SPEECH AND SLIDING TEXT AIDED SIGN RETRIEVAL FROM HEARING IMPAIRED SIGN NEWS VIDEOS

Oya Aran [1], *Ismail Ari* [1], *Pavel Campr* [2], *Erinç Dikici* [3], *Marek Hruz* [2], *Deniz Kahramaner* [4], *Siddika Parlak* [3], *Lale Akarun* [1], *Murat Saraçlar* [3]

[1] PILAB, Boğaziçi University, İstanbul, Turkey
[2] University of West Bohemia, Pilsen, Czech Republic
[3] BUSIM, Boğaziçi University, İstanbul, Turkey
[4] Robert College, İstanbul, Turkey

ABSTRACT

The objective of this study is to automatically extract annotated sign data from the broadcast news recordings for the hearing impaired. These recordings present an excellent source for automatically generating annotated data: In news for the hearing impaired, the speaker also signs with the hands as she talks. On top of this, there is also corresponding sliding text superimposed on the video. The video of the signer can be segmented via the help of either the speech or both the speech and the text, generating segmented, and annotated sign videos. We aim to use this application as a sign dictionary where the users enter a word as text and retrieve sign videos of the related sign with several examples. This application can also be used to automatically create annotated sign databases that can be used for training recognizers.

KEYWORDS

Speech recognition – Sliding text recognition – Sign language analysis – Sequence clustering – Hand tracking

1. INTRODUCTION

This project aims to exploit TRT news for the hearing impaired programs in order to generate usable data for sign language education. The news video consists of three major information sources: sliding text, speech and signs. Fig. 1 shows an example frame from the recordings.

Figure 1: *An example frame from the news recordings. The three information sources are the speech, sliding text, signs.*

The three sources in the video convey the same information via different modalities. The news presenter signs the words as she talks. It is important to note that sign languages have their own grammars and word orderings. Thus, it is not necessary to have the same word ordering in a Turkish spoken sentence and in a Turkish sign sentence [1]. Thus, the signing in these news videos is not Turkish sign language (Turk Isaret Dili, TID) but Signed Turkish: the sign of each word is from TID but their ordering would have been different in a proper TID sentence. In addition to the speech and sign information, a corresponding sliding text is superimposed on the video. Our methodology is to process the video to extract the information content in the sliding text and speech components and to use either the speech alone or both the speech and the text to generate segmented and annotated sign videos. The main goal is to use this annotation to form a sign dictionary. Once the annotation is completed, unsupervised techniques are employed to check consistency among the retrieved signs, using a clustering of the signs.

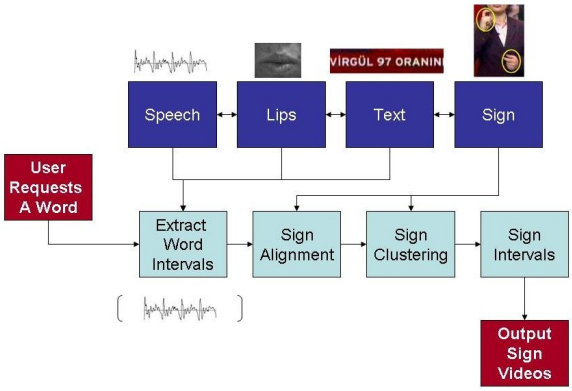


Figure 2: *Modalities and the system flow.*

The system flow is illustrated in Fig. 2. The application receives the text input of the user and attempts to find the word in the news videos by using the speech. At this step the application returns several intervals from different videos that contain the entered word. Then, sliding text information may optionally be used to control and correct the result of the retrieval. This is done by searching for the word in the sliding text modality during each retrieved interval. If the word can also be retrieved by the sliding text modality, the interval is assumed to be correct. The sign intervals will then be extracted by analyzing the correlation of the signs with the speech. Sign clustering is necessary for two reasons. First, there can be false alarms of the retrieval corresponding to some unrelated signs and second, there are homophonic words that have the same phonology but different meanings thus possibly different signs.

2. SPOKEN TERM DETECTION

Spoken utterance retrieval part of this project will be used as a tool to automatically segment the signs in a broadcast news video for the disabled and to display the desired sign to the user after the alignment. The general system diagram is given in Fig. 3 where the input to the search part is the query and the output is the occurrence time, duration and the program name in which the query appears in.

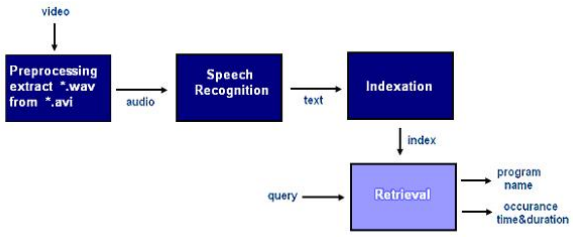


Figure 3: *Block diagram of the spoken term detection system.*

Three main modules of the system are speech recognition, indexing and retrieval which will be explained in detail. Currently we have 127 broadcast news videos, each with a duration of 10 minutes and sampled at 32 kHz. As a preprocessing operation, the audio information is extracted and the sampling rate is reduced to 16 kHz. The speech recognizer converts this audio data into textual information (in terms of weighted finite state automata). Indexation of the text is done via weighted finite state transducers [2]. The index is built in such a way that when it is composed with the query, the output is the search hits with program name, occurrence time and duration as well as the expected count. This type of indexation introduces several benefits as will be explained later.

2.1. Speech recognition

The speech recognizer takes a collection of audio files and converts them into text. Before recognition, audio data is segmented based on the energy constraint. Since the background does not include music or noise it was adequate to identify the speech and non-speech portions only. The method explained in [3] is applied and some further arrangements are made on the output. These post modifications were mainly about setting a minimum duration limit on segments and merging them if they are smaller.

An HMM-based large vocabulary continuous speech recognition (LVCSR) system is used for recognition. The feature vector consists of 12 MFCC components, the energy component as well as delta and delta-delta components. The acoustic models consist of decision tree clustered triphones and the output distributions are GMMs. The acoustic model used in this project was previously trained on approximately 100 hours of broadcast news data. The language models are pruned back off trigram models which are based on words. The recognition networks are represented as weighted finite state machines (FSMs). The output of the ASR system is also represented as an FSM and may be in the form of a best hypothesis string or a lattice of alternate hypotheses. To illustrate this, the lattice output of a recognized utterance is shown in Fig. 4. [4].

The labels on the arcs are the word hypotheses (or subwords such as morphemes) and the values next to the labels are the probabilities of each arc. It is obvious that there is more than one path from the initial state to the final. However, the most probable path is "iyi gUnler" ("good afternoon"), which is the correct transcription.

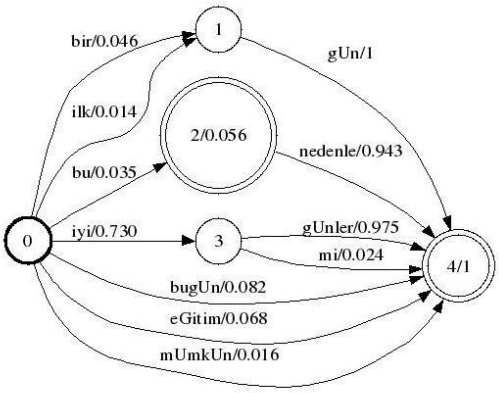


Figure 4: *An example lattice output.It belongs to the utterance "iyi gUnler".*

For the lattice output, more than one hypothesis is returned with corresponding probabilities. Indexation estimates the expected count using path probabilities. By setting a threshold on the expected count, different precision-recall points can be obtained which results in a curve. On the other hand, the one-best hypothesis is represented with only one point. Having a curve allows choosing an operating point by setting the threshold. Use of a higher threshold improves the precision but recall falls. Conversely, a lower threshold value causes small expected counts to be retrieved. This increases recall but decreases precision.

