Video Indexing using Motion Estimation

Sarah Porter and Majid Mirmehdi and Barry Thomas
Department of Computer Science
University of Bristol
Bristol, BS8 1UB, UK
{porter,majid,barry}@cs.bris.ac.uk

Abstract

Summarising video data is essential to enable content-based video indexing and retrieval. A novel graph theoretic approach is presented to extract representative key frames corresponding to the shortest path of the graph for each shot. We distinguish further amongst paths of similar weight by examining the standard deviation of their constituent edge weights which improves the distribution of the selected key frames. The perceived camera motions contained within each shot are also annotated to introduce a further level of indexing and searching video content.

1 Introduction

There is currently a distinct lack of efficient authoring and querying tools inhibiting full exploitation of online and archived video data. Manually indexing the wealth of such video content is currently the most accurate method, but it is a very laborious and time consuming process. To enable users efficient access it is necessary to develop tools that facilitate searching based on non-sequential browsing and visual content-based indexing.

A predominant approach to this problem is to use a video abstract for indexing and retrieval. A video abstract is defined as a short sequence of images, extracted from a longer video whilst still preserving the essential message [8]. The difficulty in composing such an abstract is determining which frames best represent the video contents. A common approach is to segment a sequence into shots and then select a single key frame to represent each shot. This is often referred to as a filmstrip [2]. However, this is not always sufficient as a shot can potentially contain camera motions that could drastically change its content.

In this paper, we propose an algorithm to generate a video index which assumes that a video sequence has already been segmented into individual shots using the edit effect detection algorithm outlined in [9]. Then, estimates of the dominant motion between each frame pair are used to decide when there has been sufficient camera motion to require another key frame. A weighted directed graph is formed for each shot where the frames corresponding to the vertices forming the shortest path through the graph are used as representative key frames for each shot. Thus, a small subset of frames can be used to retrieve information from the video and enable content-based video browsing. We also extend this algorithm to characterise and textually annotate the perceived camera motion using a rule-based approach to augment the indexing and searching process.

2 Key Frame Extraction

Different approaches to video summarisation often depend on the context of the application. Lienhart et al. [8] concentrate on the generation of trailers for movies resulting in abstracts that are short and designed to attract the attention of the viewer without revealing too much of the story line. In contrast, an abstract for a documentary or a digital video library should capture all the video content. Two approaches to this are filmstrips, as mentioned earlier, and skims [2] which incorporate both video and audio information and are played rather than viewed statically. Whereas many of the approaches to video summarisation rely on the explicit detection of shot changes [1, 7], other techniques have been proposed without. Commonly images are transformed into a lower dimensional space, e.g. using PCA [6], and then grouped using a clustering algorithm [6] or curve simplification [4]. The frame closest to each cluster centroid is chosen as a key frame.

In order to be an efficient index, a set of key frames should represent all the video content (e.g. objects and background) and describe the order of events (e.g. camera motions) whilst minimising redundancy. We propose an approach to summarise a video that assumes the sequence has been temporally segmented into shots using the edit effect detection algorithm outlined in [9] which uses block matching motion compensation to generate an inter-frame difference metric. For each block in frame $f(n-1)$, the best match in a neighbourhood around the corresponding block in frame $f(n)$ is sought. This is achieved by calculating the normalised correlation between blocks in the frequency domain and locating the maximum correlation coefficient, the value of which is used as a goodness-of-fit measure for each block. The estimated motion vectors are then used to track the blocks over time. In the present work, we use these motion vectors to estimate the dominant motion between frame pairs. Assuming the dominant motion was caused by camera motion, these estimates can then be used to identify shots containing significant camera motion that may require more than one key frame to represent their content.

