IMPROVING AUTOMATIC DUBBING WITH SUBTITLE TIMING OPTIMISATION USING VIDEO CUT DETECTION

Jindřich Matoušek and Jakub Vít

University of West Bohemia
Faculty of Applied Sciences, Dept. of Cybernetics
Univerzitní 8, 306 14 Plzeň, Czech Republic

ABSTRACT

This paper presents improvements to an automatic dubbing system in which text-to-speech technology is used to synthesise speech from subtitles. Spring-based subtitle timing optimisation was proposed to reduce the need for speeding up synthetic speech to fit it into corresponding subtitle slots. Video cut detection algorithm was also introduced, and the cuts were then used to prevent stretching subtitles across the cuts. Results show that after the optimisation smaller speeding-up factors are applied on synthetic speech while keeping optimised subtitle start and end times close to original positions.

***Index Terms*—** text-to-speech, automatic dubbing, video cut detection, subtitle timing optimisation, spring-based model

1. INTRODUCTION

In this paper, improvements to a baseline automatic dubbing system introduced in [1] are presented. The automatic dubbing system is developed within the ELJABR[1] project and is aimed at TV watchers with minor hearing impairments like seniors (people over 50 years of age), people with dyslexia or minor mental retardation and also for people who simply are not able to follow the complex sound structure of modern TV programmes—they do mind the lower intelligibility of real dialogues, the alternation of very fast and normal speech, rapid changes in both voice quality and identity, or possibly also music or effect component present in the original audio track. The automatic dubbing system exploits text-to-speech (TTS) technology as it produces speech from subtitles. The resulting TTS-generated track contains only (synthetic) speech without background music, effects, emotions, etc.

The problem of the baseline system was that, employing a general unit-selection TTS system (a Czech TTS system ARTIC in our case [2]), synthesised speech exceeded the corresponding subtitle slots, resulting thus in significant audio/video synchronisation issues [1]. A solution can be to speed up (or, time-compress, respectively) the output speech signal in order to fit it into the given subtitle slot, e.g. by means of a time-scale modification algorithm like WSOLA [3]. In this way video and audio tracks are kept synchronised. However, as shown in Section 5, many subtitles are then speeded up with high factors. As a result, very fast speech would be synthesised which is in conflict with our objective to produce a less-dynamic and more intelligible audio track. In this paper, another solution, to modify subtitles timing utilising a spring-based model, is proposed.

Support for this work was provided by the Technology Agency of the Czech Republic, project No. TA01011264, and by the grant of the University of West Bohemia, project No. SGS-2010-054.

[1]ELJABR is a Czech acronym for "Elimination of the Language Barriers Faced by the Handicapped Watchers of the Czech Television".

A similar project was presented in [4] where the audio track was created from subtitles using TTS technology in a stand-alone box. Therefore, the synthetic speech is listened together with the original audio track. On the other hand, in our project the TTS-based dubbing system will be operated in the Czech Television, a public service broadcaster, and an alternative audio track will be delivered to ordinary home TV sets. The mixing of the TTS-generated and the original audio track will be avoided because every TV watcher will be able to choose the track according to his/her preference.

The paper is organised as follows. Experimental data are presented in Section 2. The improvements to the baseline automatic dubbing system are described in the next sections—an algorithm for automatic video cut detection in Section 3, and a utilisation of the detected cuts for spring-based subtitle timing optimisation in Section 4. Finally, results of the optimisation are shown in Section 5, and conclusions are drawn in Section 6.

2. EXPERIMENTAL DATA

In TV broadcasting, *subtitles* (also known as *closed captions*, or subtitles for the hearing impaired) could be viewed as an extra service, which supplements the standard video and audio tracks with a transcript (although not always verbatim) of the audio track. The subtitles present the only source of information that could be exploited when generating the supplementary audio track for TV broadcasting (Czech Television currently broadcasts the subtitles using a teletext page 888).

At present, the EBU Subtitling data exchange format is used for storing subtitles of particular programmes in binary data files. A subtitle file consists of a sequence of subtitles; each subtitle is described by its text, start and end times, position on the screen, etc.

