Audio-Video Based Character Recognition for Handwritten Mathematical Content in Classroom Videos

Smita Vemulapalli and Monson Hayes

Abstract. Recognizing handwritten mathematical content is a challenging problem, and more so when such content appears in classroom videos. However, given the fact that in such videos the handwritten text and the accompanying audio refer to the same content, a combination of a video and an audio based recognizer has the potential to significantly improve the content recognition accuracy. In this paper, using a combination of video and audio based recognizers, we focus on improving the character recognition accuracy for mathematical content in such videos and propose an end-to-end recognition system - complete with components for video preprocessing, selecting the characters that may benefit for audio-video based combination, establishing correspondence between handwritten and the spoken content and finally combining the corresponding recognition results from audio and video based recognizers. The current implementation of the system makes use of a modified open source text recognizer and a commercially available phonetic word spotter. For evaluation purposes, we use videos recorded in a classroom-like environment and our experiments demonstrate the significant improvements in character recognition accuracy that can be achieved using our techniques.

Keywords: Video Preprocessing, Handwriting Recognition, Speech Recognition, Classifier Combination

1. Introduction

Recent years have witnessed a rapid increase in the number of e-learning and advanced learning initiatives that either use classroom videos as the primary medium of instruction or make them available online for reference by the students. As the volume of such recorded video content increases, it is clear that in order to efficiently navigate through the available classroom videos there is a need for techniques that can help extract, identify and summarize the content that is contained in such videos. In this context, and given the fact that the whiteboard continues to be the preferred and effective medium for teaching complex mathematical and scientific concepts [1, 2], this paper focuses on improving the character recognition accuracy for handwritten mathematical content in classroom videos.

There is a significant body of research devoted to handwritten mathematical content recognition [3-8], and to the extraction and recognition of textual content from video [9-14]. While the research presented in this paper is closely related and dependent on the advances made in these fields, its focus is on the use of an audio based recognizer in combination with a video based recognizer to improve the character recognition accuracy. Specifically, this paper presents a recognition system for audio-video based character recognition for handwritten mathematical content in classroom videos that includes novel techniques for each component in the system.

2. Related Work

The research presented in this paper lies at the intersection of four distinct and well-studied specializations: video preprocessing, handwritten text recogni-
tion, speech recognition and classifier combination. In the following sections we briefly review the relevant literature for each of these specializations. We also review recent work on using a combination of audio and video signals for content recognition.

2.1. Video Preprocessing for Text Extraction

A method for detecting and tracking text in digital video is proposed in [11], which implements a scale-space feature extractor that feeds an artificial neural processor to detect text blocks. Researchers have also proposed methods to partially process video (e.g. detect key frames) for easy retrieval, to acquire data about the movement of the tip of the pen (or pen strokes) on the whiteboard and to augment the whiteboard environment with cameras for tasks such as controlling the computer [9, 10]. A system that automatically produces a set of key frames representing all the written content on the whiteboard before each erasure is described in [12]. A video camera based method to capture annotations made on PowerPoint slides projected on a whiteboard, is presented in [15, 16]. In the context of videos, indexing and retrieval of lecture recordings, e-learning initiatives [17], video-taped presentations [18] and commercial and personal videos are a few of the application domains that are making it ever important to extract and recognize the textual content that is contained in such recordings. An advanced technique for extracting text from faded historic documents that is arranged as a complex pattern, with parallels to mathematical content and out of focus handwritten content, is presented in [19]. The video preprocessing techniques proposed in this paper extend several known video preprocessing and text-extraction techniques (e.g. key frame detection and connected component analysis) and are geared specifically towards classroom video and handwritten mathematical content.

2.2. Handwritten Mathematical Text Recognition

There is a vast collection of literature related to the recognition of handwritten text. Plamondon et al. [20] provide a comprehensive survey of the research in this field. Handwriting recognition could be defined as the ability of a computer to interpret handwritten textual content from input sources such as paper documents [21-24], pen-based inputs that write on an electronic screen [25], videos [26] and other specialized input devices [9]. Mathematical content recognition [4, 27] presents challenges that, in some aspects, are quite distinct from the challenges of recognizing textual content. This is due to the fact that mathematical characters have different sizes and, more importantly, they may be arranged spatially in a complex two-dimensional structure. For mathematical character recognition [7], researchers have addressed issues that arise from factors such as the large number of similar symbols that must be distinctly recognized and the lack of a lexicon that could be used for final validation. The focus of the research presented in this paper is not to develop a better character recognition technique but to utilize existing character recognition techniques in combination with audio recognition tools.

Research has also shown that the recognition accuracy of mathematical content can be improved by combining two or more stages [28]. Prusa et al. [29] have proposed a method that combines the character segmentation and structural analysis stages to achieve better recognition accuracy for mathematical equations. Similarly, a hidden Markov model based method that avoids segmentation during preprocessing by making use of simultaneous segmentation and recognition capabilities is presented in [30, 31]. Awal et al. [32] describe an interesting approach for simultaneous optimization of segmentation, recognition and structure analysis under the restriction of a mathematical expression grammar. In this context, the audio-video based handwritten mathematical character recognition techniques proposed in this paper are both unique and novel.

2.3. Speech Recognition

Speech recognition is the process of converting an acoustic signal captured by a microphone or a similar device into a sequence of words. Over the last few decades speech processing research has made significant advances that have led to the development of a number of commercial speech recognizers including Dragon Naturally Speaking [33], Microsoft Speech [34] and IBM ViaVoice [35]. There have also been some very active research contributions from the academic community in the form of the HTK [36] project from the University of Cambridge and the Sphinx [37] project from CMU. In this paper, we utilize the capabilities of existing speech recognition systems to resolve any ambiguities that exist in the output of a character recognizer. Since our aim is to use the audio content to disambiguate the recognition results from a character recognizer and the fact that extensive language models do not exist for mathe-
maternal content, we have chosen to use Nexidia's word spotting tool [38] which works well for non-standard grammar patterns without any training.

2.4. Classifier Combination

The field of classifier combination has been constantly evolving to address the challenges posed by new application domains. A comprehensive survey of classifier combination techniques is presented in [39], which partitions the classifier combination methods along several distinct dimensions, some of which include the way in which the output of the classifiers is combined, whether the number of classifiers is fixed or if the combination methods use classifiers from a large pool of classifiers, etc. There is also a large body of existing research focusing on generic methods for classifier combination. Lucey et al. [40], for example, have proposed a theoretical framework for independent classifier combination. While some classifier combination methods use another classifier to combine the output of multiple classifiers, others make use of rules and functions to combine the output. Some of the combination techniques proposed in our research are adaptations of well known methods such as weighted combination, Borda count [41], and other decision combination techniques [42]. In the context of handwriting and speech recognition, classifier combination techniques have also been used to improve the recognition accuracy of handwriting recognizers [43-46] and the accuracy of speech recognizers [47].

