparent warp as given by initial tentative feature descriptor matches. Descriptor matches that do not conform to the model are rejected. Using only a single transform model, this method assumes that the entire object is rigid, and will often reject good matches relating to the deformable parts of the object. This can be accounted for by using more than one transform model to describe the apparent object motion, however, success is highly dependent on correctly tuning the number of models.

A simpler method given in [22] applies the constraint that a match can only be accepted if it has at least N matches within a given radius. This constraint allows for arbitrary object deformations, and works surprisingly well. However, the spatial coherency methods of [22, 16, 19] are rejective, that is, given a set of tentative matches, they can only reject candidate matches. For objects or scenes with low numbers of feature-points, the resulting number of matches following the applied spatial constraints may be too low to estimate a rough shape of the object n the image. A method is presented that both rejects poor matches, and promotes alternative tentative matches.

To encourage spatial coherency, we extend the star-graph probabilistic model, as originally outlined in [20] and [4]. Since we are using a code-book to represent features, it is probable that we find ambiguous feature matches. For example, a query feature has a code-book number of 100. In the candidate image, three features may have the same code-book number 100. In this case, what candidate feature matches the query feature? If we figure out where the query feature would be in the candidate image, the distance between this point and each of the three candidate points is calculated. The lowest distance indicates a more probable match to the corresponding candidate feature. This is the idea behind star-graph voting. Our version of the algorithm proceeds as follows. Given a set selection of query points P, an arbitrary point c is chosen, such as the centroid of P. The polar co-ordinate translation, (ρ, θ), from each query point p

\[ \rho_n = \frac{\max(p - c)}{s_{p,n}} = \sqrt{(p_n_x - c_x)^2 + (p_n_y - c_y)^2} \]

\[ \theta_n = \arctan\left(\frac{p_n_y - c_y}{p_n_x - c_x}\right) - \alpha_{p,n} \]

An example of this is shown in Figure 2(a). A candidate video frame containing the object is supplied, and features of tentatively matched points Q in the frame are identified, shown in Figure 2(b). For each candidate point q

\[ o_{m,n} = \begin{pmatrix} q_{m,x} \\ q_{m,y} \end{pmatrix} = \rho_n s_{q,m} \begin{pmatrix} \cos(\theta_n - \alpha_{q,m}) \\ \sin(\theta_n - \alpha_{q,m}) \end{pmatrix} \]

where \( o_{m,n} \) is the location of c as projected by \( (\rho_n, \theta_n) \) from candidate feature point q

\[ D_{p,q}(n, m) = \begin{cases} 1, & c(n) = c(m) \\ 0, & c(n) \neq c(m) \end{cases} \]

where \( c(n) \) and \( c(m) \) are the code-book values of features n and m. The projected points \( o_{m,n} \) are plotted on an image of identical size as the candidate image. Large clusters of points indicate the probable location of c relative to the candidate point set P. This is denoted by \( c' \), and is shown in Figure 2(c). This projected point image is smoothed, and non-maximal suppression is performed to detect the most likely candidates of \( c' \). At a candidate location of \( c' \), projections \( P' \) of the original point set of \( P \) are calculated using inverses of the query point polar co-ordinates as follows.

\[ p_n' = c' = \rho_{s, p, n} \begin{pmatrix} \sigma_Q^2 \\ \sigma_P^2 \end{pmatrix} \begin{pmatrix} \cos((\theta_n - \pi) - \alpha_{q,m} - (\phi_Q - \phi_P)) \\ \sin((\theta_n - \pi) - \alpha_{q,m} - (\phi_Q - \phi_P)) \end{pmatrix} \]

where \( p_n' \) is the projected point of \( p_n \) about the point \( c' \). \( \sigma_Q^2 \) and \( \sigma_P^2 \) are the sample variances of \( \rho \) and \( \phi \). \( \phi_Q \) and \( \phi_P \) are the expected values of \( \theta_Q \) and \( \theta_P \) for the point sets Q and P. \( \sigma^2 \) and \( \phi \) correct for changes in scale and rotation between the point configurations. This can be seen in Figure 2(d).

