Semi-automatic Annotation of Sign Language Corpora

Marek Hrúz, Pavel Campr, Miloš Železný

Department of Cybernetics, Faculty of Applied Sciences, University of West Bohemia
Univerzitní 22, Pilsen, Czech Rep.
mhruz@kky.zcu.cz, campr@kky.zcu.cz, zelezny@kky.zcu.cz

Abstract

Automatic Sign Language Recognition is a problem that is being solved by many research institutes in the world. Up to now there is a deficiency of corpora with good properties such as high resolution and frame rate, several views of the scene, detailed annotation etc. In this paper we take a closer look at the annotation of available data.

1. Introduction

The first step of automatic sign language recognition is feature extraction. It has been shown which features are sufficient for a successful classification of a sign (Ong and Ranganath, 2005). It is the hand shape, orientation of the hand in space, trajectory of the hands and the non-manual component of the speech (facial expression, articulation). Usually the efficiency of the feature extracting algorithm is evaluated by the rate of recognition of the whole system. This approach can be confusing since the researcher cannot be always sure which part of the system is failing. However if the corpora were available with a detailed annotation of these features the evaluation could be more precise. A manual creation of the annotation data can be very time consuming. We propose a semi-automatic tool for annotating trajectory of head and hands and the shape of the hands.

2. Goal of the paper

The goal of this paper is to introduce a system for semi-automatic annotation of sign language corpora. There is some annotation software available (for example ELAN) but the possibilities of these programs are limited. Usually we are able to select a region in a video stream where a sign is performed and note some information about this sign. This process is inevitable for sign recognition and sign language understanding. However if we want to evaluate the feature extracting algorithm we need a lower level annotation of the features themselves in every frame. This annotation has several benefits. We can use the features from the annotation to build and test a recognition system. We can use the features to train models of movement and hand shape. And finally we can compare a set of automatically detected features with the features from annotation.

3. Annotation of features

There are many ways to describe the features needed for an automatic sign language recognition. We chose the following description:

- trajectory - a set of 2D points representing the mean of the contour of an object (or center of mass) for every frame
- hand shape and orientation - we use seven Hu moments (Hu, 1962)
- non manual component - a gray-scale image of the face

From this set of features we derived that the needed annotation of the image data is a countour of the hands and the head. Detecting the contour can be very time expensive for a human but there are many methods for extracting the contour automatically. Next step is to decide which object is represented by the contour. It is a very easy task for a human but again can be time consuming. That is why we developed a tracker for this purpose.

3.1. Tracking process

The tracker is based on a similarity of the scalar description of the objects. We describe the objects by:

- seven Hu moments of the contour
- a gray scale image (template)
- position
- velocity
- perimeter of the contour
- area of the bounding box
- area of the contour.

For every new frame all objects in the image are detected and filtered. Every tracker instance computes the similarity of the tracked object and the evaluated object.

$$S_{Hu} = \sum_{i=1}^{7} \frac{1}{m_i^A} - \frac{1}{m_i^B} \tag{1}$$

where A denotes the first shape (tracked object in the last frame), B denotes the second shape (object in actual frame),

$$m_i^A = sign(h_i^A) \cdot log(h_i^A) \tag{2}$$

$$m_i^B = sign(h_i^B) \cdot log(h_i^B) \tag{3}$$

where h_i^A is the i-th Hu moment of the shape A and analogical for h_i^B. S_{Hu} then denotes the shape (contour) similarity. Next we present the similarity of the template. For this purpose we have to compute the correlation between the template of the tracked object and the evaluated object.

$$R(x,y) = \frac{\sum_{x'}\sum_{y'} T'(x',y') \cdot I'(x+x',y+y')}{T' \otimes I'} \quad (4)$$

where

$$T' \otimes I' = \sqrt{\sum_{x'}\sum_{y'} T'(x',y')^2 \sum_{x'}\sum_{y'} I'(x+x',y+y')^2} \quad (5)$$

where

$$T'(x',y') = T(x',y') - \frac{1}{(w \cdot h) \cdot \sum_{x''}\sum_{y''} T(x'',y'')} \quad (6)$$

$$I' = I - \frac{1}{(w \cdot h) \cdot \sum_{x''}\sum_{y''} I(x+x'',y+y'')} \quad (7)$$

where I is the image we search in, T is the template that we search for, w and h are the width and height of the template respectively. Then

$$S_T = \max_{x,y} R(x,y) \quad (8)$$

is the template similarity. The other similarity functions are an absolute difference between the values in last frame and in the present frame.