2.5. Audio-Video Based Content Recognition

Audio and video signals often carry complementary information and often errors in audio recognition may not be accompanied by errors in video recognition for the same character. Therefore, a combination of both information sources can potentially lead to a significant improvement in the recognition accuracy of both the audio and the video signal. Yu et al. [48, 49] propose a classifier combination framework for grammar-guided sentence recognition, and provides the results spoken email command recognition, where an acoustic classifier and a visual classifier (for recognizing lip movement, and tongue and teeth visibility) are combined. The Speech Pen system [50], for classrooms equipped with advanced digital whiteboards, recognizes speech and handwriting in the background and provides the instructor with a list of possible next words that allow the instructor to skip manual writing. A fairly comprehensive collection of research in the field of audio-visual speech recognition is presented in [51]. Anderson et al. [52], in the context of classroom videos that utilize slides and digital ink, present an empirical basis for addressing problems such as automatic generation of full text transcripts for lectures. Their approach relies on matching spoken content with slide content, and recognizing the meaning of the content handwritten by the instructor using digital ink. An investigation of a number of strategies for combining HMM classifiers for improving audio-visual recognition is presented in [53] and a recommendation for using a hybrid of the sum and product rules is made based on empirical, theoretical and heuristic evidence. An interesting approach to using audio-visual inputs for celebrity recognition in unconstrained web-videos is presented in [54]. Humm et al. [55] present another interesting application that combines acquisition of online pen and speech signals for user authentication. It is clear that there is a significant amount of ongoing activity in the field of audio-video based recognition, which can be attributed to the complementary nature of the information contained in the audio and the video streams.

Hunsinger et al. [56] have proposed a multimodal mathematical formula editor that combines speech and handwriting recognition and describes the speech understanding module of the proposed system. In a later publication, Hunsinger et al. [57] present a multimodal probabilistic grammar that incorporates the syntactic-semantic attributes of spoken and handwritten mathematical formulas. A system for speech and handwriting data fusion based isolated mathematical symbol recognition is presented in [58]. Although neither the speech nor the handwritten data originates from video, the approach for combination techniques used for combining the output of the character recognizer and the speech recognizer share commonality with the techniques presented in this dissertation. However, issues such as ambiguity detection and A/V synchronization are not considered in the aforementioned research. Another relevant research effort, closely tied to [58], relates to the creation of a data set [59] with handwritten and spoken mathematical content. Unfortunately, the data set contains static image segments containing handwritten content and the corresponding audio content is stored as separate files. The unavailability of video (used for timestamp information) for the handwritten content make the data set unsuitable for our experiments.
3. System Overview

The end-to-end system for handwritten mathematical content recognition, shown in Figure 1, is organized as a multi-stage system consisting of three separate stages: a video preprocessing stage, an audio-video based character recognition stage, and an audio-video based structure analysis stage.

The video text recognizer (VTR), includes a video preprocessor and a character recognizer. The video preprocessor includes all processing required to extract regions of text within the video that contains handwritten mathematical content, segment the text region into characters, generate a timestamp corresponding to the first appearance of the handwritten characters in the video, and identify the location of the characters in the video frame. The character recognizer then generates a list of one or more characters from a dictionary of characters for each segmented character in the video. The (VTR) also produces a score for each character in the list that represents the recognizer’s belief in the correctness of the character name. Since the characters in this list are based only on the video, they are referred to as video options.

The audio-video based character recognizer is responsible for recognizing the segmented characters generated by the video preprocessor and performing audio-assisted character disambiguation on a subset of the recognized characters. Following the character recognizer is the character disambiguation stage that consists of three distinct components.

1. Ambiguity Detection. For each segmented character, the ambiguity detector determines whether or not character has a high probability of being incorrectly recognized. If so, then the character is classified as ambiguous, and a set of video options are selected and then forwarded to the A/V synchronization component.

2. Audio-Video Synchronization. This component consists of an audio text recognizer and an audio-video synchronizer. For each video option that is generated by the ambiguity detector, the audio text recognizer identifies a set of audio options, which are occurrences in the audio stream of the video option’s character name. Each audio option consists of the audio search term, an audio timestamp, and an audio match score. The audio-video synchronizer then prunes the set of audio options and produces at most one audio option for each video option.

3. Audio-Video Combination. This component combines the output of the video text recognizer and the synchronized output from the audio text recognizer to produce the final recognized character for each of the ambiguous characters.

The audio-video based structure analysis stage uses the set of recognized characters, the location, the timestamps and a mathematical grammar to disambiguate the structure associated with the recognized mathematical content. More details of this stage may be found in [60].

4. Notation

In this section we present the notation that will be used to describe the various components in the A/V character recognition system. First, the dictionary of symbols that are to be recognized will be denoted by $C$, and a specific symbol within the dictionary will be denoted by $c$. It is assumed that the dictionary contains $L$ characters.
Each character that is extracted from the video by the text extraction and segmentation module will be denoted by \( s \), which is the actual image of the character as illustrated in Figure 2. Associated with each character \( s \), is a video timestamp, \( s_t \), that indicates the time at which the character is first detected in the video, and the location, \( s_l \), of the character on the whiteboard. The character \( s \) along with \( s_t \) and \( s_l \), form a feature vector for the segmented character, \( f = [s, s_t, s_l] \).

We define \( G \) to be a function that returns the ground truth for a given segmented character \( s \), i.e.,

\[
G(s) = c
\]

where \( c \) is the character corresponding to \( s \).

When an image of a character is passed to the character recognizer, a set of video options are produced. Each video option consists of a character name from the dictionary \( C \), and a recognition score. For simplicity, we assume that every character in the dictionary is a video option, although many of the scores may be close or equal to zero. Thus, for each character image \( s \), we have a set of \( L \) video options,

\[
V(s) = [v_1(s), v_2(s), \ldots, v_L(s)]
\]

Each video option, \( v_j(s) \), is an ordered pair

\[
v_j(s) = [v_j^c(s), v_j^p(s)]
\]

where \( v_j^c(s) \) is a character from the dictionary \( C \), and where \( v_j^p(s) \) is the video match score generated by the character recognizer. It is assumed that the video options are ordered in terms of their video match score with the first one having the highest score. Figure 2 shows a segmented character \( s \), the feature vector, and the set of video options generated by the video text recognizer.

For each video option \( v_j(s) \), passed to the A/V synchronization component, the audio text recognizer outputs a set of audio options vectors, \( a_{j,k}(s) \),

\[
a_{j,k}(s) = [a_{j,k}^c(s), a_{j,k}^p(s), a_{j,k}^l(s)]
\]

The first element in this vector, \( a_{j,k}^c(s) \), is the audio search term that is the audio equivalent of \( v_j^c(s) \), the second, \( a_{j,k}^p(s) \), is the audio timestamp, and the third, \( a_{j,k}^l(s) \), is the audio match score.

5. Video Text Recognition

In capturing video for video text recognition, a few assumptions are made about the recording setup and recording process. Although these assumptions are not very restrictive, they simplify many of the preprocessing tasks. First, we assume that the entire frame of the whiteboard is within the field of view of the camera. Second, it is assumed that the whiteboard is the largest object in the video frame and that the edge of the whiteboard is significantly darker than the whiteboard. This allows for the region of interest (the whiteboard) to be easily detected. It is also assumed that the beginning of every recording session has at least one calibration frame, which is a video frame with a clean whiteboard without the instructor. In the recording of a lecture, it is assumed that no text is written close (within a few pixels) to the edge of the whiteboard, that the whiteboard is erased completely before the instructor starts to write again, and that the instructor briefly steps away from the whiteboard after a complete erasure so that the entire region of interest is unobstructed. Although partial erasures of the whiteboard should be avoided, corrections are allowed.