Ten different TV programmes with manually detected video cuts (both for the purposes of setting-up the parameters of video cut detection algorithm and its evaluation in Section 3) shown in Tab. 1 were used in our experiments.

3. VIDEO CUT DETECTION

In the baseline system, only audio track and subtitles were exploited for automatic dubbing. Information from another medium, the video track, was not utilised. To optimise subtitle timing, knowledge of cuts, positions in a video where the video content changes, will be utilised to prevent subtitles from stretching across the cuts.

Let video be described as a sequence of $N^{(\mathrm{f})}$ frames f_k, $k = 1, \dots, N^{(\mathrm{f})}$. Each frame f_k is an $N^{(\mathrm{r})} \times N^{(\mathrm{c})}$ matrix of pixels, in which $N^{(\mathrm{r})}$ is a number of rows and $N^{(\mathrm{c})}$ is a number of columns. Using the RGB colour model, each pixel $p_{x,y}$ can be represented as a

Table 1. List of TV programmes (movies in our case) used in our experiments. The superscripts stand for genres (according to IMDb): action (1), adventure (2), animation (3), documentary (4), drama (5), fantasy (6), history (7), sci-fi (8), thriller (9), western (10). The columns denote title, ID, number of subtitles, subtitle time in minutes, and number of cuts.

Title	ID	# stl.	Stl. time	# cuts
12 Angry Men^{5}	P01	1304	75	357
300^{1,6,7}	P02	938	35	1502
The Fifth Element^{1,8,9}	P03	1022	41	2388
Frona^{5}	P04	903	42	651
Futurama: Bender's Game^{3,2,1}	P05	1148	53	1181
Gran Torino^{5}	P06	1360	55	1357
Once Upon a Time in the West^{10}	P07	749	27	1364
Pirates of the Caribbean I^{1,2,6}	P08	1174	50	2697
Planet Earth (Episode 2)^{4}	P09	333	15	369
X-Men: First Class^{1,2}	P10	1250	49	2315
Total		10181	442	14181

combination of the RGB components. In our experiments, different weights were used to compensate for the different effect of the RGB components on the human eye [5]

$$p_{x,y} = 0.3R_{x,y} + 0.59G_{x,y} + 0.11B_{x,y}, \tag{1}$$

where $R_{x,y}$, $G_{x,y}$, and $B_{x,y}$ are the values of each component for pixel $p_{x,y}$, i.e. for pixel with (x, y) coordinates. To indicate a pixel of a concrete frame f_k, the notation $p^{(k)}_{x,y}$ will be used hereafter.

Two-phase cut detection algorithm was utilised. Firstly, in the *scoring phase*, each pair of consecutive video frames is given a score that represents the similarity/dissimilarity between these two frames. Then, in a *decision phase*, all previously calculated scores are evaluated and a cut is detected if the score is considered high.

3.1. Scoring Phase

Two *scoring methods* were used in our experiments:

Sum of absolute differences (SAD). Consecutive frames f_k and f_{k-1} are compared pixel by pixel, summing up the absolute values of the differences of each two corresponding pixels

$$SAD_k = \sum_{x=1}^{N^{(\mathrm{r})}} \sum_{y=1}^{N^{(\mathrm{c})}} \left| p^{(k)}_{x,y} - p^{(k-1)}_{x,y} \right|, \qquad k = 2, \ldots, N^{(\mathrm{f})}. \tag{2}$$