The opportunity of choosing the operating point is crucial. Depending on the application, it may be desirable to retrieve all the documents or only the most probable ones. For our case, it is more convenient to operate at the point where precision is high.

Although this introduces some redundancy, using lattice output of the ASR, improves system performance as shown in the experiments. This improvement is expected to be much higher for noisy data.

The HTK tool[5] is used to produce feature vectors and AT&T's FSM and DCD tools[6] are used for recognition. Currently, the word error rate of the speech recognizer is approximately 20%. Using a morpheme based-model instead of the word-based one reduces the WER and will be left as a future work.

2.2. Indexation

The output of the speech recognizer is a weighted automaton where each speech utterance of the dataset is represented as an FSM and they are concatenated. Since input to indexation is a finite state machine, it is advantageous to represent the indexer as a transducer FSM. Another advantage of this approach is that it simplifies dealing with arc weights or path probabilities in the lattice. Since the input to indexation is uncertain, it is important to keep log-likelihoods. This information should be passed to the search module by the indexer for ranking purposes. Since FSMs are compact representations of alternative hypotheses with varying probabilities, designing the indexer as an FSM is favorable.

The problem is to build an index which accepts any substring of the utterance transcriptions. Thus the index can be represented by a weighted finite-state transducer mapping each factor of each utterance to the indices of the automata. This in-

formation comes from the ASR output and is defined as:

$$T(x,i) = -\log(E_{P_{|i|}}[C_{[x]}]) \quad (1)$$

where x is the substring, i is the automaton index, P is the probability coming from ASR and C_x denotes the number of occurrences of substring x in the utterance.

The factor selection is done by creating a new initial and final state to the automata such that each intermediate state is connected to both the initial and the final state. Thus it is possible to go from the initial state to any of the terms (phones or words) and to reach the final state from there; this is accepted by the transducer. After the new states and connections are added, the automata are optimized with epsilon removal, determination and minimization. The details of this process and the further theory of weighted finite state automata are explained in [2].

A very powerful feature of this indexing method is its speed. The resulting index also does not bother with the expansion of the database. The search time for a query is linear in the size of the query and the number of places it appears in the database. Search duration is currently 1-1.8 seconds depending on the frequency of the query and expanding the database from 20 videos to 120 videos does not make any difference.

2.3. Retrieval

The top three blocks in 3, namely, preprocessing, recognition and indexation are performed only once. When the query is entered, only the retrieval module in the diagram is activated.

First the query is converted into an FSM and then composed with the index FSM. After the optimization, the list of all indices where the query appears and the corresponding log-likelihoods are acquired. Since the probabilities are known, it is possible to select the most probable outputs and rank the documents [2].

Now we have the utterance indices; however this includes not only the query but other words. To clarify each word's starting time and duration we apply forced alignment. The final output is the program name, starting time and duration of the query in seconds.

For the client-server interaction, a server application using sockets is written in PERL. Client side will be explained later. When the server gets the query from the client, the search operation is initiated and the standard output is sent back to the client. If the user asks for a word which is not in the vocabulary, an error message is displayed and no search is performed.

2.4. Experiments and results

2.4.1. Evaluation Metrics

Speech recognition performance is evaluated by the WER (word error rate) metric which was measured to be around 20% in our experiments. For evaluation of the retrieval system precision-recall values and F-measures are used.

Let the reference transcriptions include R(q) occurrences of the search term q. A(q) is the number of retrieved documents and C(q) is the number of documents which are related to the query from the retrieved ones.

Then

$$Precision(q) = \frac{C(q)}{A(q)} \quad (2)$$

$$Recall(q) = \frac{C(q)}{R(q)} \quad (3)$$

and the F-measure is:

$$F(q) = \frac{2 * Precision(q) * Recall(q)}{Precision(q) + Recall(q)} \quad (4)$$

Precision and recall values for each word in the query set will be determined and averaged. Out-of-vocabulary words will be discarded in the precision averaging. However their recall is assumed to be zero for recall averaging. This is also the case for words which exist in the reference but could not be retrieved. The reason behind this assumption is that retrieving a wrong document and retrieving nothing cannot be judged as if they are the same.

2.4.2. Corpora

Although the whole collection consists of 127 videos, we used only 15 for the evaluation.Since we have the reference manual transcripts only for these files. The evaluation task begins with the forced alignment of the manual transcriptions which will be used as the reference. Start times and durations of each word - and the silence - are identified and kept in the format (*.ctm), shown in Fig 5, where the columns represents program name, channel (studio, telephone etc.), start time(in sec.), duration(in sec.) and spoken word (or silence) respectively.

```
2007-06-29-14-40 1 10.06 0.08 Z1
2007-06-29-14-40 1 10.14 0.23 iyi
2007-06-29-14-40 1 10.37 0.57 gUnler
2007-06-29-14-40 1 10.94 1.11 Z1
```

Figure 5: **.ctm file format. It is constructed after the forced alignment of manual transcriptions.*

A query set is created from the reference files by taking each word uniquely. Each item in the query set is searched in the database and search results are kept in a file with the format, shown in Fig. 6. In this format, columns represent the program file name in which the query is claimed to occur, start time (in ms.), duration (in ms.) and relevance respectively. After these files are obtained, precision and recall values are calculated and averaged.

```
2007-06-29-14-40.avi, 255477, 480, 1
2007-06-29-14-40.avi, 271494, 460, 1
2007-06-29-17-40.avi, 530545, 500, 0.0118179
```

Figure 6: *Search result file format. For each query, this file is created and compared with the reference ctm file.*

The retrieval result is checked with the reference ctm file with a margin time. If the beginning and end times are found to agree with margin seconds or less from the correct, search is assumed to be successful.

2.4.3. Results

The index is created for the lattice and one-best output of the recognizer. Precision-recall (no limit is set on the number of retrieved documents) and precision-recall at 10 (maximum number of retrieved documents is limited to 10) graphs are depicted as in Fig. 7 and Fig. 8.

It is obvious from the plots that, use of lattices performs better than one-best. This is also the case for precision and recall at 10. The arrows point at the position where the maximum F-measure is achieved for lattice. Comparison of F-measures of lattice and one-best output of ASR are given in 1. Use of lattices introduces 1-1.5 % of improvement. Since the broadcast news corpora are fairly noiseless, the achievement may seem minor. However for noisy data the difference is much higher [4].