Note that a second instance of the duck in the target image of Fig. 2(c) will produce a second strong candidate location \( c' \). This case is ignored for the moment and assume there is at most one instance of the object per frame, but will need to be addressed in future.
The spatial distances between the projected points \( P' \) and the candidate points \( Q \) are calculated, and are given by \( S_{p,q}(n,m) \).

There now exists two distance matrices. The first describes the likelihood of feature descriptors from query points \( p_n \) matching to candidate points \( q_m \), given by the binary matrix \( D_{p,q}(n,m) \). The second measures the spatial distances between where the candidate points and query points are expected to be \( p_n \) in the candidate image, denoted \( S_{p,q}(n,m) \). A balance between feature likelihood and spatial information is found by combining the two matrices

\[
E_{p,q}(n,m) = D_{p,q}(n,m) + \lambda S_{p,q}(n,m)
\]

by a constant \( \lambda \), as \( E_{p,q}(n,m) = D_{p,q}(n,m) + \lambda S_{p,q}(n,m) \). \( \lambda \) was shown experimentally to give good results at a value around 0.25. Matches between query and candidate points are now identified by selecting entries in the weighted distance matrix \( E_{p,q}(n,m) \) using the “nearest-neighbour” technique, followed by a ratio test as used by Lowe in [13]; Given a query point \( p_n \), find the candidate point \( q_m \) in \( E_{p,q}(n,m) \) with the lowest distance, and with a ratio to second-lowest distance less than 0.8.

### 3.4 Accurate Object Segmentation

Feature-points that represent an object do not necessarily reside within the boundaries of the object. For example, given an image of a fork, a corner detector will likely detect the image regions between the prongs of the fork as being long, elongated corners. Although these particular corners are crucial in depicting the shape of the fork, the location of the detected corner will reside on a region that would be classified as background. This indicates that feature-points cannot be used directly to classify object pixels, but can indicate a region where object pixels are probable. The recent work of [1, 25] demonstrates a fast, and accurate user-driven object segmentation scheme. Unfortunately, it requires that the user-supplied foreground / background strokes correspond exactly to the object or non-object respectively. Any slight ambiguity in the manual labeling stage can cause the algorithm to fail drastically. To segment the object accurately, the image region enclosing the feature-points is supplied to an object segmentation process that is able to deal with the foreground / background label ambiguity.

A modification of the popular segmentation process Grab-Cut [18] is employed, as it naturally handles label ambiguity. Our algorithm proceeds as follows; a rough mask from the feature-point object detection stage is supplied to the Grab-Cut algorithm [18]. The mask contains pixels belonging to both object and background. The background colour distribution is sampled from a single pixel border around the rectangular box. The initial colour foreground distribution is sampled from inside the bounded region. In the presented version of the algorithm, the foreground and background distributions are partitioned using a weighted mean-shift algorithm. Initially, all samples are weighted equally. The partitions are then modeled as GMMs, with the colour energies defined as the negative-logs of the label likelihood. As shown in [18], colour (data likelihood) and gradient (spatial prior) energies are minimized in the traditional graph-cut framework, resulting in a pixel labeling of either object or not-object. The weights of the pixels classed as object are incremented, and the mean-shift is run again to partition only the foreground samples. The foreground GMM parameters are updated and the graph-cut is run again to give an updated labeling. This re-weighting and re-labeling encourages a separation in the sampled colour distributions, and proceeds until convergence or until a maximum number of iterations has passed.

### 4 Results

The presented video segmentation algorithm has been tested on several real-world videos with varying properties, for example, varying types of camera and object motion, photometric conditions such as colour quality and brightness, and apparent video artifacts such as as interlacing and colour channel decimation. In some cases, a manual Grab-Cut performed on the image is shown for comparison, allowing the reader to isolate the contribution of the object detection stage on the overall segmentation.

Figure 4 illustrates the propagation of user-supplied rectangles. In this case, objects are selected up to 50 frames apart, detected and segmented. It can be seen that the reliability of the object detection masks becomes more accurate the closer in time between the selected frame and the frame to be segmented. As a result, the segmentation results also improve. This behavior is expected, and is typical of most video segmentation algorithms. Although the segmentations in Figures 4(b) and 4(d) are not perfect, the amount of extra work needed to correct and tighten the segmentation is considerably less than without the automatic part of the process.

