Tradeoffs in the Efficient Detection of Sign Language Content in Video Sharing Sites

CAIO D. D. MONTEIRO, FRANK M. SHIPMAN, SATYAKIRAN DUGGINA, and RICARDO GUTIERREZ-OSUNA, Texas A&M University

Video sharing sites have become keepers of de-facto digital libraries of sign language content, being used to store videos including the experiences, knowledge, and opinions of many in the deaf or hard of hearing community. Due to limitations of term-based search over metadata, these videos can be difficult to find, reducing their value to the community. Another result is that community members frequently engage in a push-style delivery of content (e.g., emailing or posting links to videos for others in the sign language community) rather than having access be based on the information needs of community members. In prior work, we have shown the potential to detect sign language content using features derived from the video content rather than relying on metadata. Our prior technique was developed with a focus on accuracy of results and are quite computationally expensive, making it unrealistic to apply them on a corpus the size of YouTube or other large video sharing sites. Here, we describe and examine the performance of optimizations that reduce the cost of face detection and the length of video segments processed. We show that optimizations can reduce the computation time required by 96%, while losing only 1% in F1 score. Further, a keyframe-based approach is examined that removes the need to process continuous video. This approach achieves comparable recall but lower precision than the above techniques. Merging the advantages of the optimizations, we also present a staged classifier, where the keyframe approach is used to reduce the number of non-sign language videos fully processed. An analysis of the staged classifier shows a further reduction in average computation time per video while achieving similar quality of results.

Categories and Subject Descriptors: I.4.9 [**Image Processing and Computer Vision**]: Applications; K.4.2 [**Computers and Society**]: Social Issues—*Assistive technologies for persons with disabilities*

General Terms: Algorithms, Design, Experimentation, Human Factors

Additional Key Words and Phrases: Sign language, ASL, video analysis, video sharing, metadata extraction

ACM Reference format:
Caio D. D. Monteiro, Frank M. Shipman, Satyakiran Duggina, and Ricardo Gutierrez-Osuna. 2019. Tradeoffs in the Efficient Detection of Sign Language Content in Video Sharing Sites. *ACM Trans. Access. Comput.* 12, 2, Article 5 (June 2019), 16 pages.
https://doi.org/10.1145/3325863

1 INTRODUCTION

Identification of moments where communication takes place in a recording is crucial for many applications. This involves voice detection in an audio signal for spoken languages and detecting

Author's address: C. D. D. Monteiro, F. M. Shipman, S. Duggina, and R. Gutierrez-Osuna, Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843-3112; emails: {shipman, rgutier}@cse.tamu.edu.

1936-7228/2019/06-ART5 $15.00
https://doi.org/10.1145/3325863

sign language in a video signal for sign languages (SLs). This article describes and evaluates optimizations for SL detection algorithms aimed at reducing computation without severely impacting accuracy. This article is a revised and expanded version of Shipman et al. (2017), exploring continuous video-based and keyframe-based classification; it includes the addition of the design and evaluation of a cascading classifier combining these techniques and metadata features with the goal of bringing together the greater improvement in efficiency of keyframe-based and textual techniques with the accuracy of continuous video-based techniques.

Many people who are deaf or hard of hearing communicate via sign language. With webcams and free video sharing sites like YouTube and Vimeo, the volume of SL content recorded and made publicly available is growing rapidly. In practice, members of the SL community often directly share links to these videos with friends and acquaintances through email or posts in social media. However, when an information consumer, rather than an information provider, wants to locate SL content on a particular topic, they must rely on the existence of metadata that identifies both the topic and language used in the video. Our prior studies of SL access via textual queries have shown that, due to inconsistent metadata (Marshall 2009), metadata-based access to SL videos on particular topics had precision rates averaging 43% (Shipman et al. 2014). The retrieval and presentation of non-SL content, even when queries included terms like ASL or "American sign language," indicates that automatic SL detection would help this situation.

2 PRIOR WORK

Since 2011, we have been exploring how to identify the SL videos uploaded to video sharing sites. Our approach relies on using face detection to define regions of interest (RoI) in which to examine aggregate features of human gesturing, rather than trying to identify particular hand shapes or signs. The earliest work explored the relative value of a set of video features (Monteiro et al. 2012), including the quantity, location, speed, and symmetry of motion in the RoIs. The videos used to assess the classifiers were selected from YouTube and other video sharing sites with the videos in the non-SL corpus being likely false positives due to including individuals gesturing or otherwise moving their hands and arms. The results indicated that video features alone could be used to detect SL content. The results also showed that symmetry of motion was more valuable in differentiating between signing and other forms of human gesturing than were the other video features.

The initial study was constrained to videos containing a single person with moving arms and hands facing the camera and thus does not represent the set of videos found on video sharing sites. Following on the initial work, we developed a second corpus of SL and non-SL videos that were collected by being returned via queries including ASL or "sign language" along with other query terms. The resulting videos could include multiple visible people, sometimes engaged in a sign language conversation. Thus, this video corpus better represents the expected use case for the SL detection techniques being developed.

Based on the value of symmetry of motion, we developed a richer set of video features that preserves more information about the location of movement in the RoIs. In this later approach, likely signing activity in each RoI in a video is modeled using polar coordinates (angle and distance) in a *polar motion profile* (PMP), which relies on a Gaussian Mixture Model for background modeling and subtraction (Zivkovic 2004) and face detection using Haar features (Viola and Jones 2001). The result improved recall but with considerable computational cost (Karappa et al. 2014). Reduction in computation time is needed to scale the solution to the quantity of videos uploaded to video sharing sites. Of course, differentiating between SL and non-SL videos is only a first step in providing videos in a particular sign language. Once a fast approach to detecting sign language in videos is available, more computationally expensive techniques can be applied to detect which sign language it is (Monteiro et al. 2016).

This article examines a range of techniques to reduce the amount of computation needed to generate PMPs for a video. We compare the accuracy and computational costs of alternate face detectors and examine executing face detection on sampled video frames to reduce computation. We also examine how shortening the length of the video segments analyzed effects the quality of resulting classifications. These results lead to a recommended combination of these optimizations, which is analyzed in terms of both computation costs and accuracy of results. Beyond processing continuous video, we explore the potential for computing PMPs from sampled frames without generating a background model – an approach that greatly reduces computation but comes with a greater reduction in accuracy. Finally, we present and evaluate alternative designs for a staged classifier that initially uses the least-costly keyframes approach to select videos for follow-on processing using more costly techniques and show how we can achieve high-quality results while greatly reducing the average time per video classified.