Histogram differences (HD). A colour histogram is a representation of the distribution of colours in a frame, i.e. a colour histogram represents the number of pixels that have colours in each of a fixed list of colour ranges, that span the frame's colour space. In our case, a histogram can be viewed as a function, or a table, $h^{(i)}$ that counts the number of pixels that have colour in each of the fixed colour ranges (known as bins). Thus, if we let $N^{(\mathrm{b})}$ be the total number of bins, the histogram $h^{(i)}$ meets the following conditions:

$$N^{(\mathrm{r})} \cdot N^{(\mathrm{c})} = \sum_{i=1}^{N^{(\mathrm{b})}} h^{(i)}, \tag{3}$$

where $N^{(\mathrm{r})} \cdot N^{(\mathrm{c})}$ is the number of pixels in each frame. The HD scoring method detects how much the distribution of colours changed from frame to frame as it computes the difference between the histograms $h^{(i)}_k$ of two consecutive frames f_k and f_{k-1}

$$HD^{(i)}_k = h^{(i)}_k - h^{(i)}_{k-1}, \qquad i = 1, \ldots, N^{(\mathrm{b})}, k = 2, \ldots, N^{(\mathrm{f})}. \tag{4}$$

A single value

$$HD_k = \sum_{i=1}^{N^{(\mathrm{b})}} |h^{(i)}_k - h^{(i)}_{k-1}| \tag{5}$$

can then represent a rate of colour distribution change between two consecutive frames.

3.2. Decision Phase

In the *decision phase*, a *threshold-based method*, in which the scores between each consecutive frames are compared to a threshold, was employed. If the score is higher than the threshold, a cut is declared. In order to adapt the threshold to the properties of the current frames f_k and f_{k-1} and their neighbours in a video, the context of $2N^{(\mathrm{n})}$ frames with scores s_k (SAD or HD)

$$C_k = s_{k-N^{(\mathrm{n})}}, \ldots, s_{k-1}, s_{k+1}, \ldots, s_{k+N^{(\mathrm{n})}} \tag{6}$$

was used.

Now, the cut detection algorithm can be introduced:

1. For each frame f_k, $k = N^{(\mathrm{n})} + 1, \ldots, N^{(\mathrm{f})} - N^{(\mathrm{n})}$, a vector C_k of scores for the neighbouring frames is computed.
2. The following rules are applied to the score s_k of the current frame f_k and to the neighbouring scores in C_k:
$$s_k > T_1 \tag{7}$$
$$\max\{C_k\} < T_2 \cdot s_k \tag{8}$$
$$\mathrm{avg}\{C_k\} < T_3 \cdot s_k, \tag{9}$$
where $\max\{C_k\}$ and $\mathrm{avg}\{C_k\}$ are maximum and average scores, respectively.
3. If all rules are satisfied, a cut is declared to be between frames f_k and f_{k-1}.

T_1, T_2, and T_3 are thresholds. They must be determined for each score s_k (i.e. SAD and HD) separately. Thresholds values used in our experiments were determined by a grid-search algorithm within 5-fold cross-validation using 80 % of all frames from each movie in Table 1. As shown in Tab. 2, the values are relatively stable over the runs of cross-validation. The best results were achieved for context size $N^{(\mathrm{n})} = 7$ or 8.

Table 2. Thresholds values for SAD score.

Score	min	max	avg.	std.
T_1	12.500	12.688	12.538	0.084
T_2	0.669	0.686	0.675	0.007
T_3	0.548	0.514	0.529	0.016

3.3. Cut Detection Results

The following three measures can be used to measure the quality of a cut detection algorithm (see e.g. [6]):

- *Recall* is the probability that an existing cut will be detected:

$$V = \frac{C}{C + M} \tag{10}$$

- *Precision* is the probability that an assumed cut is in fact a cut:

$$P = \frac{C}{C+F} \tag{11}$$

- *F1* is a combined measure that results in high value if, and only if, both precision and recall result in high values:

$$F1 = \frac{2*P*V}{P+V} \tag{12}$$

The symbols stand for: C, the number of correctly detected cuts ("correct hits"), M, the number of not detected cuts ("missed hits") and F, the number of falsely detected cuts ("false hits"). Five-fold cross-validation was performed using different partitions of cuts into training (80 %) and testing (20 %) data, and the validation results averaged over the rounds are shown in the upper part of Table 3. Cut detection results for each movie are given in the lower part of Table 3.

Table 3. Overall cut detection results averaged over the runs of 5-fold cross-validation (upper table) and results for each movie using $SAD|HD$ combination (lower table).