The algorithm operates well across longer videos, as shown in Figure 5, where the object is reliably detected 120 frames after the selection frame. In the case of Figures 5(e), 5(g) and Figure 6(a), part of the background has been incorrectly labeled as object. Under the implemented Grab-Cut framework, this is unavoidable as part of the object and background colour distributions overlap. At the pixel level, the Grab-Cut workflow allows incorrect pixels to be re-assigned, as shown in Figure 5(h). The current object detection framework does not allow a similar explicit assignment of background “object”, which causes accidentally selected background to be re-detected, causing the same segmentation error in every frame. We plan to address this in future work by propagating pixel level updates through object detection.

The object detection framework is robust to many photometric distortions. In the case of Figure 7, the object is detected in the presence of high amounts of motion blur, camera noise and dust, and severe interlacing artifacts. However, even small errors in the detection stage directly affect the segmentation. The detection of the quad-bike in Figure 8 proceeds well for the first 20 frames, but begins to suffer when parts of the previously unseen background are incorrectly detected as background. In this case, the accurate segmentation process is given conflicting
information, and the overall segmentation process suffers. This is of course easily fixed by re-selecting the object in a later frame. The results of this for frame 40 is shown in Figure 1(a).

A comparison of feature- and colour-based segmentation schemes against manual ground-truth segmentations is given in Table 1. Two sequences, “Eating Apple” (100 frames) and “Quad Bike” (50 frames), were chosen for comparison to highlight the kind of videos particularly suitable and not suitable for each segmentation scheme. “Quad-bike” is highly textured, resulting in many stable features, while there is a clear colour separation between the woman and background in the “Eating Apple” sequence. The object was manually segmented in frames 1, 25, and 49 of each sequence, and depending on the method, the feature points or colour samples from these frames were used to automatically segment the rest of the video. The results were compared against the ground truths, with the numbers of pixels incorrectly classed as object (Type 1 Error) and not-object (Type 2 Error) are represented as percentages of the ground truth segmentation. The mean times to process a frame in each sequence are also given. Feature-based segmentation typically takes twice as long per frame, as it first performs the matching step, followed by the colour step. The ground-truths were created by manually segmenting each frame from the two sequences using the Quick Select tool in Adobe Photoshop.

Table 1: Comparison between IR (Feature) and Grab-Cut (Colour) based segmentations against manually segmented ground truths. Object statistics, such as feature points or colour, are taken from frames 1, 25, and 49, and used to segment the other frames in each of the two sentences. The Type 1 and Type 2 errors are the mean percentages of pixels incorrectly classed as object and not-object respectively, and Time is the mean time taken to process each frame.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Method</th>
<th>Type 1 Error</th>
<th>Type 2 Error</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eating Apple</td>
<td>Features</td>
<td>8.37%</td>
<td>35.55%</td>
<td>16.9s</td>
</tr>
<tr>
<td>Quad Bike</td>
<td>Features</td>
<td>1.34%</td>
<td>2.46%</td>
<td>10.1s</td>
</tr>
<tr>
<td>Eating Apple</td>
<td>Colour</td>
<td>1.59%</td>
<td>6.10%</td>
<td>5.9s</td>
</tr>
<tr>
<td>Quad Bike</td>
<td>Colour</td>
<td>5.69%</td>
<td>12.09%</td>
<td>8.7s</td>
</tr>
</tbody>
</table>

5 Discussion

This paper has shown that object re-detection is a good way to propagate user supplied data in a video segmentation framework. Using feature points, information retrieval techniques can be used to successfully detect, localize and extract objects in future frames, reducing the amount of user interaction required. By supplying additional strokes and rectangles, segmentation accuracy is improved at both the pixel and object detection levels.

The success of the object detection is highly dependent on the number of image features available to the system. For example, in the video of the woman eating an apple in Figure 9, the flat regions of her face, neck and hair contain relatively few features compared with the background. This means that the object shape as determined by the points will be inaccurate, seen in Figure 9(b), resulting in the high errors in Table 1. Features from different feature detectors such as Harris-Corners, SIFT and MSER were combined to increase the number of points, however, flat image regions still contain relatively few points. Instead of designing a new feature detector to cope with vacant regions, a more intelligent method of estimating object shape using the sparse feature points would be useful. This will be investigated in future work.

