An Evolutionary Approach to Automatic Video Editing

Tinghuai Wang, Andrew Mansfield, Rui Hu, John P. Collomosse
Centre for Vision, Speech and Signal Processing
University of Surrey
Guildford, UK
{tinghuai.wang, r.hu, j.collomosse}@surrey.ac.uk

Abstract—Digital video has become affordable and attractive
to home users, but skill and manual labour are still required to
transform amateur footage into aesthetically pleasing movies.
We present a novel algorithm for transforming raw home video
footage into concise, temporally salient clips. We interpret the
sequence of editing operations applied to footage as a ‘pro-
gram’ comprising cutting, panning and zooming constructs.
We develop a Genetic Programming (GP) framework for rep-
resenting and evolving such programs. Under this framework,
the search for an aesthetically pleasing video edit becomes a
search for the optimal genetic program. Our aesthetic criterion
promotes the inclusion of people in shots, whilst penalising
rapid shot changes or shot changes in the presence of camera
motion. We present results on some representative home videos.

Keywords—genetic programming, consumer video, editing.

I. INTRODUCTION

Falling hardware costs have prompted an explosion in
casual digital video capture by domestic users. However,
once captured, this video is infrequently accessed and often
lies dormant on the user’s hard disk. One explanation is
that raw home video requires substantial editing to be
comparable, in terms of aesthetics and succinctness, with
professional footage. For example, amateur home videos
often contain lurching pans as the camera operator switches
subject, and subjects often suffer from poor framing. This
can lead to videos that are not enjoyable to watch, despite
the periods of interest within them.

We present an algorithm to breathe life into users’ video
repositories by editing raw video footage into salient, aes-
thetically pleasing clips. We are concerned with three types
of editing operation:

• Cut – frames are removed to shorten the video.
• Zoom – frames are spatially cropped to focus attention.
• Pan – view-port moves to follow a subject.

These operations may be applied to source video, with ap-
propriate parameters and in a specific sequence, to produce
an edited video. We interpret this sequence of operations
as a program, and state finding the “best” program under
some aesthetic criterion (Sec. III) to be equivalent to finding
an optimal edit sequence for a particular home video. We
contribute both a novel representation for such programs,
and a novel method for searching the space of programs
using a Genetic Programming (GP) framework.

GP is an evolutionary optimization method [1]. Similar
to the more common Genetic Algorithm (GA), GP creates a
population of putative solutions (individuals) and “breeds” the
best individuals together to produce successively im-
proved generations of solutions [2]. With GP, however,
the solutions are parse trees (programs) rather than points
in a fixed-dimensional search space. GP is well suited
to the problem of video editing, since the number and
order of editing operations may vary greatly between video
sequences. Furthermore, evolutionary algorithms such as
GP are well suited to large search spaces in which the
combination of distinct yet locally optimal solutions (e.g.
partial video edits) are likely to yield globally preferable
solutions. To the best of our knowledge, GP has not been
previously applied to the automated editing of home videos.

Section II outlines our GP representation of an edit
sequence. Our optimization process and aesthetic measure
are described in Section III. We present and discuss the
results of applying our algorithm to representative home
videos in Section IV, concluding in Section V.

A. Related Work

Automated video editing is closely related to research on
video summarisation, which has gained momentum in recent
years. Many such algorithms rely on shot detection to extract
representative key-frames from video [3]. Such techniques
are well suited to movies exhibiting frequent cuts between
shots, but are ill-suited to home videos (typically captured
as a single lengthy shot). An alternative is [4] who model
video as a trajectory through a high-dimensional appearance
space, cutting key frames at points of high curvature.

Techniques that summarise video into shorter videos by
‘cutting’ frames have been proposed. Lienhart defines a
visual quality metric, creating an automatic digest of home
videos by selecting portions of video with good quality and
inserting transition effects [5]. Girgensohn et al.’s semi-
automatic “Hitchcock” system [6], [7] is similar to [5], but
defines quality in terms of camera stability; we incorporate
a similar cue in our work. Hua et al. propose an automatic
video editing system that seeks to cut video to synchronise
motion in selected sub-shots with music tempos [8]. Atten-
tion models for video summarization were studied in [9],
[10], integrating visual, auditory, and linguistic cues.
Most recently researchers have looked beyond cutting, to the framing of video content (e.g. zooming/cropping). Al-Hames et al. controlled multiple cameras to select and zoom-in on meeting participants to “direct” a live video stream of a meeting [11]. Hospedales and Williams recently explored Bayesian networks to learn director preferences for similar real-time editing of streamed video [12]. Such techniques necessarily make temporally local editing decisions. Our GP approach performs global optimization over all frames of a pre-captured video.