5.1. Video Preprocessing

The first stage of the audio/video character recognition system is the video preprocessor that performs a number of tasks including finding the region of interest, extracting the characters on the board, and video time-stamping. Although this stage is important, it involves traditional and well-known image processing algorithms, and it is not one of the contributions of this paper. Therefore, only a brief over-
view of this stage is presented here. Full details may be found in [60].

The first task of the video preprocessor is to identify the region of interest (the whiteboard). The second is to detect the frames of interest in the video, which are unobstructed views of the whiteboard just prior to erasure. In other words, the frame of interest contains all of the characters that are to be detected and recognized. The process of detecting the frame of interest primarily relies on counting the number of connected components contained in video frames over the duration of the video, and selecting the frame with maximum number of connected components.

The next step is character segmentation. Assuming that a given frame of interest is free of any occlusions or shadows from the instructor, we assume that the individual characters appear as one or more distinct connected components, and use connected component analysis algorithm [61]. A post-processing step allows us to handle characters in our dataset that do not appear as a single connected component such as ‘i’ and ‘=’. An example showing the output of this stage is shown in Figure 3. The output of the character segmentation step is a collection of segmented character images along with the corresponding coordinates in the video frame. The final step is to produce a set of video timestamps for each segmented character. These are the times at which each character is written on the whiteboard.

### 5.2. Character Recognition

After a set of segmented characters has been extracted from the frame of interest, they are forwarded to a character recognizer. In our system, we used a character recognizer that is based on the optical character recognition tool called GNU Optical Character Recognition or GOCR [62]. However, we made two modifications to the existing GOCR tool. The first is to return a set of candidate characters rather than a single recognized character or no match at all. The second is to return a match score that is based on the number and the relative significance of recognition rules that have been satisfied. The set of candidate characters and their match scores are the video options for a given character that has been written on the board. The current engine recognizes alphabets (both capital and small), numbers and basic arithmetic operators. It should be pointed out that our sys-
tem is not tied to GOCR. Any character recognizer that returns a set of character options along with a match score may be used.

6. Ambiguity Detection and Option Selection

The output of the video text recognizer is a set of video options for each character with each video option consisting of a character from the dictionary $C$ and a character matching score. The next step is to decide whether or not the top-ranked video option is likely to be correct. Thus, the ambiguity detection and option selection module determines whether or not the character with the highest score is likely to be correct (ambiguity detection) and, if not, determines which set of video options (option selection) should be sent to the audio text recognizer to resolve the ambiguity.

The need for ambiguity detection is motivated by the following observation. With a two-stage recognition system where a video text recognizer produces a set of options for a given character and each of these options are forwarded to an audio text recognizer, when the top choice of the video text recognizer is correct, then the audio text recognizer may find an utterance in the audio that makes another character more likely. Similarly, if an incorrectly recognized character is not forwarded to the audio text recognizer, then there is no possibility for the audio text recognizer to correct the error. Therefore, it is important to determine which of the segmented characters have a sufficiently high probability of being incorrectly recognized, and tagging these as ambiguous. Thus, only the ambiguous characters are forwarded to the audio text recognizer.

In addition to deciding which characters are ambiguous, it is also necessary to specify the set of video options that will be forwarded to the audio text recognizer. The reason why option selection is needed is motivated by the observation that the smaller the number of options that are forwarded the smaller the probability that the correct character is included in the set, and the larger the number of options the higher the probability that the character will be incorrectly classified as a result of combination.

6.1. Score Mapping

Video match scores that are generated by a video text recognizer are often not the best metric to use to determine whether or not the output of the text recognizer should be forwarded to an audio recognizer. The reason is that these scores are not designed for the purpose of performing ambiguity detection in an audio-video text recognition system. Therefore, it is necessary to map (rescore) these values to a new set of scores with the goal of better detection of ambiguous characters, better selection of which video options to forward to the audio text recognizer, and improved overall character recognition accuracy. Some commonly used score normalization techniques are discussed in [63]. The approach that we have taken is that, for each character $s$, the video match score is replaced with the conditional probability that the character is correctly classified, $v_C^v(s) = G(s)$, given that the video match score is equal to $v^v(s)$.

$$\text{Prob} \{ v_C^v(s) = G(s) | v^v(s) \} = \frac{\text{Prob} \{ v_C^v(s) = G(s) \}}{\text{Prob} \{ v^v(s) \}}$$

Estimating these conditional probabilities is done using a training set $S_{\text{train}}$ that consists of a set of segmented characters, $s$, along with their ground truth, $G(s)$. This conditional probability, which will be used as the new video match score, is denoted by $v_C^v(s)$ and estimated follows.

$$\text{Number of times } v_C^v(s) \text{ is correctly classified with a score } v^v(s)$$
$$\text{Number of times } v_C^v(s) \text{ has a score } v^v(s)$$

After mapping, the video match scores for each video option change, and need to be reordered so that they are in decreasing order. Although mapping is optional, as we will see in Section 9, the use of a suitable mapping technique may lead to significant improvements in the recognition rate. In some cases, such as in the case of a limited training set, it may be advisable to divide the entire range of match scores into sub-ranges and then compute the value of the conditional probability given that $v_C^v(s)$ falls within a certain range of values instead of a specific value.

6.2. Thresholding

We now present a number of thresholds that may be used to determine whether or not a segmented character should be classified as ambiguous. These thresholds may be divided into two categories: simple and character-specific thresholds. The ambiguity detector will be represented by a function, $D(s)$, that places the character $s$ into the set of ambiguous characters if the video text score for $s$ does not satisfy a given threshold criterion. For a set of characters $S$ within a frame of interest, the set of ambiguous characters will be denoted by $S_D = D(S)$. 

Simple Thresholds. Simple thresholds are those that are applied uniformly to all characters. The first thresholding technique for ambiguity detection to classify a character \(s\) as ambiguous if the video match score of the first video option, \(v_1^p(s)\), is less than a fixed threshold \(T_A\).

\[
D(S) = \{ s \in S | v_1^p(s) < T_A \}
\]

Another test to use for ambiguity detection is to compare the ratio of the first and second video scores to some threshold \(T_{R^c}\). If this ratio is larger than \(T_{R^c}\) then the character is classified as ambiguous. Thus,

\[
D(S) = \{ s \in S | \frac{v_1^p(s)}{v_1^q(s)} > T_{R^c} \}
\]

This method is based on the premise that the top video option, if correct, will have a score that is significantly larger than the score for the second best video option.

Character-Specific Thresholds. Since some characters are more difficult to recognize than others, and since a given character recognizer may have different recognition rates for different characters, having the same threshold for all characters is generally not the best approach to use. Therefore, another approach is to use a different threshold, \(T(c)\), for each character \(c\) in the dictionary. An ambiguity detector with a character-specific threshold is given by

\[
D(S) = \{ s \in S | v_1^p(s) < T(c) \}
\]

where \(T(v_1^p(s))\) is a character-specific threshold for character \(v_1^p(s)\).