Camera motions can be grouped into two broad classes: (i) tripod motion and (ii) free motion. If a camera is fixed to a tripod it can only exhibit three types of motion; pan, tilt and zoom. If there is free motion of the camera it can additionally track, boom or dolly. The effect of a pan on the change of contents in a shot and the perceived image motion is very similar to that of a track. For example, if a camera pans or tracks right, the background and objects appear to move to the left and gradually leave the shot while new background and objects may appear on the right. Such similarities can also be drawn between "tilt and boom" and "zoom and dolly". For this reason, we use a simple motion model which only represents the scale and translation in x and y between two frames. The point $p_i = (x, y)$ in frame $f(n-1)$ is transformed to the point $p'_i = (x', y')$ in frame $f(n)$, with respect to a reference point (x_r, y_r) according to

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} {}^s\theta_n(x - x_r) + {}^x\theta_n \\ {}^s\theta_n(y - y_r) + {}^y\theta_n \end{pmatrix}. \tag{1}$$

In practice, the frame centre is taken as the reference point. The parameter vector $\vec{\theta}_n = ({}^s\theta_n, {}^x\theta_n, {}^y\theta_n)$ corresponds to the scale, translation in x and translation in y respectively, between the frames $f(n-1)$ and $f(n)$. The model parameters are estimated using the robust estimator MSAC [10] which provides good estimates in the presence of outliers. Outliers could possibly be present in the data where the motion equation is invalidated or where points correspond to secondary motions.

2.1 Minimising Similarity between Key Frames

To portray all of the video content, we must determine when there has been sufficient scene change due to camera motion to warrant more than one key frame to represent a shot. We formulate this as a shortest path problem. Given a directed weighted graph, the shortest path between two vertices is the path of minimum total weight. To minimise representational redundancy we need to minimise the similarity between key frames. Therefore, a graph is formed where vertices correspond to frames in the shot and edge weights are a measure of similarity between two frames. The frames corresponding to vertices forming the shortest path through the graph are then used as representative key frames for each shot.

We define the similarity metric $\psi(i,j)$ as the amount of overlap between the contents of frames $f(i)$ and $f(j)$ where $0 \leq \psi(i,j) \leq 1$, based on the assumption that the content is only significantly changed by camera motion. If there has been no camera motion between the frame pair, then they will contain the same content and $\psi(i,j) = 1$. As the amount of camera motion increases the amount of overlap will decrease until the contents of each frame are disparate and $\psi(i,j) = 0$. For each shot we have an estimate of the motion parameter vector $\vec{\theta}_n$ for each consecutive frame pair $f(n-1)$ and $f(n)$. Given any two frames $f(i)$ and $f(j)$ where $i < j$, we accumulate the motion parameters between them to obtain $\vec{\Delta}_{ij} = ({}^s\Delta_{ij}, {}^x\Delta_{ij}, {}^y\Delta_{ij})$ where

$$ {}^s\Delta_{ij} = \prod_{n=i+1}^{j} {}^s\theta_n, \quad {}^x\Delta_{ij} = {}^x\theta_j + {}^s\theta_j \cdot {}^x\Delta_{i(j-1)}, \quad {}^y\Delta_{ij} = {}^y\theta_j + {}^s\theta_j \cdot {}^y\Delta_{i(j-1)} \qquad (2) $$

and ${}^x\Delta_{i(j-1)} = {}^y\Delta_{i(j-1)} = 0$ when $i = j-1$. Hence, ${}^s\Delta_{ij}$, ${}^x\Delta_{ij}$ and ${}^y\Delta_{ij}$ are the total amount of scale, translation in x and translation in y respectively, between $f(i)$ and $f(j)$. This accumulated motion parameter vector is then used to compute the amount of overlap between the contents of the two frames. There are two cases to be considered depending on the scale parameter ${}^s\Delta_{ij}$: (i) ${}^s\Delta_{ij} \leq 1$ and (ii) ${}^s\Delta_{ij} > 1$.

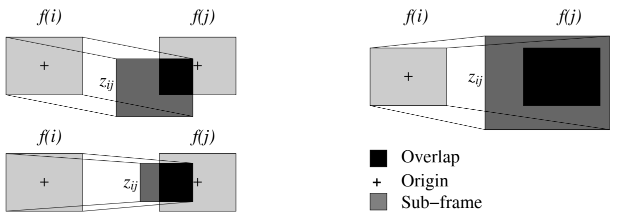


Figure 1: Computing the similarity metric between frames $f(i)$ and $f(j)$. Two cases: (left) ${}^s\Delta_{ij} \leq 1$, (right) ${}^s\Delta_{ij} > 1$ (see text for details).