II. REPRESENTATION OF VIDEO EDITS

We represent an editing sequence as a program, specifically as a parse tree in which nodes act as operators that either manipulate or combine video fragments to form the output clip. In this section we develop our tree representation.

A. Cutting

We begin by considering the basic cut operation, in which frames are removed from a video sequence in order to enhance its interest or aesthetic appeal. Under our tree representation, non-terminal nodes in the tree act as “split” operators that divide a video fragment into two sub-parts, passing the resulting fragments to their children. The point of division is governed by an operand on the node [0,1] representing the normalised length of the input video fragment. Thus split has three children: a child constant node specifying the real-valued division point, and two child operator nodes. Video fragments may be divided recursively by further non-terminal split nodes. Terminal nodes may then either “discard” a fragment, or “take” it i.e. incorporate it in the output sequence. The final edited video sequence is obtained via in-order traversal of the parse tree, appending video fragments as take nodes are encountered. We find linked lists of frames to be an appropriate data structure for managing fragments.

The split, take and discard operators form a basic editing system with cutting functionality. Fig. 1a provides an illustrative example of a terminal set comprising split, take and discard operators. It is easy to prove the sufficiency of this representation. Taking an unedited sequence of arbitrary length we can, by creating a tree comprising the right arrangement of split nodes, split the sequence into its individual constituent frames. We can then create any possible output sequence by applying take and discard operators.

B. Panning and Zooming

In addition to cutting (temporal cropping) we enable a degree of freedom in the framing of video content through a spatial cropping mechanism. The effect of the cropping mechanism is to define a window around a portion of the frame, and then to scale that region to full frame size when outputting the edited video. When the window is appropriately positioned, this has the effect of “zooming” in on interesting content (e.g. a person) and so improving the framing of the scene.

We implement this operation by modifying the take terminal operator defined above. By specifying the cropping window as operand on the take node, we are able to specify the region of interest for cropping over each video fragment incorporated into the final edited video. Absence of cropping becomes a degenerate case; the crop window is simply positioned over the entire frame. To avoid visual artifacts we constrain the aspect ratio of the window to match the frame. The window’s position is thus defined by operand [x, y, σ]; centre (x, y) and a uniform scale factor σ. Specifying the cropping window geometry in this manner also reduces our search space.

Although camera pans are technically achievable by splitting video into individual frames, and carefully specifying...
crop windows, this is not practically achievable by our
GP optimization. Instead, we explicitly incorporate camera 
“panning” through an extension of the cropping mecha-
nism. We extend the take operator again, to now have 
two operands: a crop window at the starting frame, and a 
crop window at the ending frame of the fragment. When 
outputting the final editing video, the window parameters 
are linearly interpolated between the start and end frames of 
each video fragment. Cropping thus becomes a degenerate 
case of panning, where the start and end cropping windows 
are identical. The take terminal node thus has six constant 
node operands \([x_s, y_s, \sigma_s, x_e, y_e, \sigma_e]\), where subscripts \(s\) 
and \(e\) indicate start and end frame respectively. As with 
the division point on the split non-terminal operator, these 
parameters are represented by normalised constant terminal 
nodes. Parameters \((x, y)\) are normalised to frame width and 
height, while \(\sigma\) is normalised to range from half frame size 
\((0)\) to full frame size \((1)\). Figure 1b gives an illustrative 
example.

C. Concatenation of Video Fragments

Optimizations frequently result in parse trees that split 
video into many small fragments, with similar but slightly 
different cropping windows. This can result in a distracting 
flicker and instability in the final video. To mitigate against 
this, we perform some interpolation on window parameters 
when video fragments are concatenated by the split non-
terminal operator.

Suppose two fragments \(F_1, F_2\), of durations \(t_1, t_2\), and 
with window parameters \(\omega_1 = [x_s, y_s, \sigma_s, x_e, y_e, \sigma_e]\) and 
\(\omega_2 = [u_s, v_s, \tau_s, u_e, v_e, \tau_e]\) are to be concatenated. A 
straightforward approach is to replace the end and start 
windows of \(F_1\) and \(F_2\) respectively with an interpolated 
window \(\omega_f\):

\[
\omega_f = \frac{t_1}{t_1 + t_2}(\omega_2 - \omega_1) + \frac{t_2}{t_1 + t_2}\omega_2.
\]

However, when a substantial discard has been made be-
tween fragments, it may be more appropriate to permit a 
discontinuity in the window geometry i.e. leaving \(\omega_1\) and 
\(\omega_2\) unmodified.