The current segmentation system uses object detection and accurate segmentation as two separate black-box processes. This means that to achieve a successful segmentation, both parts need to work correctly. If either fails, the system as a whole fails. For example, in Figure 9(c) the object detection system failed, resulting in a poor segmentation. In Figure 7(c), the object detection was successful, however due to poor colour the limiting factor is the Grab-Cut process. Also, as it necessary to perform both a matching step and a subsequent Grab-Cut colour segmentation step, the processing time is significant, as shown in Table 1. To improve the accuracy of this system as a whole, the object detection stage either needs to be incorporated into the probabilistic segmentation framework, or the use of colour in this system should be omitted altogether. This too is left for future work.

Acknowledgements

We would like to acknowledge and thank the AfterEffects, DMRG, and GRG groups of Adobe. This research has been funded by the Irish Research Council for Science, Engineering and Technology and Adobe Systems.

References


Figure 4: Example of propagation of user marked rectangles from different start frames to frame 50, shown here. The left column shows the object shape boundary recovered from the spatial coherency stage of the object detection. The right column line shows the resulting accurate segmentation from Grab-Cut [18]. The object extracted in Figs. 4(a) & 4(b) are from a user selected rectangle from frame 1, Figs. 4(c) & 4(d) from a rectangle in frame 25, and Figs. 4(e) & 4(f) from frame 49. Notice that although the rough object masks from the object detection are reasonably good, small variations affect the subsequent accurate segmentations. It can be seen that the closer in time to frame 50 the user rectangle is supplied, the better the resultant segmentation. Included for comparison in Figs. 4(g) & 4(h) are the input and results of using Grab-Cut [18] on a manually supplied rectangle, omitting the object detection process.

Figure 5: Example of long-range segmentation. Fig. 5(a) is frame 1 of a tennis player scene, using still a still camera and moving object, and exhibiting high amounts of motion blur. We select the tennis player, and find the associated features, Fig. 5(b). Fig. 5(c) shows the object shape boundary for frame 1 as given from the features. Fig. 5(d) shows the object shape boundary for the player in frame 40, using features matched from the selection in frame 1, with the resultant segmentation shown in Fig. 5(e). Fig. 5(f) is the segmentation of the player in frame 80. Notice the player is still detected and reasonably segmented despite the high amount of motion and interlacing artifacts. Fig. 5(g) shows the player at frame 100, with the mask manually corrected with additional user strokes shown in Fig. 5(h).
Figure 6: Example of a manual correction to the refined segmentation. Fig. 6(a) shows an coarsely segmented tennis player in frame 200. The player was originally marked by a rectangle in frame 1. The coarse segmentation can be corrected with some paint strokes explicitly marking the background, shown in Fig. 6(b). The cleaner segmentation is given in Fig. 6(c).

Figure 8: An example of detecting and segmenting a quad-bike from a moving camera, moving object scene. Figs. 8(a), 8(b), 8(c) & 8(d) show the object detected in frames 20, 30, 40 and 50 following an initial selection in frame 1. In this case, the segmentation proceeds well for the first 20 frames, but begins to fail as the object detection incorrectly identifies the darker background entering the scene as part of the object.

Figure 7: An example of detecting and segmenting a robotic arm of a lunar explorer. In this scene, the object is stationary relative to the moving camera. High amounts of motion, interlacing artifacts, and poor colour make this a difficult scene to segment. Figs. 7(a), 7(b) & 7(c) show the object detected in frames 25, 100, and 120 following an initial selection in frame 1. Notice in Fig. 7(c) that despite severe ghosting due to interlacing, the object is still successfully detected. Fig. 7(d) is a comparative Grab-Cut [18] on the frame 120.

Figure 9: An example of the object detection system failing. Features are selected in frame 40, shown in Fig. 9(a). The corresponding shape outline for the selected frame 40 features is shown in Fig. 9(b). Notice that the lack of features on the face and neck makes it difficult to estimate a good shape. Fig. 9(c) shows the segmentation using the shape input from Fig. 9(b). This particular image has good colour separation between object and background, and is a trivial task for Grab-Cut alone, shown in Fig. 9(d).