To find a set of values for the character-specific thresholds, for each character \(c\) in the dictionary, we use a training set \(S(c)\). The goal is then to find a character-specific threshold \(T(c)\) that maximizes the audio-video recognition accuracy for the training set \(S(c)\). If we assume that only the top \(N\) video options are forwarded to the audio text recognizer then, in order to evaluate the recognition accuracy, the training set \(S(c)\) is partitioned into three sets. The first, \(S_1(c)\), is the set of all elements in \(S(c)\) whose first video option is correct. The second, \(S_N(c)\), is the set of all elements that have the correct video option within the first \(N\) video options but is not the first video option. Finally, the third set, \(S_\infty(c)\), is the set of all elements that do not have the correct video option in the first \(N\) options.

Ideally, the characters in \(S_1(c)\) will be classified as non-ambiguous and will not be forwarded to the audio text recognizer. Since the decision whether or not to forward a character depends on the threshold that is used, we define four disjoint subsets of \(S(c)\) as follows. The first is the set of True Positives, \(T_P(c)\), that contains those characters that are correctly recognized by the video text recognizer and are tagged as non-ambiguous. In other words, these characters exceed a given character-specific threshold and are not passed to the audio text recognizer. This set of characters has a recognition accuracy of 100%.

The next two subsets are those that contain characters that do not exceed the character-specific threshold and are therefore classified as ambiguous and forwarded to the audio text recognizer. The first is the set of False Negatives, \(F_N(c)\), which consists of those characters that are correctly recognized by the video text recognizer but are tagged as ambiguous. The second is the set of True Negatives \(T_N(c)\), which are those characters that are not correctly recognized by the video text recognizer but are tagged as ambiguous and have the correct character within the first \(N\) options. It is assumed that when the audio-video combination component is given a set of \(N\) video options with one of the options being correct, the recognition accuracy is equal to \(\alpha_{F_N}\). This value may be estimated from the training data set.

The fourth subset is the set False Positives, \(F_P(c)\), which is the set of all characters that are not correctly recognized, have the correct option within the first \(N\) options, and have been tagged as non-ambiguous. This subset is not passed to the audio text recognizer and will have a recognition accuracy of 0%.

Therefore, it follows that the overall recognition accuracy is given by

\[
\alpha(S(c), T) = \frac{|T_P(c)| + \alpha_{F_N}(|F_N(c)| \cup T_N(c))}{|S_1| + |S_N|}
\]

where \(|A|\) is used to represent the number of elements in the set \(A\). The value for character-specific thresholds \(T(c)\) can now be calculated by finding the value of \(T(c)\) that maximizes \(\alpha(S(c), T)\).

6.3. Option Selection

An important factor that impacts the audio-video character recognition accuracy is the number of video options that are passed to the A/V combination unit. Although forwarding all video options increases the chance of having the correct output among them, when the correct output is the top-rated video option within a short list of video options, each additional option that is passed increases the chances of an error in recognition. On the other hand, if the number of
options passed for combination is too low, then there is an increased chance that the correct video option is not included within the set. The goal of option selection is to choose the video options in such a way that the probability of having the correct video option in the list of options is maximized while, at the same time, minimizing the number of video options that are included in the list of forwarded options.

We have considered three different option selection strategies. The first is to select the top $N$ video options,

$$O(s) = \{v_1(s), v_2(s), \ldots, v_N(s)\}$$

The second is to select those video options that have a video match score that exceeds a threshold $T$,

$$O(s) = \{v_i(s) \in V(s) | v_i^p(s) > T\}$$

The third is to select the video options that have a video match score that exceeds some fraction, $\beta$, of the top video score,

$$O(s) = \{v_i(s) \in V(s) | v_i^p(s) > \beta v_1^p(s)\}$$

The performance of each of these video option selection strategies are discussed in Section 9.

7. Audio-Video Synchronization

The ability to correctly synchronize the output from the video and the audio text recognizers is critical to the accuracy of the end-to-end audio-video based recognition system. However, due to issues such as shadows that affect the quality of video timestamping, occlusions that are caused by the instructor, the skew between the writing and the utterance of a character, and errors in the output of the video and the audio text recognizers, establishing the correspondence between the handwritten content and the spoken content is a difficult problem. In fact, techniques that work well for one scenario may fail for another.

We have developed a suite of audio-video synchronization techniques that range from simple techniques that are geared for a specific scenario, to more general techniques that perform well under a range of scenarios. We assume that everything that is handwritten is spoken, and focus on the issues that arise due to occlusions, shadows, the skew between the writing and the utterance of a character, and errors in the audio and video recognizers. We begin by defining the various timestamps that are associated with a character, present the features that are used in the various synchronization methods, and describe a method that is used for initial pruning of the audio options. This is followed by a description of the synchronization techniques.

7.1. Timestamps

There are four timestamps associated with each character $s$ that is extracted from the video. The first is $TS_v^o(s)$, which is the timestamp that is generated automatically by the video time-stamping algorithm, and the second is $TS_v^m(s)$, which is the timestamp that is determined manually as part of the labeling process for evaluation, and corresponds to the time when the segmented character first becomes visible in the video. Similarly, we define $TS_a^o(a_{j,k}(s))$ to be the timestamp that is generated automatically by the audio text recognizer for the audio option $a_{j,k}(s)$, and we let $TS_a^m(s)$ to be the timestamp of the audio option that is manually selected to be the correct one, i.e., the one corresponding to the utterance of the character. The absolute difference between the automatic and manual video timestamps,

$$\delta T(s) = |TS_v^o(s) - TS_v^m(s)|$$

is a measure of the video time-stamping accuracy. Similarly, the absolute difference between the manually labeled video and audio timestamps,

$$\delta R(s) = |TS_v^m(s) - TS_a^m(s)|$$

which is a measure of the recording alignment between the written and spoken content.

7.2. Audio Features

For each audio option, $a_{j,k}(s)$, of a given video option $v_j(s)$, four audio features are computed. The first is the audio match score that is generated by the audio text recognizer,

$$F_1(a_{j,k}(s)) = o_{j,k}^a(s).$$

The second is the difference between the audio and video timestamps,

$$F_2(a_{j,k}(s)) = |TS_v^o(s) - TS_a^o(a_{j,k}(s))|$$

The third and fourth features are related to the number of audio options corresponding to neighboring video options that are found within a given window of time around $a_{j,k}(s)$. More specifically, let $[w_B, w_A]$ be a time window around audio option $a_{j,k}(s)$, and let $n_l$ and $n_A$ be integers. Feature $F_3(a_{i,k}(s))$ is then defined to be the number of audio
options that are found within the time window $[w_B, w_A]$ that corresponds to $n_B$ video options before and $n_A$ video options after video option $v_j(s)$. For example, suppose that the equation

$$a + b = c$$

is written on the board, and let $a_{i,k}(b)$ be an audio option for the character ‘b’. If

$$[w_B, w_A] = [2, 2]$$

and $n_A = n_B = 1$, then $F_3 = 0$ if no audio options for any of the video options associated with the characters ‘+’ and ‘=’ are found within +2 seconds of audio option $a_{i,k}(b)$. Similarly, $F_3 = 1$ if there is a video option for either ‘+’ or ‘=’ that has an audio option within the given time window, and $F_3 = 2$ if audio options are found within the given time window for a video option for both characters.