Our solution is to update the windows using a weight 
derived from the temporal distance \(d\) between the start and 
end of \(F_2\) and \(F_1\) respectively:

\[
\omega_1 \leftarrow \omega_1 + e^{-kd}(\omega_2 - \omega_1) \\
\omega_2 \leftarrow \omega_2 + e^{-kd}(\omega_1 - \omega_2)
\]

where \(k = 0.5\) provides interpolation over cuts up to \(d \leq 10\) 
frames (i.e. \(\sim 0.5\) second duration).

III. GENETIC SEARCH FOR AN OPTIMAL EDIT

We first describe the fitness function by which we measure 
the aesthetics of a video edit, and then provide the specifics 
of our GP optimization process.

Figure 2. Video meta-data is extracted as a pre-process; we measure 
interest through detection of people (left), and inter-frame motion via optical 
flow (right). Result V4 (Sec. IV-B)

A. Fitness of a Video Edit

Our fitness measure for a putative video edit seeks to 
estimate both the level of interest, and the aesthetics of the 
edited output video.

Our fitness function incorporates two terms for measuring 
interest; the total captured interest and the average interest 
captured over selected frames. The first term promotes com-
pleteness of interests selected from the raw video footage, 
while the second term promotes removal of “interest sparse” 
frames to produce feature rich video. The second term also 
encourages subjects of interest to be framed such that they 
occupy most of the scene. With respect to aesthetics, Arijon 
[13] notes that frequent short-term cuts within a sequence 
are unpleasant for the viewer. In some situations such cuts 
are appropriate, e.g. fast action shots, but these are too 
specific for general home video editing. Scene and camera 
motion should also be minimal at the points where shot 
boundaries are introduced. To incorporate these preferences, 
we introduce penalty terms for short cut sequences or cuts 
made in the presence of large-scale motion.

In line with these heuristics, we specify the following 
fitness function over all frames \(\{E_1, E_2, ..., E_N\}\) included 
in the edited video:

\[
F(E) = \frac{P^{SC}}{N} \sum_{i=1}^{N} \left( w_1 \cdot CI(E_i) + w_2 \cdot \frac{CI(E_i)}{N} \right) e^{-\gamma M(E_i)}
\]

Where \(CI(.)\) is a normalised operator evaluating the 
captured interest within a frame (subsection III-A1). \(M(.)\) 
is a sum of the optical flow vector magnitudes within a 
frame (Figure 2, right). \(SC(.)\) is a count of the number of 
short fragments within the edited sequence (below 0.5 
second), and constitutes a penalty term on short clips when 
\(0 < P < 1\). The pairs of parameters \(P, \gamma\) and \(w_{1,2}\) are 
weights on the aesthetics and interest terms respectively, 
and may be adjusted to user preference. The latter weights 
are empirically selected to find the trade-off between the 
completeness and richness of captured interests. We give 
typical values with results in Section IV.

1) Captured Interest: Home video is predominantly used 
to capture life events, and people (e.g. friends and family) 
are frequently the objects of interest in such footage. In our
system we correlate interest with the presence of people in a shot. Specifically, the greater the viewing area occupied by images of people, the more “interesting” and thus optimal our video is deemed to be. Person detection can be achieved in a number of ways, such as human face detection [14] and upper-body detection [15]. We opt for the latter, since face detection systems tend to perform poorly over the wide variations in pose, scale and lighting typical in home movies. Figure 2 (left) shows application of a popular cascade based person detector [15] to typical source footage. We obtain our value for \( CI(.) \) by averaging the probabilities of pixels belonging to a person over the cropped window within the editing frame.

More sophisticated definitions of interest exist — for example considering temporal [16] and auditory [9] cues, or even models of linguistic semantics [10]. Although other normalised measures might be substituted, we find our measure suitable for the domain of home video. Our method also has the advantage that \( CI(.) \) and \( M(.) \) may be efficiently pre-computed by finding bounding regions for people in each frame of video, and intersecting those polygons with the cropping window to obtain the area of overlap during optimization. However we emphasise that our technical contribution is not in the interest measure \textit{per se}, but rather in demonstrating the feasibility of a GP framework for identifying optimal video edits.