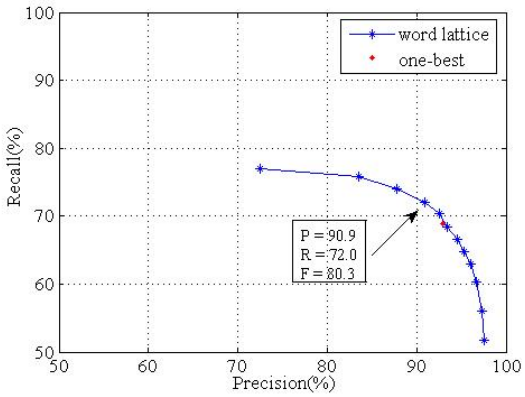


Figure 7: *Precision-Recall for word-lattice and one-best hypotheses when no limit is set on maximum number of retrieved documents.*

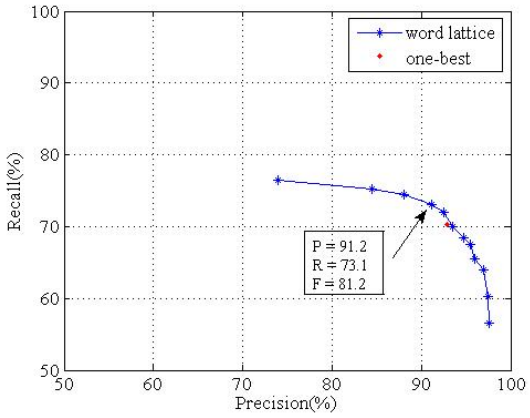


Figure 8: *Precision-Recall for word-lattice and one-best hypotheses when maximum number of retrieved documents is set to 10 (precision at 10).*

3. SLIDING TEXT RECOGNITION

3.1. Sliding Text Properties

The news videos are accompanied by a sliding text band, which includes simultaneous transcription of what is being said. It is placed at the bottom of the screen and contains characters with a specific font, shown with white pixels over a solid background. Speed of the sliding text is approximately constant throughout the whole video sequence (usually 4 pixels/frame), which allows each character to appear on the screen for at least 2.5 seconds. An example of a frame with sliding text is shown in Fig. 9.

3.2. Baseline Method

The method we propose to obtain sliding text information consists of three parts: Extraction of the text line, character recognition and temporal alignment.

3.2.1. Text Line Extraction

Size and position of the sliding text band does not change throughout the video. Therefore, it is found at the first frame and used in the rest of the operations. To find the position of the text, first, we convert the RGB image into a binary image, using grayscale

Table 1: *Comparison of lattice and one-best ASR outputs on maximum F-measure performance.*

	Max-F(%)	Max-F@10(%)
Lattice	80.32	81.16
One-best	79.05	80.06

Figure 9: *Frame snapshot of the broadcast news video.*

quantization and thresholding with the Otsu method [7]. Then we calculate horizontal projection histogram of the binary image, i.e., the number of white pixels for each row. The text band appears as a peak on this representation, separated from the rest of the image. We apply a similar technique over the cropped text band, this time on the vertical projection direction, to eliminate the program logo. The sliding text is bounded by the remaining box, whose coordinates are defined as the text line position. Fig. 10 shows an example of binary image with its horizontal and vertical projection histograms.

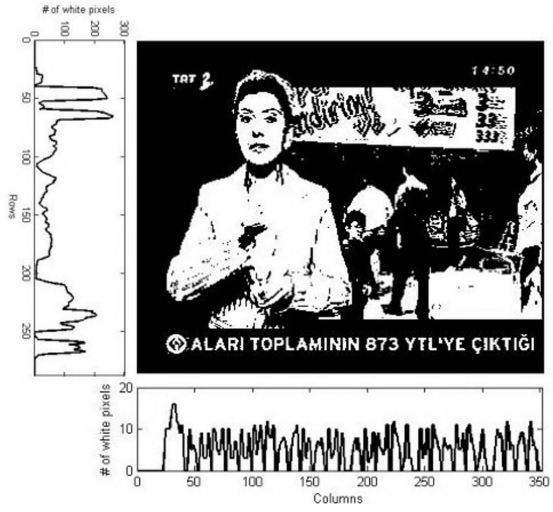


Figure 10: *Binary image with horizontal and vertical projection histograms.*

Since there is redundant information in successive frames, we do not extract text information from every frame. Experiments have shown that updating text transcription once in every

10 frames is optimal for achieving sufficient recognition accuracy. We call these the "sample frames". The other frames in between are used for noise removal and smoothing.

Noise in binary images stems mostly from quantization operations in color scale conversion. Considering the low resolution of our images, it may cause two characters, or two distinct parts of a single character to be combined, which complicates the segmentation of text into characters. We apply morphological opening with a 2x2 structuring element to remove such effects of noise. To further smooth the appearance of characters, we horizontally align binary text images of the frames between two samples, and for each pixel position, decide on a 0 or 1, by voting.

Once the text image is obtained, vertical projection histogram is calculated again to find the start and end positions of every character. Our algorithm assumes that two consecutive characters are perfectly separated by at least one black pixel column. The threshold of an inter-word space is determined adaptively, by searching for outliers in character spacing. For proper alignment, only complete words are taken for transcription.

Each character is individually cropped from the text figure and saved, along with its start and end horizontal pixel positions.

3.2.2. *Character Recognition*

For character recognition, we implement the template matching method. Each binary character image is compared to each template, pixel by pixel. The total number of matching pixels are divided by the size of the character image and used as a similarity score. Matched characters are stored as a string. Fig. 11 depicts a sample text band image, its transcribed text and corresponding pixel positions, respectively.

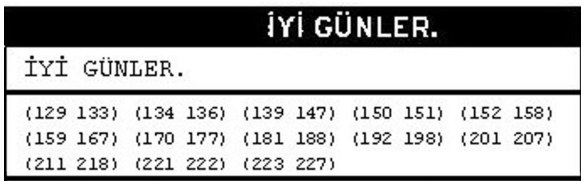


Figure 11: *Sample image and transcription.*

3.2.3. *Temporal Alignment*

The sliding text is interpreted as a continuous band throughout the video. Since we process only selected sample frames, the calculated positions of each character should be aligned in space (and therefore in time) with their positions from the previous sample frame. This is done using frame shift and pixel shift values. For instance, a character which appears in positions 180-189 in the first sample and the one in 140-149 of the second refer to the same character, since we investigate each 10^{th} frame with 4 pixels of shift per frame. Small changes in these values (mainly due to noise) are compensated using shift-alignment comparison checks. Therefore, we obtain a unique pixel position (start-end) pair for each character seen on the text band.

We determine the final transcript as follows: For each frame, we look for the occurrence of a specific start-end position pair, and note the corresponding character to a character candidate list (for different frames, these characters may not be the same, due to recognition errors). The final decision is made by majority voting; the character that is seen the most in the list is assigned to that position pair.

3.3. Baseline Performance

We compare the transcribed text with the ground truth data and use character recognition and word recognition rates as performance criteria. Even if only one character of a word is misrecognized, we label this word as erroneous.

Applying the baseline method on a news video dataset of 40 minutes (around 3500 words), we achieved 94% character recognition accuracy, and 70% word recognition accuracy.

3.4. Discussions

One of the most important challenges of character recognition was the combined effect of low resolution and noise. We work with frames of 352x288 resolution, therefore, each character covers barely an area of 10-14 pixels in height and 2-10 pixels in width. In such a small area, any noise pixel distorts the image considerably, thus making it harder to achieve a reasonable score by template comparison.