3 RELATED WORK

Considerable research has examined the problem of transcribing sign language into written words. Sign language interpretation involves recognizing hand shape, orientation, and motion direction combined with facial expressions and thus relies on multiple types of analysis of video content (Caridakis et al. 2008). Transcription would be useful for those not in the SL community to understand the videos in SL and would also enable search over signed content. However, current techniques are often constrained to limited vocabularies, limited signing speeds, and place constraints on signer position and background. As a result, these techniques are not currently applicable to the highly variable quality of videos found on video sharing sites.

The video processing used to transcribe sign language informs our approach to detecting sign language. The use of face detection to identify the likely signing region and the inference of hand position using foreground-background separation are the initial steps for many transcription efforts. The higher-level features computed based on this initial processing must enable classification for a single frame or the short sequence of frames associated with a single sign. As a result, they use either a geometric model of the signer's hands (Somers and Whyte 2003) or a catalog of examples of particular signs used for similarity assessment with known signs (Dimov et al. 2007) or handshapes (Potamias and Athitsos 2008). The order of handshapes matter as well, resulting in the use of models that capture temporal sequences of handshapes (Starner and Pentland 1995; Vogler and Metaxas 1999; Hernandez-Rebollar 2005). Given our goal of detecting signing rather than identifying specific signs and their order, such detailed analysis is unnecessary.

Instead of transcribing SL, our focus is more modest: detecting SL. Being able to identify which videos available from a video sharing service are accessible to signers would improve access to information for those in the signing community. The earliest work related to this problem was by Cherniavsky et al. (2008). To more efficiently use the limited video connections between signers, they developed an algorithm to detect activity to provide greater bandwidth to the active signer during mobile video communication. But this approach was not meant to differentiate between sign language and other forms of human gesturing and hand/arm motion.

Similar to the fact that speech detection is a first step towards classification of an audio signal as containing a particular language, recognizing that a video contains sign language does not mean it is accessible to a member of the signing community. Once a video is believed to include sign language, techniques are needed to identify which sign language is in use. Gebre et al. (2013, 2014) and Toman (2016) have tackled the problem of distinguishing between sign languages based on features learned from a single or small series of video frames. Such techniques have the potential to be efficient in terms of processing due to the limited number of frames involved in each classification but are more likely to be susceptible to misclassification due to similarity between signing

and other human gestures when detecting sign language and similar or shared signs across languages when distinguishing between sign languages. We quantify this challenge for sign language detection later in the article when we examine the accuracy of using the PMP features for shorter video segments and keyframes.

Video features are also affected by the variety and quality of the video being analyzed. The PMP video features developed to detect sign language in our approach have been used to distinguish between pairs of sign languages (Monteiro et al. 2016) both with professionally produced videos and with more typical shared videos. The results show that classifier accuracy is considerably lower for the type of videos found on video sharing sites than they are for the professionally produced SL videos. Professionally produced videos have higher video quality (e.g., good contrast, plain backgrounds) and the signing in these videos is slower and more carefully expressed than signing in typical videos on video sharing sites.

Overall, the PMP features have been shown to enable distinguishing between SL and non-SL videos and between certain pairs of sign languages. The next section describes the PMP features and leads into the optimizations explored to reduce the overhead of generating these features.

4 POLAR MOTION PROFILES

Our SL-video classifier is composed of two components: a video processing module that generates video features, and a classification module that determines whether or not they have sign language content. Here, we explain the process of generating the polar motion profiles (PMP) video features.

PMPs are a translation and scale invariant measure of the amount of activity in the video, computed on a polar coordinate system centered on each face. Generation of the PMP requires the results of both face detection and background subtraction modules. Figure 1 shows the steps involved on the computation of a PMP for a given video.

To reduce errors in face detection as signers alter their orientation with the camera, the individual face detectors provided by the openCV library (Default, Alt, Alt2, Alt Tree, Profile) are used as an ensemble face detector in Karrapa et al. (2014). This is shown in Figure 1(a). In this approach, areas that receive a majority vote (Figure 1(b)) are considered face locations for subsequent steps. Regions of interest based on the location of the detected faces are defined in every frame of the video (Figure 1(e)). Each ROI is computed to be large enough to capture the arm and hand movements for the person detected. Activity inside each ROI is identified using foreground-background separation (Figure 1(c) and Figure 1(d)) based on the adaptive Gaussian mixture model described in Zivkovic (2004).

Given the ROI and the foreground pixels, PMPs are computed to represent the activity in a video. The resulting PMP is a feature vector with 460 elements, 360 representing each angular coordinate and 100 elements for radial coordinates (Figure 1(f)). Equation (1) details the computation for the angular coordinate (θ) features; a similar equation is used to obtain the radial coordinate features:

$$PMP(\theta) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{R(t)}\sum_{r=1}^{R(t)} PMP_r(\theta, t), \tag{1}$$

where $R(t)$ is the number of ROIs at frame t, T is the total number of frames in the video, and $PMP_r(\theta, t)$ is the ratio of foreground pixels (FG) to the sum of foreground pixels and background pixels (BG) at polar coordinate θ for region r and frame t:

$$PMP_r(\theta, t) = \frac{FG_r(\theta, t)}{FG_r(\theta, t) + BG_r(\theta, t)}. \tag{2}$$

Principal component analysis (PCA) is a statistical technique frequently used to reduce high-dimensional feature vectors to a lower dimensionality where each new dimension is a weighted