In each case, assume any frame $f(i)$ has constant width w, height h and area $A = w \cdot h$ with the position $(0,0)$ at its centre. We apply the motion transformation $\vec{\Delta}_{ij}$ to frame $f(i)$ to obtain a sub-frame z_{ij} with a new width ${}^w z_{ij} = w \cdot {}^s\Delta_{ij}$, height ${}^h z_{ij} = h \cdot {}^s\Delta_{ij}$, area ${}^A z_{ij} = {}^w z_{ij} \cdot {}^h z_{ij}$ and its centre at the position $({}^x\Delta_{ij}, {}^y\Delta_{ij})$. The sub-frame conveys the size and position of the contents of frame $f(i)$ relative to frame $f(j)$. In other words, computing the amount of physical overlap between $f(i)$ and the sub-frame z_{ij}, defined by $\phi(i,j)$, is equivalent to computing the amount of overlap between the contents of frame

$f(i)$ and frame $f(j)$. If ${}^{s}\Delta_{ij} \leq 1$, there has either been no scaling or a scale down plus possibly translation in x and y. In this case, we must compute what proportion of $f(j)$ is taken up by the contents of $f(i)$ hence, $\psi(i,j) = \phi(i,j)/A$, as illustrated in Figure 1(left). In case (ii) when ${}^{s}\Delta_{ij} > 1$, there has been a scale up plus possibly translation in x and y so we must compute what proportion of the contents of $f(i)$ are still in $f(j)$, thus $\psi(i,j) = \phi(i,j)/{}^{A}z_{ij}$ as shown in Figure 1(right).

2.2 Graph-based Shot Representation

We now use the similarity metric $\psi(i,j)$ described above to represent each individual shot as a graph. Let us define a graph $G = \{V, E\}$ comprised of a set V of N vertices, $\{v_1, ..., v_N\}$, and a set $E \subseteq V \times V$ of directed weighted edges connecting vertices in V. In a directed graph, each edge also has a direction, so edges (v_i, v_j) and (v_j, v_i), where $i \neq j$, are distinct. The weight of an edge connecting vertices v_i and v_j is defined by $\omega(v_i, v_j)$. A path from v_i to v_m is a set of connected edges $\{(v_i, v_j), (v_j, v_k), ..., (v_l, v_m)\}$ from E. The weight of path $p = \langle v_0, v_1, ..., v_k \rangle$ is the sum of the weights of its constituent edges:

$$\Omega(p) = \sum_{i=1}^{k} \omega(v_{i-1}, v_i). \tag{3}$$

If one or more paths exist from v_0 to v_k the shortest path is defined as the path p with the minimum total weight, $min\{\Omega(p)\}$ [3].

The connectivity of a graph can be represented as an adjacency matrix M in which each element (i,j) represents the edge between vertices v_i and v_j. If there exists an edge (v_i, v_j) then $M_{ij} = \omega(v_i, v_j)$ otherwise $M_{ij} = 0$. Our goal is to form an adjacency matrix for each shot where vertices correspond to individual frames and $\omega(v_i, v_j) = \psi(i,j)$. Hence, the shortest path from the first to the last frame will minimise the amount of overlap between the contents of the representative key frames. If there is no overlap between two frames $f(i)$ and $f(j)$ then by definition, $M_{ij} = \psi(i,j) = 0$. This implies that the key frames corresponding to the vertices in the shortest path must always have some overlap of their contents. In fact, we define a threshold T_{min} to specify the minimum amount of overlap there must be between key frames. Thus an edge (v_i, v_j) only exists if $\psi(i,j) \geq T_{min}$. Additionally, to preserve temporal coherence in the video index, a directed edge (v_i, v_j) can only exist if $f(j)$ succeeds $f(i)$ in the video sequence.