Noise cancellation techniques created another disadvantage since they removed distinctive parts of some Turkish characters, such as erasing dots (İ, Ö, Ü), or pruning hooks (Ç, Ş). Fig. 12 shows examples of such characters, which, after noise cancellation operations, look very much the same.

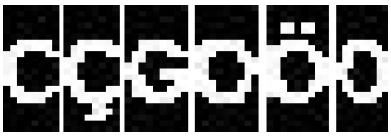


Figure 12: *Highly confused characters.*

3.5. Improvements over the Baseline

We made two major improvements over the baseline system, to improve recognition accuracy. The first one is to use Jaccard's distance for template match score. Jaccard uses pixel comparison with a slightly different formulation: Let n_{ij} be the total number of pixel positions, where binary template pixel has value i and character pixel has value j. Then, the comparison score is formulated as [8]:

$$d_j = \frac{n_{11}}{n_{11} + n_{10} + n_{01}} \tag{5}$$

Using Jaccard's distance for template match scoring resulted in 96.7% accuracy for character recognition, and 80.0% for word recognition accuracy.

The highest rate in the confusion matrix belongs to discrimination of the letter "O" and the number "0". To distinguish these, the knowledge that a zero should be preceded or succeeded by another number, is used as a postprocessing operation. Adding this correction resulted in 98.5% character accuracy and 90% word recognition accuracy, respectively. The confusion rates are listed in Table 2.

4. SIGN ANALYSIS

4.1. Skin color detection

4.1.1. *Skin color detection using a probabilistic model*

Skin color is widely used to aid segmentation in finding parts of the human body [9]. We learn skin colors from a training set and then create a model of them using a Gaussian Mixture Model (GMM). This is not a universal skin color model; but rather, a model of skin colors contained in our training set.

Table 2: *Character confusion rates.*

Character (Original)	Character (Recognized)	Confusion Rate (%)
Ç	C	8.33
Ğ	G	1.49
Ğ	Ö	2.99
H	M	2.94
I	1	0.85
N	M	0.34
Ö	O	9.68
Ş	S	2.73
Ü	U	2.47
0	O	36.36
2	Z	7.14

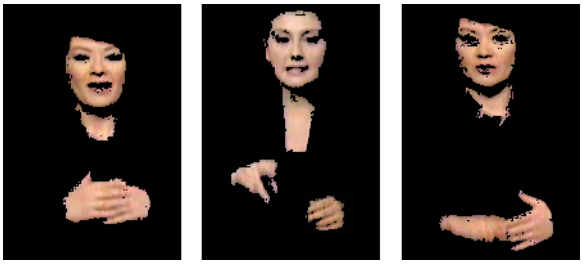


Figure 13: *Examples from the training data*

Training of GMM. We prepared a set of training data by extracting images from our input video sequences and manually selecting the skin colored pixels. We used images of different speakers under different lighting conditions. In total, we processed six video segments. Some example training images are shown in Fig. 13.

For color representation we use the RGB color space. The main reason is that this color space is native for computer vision and therefore does not need any conversion to other color space. The collected data are processed by the Expectation Maximization (EM) algorithm to train the GMM. After inspecting the spatial parameters of the data we decided to use a five Gaussian mixtures model. The resulting model can be observed in Fig. 14.

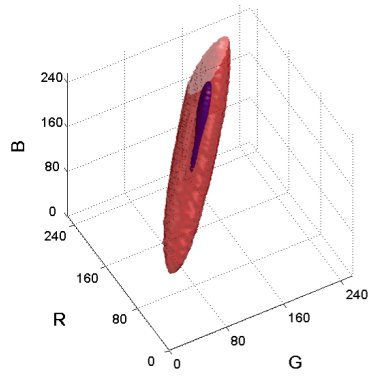


Figure 14: *General look up table. Levels of likelihood can be seen in different colors. Lower level likelihood of 128, higher level likelihood of 220.*

Using GMM for image segmentation. The straight forward way of using the GMM for segmentation is to compute the probability of belonging to a skin segment for every pixel in the image. One can then use a threshold to decide whether the pixel color is skin or not. But this computation would take a long time, provided the information we have is the mean, variance and gain of each Gaussian in the mixture. We have precomputed the likelihoods and used table look-up.

4.1.2. Skin color detection using look up table

Look up table model description. We decided to create a look-up table to store the likelihood that a color corresponds to a skin. In this case the values of the look-up table are computed from the probability density function given by GMM. The range of likelihoods is from 0 to 255. A likelihood of 128 and more means that the particular color belongs to a skin segment. With this procedure we obtain a 256x256x256 look-up table containing the likelihood of all the colors from RGB color space.

The segmentation is straightforward. We compare the color of each pixel with the value in the look-up table. According to the likelihood of the color, we decide whether it belongs to skin segment or not. For better performance, we blur the resulting likelihood image. That way, the segments with low probability disappear and the low probability regions near the high probability regions are strengthened. For each frame, we create a mask of skin color segments by thresholding the likelihood image (See Fig. 15).

Figure 15: *The process of image segmentation. From left to right: original image, probability of skin color blurred for better performance, binary image as a result of thresholding, original image with applied mask.)*

The look up table is created using color information from more than one speaker. Generally we want to segment a video sequence with just one speaker. This means that with the look up table we will obtain some undesirable pixels identified as skin color pixels which do not belong to the particular speaker. Thus, we need to adapt the look up table for each video. Since manual data editing is not possible at this stage, a quick automatic method is needed.

Training of the look up table. We refer to the general look up table modeled from several speakers as the background look-up table and describe how we adapt it to the current speaker. First we need to extract the skin color pixels from the current speaker. Using the OpenCV's Haar classifier [10] we detect the face of the speaker and store it in a separate image. This image is then segmented using the background look up table and a new look up table is created. We process all the pixels of the segmented image and store the pixels' color in the proper positions in the look up table. Information from one image is of course not sufficient and there are big gaps in the look up table. To solve this problem, we use a convolution with a Gaussian kernel to smooth the look-up table. To save time, we process color components separately. At the end of this step, we obtain a speaker dependent look up table.

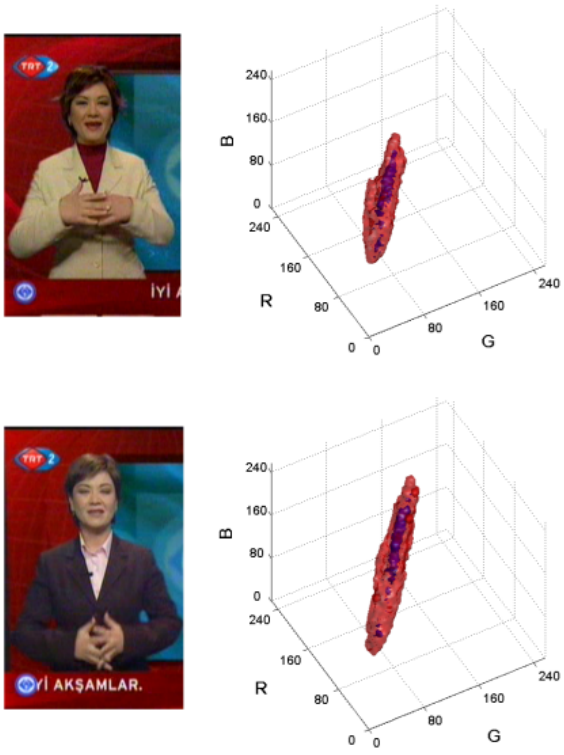


Figure 16: *Examples of different lighting conditions resulting into different skin color of the speaker and the corresponding adapted look up table.)*

4.1.3. Adaptation of general skin color model for a given video sequence

Now we need to adapt our background look up table to the new speaker dependent look up table. We use weighted averaging of the background and the speaker dependent look up tables. The speaker is given more weight in the weighted averaging. This way we eliminate improbable colors from the background look up table thus improving its effectiveness. Some examples of adapted look up tables can be seen in Fig. 16.