Score	C	M	F	V	P	$F1$
HD	2747	157	103	0.946	0.964	0.955
SAD	2788	107	42	0.963	**0.985**	0.974
$SAD\&HD$	2765	148	**41**	0.949	**0.985**	0.967
$SAD|HD$	**2825**	**85**	47	**0.971**	0.984	**0.977**

	V	P	$F1$		V	P	$F1$
P01	0.997	0.994	0.996	P06	0.988	0.996	0.992
P02	0.975	0.982	0.979	P07	0.985	0.992	0.989
P03	0.941	0.976	0.958	P08	0.961	0.989	0.975
P04	0.969	0.983	0.976	P09	0.978	1.000	0.989
P05	0.987	0.942	0.964	P10	0.969	0.991	0.980

As expected, different results were achieved for individual scores. Better results were obtained for the SAD score in all aspects—only a minimum number of false hits, and a relatively small number of missed hits (approx. 4 % of all hits) were achieved with this score. As for the combinations of both scores, slightly better results were obtained for the combination $SAD|HD$ which means that a cut is detected if it is detected at least by one of the scores.

4. OPTIMISATION OF SUBTITLES TIMING

As mentioned in Section 1, to keep synthetic audio track synchronised with video, speech synthesised from a subtitle text has to fit into the corresponding subtitle slot. This requirement causes synthetic speech to be speeded up, possibly with high speeding-up factors. After the speedup, otherwise clear and perfectly intelligible synthetic speech can become less intelligible which is in conflict with our objective to produce a less-dynamic and more intelligible audio track.

The idea behind the optimisation of subtitles timing is that a reasonable change in a position of a subtitle (i.e., in its start and/or end times) can reduce the speeding-up factor. To keep the synthetic audio track synchronised with video, the following requirements has to be respected when changing the timing of a subtitle:

- a subtitle must not overlap with the neighbouring subtitles;

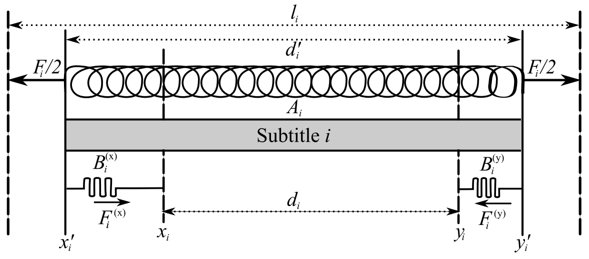

Fig. 1. An illustration of subtitle timing optimisation using a spring-based model.

- a subtitle must not exceed a cut;
- the changed subtitle timing should be as close to the original timing as possible;
- the speeding-up factor needed to fit synthetic speech into the (changed) subtitle slot should be kept at minimum.

Such a formulated problem can be modelled using a *physical simulation of springs* (see Fig. 1). For each subtitle i (where i is its order in a TV programme) with length d_i (how long it is shown on screen), the corresponding TTS-generated speech can be viewed as a spring A_i with a certain *rate/constant* (i.e. stiffness or stretchability) k_A and with length equal to the length l_i of the synthesised (not speeded up) speech. To fit the spring A_i into the given subtitle slot (i.e. to speed up the speech), the spring has to be compressed to the length d_i. After the compression is released, the spring starts, following Hooke's law (with spring displacement expressed in terms of speeding-up factors $f_i = \frac{l_i}{d_i}$ and 1), to stretch with a restoring force

$$F_i = -k_A \cdot (f_i' - 1) \tag{13}$$

with $f_i' = \frac{l_i}{d_i'}$ and d_i' being the current stretched length of the spring/subtitle, causing subtitles start x_i' and end y_i' positions recede from their original positions x_i and y_i. At the same time, speeding-up factor f_i' starts to reduce. To stay synchronised, i.e. to have x_i' and y_i' as close to the original positions x_i and y_i as possible, other springs, $B_i^{(x)}$ and $B_i^{(y)}$, are used at the beginning and end of the spring A_i, compressing back the spring A_i to its original positions with forces