Figure 2 shows a visualisation of the adjacency matrix representing a shot where the camera pans continuously to the right with $T_{min} = 0.2$. The shortest path from the first to the final vertex is $p = \langle 0, 38, 74 \rangle$, with the corresponding key frames shown in Figure 3(left). The weight of the shortest path edges are $\omega(0,38) = 0.432$ and $\omega(38,74) = 0.439$ and the total weight is $\Omega(p) = 0.871$. For comparison, the frames representing the second shortest path $p' = \langle 0, 29, 55, 74 \rangle$ are shown in Figure 3(right) with $\Omega(p') = 1.850$. It can be seen that applying the shortest path algorithm results in less representational redundancy.

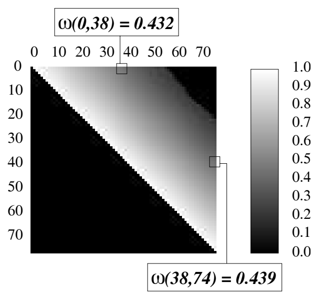
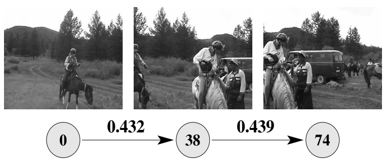


Figure 2: Adjacency matrix representing a panning shot at $T_{min} = 0.2$.



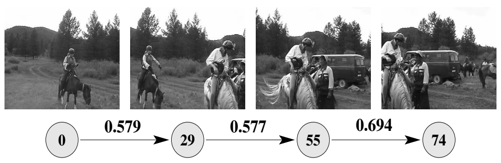


Figure 3: (left) shortest path representation, (right) next shortest path (has redundancy).

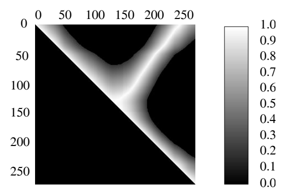
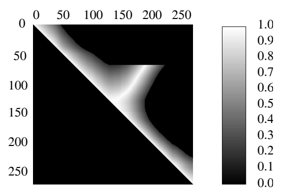



Figure 4: Adjacency matrices for a shot with significant camera motion: (left) any edge exists if $\omega(v_i, v_k) \geq T_{min}$, (right) an edge only exists if $\omega(v_i, v_k) \geq T_{min}, \forall\ i < k < j$.

Figure 4(left) shows the adjacency matrix representing a shot of 289 frames where the camera pans to the right followed by a pan left returning just past the origin. It can be seen where the latter frames start to overlap again with those earlier in the sequence. The shortest path in this graph is $p = \langle 0, 288 \rangle$ and $\Omega(p) = 0.564$. To be an efficient index into a video, the key frames must depict all of the content and convey the temporal order of events in the shot i.e. the camera motion. The key frames here (i.e. 0 and 288) can be seen in Figure 5. In this example, the two selected key frames would not represent any of the video content when the camera pans to the right. We therefore add a final constraint to forming an adjacency matrix. In addition to the earlier condition that an edge only exists if $\omega(v_i, v_j) \geq T_{min}$, the edge (v_i, v_j) only exists if $\omega(v_i, v_k) \geq T_{min}$ for all $i \leq k \leq j$. Figure 4(right) shows the adjacency matrix representing this shot after this constraint has been added and the shortest path is now $p = \langle 0, 49, 100, 214, 245, 288 \rangle$ with $\Omega(p) = 1.363$ which represent all of the shot content, as shown in Figure 5.

In summary, we form an adjacency matrix for each shot where the vertices represent each frame of the sequence and $\omega(v_i, v_j) = \psi(i, j)$. Table 1 outlines the conditions for which a directed edge (v_i, v_j) exists.

Condition 1	$j > i$
Condition 2	$\omega(v_i, v_j) \geq T_{min}$
Condition 3	$\omega(v_i, v_k) \geq T_{min}$ for all $i < k < j$

Table 1: Conditions for which a directed edge (v_i, v_j) exists.