4.2. Hand and face tracking

The movement of the face and the hands is an important feature of sign language. The trajectory of the hands gives us a good idea about the performed sign. The position of the head is used for creating a local system of coordinates and normalizing the hand position.

There are many algorithms that are able to detect a face with very good results. We use OpenCV's Haar feature based face detector [10]. From the resulting bounding rectangle we calculate an ellipse around the face. The center of the ellipse is assumed to be the center of mass of the head. We use this point as the head's tracked position.

The input of our tracking algorithm is a segmented image containing only blobs of skin colored objects. As a result of skin color detection, we have an image that contains only skin colored regions. To retrieve the separate blobs from the image we use cvBlobsLib [11]. There can be some blobs which appear in our segmented image but they do not belong to the signer. That means the blob is neither the signer's head nor the signer's hand. We want to eliminate these blobs.

4.2.1. Blob filtering

First we eliminate the blobs with small area. Next we eliminate blobs which belong to the background. The second step is to take into account only the three biggest blobs in the image. They should belong to the head and the hands of the signer as the other false blobs are a result of noise. Sometimes when an occlusion occurs there can be fewer than three blobs belonging to the signer. We need to take this into account. Third, we eliminate all the blobs that are too far away from the previous position of already identified blobs. As a threshold, we use a multiple of the signer's head width. This solves the problem when a false blob appears when a hand and the head are in occlusion.

Generally the biggest challenge in hand tracking is when hands occlude each other, or the head. In this case, two or three objects form a single blob. This situation can be easily detected, but this information alone is not enough to resolve the occlusion. We need to somehow know which objects of interest are in occlusion or whether they are so close to each other that they form a single blob. For this purpose, we try to predict the occlusion. When two objects occlude in the next frame, we assume that they are the predicted ones.

4.2.2. Occlusion prediction

We apply an occlusion prediction algorithm as a first step in occlusion solving. We need to predict whether there will be an occlusion and among which blobs will the occlusion be. For this purpose, we use a simple strategy that predicts the new position, p_{t+1}, of a blob from its velocity and acceleration. The velocity and acceleration is calculated by the first and second derivatives of the position at the previous frames.

$$v_t = p_{t-1} - p_t \quad (6)$$

$$a_t = v_{t-1} - v_t \quad (7)$$

$$p_{t+1} = p_t + v_t + a_t \quad (8)$$

The size of the blob is predicted with the same strategy. With the predicted positions and sizes of the blobs, we check whether these blobs intersect. If there is an intersection, we identify the intersecting blobs and predict that there will be an occlusion between those blobs.

4.2.3. Occlusion solving

We decided to solve the occlusions by trying to find out which part of the blob belongs to one of the objects and divide the blob into two (or three in the case that both hands and head are in occlusion). We divide the blob by drawing a black line or ellipse at the location we think is the best. Thus we always obtain three blobs which can then be tracked as will be shown later. Let us describe the particular cases of occlusion.

Two hands occlude. In this case we separate the blob with a line. We find the bounding ellipse of the blob of occluded hands. The minor axis of the ellipse is computed and a black line is drawn along this axis. For better results we draw the line several pixels wide as can be seen on Fig. 17.

One hand occludes with the head. Usually the blob of the head is much bigger than the blob of the hand. Therefore a division of the blob along the minor axis of the bounding ellipse would have an unwanted effect. We use a template matching method instead [12]. In every frame when the hand is visible we collect its template. The template is a gray scale image defined by the hand's bounding box. When the occlusion is detected a region around the previous position of the hand is defined. We

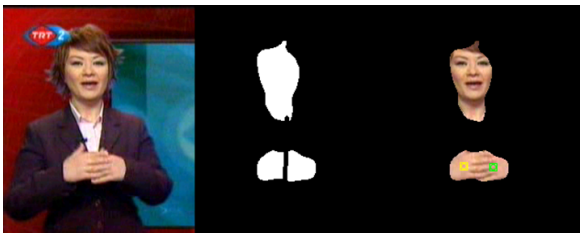


Figure 17: *An example of hand occlusion. From left original image, image with the minor axis drawn, result of tracking.*

calculate the correlation between the hand template and the segmented image. We use a squared difference correlation method to calculate the correlation,

$$R(x,y) = \sum_{x'} \sum_{y'} (T(x',y') - I(x+x', y+y'))^2 \qquad (9)$$

where T is the template, I is the image we search in, x and y are the coordinates in the image, x' and y' are the coordinates in the template. After that we find the minimum in the result of the correlation but we limit the search only to the region around the last position. We draw a black ellipse at the estimated position of the hand as in Fig. 18. The parameters of the ellipse are taken from the bounding ellipse of the hand. As a last step we collect a new template for the hand. Previously the last step was omitted and the last known template of the hand was used. But often the shape of the hand changes in front of the head, so it is necessary to collect the template all the time be able to track the hand properly. The remaining parts of the head in the template image do not have a significant impact on the result of the correlation.

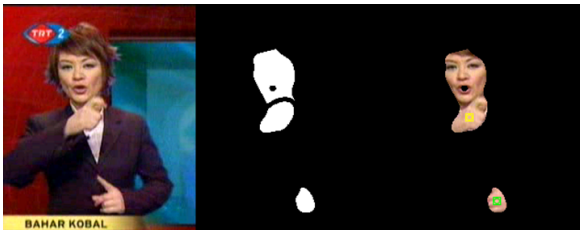


Figure 18: *An example of hand and head occlusion. From left original image, image with the ellipse drawn, result of tracking.*

Both hands occlude with the head. In this case the template matching method proved to be successful. We just need to apply it to both hands.

4.2.4. Tracking of hands

For the tracking of the hands we are able to apply several rules. First, we need to decide which blob belongs to what part of the signer's body. In principle, we assign the blobs in the current frame to the hands and the head by comparing their previous positions and velocities. We assume that the biggest blob closest to the previous head position belongs to the signer's head. The other two blobs belong to the hands.

After examining the signs we found out that the right hand of the signer is most of the time in the left part of the image and the left hand is in the right side of the image. We use this assumption when there is no hand in the previous frame: for the current frame, we set the blob whose center is more left than the other (whose x coordinate is smaller) as the right hand and vice versa.

4.3. Feature extraction

Several features have been extracted from processed video sequences [13] for further use, for example for sign recognition, clustering or alignment. These features can be separated into three groups according to their character.

4.3.1. Tracking features

In the previous section the method for head and hands tracking was introduced. The output of this algorithm is the position and a bounding ellipse of the hands and the head during the whole video sequence. The position of the ellipse in time forms the trajectory and the shape of the ellipse gives us some information about the hand or head orientation. The features extracted from the ellipse are its width, height and angle between ellipse's major axis and x-axis of the image's coordinate system. The tracking algorithm provides five features per object of interest, 15 features in total.