B. GP Optimization

Ideally a GP operator set should fulfil three criteria identified in [17]. First, any operator should return a value on any input, called evaluation safety. Second, the operator set should be sufficient; it should have enough expressive power to generate any possible solution to the problem. Third, the operators should be type consistent, i.e., return values of the same type so as they can be freely interchangeable in breeding.

Criteria one and two are satisfied (Section II) however our constant terminal nodes return a different type \((\mathbb{R})\) to that of the non-constant terminal and non-terminal nodes (video). This breaks the third condition of “type unity”. Koza et al. suggest use of a \textit{constrained semantic structure} in such cases; effectively performing separate cross-over and mutation for constant and non-constant nodes [18]. We follow this strategy in subsection III-B3.

An overview of the optimization is shown as a flowchart in Fig. 3. We begin by randomly generating a large set of programs (or “individuals”, collectively referred to as the “population”). Each individual represents a putative solution, in the form of our edit tree representation (Section II). GP is an iterative process, in which pairs of individuals are selected from each generation stochastically — with a bias to fitness — and combined via a breeding process of “cross-over” and “mutation” to create a population for the next generation. Thus at each iteration, the fitness of all individuals in the population must be evaluated using eq. (3) to enable fitness-proportionate selection. Optimization can be halted when maximum fitness within the population shows negligible improvement over several successive generations.

1) \textit{Initialization:} Individuals within the first generation are initialised independently and randomly. In our experiments we use a generation size of 500. An individual’s parse tree is constructed recursively by picking a node from the set of possible operators \{\textit{take, discard, split}\}. Operators requiring constant operands will have appropriate child nodes created. In the case of a non-terminal operator being picked, further operators must be generated for the remaining child operands. The process recurses in a depth first manner until a terminal operator is generated. When choosing an operator for a non-constant node, the decision on type of node is made stochastically according to depth of recursion. Non-terminal nodes are less likely to be generated at deeper points on the tree. When generating a constant node, a value is picked uniformly at random, in range \([0,1]\) as all operands are normalised by design (Section II).

2) \textit{Elitism:} At each iteration, the top \( \sim 1\% \) fittest individuals pass through directly to the next generation. To

![Figure 3. Schematic of the GP optimization algorithm.](image-url)
maintain population diversity, \( \sim 5\% \) of the next generation is
reinitialised at random. The remainder of the next generation is
bred from the current, using the processes of cross-over
and mutation.

3) Cross-over: Cross-over is the mechanism by which
elements of parent individuals are mixed to produce off-
spring for the next generation. In GP this is achieved by
constructing two new parse trees using portions of the parent
parse trees.

Given two parents \( A \) and \( B \) we create two new individuals
\( N_1 \) and \( N_2 \), initially by duplicating \( A \) and \( B \). We then tra-
verse \( N_1 \) in a depth-first manner, simultaneously traversing
\( N_2 \) to create a one-one correspondence between nodes in
\( N_1 \) and \( N_2 \). Where such a correspondence is possible (i.e.
moves are possible from a parent node to a child node in
both trees), we may swap the node and subtree below it in
\( N_1 \) with the corresponding node and its subtree in \( N_2 \).
The swap is made with probability 0.2 in our experiments.
Figure 4 illustrates the process.

As our representation lacks type unity, evaluation prob-
lems will be encountered if constant nodes are substituted
with non-constant nodes during swapping. Thus when a
child node is swapped, its constant nodes are carried from
the source to the destination tree in situ (as if logically
part of the child node). Any non-constant operands are then
recursively descended and swapped stochastically as before.

Mutation introduces diversity into the population, en-
abling exploration of the solution space. Again, due to
the lack of type unity we must mutate constant and non-
constant nodes using a separate mechanism. In the case of
constant nodes, we iterate through nodes in \( N_1 \) and \( N_2 \), and will generate
an entirely new subtree for a node (using the method of
subsection III-B1). Figure 4 illustrates this process. The
probability of making such a mutation is 0.1 for all our
experiments.

IV. RESULTS AND DISCUSSION

To evaluate the video editing system, we captured home
videos covering a variety of events. Here we present the
results of five videos (\( V_1 \) – \( V_5 \)). In \( V_1, V_2 \) we disabled
our zooming/panning mechanism to show the effects of
the cutting operator alone. In \( V_3 - V_5 \) the full system is
evaluated.

A. Cutting Only

Figure 5 depicts frames from our source videos, regularly
sampled along a time-line running left-right. The presence
of blue below the time-line indicates detection of interests
(people), and red indicates portions of the source video time-
line that were selected and concatenated to create the editted
output.