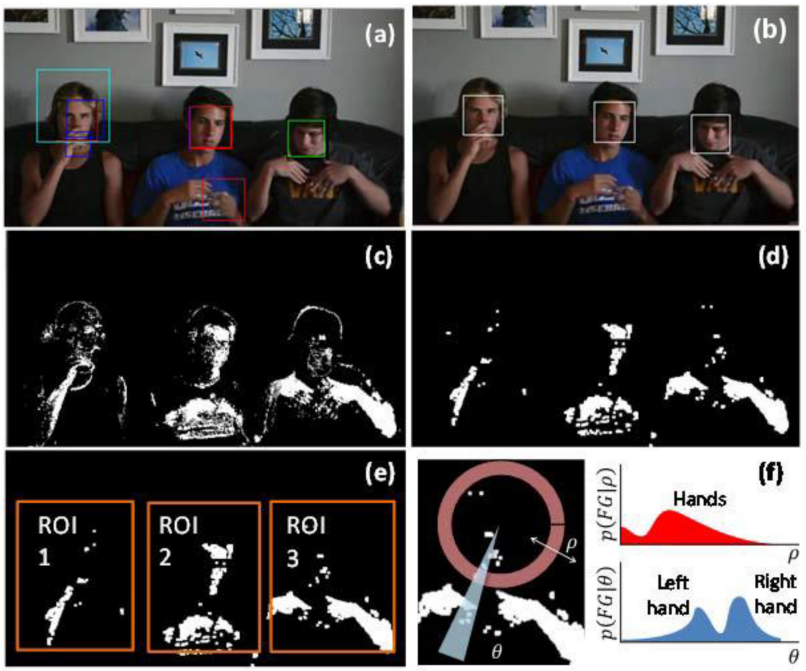

Fig. 1. (a) Faces identified by each face detector. (b) Faces from the ensemble of the detectors. (c) Foreground (FG) pixels returned by the background subtraction. (d) Refined FG after morphological de-noising. (e) Regions of interest (ROI) defined for each face detected in the frame. (f) Computation of PMPs for a video frame.

sum of the original dimensions. The weights are generated to preserve as much of the original data as possible. Here, we apply PCA to reduce the 460 initial video features to 6 features that are then passed to a support vector machine (SVM) during training and testing.

5 OPTIMIZATIONS

There are a variety of approaches available to reduce the computation time associated with PMP generation. Each of these techniques may detrimentally affect classification accuracy. We initially explored (1) reducing face detection computation time by using a single face detector instead of the ensemble approach, (2) applying face detection to a reduced number of frames, and (3) reducing the length of the video segments used for classification. The initial goal was to match the performance of background subtraction, which happens at a fraction of real-time, i.e., 1s of a video is processed in less than 1s. If the face detection time can match the background subtraction, then the lag to generate PMPs can be minimized, since the two efforts can be executed in parallel.

5.1 Face Detection Computation Time

Face detection is computationally expensive. The ensemble of five face detectors proved to be effective in reducing false positives but at the expense of computational cost. False positives in face detection can introduce PMPs with no signer in the region. When there is no signer, the activity in the region of interest might become trivial due to morphological opening (erosion and dilation) after background subtraction. Hence, the PMP for false positive faces may not significantly affect the performance of the classifier, allowing the use of a single face detector to reduce computation

time. To support this hypothesis, we replaced the ensemble of face detectors with each of the individual face detectors, and measured precision, recall, and computation time.

A second approach to reducing computation for face detection examines how performing face detection on every Nth frame affects SL detection. We had reason to believe that sub-sampling would have limited effects, since signers' faces and bodies tend to be relatively stationary in many of the SL videos on video sharing sites. Initially, we interpolated the location of the face between the sampled frames but found that this generated additional computational overhead without improving results. Thus, the ROIs for the frames between sampled frames are the last computed ROI—that is the ROIs are in static positions for the frames in between frames where face detection is computed.

5.2 Shorter Video Segments

A 1min segment of the original video, selected to be centered around the midpoint of the video, was used to perform classification in our earlier studies. This means that face detection and background subtraction was performed on each frame in that segment. Reducing the length of the chosen segments reduces computation but the resulting PMPs may be less representative of the overall video. How short is too short? Answering how varying the length of the segment affects SL detection not only informs the design of optimized SL detectors but helps answer to what degree fine-grained diarization (i.e., discriminating segments of a video that include SL from those that do not) can be achieved.

5.3 Recommended Approach

Using the results from the above assessments, we identify a recommended configuration that substantially lowers computation time while not sacrificing precision and recall. We report the computation time and accuracy of this configuration.

6 EVALUATION

The dataset used for assessing the current work is the same as in Karappa et al. (2014). It was collected from online video sharing sites and each video was manually labeled as SL or Non-SL video. This corpus contains 111 SL videos and 116 non-SL videos considered related to the SL videos in the corpus by the video sharing site—that is the non-SL videos were returned as results for queries including terms like "sign language" or were recommended as related to known SL videos. The videos in this corpus are not limited to videos with a single potential signer facing the camera as in our earliest evaluations. As such, this dataset closely resembles the set of videos that need to be classified in real-world scenarios.

For all experiments, we compare the recall (a measure of false negatives) and precision (a measure of false positives) achieved by our new approaches with the earlier ensemble approach. The F1 score is the harmonic mean of precision and recall. Each value reported is the average of 50 iterations, with each iteration dividing the dataset into training and testing samples randomly. Random sampling was chosen as a technique to separate the labelled corpus into training and testing data for historical reasons. We initially used this approach to determine how increasing the size of the training set affected accuracy.

6.1 Time to Process a Minute of Video

We ran the five individual face detectors in openCV and the ensemble technique on the videos from the dataset to determine their computational time requirements. The length of each video segment was chosen to be 1min and the resolution 240p at 30 frames per second. The results are shown in Table 1. The focus here is on the relative time requirements as the computation was

Table 1. Average time for face detectors for 1min video

Ensemble	Default	Alt	Alt 2	Alt Tree	Profile
896s	94s	161s	130s	107s	174s

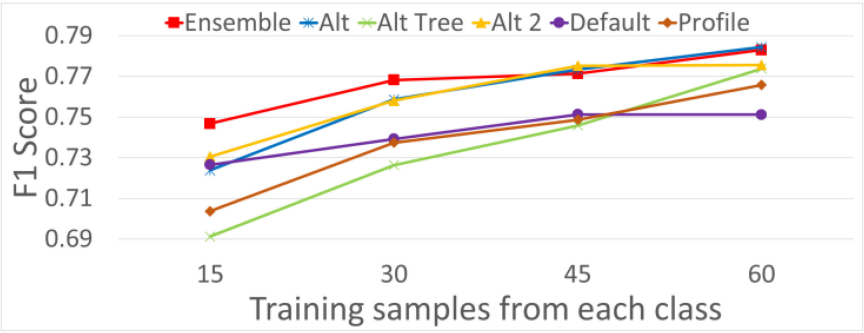

Fig. 2. F1 scores for face detectors and training set sizes.

performed on a quad-core 3.5GHz Windows PC with 8GB of memory so may not be indicative of performance on a modern server.

Using a single face detector instead of the 5-detector ensemble can reduce face-detection time by a factor of 5–10, depending on the particular selection. The time required for the ensemble also indicates that running all five face detectors in parallel takes more time than the sum of time when running them individually.