The fourth feature differs from $F_3$ in that instead of counting how many video options are found before and after $v_j(s)$ within a given time window around $a_{i,k}(s)$, here the window is expressed in terms of some specified number of segmented characters before and after a given video option $v_j(s)$. More specifically, the window is expressed as $[n_B/t_B, n_A/t_A]$ and involves the selection of $n_B$ out of $t_B$ neighbors before the given video option $v_j(s)$, and $n_A$ out of $t_A$ neighbors after $v_j(s)$. For example, if

$$[n_B/t_B, n_A/t_A] = [2/3, 2/3]$$

then two out of three neighbors are selected before the video option and two out of three after it. The value of the feature $F_4$ depends on which neighbors are selected.

### 7.3. Initial Pruning

The first step in each A/V synchronization technique is to prune the audio options that are unlikely to be the final synchronized audio option. There are two pruning used, and they are based on $F_1$ and $F_2$. The first is to remove any audio option for which $F_1$ is less than some threshold, $P_{th}$. Thus, any audio option that has an audio match score that is less than $P_{th}$ will be pruned. The second is to remove any audio option that has a value for $F_2$ that falls outside a given window, $[P_{th}, P_{th}]$. Since $F_5$ is the difference between the audio and video timestamps, then an audio option will be pruned if the difference between these two times falls outside a given range of values. For example, a pruning window $[P_{th}, P_{th}] = [-8, 8]$ will remove all audio options that are more than 8 seconds after or 6 seconds before the video option.

We now present a number of approaches that may be used for audio-video synchronization.

### 7.4. Time-Difference Based Synchronization

For each video option $v_j(s)$, the time-difference based approach finds the audio option $a_{i,k}(s)$ in the set of pruned audio options that is the closest in time to $v_j(s)$. Thus, the selected audio option is the one with the smallest value of $F_2$, and the index $k_0$, is

$$k_0 = \arg\min_k F_2(a_{i,k}(s))$$

### 7.5. Neighbor-Based Synchronization

With neighbor-based synchronization, the audio option that is selected, $a_{i,k_0}(s)$, is the one that has the largest number of video neighbors of $v_j(s)$ that are spoken within a time window centered around $a_{i,k_0}(s)$. Thus, from the set of pruned options, the audio option with the largest value of $F_3$ is selected,

$$k_0 = \arg\max_k F_3(a_{i,k}(s))$$

When there are two or more audio options that maximize $F_3$, the audio option that is selected is the one that has the smallest value of $F_3$.

Selective Neighbor Based Synchronization

In the neighbor-based synchronization technique, the top video option of the neighbors is assumed to be the correct character, and the audio search term for this option is used when searching for neighbors and in finding the value of feature $F_3$. However, since the character recognition accuracy may be relatively low, many of these neighbors will be incorrectly recognized and lead to poor synchronization. Therefore, it is preferable to select neighbors that have a higher probability of being correct. Thus, from the set of pruned options, the selective neighbor based technique selects the audio option that has the highest value of $F_3$, and the index, $k_{th}$, for the selected audio option is given by

$$k_0 = \arg\max_k F_4(a_{i,k}(s))$$

The selection of the neighbors may be either video-based or audio-based as described below.
Video-Based Neighbor Selection

Video-based neighbor selection focuses on selecting the neighbors that have a highest chance of having been correctly recognized by the video text recognizer. Since ambiguity detection techniques are designed to detect those characters that have a higher chance of being incorrectly recognized by the video text recognizer, the same approach is used to select neighbors.

One approach is to select \( n_R \) out of the \( t_R \) neighbors before the character under consideration that have the largest video match score and, similarly, \( n_A \) out of the \( t_A \) neighbors after the character that have the largest video match score. Other approaches would be to use relative or character-specific thresholding rather than absolute thresholding to select the neighbors.

Audio-Based Neighbor Selection

Audio based neighbor selection focuses on selecting neighbors from among the \( n_R \) neighbors that result in a lesser number of false positives when we search for their corresponding audio search term in the audio component. This is due to the fact that those audio search terms that have too many false positives may not be very reliable neighbors as they may be found within the neighbor audio time window \([w_B, w_A] \) of several audio options even when they were not actually spoken.

Audio search terms with larger number of phonemes lead to lesser false positives and therefore result in higher audio text recognizer accuracy. We use this observation to perform audio based neighbor selection. For the number of neighbors parameter \([n_B/t_R, n_A/t_A] \), from among \( t_R \) neighbors before the character under consideration, we select \( n_R \) neighbors that have a higher number of phonemes in the audio search term for their top video option and \( n_A \) neighbors from \( t_A \) neighbors after the character.

7.6. Feature Rank Sum Based Synchronization

We assign four feature ranks \( R_i(a_{j,k}(s)) \) for \( i = 1, 2, 3, 4 \) to each audio option \( a_{j,k}(s) \) based on the ranks of the audio features \( F_1, F_2, \) and \( F_3 \). For example, if audio option \( a_{j,k}(s) \) has the \( k \)th largest value of feature \( F_i \) among all audio options, then the rank \( R_i \) for this audio option will be equal to \( k \). Similarly, if audio option \( a_{j,k}(s) \) has the \( k \)th smallest value of feature \( F_i \), then the rank \( R_i \) will be equal to \( k \).

Finally, if \( a_{j,k}(s) \) has the \( k \)th largest value of feature \( F_3 \), then the rank \( R_3 \) will be equal to \( k \). The audio option with the minimum value of the rank sum, \( R_1 + R_2 + R_3 \) is then selected as the output. Thus, the index \( k_0 \) for the chosen audio option is given by

\[
k_0 = \arg \min_k \sum_{i=1}^{3} R_i(a_{j,k}(s))
\]

with \( R_2 \) and \( R_3 \) being used in the case of a tie. Another implementation of the feature rank sum based synchronization technique uses the neighbor selection based audio feature \( F_4 \) instead of \( F_3 \).

8. Audio-Video Combination

There are many techniques that can be used to combine the outputs from multiple recognizers to improve the overall recognition accuracy. These techniques include weighted combinations, rank based combinations, and methods that rely on sophisticated machine learning algorithms. Such combination methods provide an opportunity to incorporate various recognizer, character and instructor specific nuances, learned or otherwise, into the end-to-end recognition system. In this section, we describe several audio-video combination methods that could be used by the proposed recognition system.

The input to the audio-video combination stage is a set of segmented characters that have been designated as ambiguous, with each one having a set of video options \( v_j(s) \) along with a single audio option for each video option. The audio-video combination stage generates the final recognized output based on the match scores generated by the two recognizers along with some audio-video features. The remainder of this section describes the specific audio-video combination techniques that we have considered.