The \( V_1 \) and \( V_2 \) source footage depicts family members
at the park. In \( V_1 \) the cameraman periodically becomes
distracted and points the camera at the floor or at un-
interesting objects. The system has identified contiguous
blocks of interest in the video, and cut three sections of
the source time-line for concatenation into the final edited
video. Virtually all of the interest is captured in a minimal
number of cuts. In \( V_2 \) cuts have been made not only to
maximise the density of interest in the clip, but also to
prohibit rapid cutting in frames where detection of people
is intermittent. This is frequently the case using [15] when
people’s backs are turned to the camera, or are of small

\(^1\)The source and edited videos are included in the supplementary material
Figure 6. Optimization results for videos $V_1 - V_5$, plotting maximum fitness in each generation.

Figure 7. Optimization results for videos $V_1 - V_5$, plotting standard deviation (fitness diversity) for each generation.

For these results, system parameters were set such that the ratio $w_1 : w_2$ was $1 : 10$, $P = 0.99$, and $\gamma = 10^{-5}$. Figures 6,7 show convergence with negligible change in population fitness or diversity after $\sim 20$ iterations.

B. Cutting, Zooming and Panning

For videos $V_3 - V_5$ we re-enabled the zooming/panning mechanism to run the system with full functionality. Figure 5 shows the cuts made in the source video to isolate “interesting” parts of the time-line. Again, source video frames exhibiting a negligible or intermittent response from the interest detector have been cut. Figure 5 also shows the position of the cropping window (red box) within frames. Footage within the window is scaled to create the rendered output footage shown in Figure 8. In the cases of $V_3$ and $V_4$, a crop window is created around the main subject which pans to follow the movement of the subject in the video. In the case of $V_5$ a cropping window is also introduced to zoom in and improve framing of the subject; however since the camera is already panning to follow the subject, no additional panning is introduced. For these results, system parameters were set as in subsection IV-A, but with ratio $w_1 : w_2$ set to $1 : 100$. Figures 6,7 again show quick convergence, with negligible change in population fitness or diversity after $\sim 50$ iterations. For our experiments we ran the optimizations up to 1500 generations (300 are shown).

Figure 8. Final edited clip results from footage $V_3 - V_5$. Upper strip: Blue box indicates interest detection, red box indicates cropping window. Lower strip: Footage is rendered from within the red window to output the final clip.

V. Conclusion

We have presented a novel tree representation for home video editing, suitable for use in a Genetic Programming (GP) optimization framework. Our representation incorporates cutting, zooming and panning operations. Uniquely, we search for a globally optimal video edit using GP, maximising both aesthetics and interest within the final clip. Our measures for aesthetics are grounded in common directing practice, and our measure for interest is based on the presence of people; the most common subject of interest for home videos.

We have demonstrated the efficacy of our approach over some representative examples of home video footage. Our system quickly converges to an acceptable edit sequence, requiring $\sim 50$ generations / minute of source video. To capture the subjectivity of video aesthetic, our fitness function is governed by user parameters weighting desire for objects of interest against frequency of cuts, and motion. The short optimization times enable user experimentation to taste. This paper has focused on GP optimization as a means for generating edit decisions. It has not explored the visual rendering of those edits. Transition effects might be
Figure 5. Evaluating our system over videos (V1 – V5), from top to bottom. For V1 and V2 we disabled zooming/panning. A time-line (in frames) runs from left to right, annotated in blue to show presence of interest, and in red to show segments of video selected for output by our editing process. Frames have been sampled from the source video at regular intervals; the blue box indicates the areas of interest detected. In the case of V3 – V5 the red box shows the cropping window.
introduced e.g. cross-fades when cutting. Future work may explore alternative cropping operators, for example seam-carving, to accommodate multiple disjoint regions of interest within a frame.

Although our fitness measure lacks the sophistication of [9], [10], we find it suitable for demonstrating value in our GP editing framework, and for the purposes of general home video editing. Extensions to this measure are a possible route for future work. A higher level temporal constraint (e.g. preferring alternating cuts between subjects during dialogue) might further enhance the aesthetic terms within fitness function. However, within a subject domain as broad as home video, care should be taken to draw a sensible compromise between the complexity of editing heuristics and the generality of footage that may be processed.

ACKNOWLEDGEMENT
We are grateful to HP Labs Bristol for funding this research under HP’s IRP programme. Thanks to David Slatter, Phil Cheatle and Darryl Greig for fruitful discussions.

REFERENCES