6.2 Using Single Face Detector

While using a single face detector is much faster, such a choice may negatively affect the accuracy of results. To answer this question, we evaluated classifiers based on each of the five individual face detectors and a classifier that used the 5-detector ensemble. Figure 2 presents the F1 score as a function of the training set size. As shown, the ensemble is the best face detection technique to be used when training with a limited number of samples and performs well overall. As the number of training samples increase, classifiers with frontal face detectors employing a cascade of stage classifiers and adaptive boosting, i.e., alt and alt2, performed best. Overall, the range of F1 scores shows that using a single face detector instead of the ensemble detector does not substantially impede SL detection.

Taking both accuracy and computation time into consideration, we chose the alt2 frontal face detector over the alternatives for our recommended configuration.

6.3 Sampling Frames for Face Detection

As already mentioned, the body and head of signers in SL videos tend to be relatively stationary. Hence, instead of detecting faces at each frame, we tested sub-sampling frames at regular intervals and detecting faces at only those frames. The frame rate of the videos in the dataset is 30 frames per second. We tested the effect of sampling intervals ranging from 1 (each frame) to 120 (one frame every 4s) for each of the face detectors. The alt and alt2 face detectors consistently performed better than the other individual face detectors. Figure 3 shows that the ensemble and alt2 face detectors performance was relatively stable up to sampling every 20th frame, at which point the performance gradually decreased and became inconsistent.

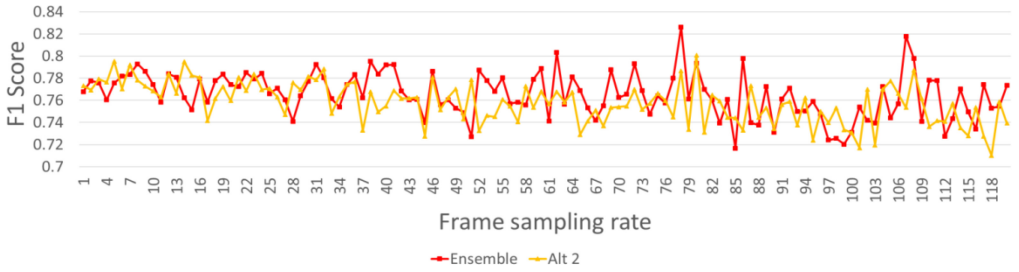

Fig. 3. F1 scores for applying face detection to sampled frames. The erratic nature of the results are likely due to the deterministically sampled frames being more or less representative of the video as the starting frame was fixed for each iteration.

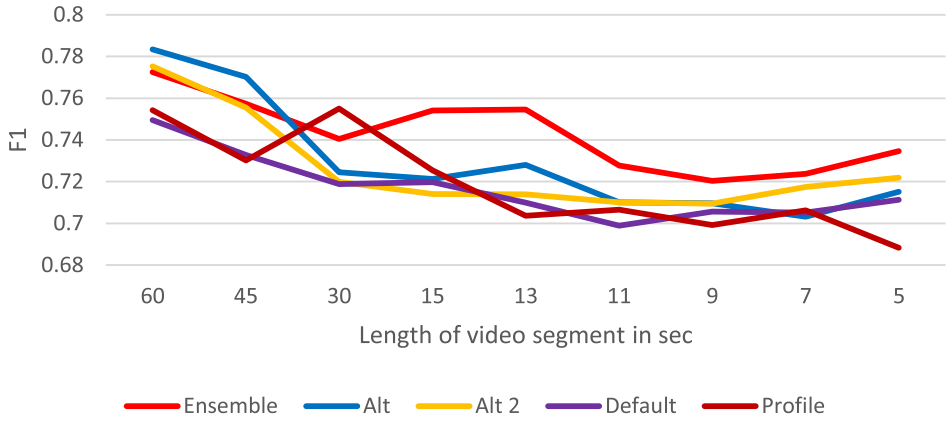

Fig. 4. F1 scores for different segment lengths.

This indicates that a sub-sampling rate of 1/20 significantly reduces computation time without losing much accuracy. Additionally, the slow degradation in F1 score as the number of frames sampled is reduced indicates the potential for frame sampling at much longer intervals, as is explored in Section 7.

6.4 Processing Shorter Video Segments

Shortening the length of video segments for feature extraction and classification provides two advantages: first, the computation time for feature extraction can be substantially reduced; second, it enables the identification of shorter segments of SL content in videos (i.e., diarization). We evaluated classifiers with the individual face detectors to find how they performed relative to the ensemble classifier with shorter segments of videos. To select the shorter segment, we took the segment at the center of the first-minute segment of the full video. For training the classifier, we used 50 samples from each of the SL and non-SL corpus and the remainder of the labelled corpus was used for testing.

Figure 4 shows the F1 scores for the face detectors as the segment lengths vary. The performance degrades as the segment length is decreased. The results show similar performance for four individual face detectors and alt2 performs comparatively well for the shortest segments tested. Given the slight degradation in performance, a shorter segment length optimization is not included in the recommended model discussed next.

Table 2. Evaluation of Recommended Approach

	Karappa et al. Approach	Recommended approach
Average face detection time	896s	31s
Precision	85%	83%
Recall	71%	71%
F1 Score	78%	77%

6.5 A Recommended SL Detection Approach

Based on the above findings, we chose alt2 frontal face detector as the face detector to compare against the original voting scheme. The other design choices considered are detecting faces at a sub-sample rate of 1/20 with 60 training samples from each of the SL and non-SL corpus. Table 2 presents the results. Results were about 2% lower on precision, had the same recall, and a 1% lower F1 score when compared with the original approach. Yet these optimizations reduced computation time for processing the video by 96% (from 896s to 31s) for the 1min segments of video.

Our recommended approach did not explore how segment length would affect computation time and accuracy when combined with sampling. Shortening the segment lengths tended to have a more significant impact on performance but is clearly crucial for diarization, a topic for future work.

7 KEYFRAME-BASED SL DETECTION

Given the huge volume of videos that are uploaded to video sharing sites every day, the optimizations above improve the applicability of SL detection to this context but do not solve the problem. What is needed is a pre-processing stage that identifies videos that warrant such analysis. Toward this end, and based on the results from frame sampling for face detection, we developed a keyframe-based SL detector that uses the same PMP features as before but computes these features for a small set of frames within the video rather than computing them for each frame in the video. Because this approach does not examine every frame, foreground pixels must be identified by alternative means.