8.1. Rank Based Techniques

One approach that we have used for AV combination is the simple rank sum (Borda count [39]), accompanied by a suitable tie-breaking strategy. One of the advantages of using a rank based technique is that there is no need to compute weights for the audio and video components or normalize the scores generated by different recognizers. We have implemented rank sum for different combinations of the audio and the video recognizer scores and the features synchronization features \( F_1, F_2, F_3, \) and \( F_4 \).
Recognizer-Specific Weight Based Techniques

Another way to combine the audio and video based recognition scores is to simply form the sum [39]. However, given the fact that the video and the audio recognizers may have different performance in terms of accuracy it is necessary to take this into account when using the sum rule. Therefore, we use recognizer-specific weights, $w_V$ and $w_A$, for the video and the audio recognizers, respectively, to incorporate performance, into the sum rule based score.

8.2. Character-Specific Weight Based Techniques

As the accuracy of the audio and video text recognizers may be significantly different for each character, using different weights for each character has the potential to further improve the recognition accuracy. One way to assign a weight $w_V(v_f^j(s))$ to the $j$th video option for the character $s$ is to use the video text recognizer’s accuracy-related metrics, such as precision and sensitivity for the character label $v_f^j(s)$. The value of the audio weight $w_A(v_f^j(s))$ can either be computed in a similar fashion with the weights being normalized so that they sum to one, or we can set $w_A(v_f^j(s)) = 1 - w_V(v_f^j(s))$.

8.3. Recognizer Ensemble Based Techniques

This is a two-level combination technique. First, we create an ensemble of recognizers i.e. the first stage of combination, by configuring multiple instances of the audio different audio and video combination weights, different subsets of the audio feature set and different combination techniques. Next, we perform the second level of combination i.e. combining the outputs of the ensemble of recognizers. A number of combination techniques such as majority voting, rank sum and character-specific recognizer selection may be employed. We have used rank sum as the second level combination technique and the results are shown in Section 9.4. To select the first-level recognizers, we use various permutations of recognizers and prune away those recognizers that show no potential to improve the final accuracy when evaluated on a training set.

9. Experiments

This section presents the results of some of the experiments that were conducted to evaluate the various techniques proposed in this paper. Since the mapping of video matching scores to conditional probabilities results in consistently better audio/video character recognition, score mapping is used in each of the experimental results presented here. A more complete discussion of recognition rates with and without mapping may be found in [60].

9.1. Setup: Data Set & Implementation

The recording equipment used for capturing the videos consisted of a commercially available off-the-shelf video camera (Sanyo VPC-HD1A 720p High-Definition Digital Media Camera) and a wired microphone. The videos were recorded in a classroom-like setting with mathematical content being written on the whiteboard and being spoken by the instructor. The camera was configured to capture videos with a resolution of 1280 x 720 pixels at 30 frames per second. The main data set is organized into two sets, one for training and one for evaluation. The entire data set has about 6,600 characters, 3,000 of which are in the training data set and 3,600 in the test data set. The videos were recorded with good recording alignment with most of the characters having a misadjustment, $\delta R(s)$, of less than two seconds. To evaluate the performance of the synchronization techniques for videos with poor recording alignment, we also recorded a set of videos with poor recording alignment where $\delta R(s) > 4$ seconds. This data set contains about 500 characters. Sample data sets are available online [64], along with the instructions for obtaining the complete data set.

The recognition system described in the paper has been implemented in C/C++ and makes use of open source libraries like OpenCV [65] for video preprocessing and character segmentation. For the handwritten character recognizer, we use the open source tool GNU Optical Character Recognizer (GOCR) [62] that was modified so that a list of options for each handwritten character is generated along with a video match score. For the audio recognizer we used Nexidia [38], a commercially available phonetic word spotter. Nexidia’s libraries were used to locate all occurrences of a given word along with its audio match scores within a given video segment.

9.2. Baseline System

We established two set of baseline recognition accuracies based on the use of mapping technique. For the first set, the score mapping technique was not
used. In this set, the first baseline accuracy is for the scenario when all segmented characters in the test data set were classified as non-ambiguous. The recognition accuracy in this case is equivalent to the accuracy of the video text recognizer and was found to be 53.7%. The second baseline accuracy for the above set corresponds to the case when all characters are classified as ambiguous and undergo audio-video based combination (AVC) using rank sum based synchronization and using recognizer-specific weights based technique with $w_V=0.6$, $w_A=0.4$. The character recognition accuracy in this case is the accuracy of the audio-video based character recognizer without thresholding or option selection and is equal to 50.0%. With mapping, the second set of corresponding set of baseline accuracies were found to be 62.2% and 61.4%. For this set of baseline value, the recognizer-specific weights based technique was configured with $(w_V=0.8, w_A=0.2)$. For the two set of baselines, the lower recognition rate for the AVC based recognizer illustrates the importance of thresholding and option selection.

9.3 Thresholding and Option Selection

Extensive tests have demonstrated that the recognition rate increases when a threshold is used to determine whether or not a character is ambiguous [60]. One set of experiments, for example, showed that the recognition accuracy is 67.6% (compared to 62.2% for the baseline system with mapping) when an absolute threshold value of $T_A=0.98$ is used, and is 65.9% when a relative threshold value of $T_R=0.97$ is used. What was also observed is that as the absolute threshold $T_A$ is increased there is an increase in the recognition accuracy of both the ambiguous and the non-ambiguous characters. This is due to the fact that when the absolute threshold is increased, the number of non-ambiguous characters decreases, but those characters that remain classified as ambiguous have a high video match score for the first video option and, therefore, have a high probability of being correct. In fact, as $T_A$ approaches one the recognition rate for the non-ambiguous characters approaches 90.0%. The other hand, as the value of $T_A$ is increased the number of ambiguous characters will increase, and these additional characters have a higher chance of being correctly recognized.

Since the character-specific thresholds $T_C$ depend on $\alpha_N^C$, the character recognition accuracy of the audio-video combination when there are $N$ video options with one of them being correct, the end-to-end character recognition accuracy was found to be the highest when $\alpha_N^C$ was between 40.0% and 50.0%. Interestingly, it was found that the maximum value achieved using character-specific thresholds was 62.2%, which is the same as one of the baselines.

Extensive testing has also shown that option selection without thresholding results in small but important gains in the recognition accuracy. For example, the recognition accuracy is 61.4% when all options are forwarded, it increases slightly to 61.6% when $N=3$ options are selected. When the video options must have a video match score that exceeds a threshold of $T=0.8$ the accuracy is 62.0% and when the video option must have a match score that exceeds $\beta=0.8$ times the top score the accuracy is 64.0%.

Table 1 shows a consolidated version of the baseline recognition rates as well as the improvements in the end-to-end character recognition rate that can be achieved with thresholding and option selection. The accuracy of the VTR with mapping but without thresholding and option selection (row 1) serves as the primary baseline. The improvements due to thresholding can be seen by comparing rows 3 and 6 to row 1. The gains that are achieved with option selection are seen by comparing rows 4 and 7 when compared to row 1.

It is evident from each of the sets of experiments that using a suitable combination of thresholding and option selection prove to be much better than using them separately. We also observe that the best value for end-to-end character recognition accuracy occurs for setups that make use of a suitable combination of thresholding and option selection techniques such as those in rows 5 and 8.