$$F_i^{(x)} = -\, k_B \cdot (x_i' - x_i), \tag{14}$$
$$F_i^{(y)} = -\, k_B \cdot (y_i' - y_i), \tag{15}$$

where k_B is rate of $B_i^{(x)}$ and $B_i^{(y)}$. The ratio $\frac{k_A}{k_B}$ allows to control between preferences either towards synthetic speech with small speeding-up factors but with start and end positions somewhat receded from the original positions ($k_A < k_B$) or towards synthetic speech at original positions, but more speeded up ($k_A > k_B$). The ratio could be set up, for instance, according to the genre of a TV programme. The optimal timing of the subtitle, i.e. x_i^* and y_i^*, and the optimal speeding-up factor f_i^* are achieved when equilibrium of forces F_i, $F_i^{(x)}$ and $F_i^{(y)}$ is reached.

The iterative algorithm can be summarised in the following steps (with Δ being a simulation step; iterations are illustrated in Fig. 2):

1. Set up k_A, k_B arbitrarily, set $x_i' = x_i$, $y_i' = y_i$, and l_i set to length of not speeded up synthesised speech.

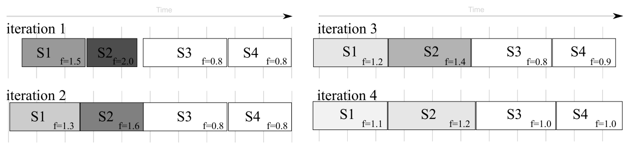

Fig. 2. An illustration of the iterative algorithm. The darker the colour, the higher speeding-up factor has to be used to fit synthetic speech into the corresponding subtitle slot ("S" stands for subtitle, "f" for speeding-up factor).

2. Compute $d'_i = y'_i - x'_i$, F_i (Eq. 13), $F_i^{(x)}$ (Eq. 14), $F_i^{(y)}$ (Eq. 15) and total forces at subtitle beginning $F_i^{(B)} = \frac{F_i}{2} + F_i^{(x)}$ and end $F_i^{(E)} = -\frac{F_i}{2} + F_i^{(y)}$.
3. Update $x'_i = x'_i + \Delta \cdot F_i^{(B)}$, $y'_i = y'_i + \Delta \cdot F_i^{(E)}$. If x'_i (or y'_i) exceeds a cut, x_i (or y'_i, respectively) is set to the position of the cut. Also ensure that x'_i and y'_i do not overlap with neighbouring subtitles.
4. If $F_i^{(B)} = F_i^{(E)} = 0$ (equilibrium reached), then $x^*_i = x'_i$ and $y^*_i = y'_i$ are the optimal start and end times of subtitle i (the optimal length of the corresponding synthesised speech is $d^*_i = d'_i$; thus the optimal speeding-up factor is $f^*_i = \frac{l_i}{d^*_i}$). Otherwise continue with step 2.

In reality, the situation is more complex because a sequence of springs with the number of springs corresponding to the number of subtitles in a TV programme has to be taken into account. The optimisation process starts after all forces F_i begin to compress the springs A_i en bloc. The optimal timing of all springs/subtitles in a TV programme is achieved when an equilibrium of all forces is reached.

To avoid gross synchronisation errors, video cuts detected as described in Section 3 are used to prevent from stretching subtitles (or the corresponding synthetic speech, respectively) across different scenes during the update of x'_i and y'_i which would be otherwise perceived as very disturbing and unnatural.

5. EVALUATION AND RESULTS

Well-tuned Czech unit-selection TTS system ARTIC [2] was used to synthesise the audio track, both from original (the baseline system) and optimised (the optimised system) subtitles. Since the TTS system performs very well (and is continuously upgraded) when synthesising with original (not speeded up) speech rate, high quality synthetic audio track can be guaranteed only when speeding-up factors will be kept minimal. Therefore, the baseline and optimised systems were evaluated with respect to speeding-up factors used during synthesis. As can be seen in Fig. 3, the optimised system employs smaller speeding-up factors; thus, it should yield a less-dynamic, more intelligible audio track. To speed up synthetic speech, WSOLA algorithm with non-linear time-scale distribution scheme [7] was applied.