Gaussian smoothing of measured data in time is applied to reduce the noise. The width of the Gaussian kernel is five, i.e. we calculate the actual smoothed value from two previous, actual and two following measured values.

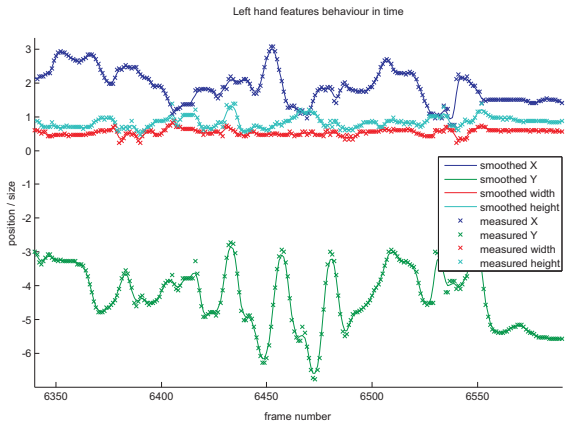


Figure 19: *Smoothing features - example on 10 seconds sequence of four left hand features (x and y position, bounding ellipse width and height).*

We then normalize all the features such that they are speaker independent and invariant to the source resolution. We define a new coordinate system such that the average position of the head center is the new origin and scale with respect to the average width of the head ellipse. Thus the normalization consists of translation to thehead center and scaling.

The coordinate transformations are calculated for all 15 features, which can be used for dynamical analysis of movements. We then calculate differences from future and previous frames and include this in our feature vector:

$$\hat{x}'(t) = \frac{1}{2}(x(t) - x(t-1)) + \frac{1}{2}(x(t+1) - x(t)) \qquad (10)$$

$$= \frac{1}{2}x(t-1) + \frac{1}{2}x(t+1) \qquad (11)$$

In total, 30 features from tracking are provided, 15 smoothed features obtained by the tracking algorithm and their 15 differences.

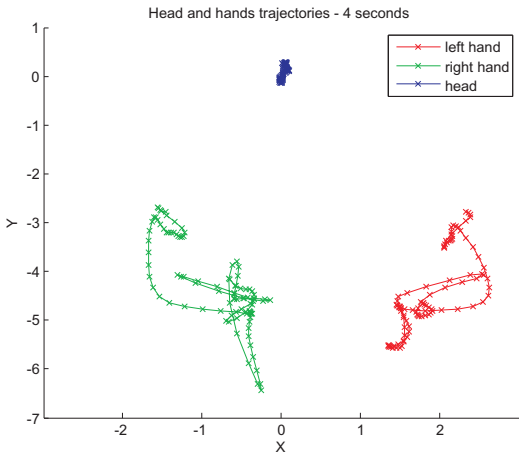


Figure 20: *Trajectories of head and hands in normalized coordinate system.*

4.3.2. DCT features for hands

The hand shape tells us a lot about the performed sign. For some signs it is the only distinguishable feature as there can be very similar signs in trajectory or the signs can be static. In every frame we take a discrete cosine transformation (DCT) of the grayscale template image for each hand. Most of the information in DCT is distributed in the lower frequencies. We extract the upper left triangular part of the DCT matrix which corresponds to those low frequencies. We use 54 features per hand (triangle matrix with width 10). We do not use DC value since this corresponds to the average gray scale value.

Figure 21: *left: reconstructed hand from 54 DCT parameters (by inverse DCT), right: original image.*

The DCT parameters are calculated from a grayscale image where the light areas correspond to the skin colors and the dark areas to the non-skin colors. When a hand is in occlusion with another hand or with the head, then other skin color areas are present in the source image and those areas are included in the DCT parameters. That way a particular hand shape without and with occlusion has different DCT parameters. Only the same hand shape with the same occlusion has the same DCT parameters. This can be either an advantage or a disadvantage depending on the further use.

4.3.3. DCT features for the whole image

At last the DCT parameters of the whole image were calculated. The DCT was calculated from the original image with the skin color mask applied and converted to gray scale. Again the upper left triangular part of the resulting DCT matrix was extracted and the DC parameter was omitted. We have experimented to find that the optimal number of the DCT parameters is 104 (triangular matrix with 14 rows and columns). This number of parameters contains sufficient information about skin color distribution in the image, and does not contain speaker dependent features (such as a specific shape of the head). This information about the specific shapes is stored in higher frequencies, which are cropped from our DCT parameter matrix. We keep only information from the lower frequencies, where the general distribution of skin color in the image is stored.

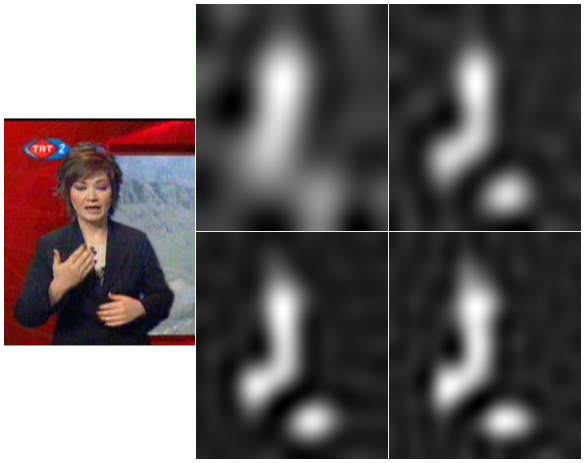


Figure 22: *a) original image, reconstructed images by inverse DCT from b) 35, c) 77, d) 104, e) 135 DCT parameters.*

4.3.4. Summary of extracted features

We have extracted a total of 242 features. Some of them have more information content than others. We extracted all of them in order to have more features for following experiments. The features are summarized in Table 3.

Table 3: *Confusion rates.*

	Number of features	Description
Tracking features	30	head and hands: x, y position width, height, angle of bounding ellipse + derivations
DCT features - hands	2 x 54	DCT parameters for left and right hand shape
DCT features - whole image	104	DCT parameters for skin color distribution in image

4.4. Clustering

Our goal is to cluster two or more isolated signs. If the signs are considered to be the same they should be added to the same cluster. The purpose of clustering is to define a distance (or similarity) measure for our signs. We have to consider that we don't know the exact borders of the sign. The sign news, which contains continuous sign speech, was split into isolated signs by a speech recognizer (see Fig. 23). In the case the pronounced word and the performed sign are shifted, the recognized borders will not fit the sign exactly. We have examined that the spoken word usually precedes the corresponding sign. The starting border of the sign has to be moved backwards in order to compensate the delay between speech and signing. A typical delay of starting instant is about 0.2 seconds backwards. In some cases,

the sign was delayed against the speech, so it's suitable to move the ending instant of the sign forward. To keep our sequence as short as possible, we shifted the ending instant about 0.1 second forward. If we increase the shift of the beginning or the ending border, we increase the probability that the whole sign is present in the selected time interval, at the expense of including parts of previous and next signs into the interval.

Figure 23: *Timeline with marked borders from speech recognizer.*

Now we take two signs from the news whose accompanying speech was recognized as the same word. We want to find out whether those two signs are the same or are different (in case of homonyms). After we extend the borders of those signs (as described above), we suppose that those intervals contain our examined sign and can contain ending part of the previous sign and beginning part of the next sign. The goal is to calculate the similarity of these two signs and to determine if they contain same sign.