2.3 Finding the Shortest Path

To find the shortest path from a starting to a final vertex, we use the A^* search algorithm which guarantees the shortest path, provided a possible path exists. Central to the A^* algorithm is the use of an evaluation function for ordering the vertices in the search space:

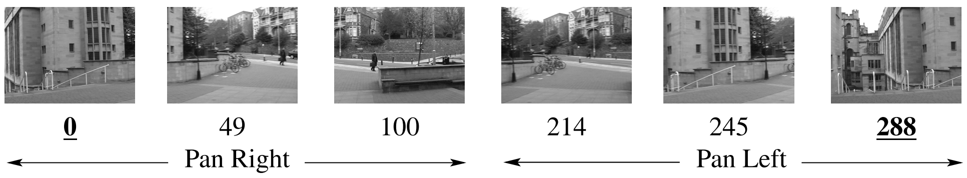


Figure 5: Shortest path key frames for the adjacency matrix in Figure 4(right). The 2 underlined frames denote the shortest path for the adjacency matrix in Figure 4(left).

$$e(v_i) = g(v_i) + h(v_i) \tag{4}$$

where $g(v_i)$ is the actual cost of reaching v_i from the starting vertex and $h(v_i)$ is a heuristic estimate of reaching the final vertex from vertex v_i which must always be an underestimate of the actual cost. It can be shown that an optimistic heuristic h always results in an optimal solution. In our algorithm, $h(v_i)$ is the minimum weight of all existing edges from v_i. Hence, the actual cost from v_i to reach the final vertex will always be greater than or equal to this estimate.

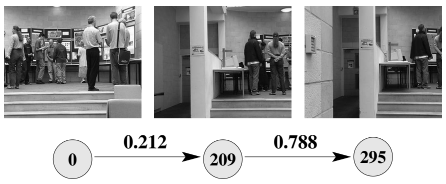
There may exist more than one shortest path in a graph, with the algorithm returning the first shortest path it finds. Given two possible shortest paths p and p' with $\Omega(p) = \Omega(p')$, ideally the one with the smallest spread of edge weights should be chosen, i.e. the path corresponding to the most evenly distributed key frames. Hence, we compute the standard deviation σ of the constituent edge weights on each path and select the path with the smallest σ. In practice, given a shortest path p there rarely exists another path p' with $\Omega(p) = \Omega(p')$ because of floating point resolution. Even so, there may exist another path p' with $\Omega(p') \approx \Omega(p)$ and $\sigma(p') < \sigma(p)$ that would be preferred as the shortest path if $\mid \Omega(p) - \Omega(p') \mid \leq \varepsilon$ and ε is small. However, comparing every possible path with the shortest path is not feasible since the search space is too large. Thus, we need to bias the search for the shortest path towards a path with a small σ of its constituent edge weights. Given a path p with $\Omega(p)$ and $\sigma(p)$, we define a new weight for p by

$$\Upsilon(p) = \Omega(p) + \delta \cdot \sigma(p) \tag{5}$$

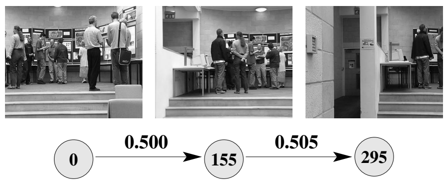
where δ is a weighting. We now choose p if $\Upsilon(p) \leq \Upsilon(p')$, otherwise we select path p'. In practice, in our algorithm, $\delta = 1$ because the standard deviations are small. Using the A^* algorithm, the vertices are ordered in the search space using $e(v_i)$. If there are several vertices for which $e(v_i)$ are approximately equal, we want the search to favour the path through the vertex where the standard deviation of the edge weights is the smallest. Therefore, we define a new heuristic

$$d(v_i) = e(v_i) + \delta \cdot \sigma(e(v_i)) \tag{6}$$

to order the vertices in the search space. Given several possible paths of approximately equal length the search algorithm is biased towards finding the path with the smallest spread of edge weights. Figures 6(a) and 6(b) show comparative key frames representing the shortest paths found using the heuristics defined in (4) and (6) for the same panning shot. It can be seen that, at the cost of a slightly longer path, the latter key frames are more evenly distributed through the shot.



(a) Key frames representing the shortest path p with $\Omega(p) = 1.000$ and $\sigma(p) = 0.288$



(b) Key frames representing the path p' with $\Omega(p') = 1.005$ and $\sigma(p') = 0.0025$

Figure 6: Two paths with approximately the same weight: the one with the smallest spread of its constituent edge weights is selected.