As with the above approaches, the keyframe-based approach relies on face detection to identify a region of interest. Within each region of interest, the foreground model is generated via frame subtraction with the prior frame—i.e., the prior frame is used as the background model. More specifically, foreground pixels in a keyframe are the result of applying a median filter to the two greyscaled frames, subtracting the frames, then thresholding the results. Finally, image opening is applied to remove noise among the identified foreground pixels. The resulting foreground image is then used to generate PMPs, which are passed to the SVM for classification.

The advantage of a keyframe-only approach when compared to the above techniques is that only a small number of frames have to be considered. The limitation of using such an approach is that, without a dynamic background model and without information about the intermediate position of the potential signers' arms, the foreground identified is likely to include change within the field of view not associated with hand/arm motions. Changes in body position, lighting, and so on, could overwhelm the foreground data associated with human gestures, creating too much noise for accurate SL detection.

To characterize the performance of keyframe-based SL detection, performance of the resulting classifier was examined as the number of keyframes selected from the video varied. These results were examined for both 30 and 60s video segments. The keyframes included in the analysis were evenly distributed in the video segment. Thus, if five frames are chosen from a 60s video segment, then the frames are chosen at the 0, 15, 30, 45, and 60s points for generating the PMPs.

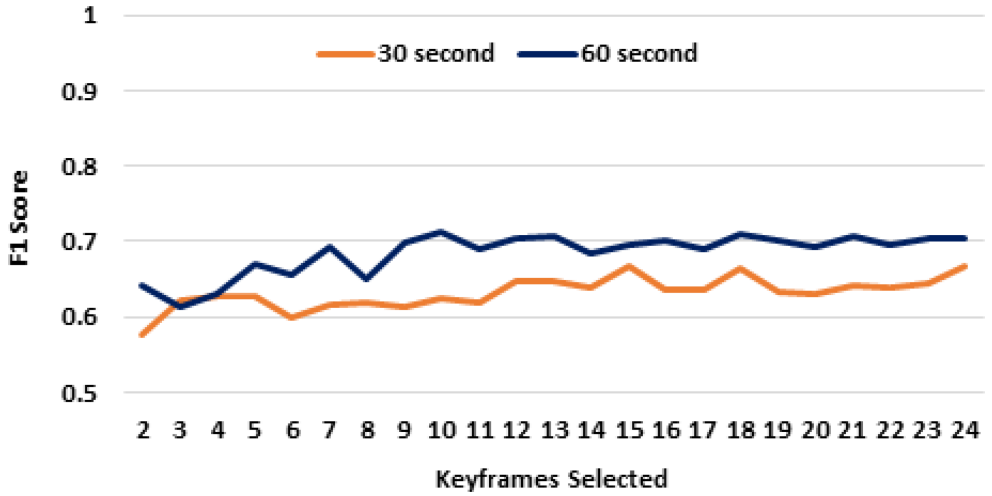

Fig. 5. SL vs. Non-SL F1 score for 30 and 60s videos.

Table 3. Comparison of Keyframe and Karappa Approach.

	Karappa et al. Approach	Keyframe Approach
Precision	85%	69%
Recall	71%	74%
F1 Score	7%	71%

The results in terms of F1 scores using the same corpus of SL and non-SL videos as before is shown in Figure 5. The F1 scores of the classifier are clearly better with the longer (60s) source videos. Since the longer video for the same number of frames doubles the time between examined frames, this indicates that changes in background are not the main cause of incorrect classifications for this approach. Overall performance in terms of the F1 score leveled off when at least 10 keyframes were selected for analysis at about 71% for the longer videos.

When the results for this approach are compared to the ensemble approach described in Karappa et al. (2014), the keyframe-based approach achieves similar or slightly better recall but at a considerable loss in precision. This is shown in Table 3.

While the performance of the keyframe-based SL detector is lower in terms of precision and F1 score, it reduces the number of frames extracted and analyzed from 1800 to 10. This makes the classifier much faster. Indeed, in our studies with this approach, feature generation is reduced to such a point that frame extraction can become a bottleneck depending on how the video is encoded.

In terms of performance within a staged classifier, given the relatively strong recall, there is the potential to include a keyframe-based classifier as an initial filter. Videos identified as likely including SL content by this filter could then be more carefully analyzed by a second, more powerful SL detector. The next section describes such a cascading classifier.

8 CASCADING CLASSIFIER

To benefit from the proposed optimizations while reducing their performance impact, we designed a cascading approach for classifying the videos. The intuition behind the cascading classifier is that videos that are easy to determine as either having or not having sign language will be classified

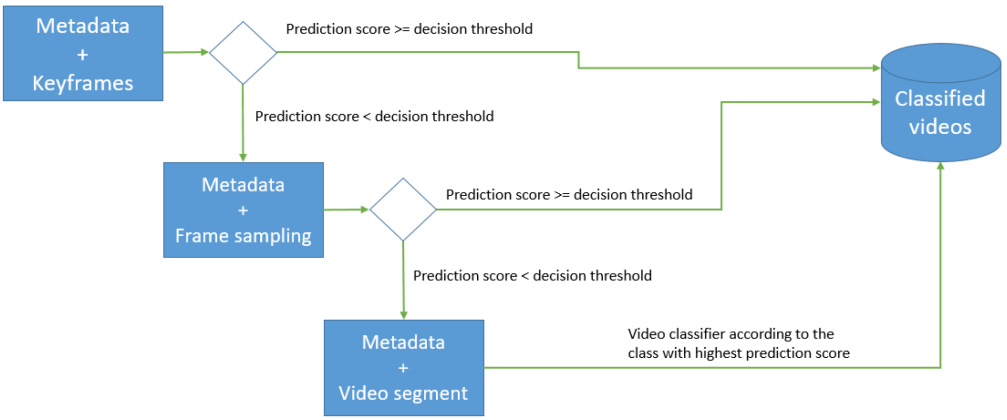

Fig. 6. Overview of the classification pipeline.

by a low-cost approach, such as the keyframe approach just described, reducing the number of videos that require full video feature extraction.

To decrease the rate of false negatives (e.g., classifying a SL video as non-SL), especially at the earlier stages of the pipeline, we incorporated metadata features in our classification process. The value of metadata features for this task and the best way to combine them with visual features was previously investigated in Monteiro et al. (2018). Figure 6 shows an overview of the cascading framework.