$$S_P = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \quad (9)$$

is the similarity of position, where $[x,y]^T$ is the center of the mass of the object.

$$S_V = \sqrt{(v_x^t - v_x^{t-1})^2 + (v_y^t - v_y^{t-1})^2} \quad (10)$$

is the similarity of velocity, where $[v_x, v_y]^T$ is the velocity of the object. The velocity can be aproximated as

$$\vec{v} = \begin{bmatrix} x_t - x_{t-1} \\ y_t - y_{t-1} \end{bmatrix} \quad (11)$$

thus the equation 10 becomes

$$S_V = \sqrt{(x_t - 2x_{t-1} + x_{t-2})^2 + (y_t - 2y_{t-1} + y_{t-2})^2} \quad (12)$$

$$S_{PC} = |p_t - p_{t-1}| \quad (13)$$

is the similarity of the perimeter of the object, where p is the perimeter of the object.

$$S_{ABB} = |abb_t - abb_{t-1}| \quad (14)$$

is the similarity of the area of the bounding box, where abb is the area of the bounding box of the object. A bounding box is a non-rotated rectangle that fits the whole object and has minimum area.

$$S_{AC} = |ac_t - ac_{t-1}| \quad (15)$$

is the similarity of the area of the object, where ac is the area of the object. Based on the values of the similarity functions the tracker has to determine the likelihood (or certainty) with which the object is the tracked object. The likelihood function can be built in many ways. We use a trained Gaussian Mixture Model (GMM) to determine the likelihood. Every similarity function responds to one dimension. There are seven similarity functions which means a 7D feature space and a 7D GMM. The training samples are collected during annotation with and untrained tracker. The untrained tracker doesn't give good results and that's why the user has to manually annotate almost every frame. This situation can be overcomed by manually setting the tracker parameters. In this case the overall similarity function can be a linear combination of the partial similarity functions. That is

$$S = \vec{w}^T \begin{bmatrix} S_{Hu} \\ S_T \\ S_P \\ S_V \\ S_{PC} \\ S_{ABB} \\ S_{AC} \end{bmatrix} \quad (16)$$

where $\vec{w}$ is the weighting vector. An expert can set the weights for better tracking performance. The weights can be then iteratively recomputed based on the data from annotation using a least squares method. After few iterations the data can be used to train the GMM.

As long as the tracker's certainty is above some threshold, the detected features are considered as ground truth. At this point all available data are collected from the object and saved as annotation. If the level of uncertainty is high, the user is asked to verify the tracking.

If a perfect tracker was available all the annotation could be created automatically. But the trackers usually fail when an occlusion of objects occurs. Because of this problem the system must be able to detect occlusions of objects and have the user verify the resulting tracking. In our system we assume that the bounding box of an overlapped object becomes relatively bigger in the first frame of occlusion and relatively smaller in the first frame after occlusion. We consider the area of the bounding box as a feature which determines the occlusion. In Figure 1 you can see the progress of the area of the bounding box of the right hand through the video stream of a sign *Brno*. Figure 2 is the difference of the area computed as

$$\Delta a = a_t - a_{t-1} \quad (17)$$

where a_t is the area of the bounding box in time (frame) t. Figure 3 shows the relative difference and thresholds.

$$\Delta a = \frac{a_t - a_{t-1}}{a_{t-1}} \quad (18)$$

The upper threshold set to 0.8 is used for the detection of first occlusion. The lower threshold set to -0.4 is used for the detection of the first frame after occlusion. The experiments were done on database UWB-06-SLR-A (Campr et al., 2007).

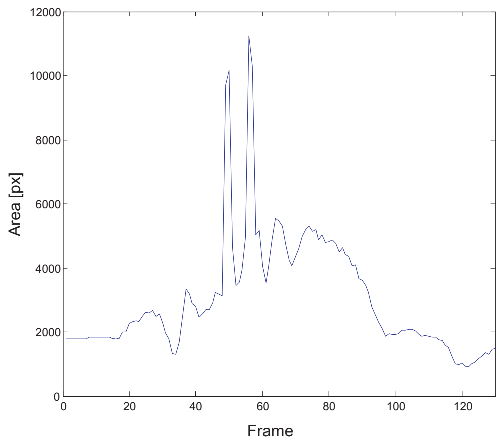

Figure 1: Area of the bounding box of the right hand in pixels.