So far, we have used the frames corresponding to the vertices in the shortest path to represent the video. It follows that we will always have a minimum of two key frames to represent each shot. However, when there is little or no camera motion a single key frame could potentially be sufficient. We introduce a second threshold T_{max} which defines the maximum amount of overlap between two key frames. If there are more than two vertices in the shortest path or there are only two vertices with $\Omega(p) < T_{max}$, then the path accurately summarises the video. However, if there are only two vertices v_i, v_j in p and $\Omega(p) \geq T_{max}$, we select the single vertex that best represents the edge (v_i, v_j), i.e. we select the frame corresponding to the vertex v_k with the most overlap with all the other frames between and including frames $f(i)$ and $f(j)$. That is the vertex v_k with the maximum sum of edge weights, $max_k\{\sum_{l=i}^{j} \omega(v_k, v_l)\} \quad \forall \quad i \leq k \leq j$. The values $T_{min} = 0.2$ and $T_{max} = 0.8$ were fixed in all our experiments and to generate the example video abstract shown in Figure 8. These parameters can be set according to user preference.

3 Motion Characterisation

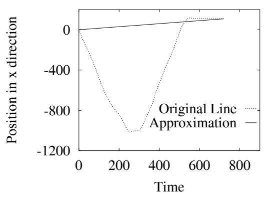
The proposed video abstraction method allows us to efficiently browse and index video through visual content. For each shot, the extracted key frames give a graphic, sequential depiction of the narrative; analogous to a storyboard. However, as well as searching for a shot containing a specific object or scene, a user may also wish to search for a shot which appears to contain a particular type of camera motion, for example, a producer of a wildlife documentary may search an archive for a “fill in” shot which must appear to pan across a particular background (but be less concerned with how the shot was filmed i.e. the actual motion of the camera). Although the key frames portray the camera motion, we extend our algorithm to characterise and textually annotate the apparent camera motion contained within each shot to facilitate this type of search.

To characterise the motion contained within each shot, we use a top-down approach starting with a crude initial guess of the camera motion which is then recursively refined. This results in less sensitivity in the presence of noise, such as camera jitter. The method employed for this refinement is the Douglas-Peucker line simplification algorithm [5]. We divide each individual shot, with motion estimates $\vec{\theta}_1, ..., \vec{\theta}_n$, into segments where the rate of change of the motion that appears in the image is constant. We start by classifying the motion in the x direction and then extend the method to classify all three types of motion.

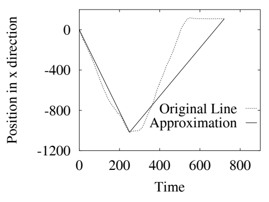
Figure 7(a) shows a dotted line in 2D which is simply the sum of ${}^{x}\theta_n$ against time.

It can be seen that the amount of translation in the negative direction increases, then the camera remains stationary for a period of time followed by a translation in the positive direction until it finally becomes constant for the remainder of the shot. In applying the DP algorithm to characterise the motion, we start with a straight line segment between the two endpoints of the original line as shown in Figure 7(a). This line segment is an initial rough approximation which describes the overall motion contained within a shot. How well a straight line approximates the original line is determined by computing the distances from all intermediate line vertices to that straight line. We compute the Euclidean distance between the estimated amount of motion and the actual amount of motion at time t. If all these distances are less than a specified tolerance the approximation is good. The endpoints are then retained and the other vertices are eliminated. However, if any of these distances exceed the tolerance the point that is the furthest away is taken as a new vertex, sub-dividing the original approximation into two shorter lines as shown in Figure 7(b).

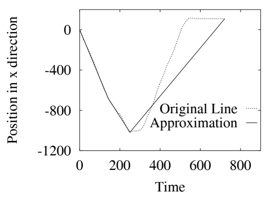
This procedure is repeated recursively until all points are within the specified tolerance and a final approximation is reached as shown in Figure 7(e). The result of the line simplification algorithm is an approximation of the original line where the average rate of change of the motion along each line segment is constant within a specified tolerance. We then use the average rate of change to classify the motion as shown in Figure 7(e). We use the term pan to characterise image translation in the x direction, tilt to characterise image translation in the y direction and zoom to characterise image scale. Each line segment represents a different type of motion or a similar motion but with a different rate of change. For example, in Figure 7(e) the camera pans left over 1.5 times faster for the first pan compared with the second. If we want to identify the changes in speed of the apparent motion the final approximation in Figure 7(e) may be used. However, once the motion has been characterised for each line segment we prefer to merge similar motions together to obtain the final approximation shown in Figure 7(f).