9.4 A/V Synchronization

To evaluate the A/V synchronization accuracy, we use a measure $\alpha_N^C(S_C)$ that is defined as follows. Let $v_a(s)$ be the correct video option for a given charac-

<table>
<thead>
<tr>
<th>#</th>
<th>Thresholding</th>
<th>Opt. Selection</th>
<th>$\alpha(N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>VTR only</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>All Options</td>
<td>61.4%</td>
</tr>
<tr>
<td>3</td>
<td>$T_A=0.95$</td>
<td>All Options</td>
<td>63.8%</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>$T=0.80$</td>
<td>62.0%</td>
</tr>
<tr>
<td>5</td>
<td>$T_A=0.95$</td>
<td>$T=0.80$</td>
<td>64.7%</td>
</tr>
<tr>
<td>6</td>
<td>$T_R=0.98$</td>
<td>All Options</td>
<td>64.5%</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>$\beta=0.80$</td>
<td>64.0%</td>
</tr>
<tr>
<td>8</td>
<td>$T_R=0.98$</td>
<td>$\beta=0.80$</td>
<td>65.9%</td>
</tr>
</tbody>
</table>

Consolidated results showing the baselines and the improvement due to thresholding and option selection when mapping is used.

Synchronization: Rank Sum, Combination: Weighted ($w_V=0.8, w_A=0.2$)
The synchronization accuracy is limited by the time-stamping errors and occlusions, which in our case is 30 seconds. When the time-stamping (TS) and recording alignment (RA) is good, a pruning window equal to \( \Delta T_s = 1 \) seconds was sufficient. However, for poor TS and RA, the best synchronization results are obtained for a pruning window \([P_B, P_A] = [-32,14]\). The table also shows that using a smaller than required pruning time window \([P_B, P_A]\) may degrade the synchronization accuracy as the correct audio option may not be present within this pruning time window.

Table 2

<table>
<thead>
<tr>
<th>#</th>
<th>Parameter Values</th>
<th>Synchronization Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>([P_B, P_A])</td>
<td>([n_B, n_A])</td>
</tr>
<tr>
<td>1</td>
<td>([-1,1])</td>
<td>-</td>
</tr>
<tr>
<td>1b</td>
<td>([-6,8])</td>
<td>-</td>
</tr>
<tr>
<td>1c</td>
<td>([-6,16])</td>
<td>-</td>
</tr>
<tr>
<td>1d</td>
<td>((-32,14])</td>
<td>-</td>
</tr>
</tbody>
</table>

**Pruning Time Window.** The pruning time window \([P_B, P_A]\) limits the set of audio options that are considered as possible candidates for A/V synchronization. In the test data sets, the difference in time between the appearance of the handwritten character and the corresponding audio utterance was less than 30 seconds, and the time difference due to video time-stamping errors and occlusions was less than 10 seconds. When the time-stamping (TS) and recording alignment (RA) is good, a pruning window equal to \([P_B, P_A] = [-6,8]\) was sufficient. However, for poor TS \(P_A\) needs to be increased to allow for a maximum time difference of 10 seconds, whereas for poor RA \(P_B\) should be increased to the maximum time difference, which in our case is 30 seconds.

The best synchronization results are obtained when the pruning time window \([P_B, P_A]\) is designed for the specific category. For example, the A/V synchronization accuracy for characters with good TS and good RA was maximum (85.1%) for the pruning time window \([P_B, P_A] = [-6,8]\) and any further increase does not result in an increase in the synchronization accuracy. For poor TS and good RA, \([P_B, P_A] = [-6,16]\) resulted in the maximum synchronization accuracy and there was no further increase when the pruning time window was further expanded (row 1d). Finally, for good TS and poor RA, the best synchronization results are obtained for a pruning window \([P_B, P_A] = [-32,14]\). As expected, for \([w_B, w_A]\) too small, audio occlusions may degrade the synchronization accuracy.

**Number of Neighbors.** The effect of changing the number of neighbors \([n_B, n_A]\) that are used for synchronization is shown in rows 2c-e. Overall, we found that \([n_B, n_A] = [2,2]\) gave the best synchronization results, and that when the number of neighbors was reduced to \([1,1]\), the synchronization accuracy decreased for poor TS and poor RA but increased when the TS and RA was good.

**Neighbor Audio Time Window.** The neighbor audio time window \([w_B, w_A]\) should be selected based on the number of neighbors \([n_B, n_A]\) that are considered for synchronization, and the average time between the utterances of consecutive characters. Although not shown in the table, we have observed both experimentally and also by inspecting the test data that for \([n_B, n_A] = [2,2]\), a neighbor audio time window of \([w_B, w_A] = [-8,8]\) seconds results in the best synchronization accuracy. As expected, if \([w_B, w_A]\) is too small, then the neighbors may lie outside the window whereas if \([w_B, w_A]\) is too large, audio occurrences (false positives) that do not correspond to the neighbor may appear in the window. In both of these cases, an incorrect value for the feature \(F_k\) will be obtained, which results in a degradation of the performance of any neighbor based technique. For the remainder of the experiments, we use a neighbor audio time window of \([w_B, w_A] = [-8,8]\) when \([n_B, n_A] = [2,2]\).

**Time Difference Based Technique.** As expected, for the time difference based technique (row 1b) the maximum synchronization accuracy was achieved for good TS and good RA. Since the time difference based technique selects the audio option based entirely on the time difference between the video option...
and the candidate audio option, the synchronization accuracy was significantly lower for characters with poor TS. The synchronization accuracy degraded even further for characters with poor RA even with a longer pruning time window (row 1d).

Neighbor Based Technique. Using neighbors for synchronization helps to improve the synchronization accuracy for characters with poor TS or poor RA. From the set of experiments shown in Table 2, we see that neighbor based synchronization (row 2b) results in an improvement in the synchronization accuracy compared to the time difference based technique (row 1c) for the same pruning time window when the TS is poor. Similarly, when the RA is poor, we see a significant increase in the synchronization accuracy (row 2d) with the A/V neighbor based synchronization compared to the time difference based method (row 1d). However, using neighbor based synchronization, the synchronization accuracy deteriorates when the TS and RA is good (row 2a) compared to the time difference based technique (row 1b) since the neighbors may not be very reliable. The results also demonstrate that increasing the \([P_B, P_A]\) further to a value of \([-50, 30]\) (row 2f) could degrade the synchronization accuracy. Increasing the pruning time window beyond the required size leads to more audio options being considered as candidates and due to factors like low video text recognizer and low audio text recognizer accuracy this may lead to an incorrect audio option being assigned a higher value for feature \(F_3\). This impacts the accuracy of synchronization techniques that make use of feature \(F_3\). The maximum A/V synchronization accuracy achieved by the A/V neighbor based technique for poor TS and good RA was 50.5% (row 2b) and for good TS and poor RA it was 47.6% (row 2d).

Feature Rank Sum Based Technique. Feature rank sum combines the advantages of the time difference based technique and the neighbor based technique, and incorporates the audio match score as a feature in the synchronization process. Experiments show that the feature rank sum based technique is a good compromise between the time difference based and the neighbor based techniques. For good TS and RA, this technique performed better (row 3a) than the time difference technique (row 1b), due to the advantage of using the audio match score values (audio feature \(F_3\)) and did not degrade as a result of using the neighbors since the small pruning time window did not allow many other audio options with a high number for audio feature \(F_3\). For poor TS and RA, the maximum synchronization accuracies achieved using feature rank sum (row 3b for poor TS and row 3c for poor RA) were higher than those of the time difference technique (rows 1c-d) but lower than those of the neighbor based technique due to the effect of A/V time difference audio feature \(F_2\). The feature rank sum technique that worked reasonably well for all three categories of TS and RA is shown in row 3c and the synchronization accuracies for these categories were 82.4%, 45.1% and 41.9%.