For the metadata classifier, we collect title, description, and keywords for each video. We then compute a TF-IDF matrix to represent the dataset metadata and reduce its dimensionality using Non-negative Matrix Factorization (NMF) (Lin 2007). Similar to the visual features, we train an SVM to detect SL based on the extracted NMF transformation.

With both metadata and PMP SVMs trained, we use a type of late fusion to classify new samples. We get the individual assessments of each SVM and compute the product rule of the classes' likelihoods using the method described in Platt (1999). If both classifiers agree on the resultant class and the computed product is greater than a set threshold, then the video is classified as the particular class and no further processing is required. If any of these two validations fail, then the video is then forwarded to the next step in the pipeline. If the video is already at the last stage of the pipeline, then it will be assigned the class with the highest computed product, regardless of agreement between the video and metadata classifiers.

It is important to note that the metadata features used in the cascade are the same in all three stages, thus, the textual to feature transformation just needs to happen once. This is not true for the visual features, where each stage uses more visual information than the previous one. Considering this, we can estimate the time required to classify a video as the following:

- For videos classified at stage 1: Time to perform keyframes computation (when using the keyframes approach, both face detection and background-foreground separation are done sequentially) plus time spent generating PMPs.
- For videos classified at stage 2: Time to perform background-foreground separation (Face detection in the sampled frames is done in parallel to background subtraction and is the fastest process between them), plus time spent generating PMPs, plus time spent at stage 1.
- For videos classified at stage 3: Time to perform face detection in every frame of the video (face detection at every frame is more costly than background subtraction, thus it will dictate

Table 4. Comparison of Keyframe and Karappa Approach

	Keyframe	Sampling	Full
Face detection + Background-Foreground separation	7.22s	18s	183.4s
PMP generation	1.56s	211s	211s
Total	8.78s	229s	394.4s

the time spent at this step), plus time spent generating PMPs, plus time spent at the previous two stages.

9 EVALUATION OF CASCADING CLASSIFIER

To assess the quality of our cascading approach, we ran experiments and analyzed their results in terms of computational costs and classifier accuracy. All the experiments were performed on the same machine, equipped with a quad-core AMD A10-7800 processor and 8GB of RAM memory. The evaluation corpus was a subset of the same dataset used for the evaluations in Sections 6 and 7, containing all the videos that were still online on YouTube, enabling us to get their metadata. This subset consists of 104 non-SL videos and 95 SL videos.

For all the experiments, 60 samples from each class were used for training during each iteration and the remaining videos were used for testing, and all the classification metrics are the average of 1,000 iterations with randomly selected training and testing sets.

Given that the costs for training the SVMs happens just once and the costs for extracting metadata features is negligible when compared to video processing, we focus our attention to the time spent by each video approach in respect to the two phases of video processing: (1) Face detection and Background-foreground separation; and (2) PMP generation. Table 4 displays the average time at each step for each one of the three video approaches (Keyframes, Frame Sampling, Whole video segment).

As is observable, the total time spent on the Keyframe approach (8.78s) is considerably lower than on the other two approaches. This means that classifying videos at the second stage (Sampling + Metadata) is still much faster than using a non-Cascading approach, taking a total time of 238s (9 for stage 1 plus the 229 for stage 2) for a stage 2 cascading classification compared to 394s for the non-cascading classification. Obviously, videos classified at the last stage of the cascading approach take longer than the non-cascading version due to having been processed for stages 1 and 2 as well as the full approach for stage 3. Therefore, for the cascading approach to reduce the average time required to classify a video, a significant number of videos must be classified in the first two stages to offset the added costs for those videos that still require full analysis.

Setting the decision threshold determines how confident the classifier must be to classify a video as either containing or not containing sign language prior to the last stage. We explore how the cascading classifier performs when the decision threshold for both stage 1 and stage 2 are varied between 80% and 20%. Figure 7 shows the average time spent to process a video as the decision threshold required for classification is varied.

We can observe that thresholds higher than 50% actually result in an average processing time that is longer than the traditional approach with no cascade. This is possibly caused by the fact that with a higher threshold, more videos reach the last stage of the cascading pipeline, increasing the average time per video. A closer look at the average number of videos classified per stage confirm this, as is shown in Figure 8.

It is interesting to note that at the 50% mark, the majority of the videos are still being classified just at the last stage, but the large computational savings of classification on the first stage makes up for the difference.

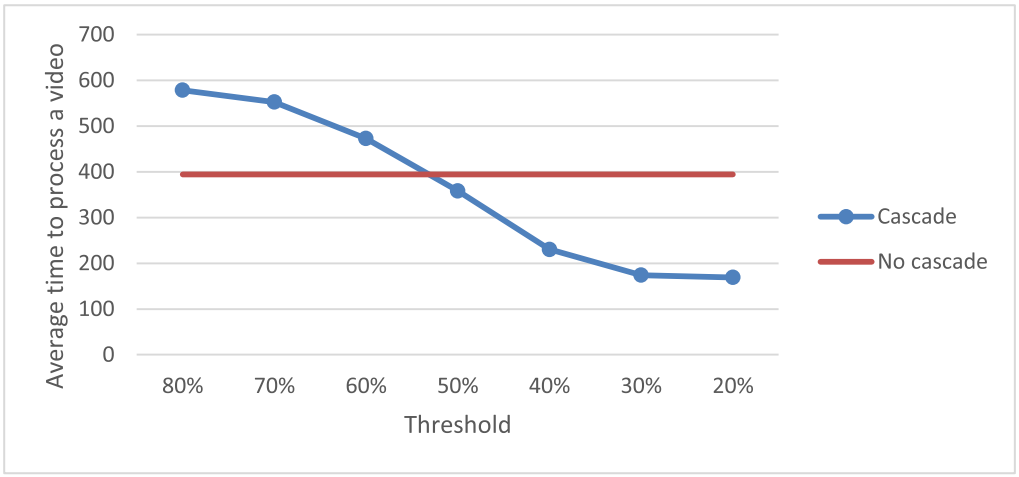

Fig. 7. Average time required to classify each video for different thresholds.