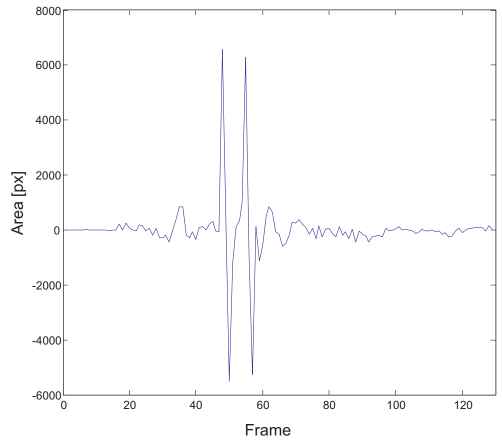

Figure 2: Difference of area of the bounding box of the right hand in pixels.

3.2. Annotation process

The annotation itself begins with loading the video file. In the first frame the trackers are initialized. There is one tracker for one object. In the case of sign language the objects are head, left and right hand. So there are three trackers in this scenario. The initialization process is as follows. The image is segmented using a skin color model. All the small objects and the very large objects are filtered out. Every tracker is created with a search window. If an object is found in this window, the tracker is initialized by this object. The result of the initialization is presented to a human. The human has to decide whether the trackers are initialized correctly and if not, he has to initialize them manually. The trackers are identified by a green contour of the tracked object, a blue bounding box of the object and a string with the class of the object (left hand, right hand, head). After

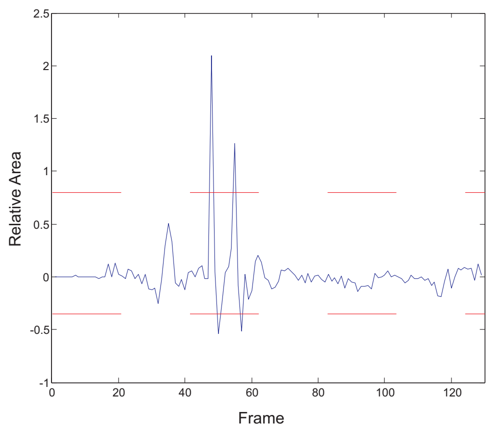

Figure 3: Relative difference of area of the bounding box of the right hand in pixels. The dashed lines are upper and lower thresholds for occlusion detection.

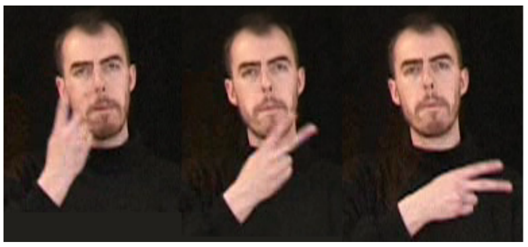

Figure 4: Selected frames (48, 49, 50) from the video stream of a sign Brno. Notice that in the frame 48 the relative difference of area is over the upper threshold and in frame 50 is below the lower threshold.

the initialization the above mentioned tracking process begins. The human operator can pause the video stream in any frame and with a key stroke he is able to view the stream frame-by-frame. If the area of the bounding box changes rapidly, the system pauses the stream automatically. Usually this is a sign where two or more objects collided with each other or were separated from each other. This state can create a confusion for the tracker and the user has to verify the correctness of the automatic annotation. If the annotation doesn't seem right, the user can modify it. In this case all the detected objects are presented to the user and he can annotate (assign a tracker to the object) the object. This way the user doesn't need to annotate every frame which means he saves a lot of time.

3.3. Verification process

After the annotation is done the user can verify it. The system loads the saved features of the video stream and presents them to the user. In every frame the system draws the detected contours and bounding boxes along with the string identifier into the image from the video stream. This way the user is able to tell whether the annotation was successfull or not. Some additional information can be seen in the verification mode. It is a line connecting the center of mass of the object in the last frame and in the present frame. The length of the line is also written on the screen.

This may be helpful when an expert is setting the tracker parameters. Again, the user can pause the stream any time and view the video frame-by-frame.