The utilisation of smaller speeding-up factors was at the expense of shifting start/end subtitle times from their original positions. The average shift for our experimental data was 70 ms (with standard deviation 47 ms and maximum shift 700 ms). Such values mean that no serious audio/video synchronisation issues should occur.

The proposed method could also be used to fix subtitles positions. The typical mistakes made by human transcribers when creating subtitles are those in which subtitles slightly overlap video cuts. Applying the subtitle timing optimisation algorithm on our experimental data, 836 subtitles (more than 8 % of all subtitles) were fixed.

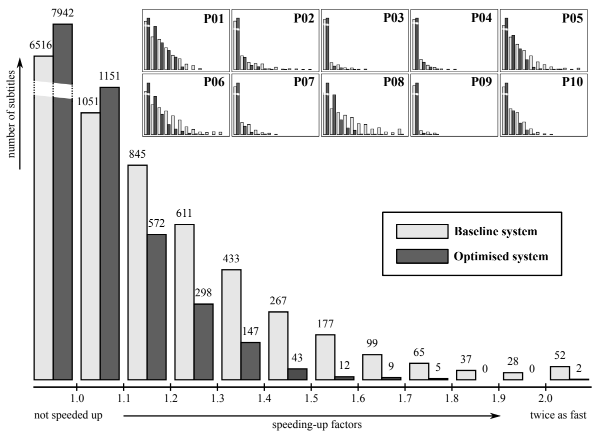

Fig. 3. Results of subtitle timing optimisation in terms of speeding-up factors: total results (left), results per movie (right).

6. CONCLUSION

Improvements to the automatic dubbing system were presented in the paper. It was shown that using only one score (SAD) or a combination of two scores (SAD or HD) yields very good video cut detection results. The detected cuts were then used to prevent stretching subtitles across the cuts during spring-based subtitle timing optimisation process. Thanks to the optimisation, smaller speeding-up factors were applied when fitting synthesised speech into the corresponding subtitle slot. At the same time, optimised subtitle start and end times were kept close to original positions.

In our future work, genre-dependent experiments will be carried out to find out whether genre-dependent settings both in cut detection and spring-based subtitle timing optimisation could further improve system performance. After testing broadcasting is launched by Czech Television, automatic dubbing will be evaluated by real users.

7. REFERENCES

[1] Zdeněk Hanzlíček, Jindřich Matoušek, and Daniel Tihelka, "Towards automatic audio track generation for Czech TV broadcasting: Initial experiments with subtitles-to-speech synthesis," in *Proc. ICSP*, Beijing, China, 2008, vol. 3, pp. 2721–2724.

[2] Daniel Tihelka, Jiří Kala, and Jindřich Matoušek, "Enhancements of Viterbi search for fast unit selection synthesis," in *Proc. INTERSPEECH*, Makuhari, Japan, 2010, pp. 174–177.

[3] Werner Verhelst, "Overlap-add methods for time-scaling of speech," *Speech Commun.*, vol. 30, pp. 207–221, 2000.

[4] Sandra Derbring, Peter Ljunglof, and Maria Olsson, "SubTTS: Light-weight automatic reading of subtitles," in *Proc. NODALIDA*, 2009, pp. 272–274.

[5] William K. Pratt, *Digital Image Processing: PIKS Scientific Inside*, John Wiley & Sons, 3rd edition, 2001.

[6] John S. Boreczky and Lawrence A. Rowe, "Comparison of video shot boundary detection techniques," *J. Electron. Imaging*, vol. 5, no. 2, pp. 122–128, April 1996.

[7] Daniel Tihelka and Martin Méner, "Generalized non-uniform time scaling distribution method for natural-sounding speech rate change," in *Text, Speech and Dialogue*, vol. 6836 of *Lecture Notes in Computer Science*, pp. 147–154. Springer, Berlin, Heidelberg, 2011.