Our first experiment was calculating distances between a manually selected short sequence which contains one sign and other same length sequences which were extracted from a longer interval. See Fig. 24, where we calculated distances between a 0.6 second long sequence and the same length sequences extracted from 60 seconds long interval (i.e. we have extracted 0.6 second long sequence starting at frame 1, than from frame 2 and so on). This way we have extracted nearly 1500 sequences and calculated their distances to our selected sequence. This experiment has shown that our distance calculation has high distances for different signs and low distances for similar signs. This was manually evaluated by comparing video sequences containing compared intervals. One may observe that in frame 500, the distance is zero, in fact this is the frame from which we have taken our first, manually selected sequence and then we have compared two same sequences.

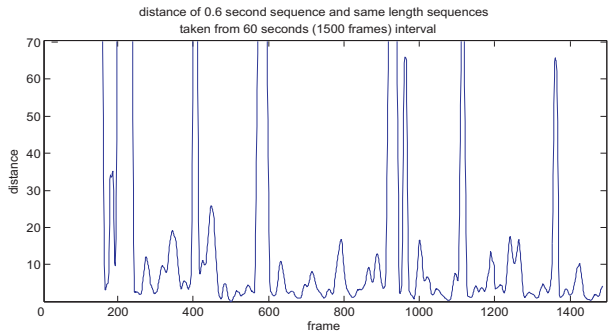


Figure 24: *Experiment: distance calculation between 0.6 second long sequence and same length sequences taken from 60 seconds long interval.*

The distance of the two same length sequences is calculated from tracking features and their derivations. We have not experimented with the hand DCT features, so when the distance is calculated as low, it means that those signs are similar in hand and head position and speed, but can differ in the hand shape.

The distance is calculated in the following way:

1. the difference between corresponding features of two signs is calculated for each frame
2. these differences are squared
3. resulting squared differences are summed over all frames and all those sums for each feature are summed together (we consider same weight for all features)
4. calculated distance is normalized by multiplication with factor 1/*length of sequence*

This distance calculation is very fast, but the problem is when compared sequences contain not only our examined sign, but parts of other signs, then the distance increases. Next problem is time warping. If two same signs are performed with different speed, then the distance increases too.

The solution for both can be usage of a multidimensional DTW (dynamic time warping) algorithm. It's obvious it can handle time warped signals, but we suppose it could solve problem with inclusion of other signs in examined intervals as well: if we calculate the lowest cost path in DTW matrix, than the part where the cost grows slowly correspond to comparing two same signs, otherwise when the cost grows fast two different signs are compared.

Another possible solution is using HMMs (Hidden Markov Models), where one sign is represented by one HMM and each HMM has some additional hidden states at the beginning and at the end, which correspond to the "noise" before and after given sign. Parts of neighboring signs represent this noise. When we cluster more than two signs, we use the following method:

1. Calculate pair wise distances between all compared signs, store those distances in upper triangular matrix
2. Group two most similar signs together and recalculate the distance matrix (it will have one less row and column)
3. Repeat step 2. until all signs are in one group
4. Mark the highest difference between two distances, at which two signs were grouped together in following steps, as distance up to which the signs are in the same cluster (see Fig. 25)

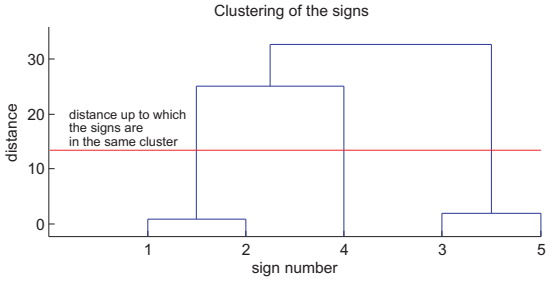


Figure 25: *Dendogram - grouping signs together at different distances.*

Clustering was implemented in Matlab as standalone and server applications. The server application receives a list of signs from a client, calculates their distances, clusters those signs and sends the cluster information for each sign back to the client.

5. APPLICATION AND GUI

5.1. System Design and Application Details

The user interface and the core search engine are separated and the communication between is being done via using TCP/IP

socket connections. This design is also expected to be helpful in the future since a web application is planned to be built using this service. This design can be seen in the Fig. 26.

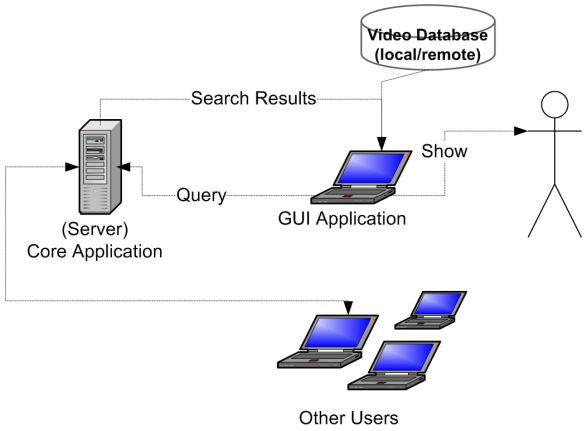


Figure 26: *System Structure.*

In Fig. 27, the screenshot of the user interface is shown. There are mainly five sections in it. The first is the "Search" section where the user inputs a word or some phrases using the letters in Turkish alphabet and sends the query. The application communicates with the engine on the server side using a TCP/IP socket connection, retrieves data and processes it to show the results in the "Search Results" section. The results being done in the same news file is grouped together and shown with date and time information in the tree structure so that the user can also use the application to scan the news archives. A "recent searches" menu is added to the search box aiming to cache the searches and increase the service time. But the user can clear the search history to retrieve the results from the server again for the recent searches. The relevance of the returned results (with respect to the speech) is shown using stars (0 to 10) in the tree structure to inform the user about the reliability of the found results. Moreover, the sign clusters is shown in paranthesis, next to each result.

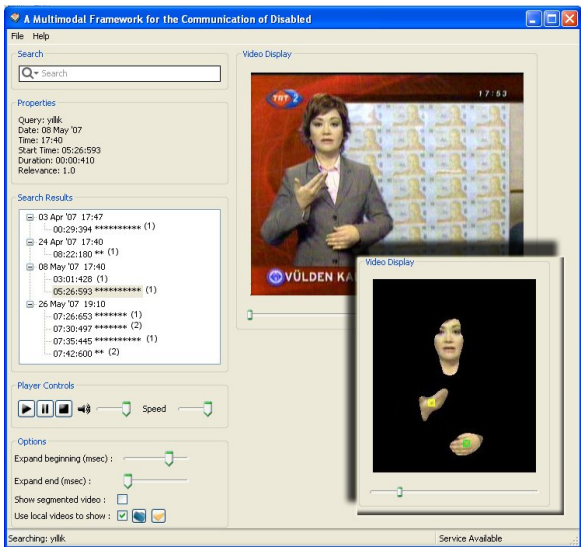


Figure 27: *Screenshot of the User Interface.*

When the user selects a result, it is loaded and played in the "Video Display" section. The original news video or the segmented news video is shown in this section according to the user's "Show segmented video" selection. The segmented video is added separately to figure above to indicate this. In addition to video display, the "Properties" section also informs the user about the date and time of the news, starting time, duration and the relevance of the result. "Player Controls" and "Options" enable the user to expand the duration to left/right or play with the volume/speed of the video to analyze the sign in detail. Apart from using local videos to show in the display, one can uncheck "Use local videos to show" and use the videos on the web. But the speed of loading video files from the web is not satisfactory since the video files are very large.