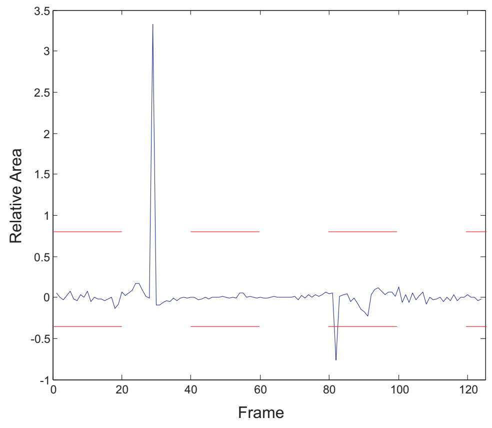

Figure 5: Relative difference of area of the bounding box of the right hand in pixels of the sign divka (girl).

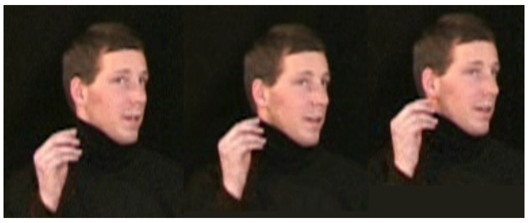

Figure 6: Selected frames from the video stream of a sign divka. You can observe a frame just before occlusion (29), the first frame of occlusion (30) and the consequent frame (31).

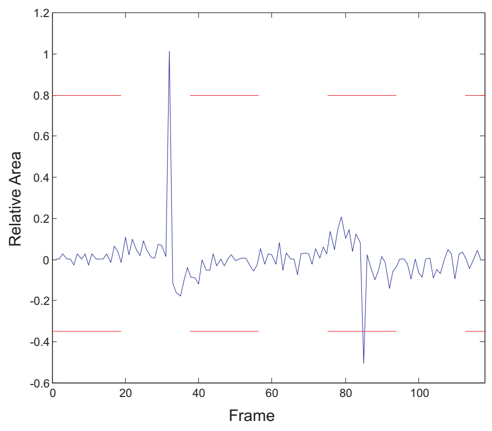

Figure 7: Relative difference of area of the bounding box of the right hand in pixels of the sign loucit se (farewell).

4. Conclusion

We present a system for semi-automatic annotation of Sign Language Corpora. The system can help experts to annotate the sing language video streams without any major time consumption. The annotation is useful for feature extraction, as the features can be computed from the annotation data. This way a system of recognition can be developed independently from the feature extracting system. New algorithms for feature extraction can be compared with the baseline system, not only in the domain of recognition but also in the correctness of the extracted features. Up to now the annotation through tracker allows us to semi-automatically obtain the trajectory of head and hands and the shape of the hands. In the future we will extend the system to be able to determine the orientation of hands and combine it with a lip-reading system which we have available (Císař et al., 2007). The verification mode is a fast way to verify your annotation and it helps experts to set the tracker parameters manually.

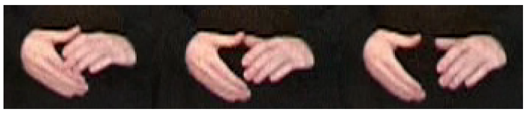

Figure 8: Selected frames from the video stream of a sign loucit se. You can observe the last frame of occlusion (83), the first frame after occlusion (84) and the consequent frame (85).

5. Acknowledgement

This research was supported by the Grant Agency of Academy of Sciences of the Czech Republic, project No. 1ET101470416 and by the Ministry of Education of the Czech Republic, project No. ME08106.

6. References

P. Campr, M. Hrúz, and M. Železný. 2007. Design and recording of signed czech language corpus for automatic sign language recognition. *Proceedings of Interspeech 2007*, pages 678–681.

P. Císař, M. Železný, J. Zelinka, and J. Trojanová. 2007. Development and testing of new combined visual speech parameterization. *Proceedings of the workshop on Audio-visual speech processing.*, pages 97–100.

M. K. Hu. 1962. Visual pattern recognition by moment invariants. *IRE Trans. Information Theory*, 8:179–187.

S.C.W. Ong and S. Ranganath. 2005. Automatic sign language analysis: A survey and the future beyond lexical meaning. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27:873–891.