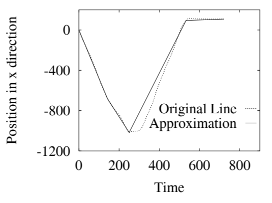
(a) 1st Approximation



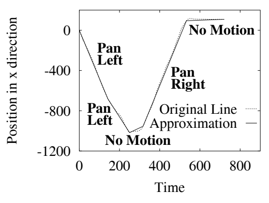
(b) 2nd Approximation



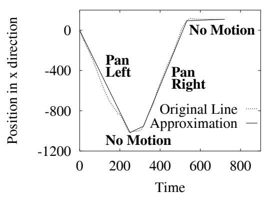
(c) 3rd Approximation



(d) 4th Approximation



(e) Final Approximation using the DP algorithm



(f) Final Approximation after merging similar camera motions

Figure 7: Motion characterisation using the Douglas-Peucker algorithm.

To characterise all three types of motion we must plot and simplify a line in 4D (scale, translation in x, translation in y, time). For this, the rate of change of each motion between each frame pair must be in the same unit of measure. To achieve this we use the amount of area removed by each motion. Hence, given a consecutive frame pair $f(n-1)$ and $f(n)$, we redefine the motion parameter vector $\vec{\theta}_n$ as $\vec{\beta}_n = ({}^s\beta_n, {}^x\beta_n, {}^y\beta_n)$ where

$$ {}^s\beta_n = \begin{cases} A(1 - {}^s\theta_n^2) & \text{if } {}^s\theta_n \leq 1 \\ A(1 - 1/{}^s\theta_n^2) & \text{otherwise} \end{cases} , \quad {}^x\beta_n = h \cdot {}^x\theta_n \quad \text{and} \quad {}^y\beta_n = w \cdot {}^y\theta_n \tag{7} $$

with A, w, and h as defined earlier. We now plot the sum of $\vec{\beta}_n$ against time to obtain a line in 4D to be simplified. Once the original line has been simplified we use the average area removed by each motion to compute the average motion parameters, $\vec{\theta}_{avg}$ for each line segment and use these parameters to classify the type of motion using a rule-based approach. The rules for this classification are shown in Table 2 where $T_{scale} = (max\{w,h\} - 2)/max\{w,h\}$. If more than one rule is satisfied then there has been a combination of motions. There are different rules for classifying a pan and tilt depending on whether there exists any zoom. If there is a zoom in then it must appear to zoom in on an object or scene not contained in the original content for there to be any pan or tilt. Likewise, if there exists a zoom out then none or only part of the original content can be present by the end of the zoom out for there to exist any pan or tilt. The direction of a pan and tilt is assigned simply by examining the sign of the parameter.

Rules	**Motion Classification**
If ${}^s\theta_{avg} \leq T_{scale}$	Zoom out
If ${}^s\theta_{avg} \geq 1/T_{scale}$	Zoom in
If $\neg\exists$ zoom and $\mid {}^x\theta_{avg} \mid \geq 1$	Pan
If $\neg\exists$ zoom and $\mid {}^y\theta_{avg} \mid \geq 1$	Tilt
If $\exists$ a zoom out and $\mid {}^x\theta_{avg} \mid \geq (w - (w \cdot {}^s\theta_{avg}))/2$	Zoom out and Pan
If $\exists$ a zoom out and $\mid {}^y\theta_{avg} \mid \geq (h - (h \cdot {}^s\theta_{avg}))/2$	Zoom out and Tilt
If $\exists$ a zoom in and $\mid {}^x\theta_{avg} \mid \geq ((w \cdot {}^s\theta_{avg}) - w)/2$	Zoom in and Pan
If $\exists$ a zoom in and $\mid {}^y\theta_{avg} \mid \geq ((h \cdot {}^s\theta_{avg}) - h)/2$	Zoom in and Tilt

Table 2: Rules for Motion Characterisation.