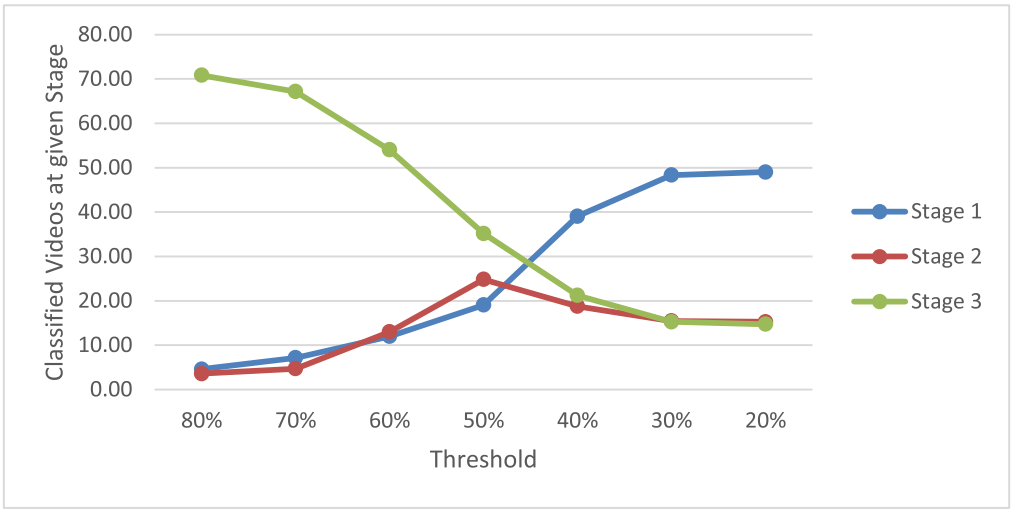

Fig. 8. Percentage of videos classified at each stage for different thresholds.

With respect to the classifier performance, we have analyzed the impact of reducing the threshold in the measures of precision, recall, and F1 score. To serve as a baseline, we first compute the results without using a cascade, Figure 9 shows that the multimodal classifier (video + metadata features) performs better than the unimodal ones, achieving an F1 score of 0.80.

The effects of varying the decision threshold on precision, recall, and F1 score are shown in Figure 10. Overall, the F1 score barely dropped as the decision threshold was reduced from 80% to 50%, indicating that the reduction in computational time between these thresholds brings very little penalty in terms of the cascading classifier's accuracy. From the 50% mark and downwards, we can then observe that the F1 score slowly declines, but even using a threshold as low as 20%, the F1 score was just about 2% below the non-cascading approach although this is the result of a combination of an ~3% increase in precision and ~7% decrease in recall. Overall, this means that with just a small hit on the classifier performance, we can classify a new video in less than half of

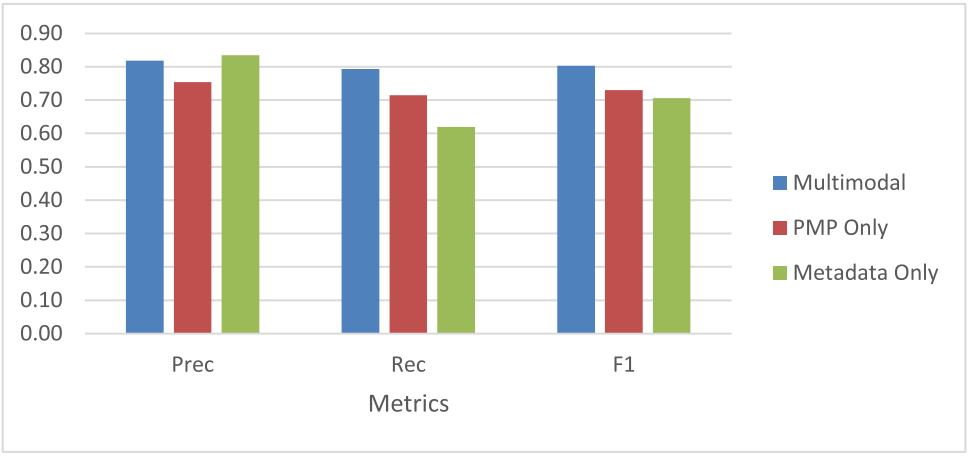

Fig. 9. Precision, recall, and F1 score for using metadata features only, video features only (PMP), and both sets of features during classification.

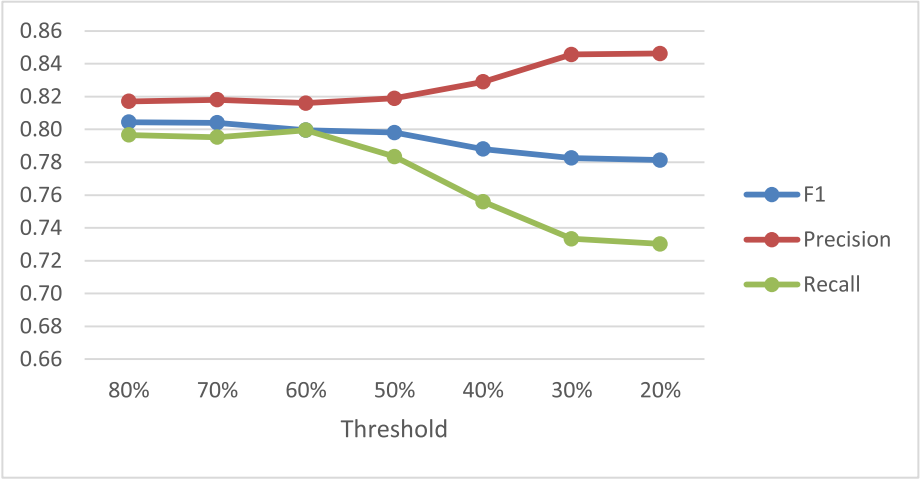

Fig. 10. Precision, recall, and F1 score for the cascading classifier for different decision thresholds.

the time required by the non-cascading approach. That said, this decrease in recall is problematic though as recall errors remove SL videos from the set of videos provided to the SL community. Further effort is needed to bias the training of the low-threshold staged classifier to prefer recall performance over precision.

These results indicate that the cascading classifier with a 20% threshold at all stages performs very similar in terms of F1 score to the original Karappa et al. (2014) approach and the single-stage video-feature recommended approach from Section 6.5. The staged classifier reduces the average computation time required for video classification by one half from that of the recommended approach.

10 DISCUSSION

The problem being examined throughout this article is how to reduce the computational costs associated with detecting sign language in videos found on video sharing sites. The original approach

for extracting the Polar Motion Profile video features depends on face detection and background subtraction, and the generation of PMPs has to wait until data from both are computed. Although background subtraction is faster than real time, face detection with the ensemble face detector could take more than 10min to compute for a video of 1min length.