One should note that in a given test data set, there may be characters that fall into each of the three categories based on the A/V recording alignment of the videos as well as the quality of the time-stamping algorithm. The ratio of characters that fall into each of these three categories determine the choice of A/V synchronization technique. The feature rank sum based technique, as shown above, is a reasonable choice for most of the test data sets.

Selective Neighbor Based A/V Synchronization. Many experiments were conducted to examine the effect of neighbor selection on the neighbor based and the feature rank sum based synchronization techniques, and these are summarized in Table 3. For various categories of characters, based on TS and RA, we observed that the use of neighbor selection improves the synchronization accuracy of both the neighbor based technique and the feature rank sum based technique. We also observed that video based neighbor selection outperforms audio based neighbor selection for both the neighbor based technique and the feature rank sum based technique. For a given test data set, if the majority of the characters have either poor TS or poor RA, then the neighbor based technique with video based neighbor selection (row 2b) would be the synchronization technique of choice. If the majority of the characters have good TS and

| Table 3: Effect of neighbor selection on A/V synchronization accuracy |
|------------------|-------------------|-------------------|-------------------|
| | Synchronization Technique | Good TS, RA | Good RA | Good TS |
| 1 | A/V Time Difference | 85.1% | 36.1% | 25.9% |
| 2a | A/V Neighbor | 78.4% | 48.3% | 47.6% |
| 2b | Audio-Based Neighbor Selection | 83.0% | 56.4% | 54.7% |
| 2c | Neighbor Based Selection | 80.3% | 51.2% | 50.6% |
| 3a | Feature Rank Sum (Audio Based Neighbors) | 82.4% | 45.1% | 41.9% |
| 3b | Feature Rank Sum (Video Based Neighbor) | 87.0% | 50.9% | 47.8% |
| 3c | Feature Rank Sum (Audio Based Neighbor) | 84.9% | 48.3% | 41.1% |

Good TS: \(\delta s \leq 2\) seconds, Poor TS: \(\delta s > 2\) seconds, Good RA: \(\delta R \leq 4\) seconds, Poor RA: \(\delta R > 4\) seconds.
of non-ambiguous characters is 79.8% and 39.2% for the ambiguous characters. Since A/V combination techniques only process ambiguous characters, the recognition accuracy for these characters should increase.

**Rank Based Techniques.** The rank sum based techniques for A/V combination in Table 4 are based on different subsets of audio features \(F_1, F_2, F_3\) and the video text recognizer match score, \(V\). What we found was that the rank sum that is based on \(V\) and \(F_1\) has the highest end-to-end system accuracy of 60.7%. The fact that the other audio features do not significantly impact the combination stage indicates that the test set is reasonably well aligned.

**Recognizer-Specific Weight Based Techniques.** Table 4 shows the results for various combination of recognizer weights, \([w_V, w_A]\). As shown, in the table, the best values for the weights is close to \([0.8, 0.2]\), which gives an overall recognition accuracy of 65.1%. It is interesting to see how the four ratios of ambiguous characters that are correctly/incorrectly recognized by the video text recognizer and the overall recognition rate changes as the weights are varied. For high values of \(w_V\), the final A/V combination output tends to be similar to the video text recognizer output and results in a high ratio for \((V_C, AV_C)\) and \((V_W, AV_W)\) and a much smaller ratio of characters change from \(V_C\) to \(AV_W\) or \(V_W\) to \(AV_C\). As expected, the reverse trend is true when \(w_A\) takes a higher value. The improvement in the final result comes from reducing the number of characters in \((V_C, AV_C)\).

**Character-Specific Weight Based Techniques.** We have generated character-specific weights (Character Wts.) in two ways. The first uses the video text recognizer sensitivity of each character as \(w_V\) and the second used the video text recognizer precision of each character as \(w_A\). Here, \(w_A = 1 - w_V\). Between these two techniques, the precision based weights seem to perform better with a recognition accuracy of 64.5%. Here, the improvement mostly comes from an increased number of characters in \((V_C, AV_C)\).

**Recognizer Ensemble Based Techniques.** Finally for ensemble based techniques, Table 4 shows that it is important to select the correct set of recognizers to combine and that increasing the number of recognizers does not necessarily increase the final system accuracy. For ensemble based techniques, we see a significant improvement in the end-to-end system character recognition accuracy. The maximum accu-

---

**Table 4**

<table>
<thead>
<tr>
<th>Technique</th>
<th>Recognition Accuracies For Ambiguous Characters (%)</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(V_C)</td>
<td>(V_C)</td>
</tr>
<tr>
<td>VTR</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[(F_1, F_2, F_3)]</td>
<td>19.0</td>
<td>6.2</td>
</tr>
<tr>
<td>[(F_1, F_2)]</td>
<td>20.5</td>
<td>4.8</td>
</tr>
<tr>
<td>(\text{Recog. Wts.})</td>
<td>19.6</td>
<td>5.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Sensitivity</th>
<th>Precision</th>
<th>Char. Wts.</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{Sensitivity})</td>
<td>19.6</td>
<td>5.5</td>
<td>14.0</td>
<td>25.0</td>
</tr>
<tr>
<td>(\text{Precision})</td>
<td>18.4</td>
<td>6.8</td>
<td>17.6</td>
<td>21.5</td>
</tr>
</tbody>
</table>

| [\(\text{Sensitivity}\)] | - | - | - | 54.9 | 63.8 |
| [\(\text{Precision}\)] | - | - | - | 58.2 | 65.9 |
| [\(\text{Sensitivity}, \text{Precision}\)] | - | - | - | 59.0 | 66.4 |

**Thresholding:** \(T=0.9\). **Option Selection:** \([\beta=0.9, \beta=0.9]\). **Synchronization:** Rank Sum.
racy achieved was 66.4% which is a 12.7% absolute improvement and a 23.6% relative improvement compared to the baseline video text recognizer.

10 Conclusions & Future Work

This paper described an end-to-end recognition system for audio-video based character recognition of handwritten mathematical content from classroom videos. We presented a comprehensive mathematical model for the end-to-end recognition system and provided description of the various stages and components of the recognition system. Experiments conducted over a large data set, consisting of videos recorded in a classroom like environment, demonstrate that significant improvements in character recognition accuracy can be achieved by making use of the proposed recognition system. While the implementation of the end-to-end recognition system made use of specific text and speech recognizers (word spotter), the proposed techniques could easily be used with other recognizers as well. We are currently in the process of investigating the use of mathematical grammar and the spoken content to improve the structure recognition accuracy associated with handwritten mathematical content. Other avenues for future work include exploring the use of a mathematical grammar for the audio-video based character recognition stage and using time-stamping information along with character recognition results to spot sequences of words instead of single words.

References