5.2. Used Tools

The ease and power of `python` language with the `wxPython`'s GUI bindings gave the most help in the user interface creation. `wxFormBuilder` enabled us to design the UI file separately. In addition, `py2exe` is used to create executables from the python code and finally `nsis` is used to create a standalone program setup from separate files and folders.

6. CONCLUSIONS

We have developed a Turkish sign dictionary that can be used as tutoring videos for novice signers. The dictionary is easy to extend by adding more videos and provides a large vocabulary dictionary with the corpus of the broadcast news videos. The application is accessible as a standalone application and will soon be accessible from the Internet.

7. ACKNOWLEDGEMENTS

This work is developed during the eNTERFACE'07 Summer Workshop on Multimodal Interfaces, İstanbul, Turkey and supported by European 6^{th} FP SIMILAR Network of Excellence.

8. REFERENCES

[1] U. Zeshan, "Aspects of Türk Isaret Dili (Turkish Sign Language)", *Sign Language & Linguistics*, vol. 6, no. 1, pp. 43–75, 2003. 37

[2] C. Allauzen, M. Mohri, and M. Saraçlar, "General Indexation of Weighted Automata- Application to Spoken Utterance Retrieval", in *HLTNAACL*, 2004. 38, 39

[3] L. R. Rabiner and M. Sambur, "An Algorithm for Determining the Endpoints of Isolated Utterances", *Bell System Technical Journal*, vol. 54, no. 2, pp. 297–315, 1975. 38

[4] M. Saraçlar and R. Sproat, "Lattice-based Search for Spoken Utterance Retrieval", in *HLTNAACL*, 2004. 38, 39

[5] "HTK Speech Recognition Toolkit". `http://htk.eng.cam.ac.uk`. 38

[6] "AT&T FSM & DCD tools". `http://www.research.att.com`. 38

[7] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE Trans. Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. 40

[8] J. D. Tubbs, "A note on binary template matching", *Pattern Recognition*, vol. 22, no. 4, pp. 359–365, 1989. 41

[9] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A Survey on Pixel-based Skin Color Detection Techniques", in *Graphicon*, pp. 85–92, 2003. 41

[10] "Intel Open Source Computer Vision Library". http://opencvlibrary.sourceforge.net/. 42, 43

[11] "OpenCV Blob Extraction Library". http://opencvlibrary.sourceforge.net/cvBlobsLib. 43

[12] N. Tanibata, N. Shimada, and Y. Shirai, "Extraction of Hand Features for Recognition of Sign Language Words", in *In International Conference on Vision Interface*, pp. 391–398, 2002. 43

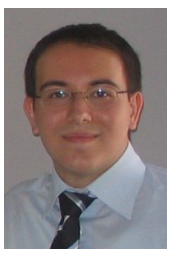
[13] S. Ong and S. Ranganath, "Automatic Sign Language Analysis: A Survey and the Future beyond Lexical Meaning", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 873–891, 2005. 44

9. BIOGRAPHIES

Oya Aran received the BS and MS degrees in Computer Engineering from Boğaziçi University, İstanbul, Turkey in 2000 and 2002,respectively. She is currently a PhD candidate at Boğaziçi University working on dynamic hand gesture and sign language recognition. Her research interests include computer vision, pattern recognition and machine learning.
Email: aranoya@boun.edu.tr

Ismail Ari received his BS degree in computer engineering from Boğaziçi University in 2006. He is currently an MS student in the same department progressing a thesis entitled "Facial Feature Tracking and Recognition for Sign Language" under the supervision of Prof. Lale Akarun. His research interests are computer vision, image processing and computer graphics.
Email: ismailar@boun.edu.tr

Pavel Campr was born in Zatec, Czech Republic. He attended Faculty of Applied Sciences, University of West Bohemia in Pilsen from 2000-2005, where he received master's degree in cybernetics with honors. As a Ph.D. candidate at the Department of Cybernetics, University of West Bohemia in Pilsen, his research interests include sign language and gesture recognition, image and signal processing.
Email: campr@kky.zcu.cz



Erinç Dikici received his B.S. degree in Telecommunications Engineering from İstanbul Technical University (ITU), Turkey, in 2006. He is currently an M.S. student in Electrical and Electronic Engineering Department at Boğaziçi University. His research interests include image and audio/speech processing, specifically, speaker clustering, music transcription, and musical instrument synthesis.
Email: erinc.dikici@boun.edu.tr

Marek Hruz was born in Martin, Slovakia. He attended Faculty of Applied Sciences, University of West Bohemia in Pilsen, Czech Republic from 2001-2006, where he received master's degree in cybernetics. As a Ph.D. candidate at the Department of Cybernetics, University of West Bohemia in Pilsen, his research interests include sign language and gesture recognition, image and signal processing.
Email: mhruz@kky.zcu.cz

Deniz Kahramaner was born in İstanbul, Turkey. He is currently attending Robert College in İstanbul. His research interests include sign language and gesture recognition.
Email: dennizk@gmail.com

Siddika Parlak received her B.S. degree in the Electrical and Electronics Engineering Department of Boğaziçi University in 2006. She is currently an M.S. student at the same department and research asistant in BUSIM (Boğaziçi University Signal and Image Processing) Laboratory. Her main interests are in the field of Speech and Language Processing. She is pursuing her M.S. degree under the supervision of Prof. Murat Saraçlar.
Email: siddika.parlak@boun.edu.tr

Lale Akarun received the B.S. and M.S. degrees in electrical engineering from Boğaziçi University, İstanbul, Turkey, in 1984 and 1986, respectively, and the Ph.D. degree from Polytechnic University, Brooklyn, NY, in 1992. From 1993 to 1995, she was Assistant Professor of electrical engineering at Boğaziçi University, where she is now Professor of computer engineering. Her current research interests are in image processing, computer vision, and computer graphics.
Email: akarun@boun.edu.tr

Murat Saraçlar received his B.S. degree from Bilkent University, Ankara, Turkey in 1994. He earned both his M.S.E. and Ph.D. degrees from Johns Hopkins University, Baltimore, MD, USA in 1997 and 2001 respectively. He worked on automatic speech recognition for multimedia analysis systems from 2000 to 2005 at the AT&T Labs Research. In 2005, he joined the Department of Electrical and Electronic Engineering at Boğaziçi University as an assistant professor. His main research interests include all aspects of speech recognition, its applications, as well as related fields such as speech and language processing, human-computer interaction and machine learning. He authored and co-authored more than two dozen papers in refereed journals and conference proceedings. He has filed four patents, both internationally and in the US. He has served as a reviewer and program committee member for various speech and language processing conferences and all the major speech processing journals. Dr. Saraclar is currently a member of ISCA and IEEE, serving as an elected member of the IEEE Signal Processing Society Speech and Language Technical Committee (2007-2009).
Email: murat.saraclar@boun.edu.tr