Once the motions have been characterised, the key frame extraction algorithm described previously may be applied between motion boundaries rather than shot boundaries and the motions can be textually annotated. If we are more interested in searching for a particular type of motion contained within a shot then the textual annotations can be used to present key frames from shots containing this type of motion and nothing else. Figure 8 shows the camera motion annotations in addition to the key frames.

4 Conclusions

We have presented a novel approach to key frame extraction by organising the similarity of frames within a shot as a graph structure. The frames corresponding to the vertices in the shortest path are then used to generate a content-based video index analogous to a storyboard. This also allows us to characterise and annotate the perceived camera motion within each shot. This facilitates a further level of indexing and searching video content.

There is no absolute measure for the quality of an abstraction. Ultimately, the effectiveness of an approach can only be evaluated by users of a video library in which the

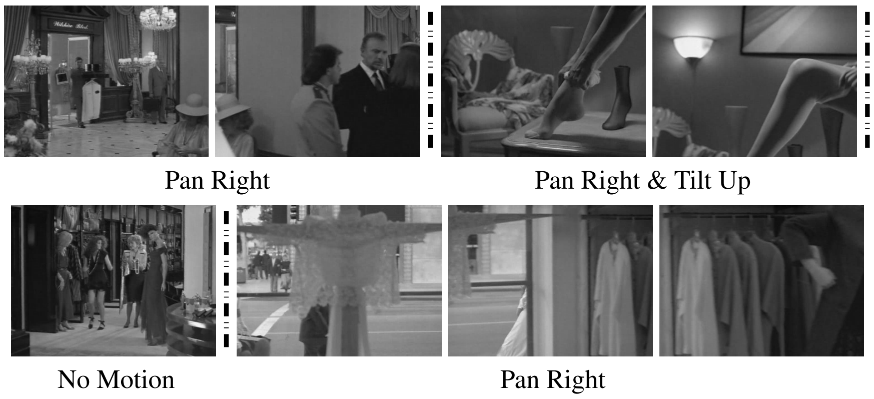


Figure 8: Excerpt from the Pretty Woman video index - shot boundaries are also shown.

system is implemented. Here, we have shown some example video abstracts to demonstrate our proposed method. Better subjective judgement can be made by viewing more video shots and key frames on-line[1]. The abstraction of video content is a very complex theme. In this work we have concentrated on using camera motion as one feature that changes scene content. In future work we will also consider semantic changes in the scene due to object motion.

Acknowledgements

The first author thanks Moving Image Research Ltd. for funding received during this work.

References

[1] J Boreczky and L Rowe. Comparison of video shot boundary detection techniques. In *SPIE volume 2670*, pages 170–179, 1996.

[2] M Christel, A Hauptmann, A Warmack, and S Crosby. Adjustable filmstrips and skims as abstractions for a digital video library. In *Advances in Digital Libraries*, pages 98–104, 1999.

[3] T Cormen, C Leiserson, and R Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[4] D DeMenthon, V Kobla, and D Doermann. Video summarization by curve simplification. In *ACM Multimedia*, pages 211–218, 1998.

[5] D Douglas and T Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[6] D Gibson, N Campbell, and B Thomas. Visual abstraction of wildlife footage using GMM and MDL criterion. In *ICPR*, pages 814–817, 2002.

[7] R Lienhart. Comparison of automatic shot boundary detection algorithms. In *SPIE volume 3656*, pages 290–301, 1999.

[8] R Lienhart, S Pfeiffer, and W Effelsberg. Video abstracting. *Communications of the ACM*, 40(12):54–62, 1997.

[9] S Porter, M Mirmehdi, and B Thomas. Detection and classification of shot transitions. In *12th BMVC*, pages 73–82. BMVA Press, 2001.

[10] P Torr and A Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *CVIU*, 78(1):138–156, 2000.

[1] http://www.cs.bris.ac.uk/home/porter/spath/video.html