The first of the three optimizations to reduce the time in the face detection replaced the ensemble of face detectors with individual face detectors. The second optimization limited face detection to sampled frames of the videos rather than application to each frame. The last optimization was to analyze shorter video segments. Based on the evaluation of the individual optimizations, a recommended configuration was identified that achieves performance quite similar that of the original model approach while reducing the computation time in the face detection module by 96%.

Further exploration of a frame-based classifier removed the need for background modeling. While this resulted in a considerable loss in precision, the frame-based classifier had a relatively high recall with computation costs similar to that required for frame extraction. This performance makes it a good candidate as an early filter in a cascading classifier.

Finally, a three-stage cascading classifier composed of a keyframe-based classifier as the first stage, a classifier applying sampled frames face detection as a second stage and the full video feature analysis for the third stage was explored. The performance and computational costs vary largely based on the number of videos classified at each stage of the cascading classifier, which is controlled by setting decision thresholds. Evaluation of alternative thresholds showed that a 50% threshold was required to have any improvement in computational costs and also resulted in little reduction in the quality of the results. When a 20% decision threshold is used, the computational costs are reduced by about half with a 2% drop in F1 score but that included a 7% drop in recall. It may be possible to alter the training to bias recall performance over precision performance.

It is worth noting that the evaluation corpus has approximately the same number of sign language and non-sign language videos to balance the costs of false positives and false negatives. This distribution may result in lower improvements in computational costs from a cascading classifier than would be seen in practice. This is because the set of videos needing classification may well have many more videos without sign language than with sign language and many of these non-SL videos may be easier to classify than the videos in our corpus, resulting in a greater percentage of videos being classified by early stages of cascading classifier. In the explored design, the full video feature extraction was included as the final stage. Given the similarity in performance between the full video feature classifier and the recommended configuration described previously, it is likely that a two-stage classifier may perform quite close to the accuracy reported here but with considerably greater reduction in average computational costs.

The 50/50 mixture of SL and non-SL videos used in this evaluation is closer to what we would expect if an initial metadata-based crawler is run on the broader corpus of shared videos collecting those that are considered candidates for including sign language. Our efforts on the design of crawlers that collect videos considered as possibly including sign language use a combination of content-focused metadata (e.g., descriptions and keywords) and structure metadata (e.g., channel id, co-occurrence on video lists.) A second source of videos to be classified will come from user queries augmented with terms such as "sign language," ASL, and so on. Together, these methods will generate the set of videos classified using the techniques described in this article.

A final consideration is that the videos identified as including sign language content will afterwards be processed by similarly designed staged classifiers for identifying which sign language is being used. By sharing metadata and video features between the SL-detection and SL-identification components, the overall computational costs per video will be reduced.

REFERENCES

G. Caridakis, O. Diamanti, K. Karpouzis, and P. Maragos. 2008. Automatic sign language recognition: Vision-based feature extraction and probabilistic recognition scheme from multiple cues. In *Proceedings of the International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08)*. 8.

N. Cherniavsky, R. Ladner, and E. Riskin. 2008. Activity detection in conversational sign language video for mobile telecommunication. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*. 1–6.

D. Dimov, A. Marinov, and N. Zlateva. 2007. CBIR approach to the recognition of a sign language alphabet. In *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'07)*. ACM, New York, NY, 9.

G. Gebre, O. Crasborn, P. Wittenburg, S. Drude, and T. Heskes. 2014. Unsupervised feature learning for visual sign language identification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 370–376.

B. Gebre, P. Wittenburg, and T. Heskes. 2013. Automatic sign language identification. In *Proceedings of the IEEE International Conference on Image Processing*. 2626–2630.

J. L. Hernandez-Rebollar. 2005. Gesture-driven American sign language phraselator. In *Proceedings of the 7th International Conference on Multimodal Interfaces (ICMI'05)*. 288.

V. Karappa, C. Monteiro, F. Shipman, and R. Gutierrez-Osuna. 2014. Detection of sign-language content in video through polar motion profiles. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'14)*. 1290–1294.

C.-J. Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Comput.* 19, 10 (2007), 2756–2779.

C. Marshall. 2009. No bull, No spin: A comparison of tags with other forms of user metadata. In *Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'09)*. 241–250.

C. Monteiro, C. Mathew, R. Gutierrez-Osuna, and F. Shipman. 2016. Detecting and identifying sign languages through visual features. In *Proceedings of the IEEE International Symposium on Multimedia*. 287–290.

C. Monteiro, F. Shipman, and R. Gutierrez-Osuna. 2018. Comparing visual, textual, and multimodal features for detecting sign language in video sharing sites. In *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval*. To appear.

C. Monteiro, R. Gutierrez-Osuna, and F. Shipman. 2012. Design and evaluation of classifier for identifying sign language videos in video sharing sites. In *Proceedings of the ACM Conference on Computers and Accessibility*. 191–198.

J. C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*. MIT Press, 61–74.

M. Potamias and V. Athitsos. 2008. Nearest-neighbor search methods for handshape recognition. In *Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08)*. 30.

F. Shipman, R. Gutierrez-Osuna, and C. Monteiro. 2014. Identifying sign language videos in video sharing sites. *ACM Trans. Access. Comput.* 5, 4 (2014), 1–14.

F. Shipman, S. Duggina, C. Monteiro, and R. Gutierrez-Osuna. 2017. Speed-accuracy tradeoffs for detecting sign language content in video sharing sites. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'17)*, 185–189.

G. Somers and R. N. Whyte. 2003. Hand posture matching for Irish sign language interpretation. In *Proceedings of the 1st International Symposium on Information and Communication Technologies (ISICT'03)*. 439–444.

T. Starner and A. Pentland. 1995. Real-time American sign language recognition from video using hidden Markov models. In *Proceedings of the International Symposium on Computer Vision*. Springer, 265–270.

P. Toman. 2016. Complexity Beyond the Trigram: Identifying Sign Languages from Video Using Neural Networks, project report. Retrieved from http://cs231n.stanford.edu/reports/2016/pdfs/207_Report.pdf.

P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'01)*. I–511.

C. Vogler and D. Metaxas. 1999. Parallel hidden Markov models for American sign language recognition. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, vol. 1. 116–122.

Z. Zivkovic. 2004. Improved adaptive Gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition*. 28–31.

Received April 2018; revised February 2019; accepted April 2019