Selective Text Removal from Videos

Snehil Sharma
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
ss7696@cs.rit.edu

***Abstract*—This paper introduces an automated system designed to enhance content moderation on online video platforms by effectively removing sensitive text. Leveraging the FAST text detection model, PARSeq text recognition model, and Aho-Corasick and Levinstein Distance string matching techniques, the system successfully identifies and flags inappropriate or sensitive texts. This ensemble system achieves 80.27% accuracy on text detection, 82.65% accuracy in text recognition, and 78.32% accuracy in text spotting tasks, on ICDAR '19 ArT dataset. Moreover, results of an informal usability study shows that the system achieves an average user rating of 4.30 in blurring performance and 4.55 in detection performance.**

***Index Terms*—Text Removal; Video Censoring; Text Recognition**

I. Introduction

Selective text removal in videos addresses the challenging task of content moderation, particularly within the context of the vast landscape of user-generated video content. The diverse nature of video uploads on online platforms emphasize the importance of safeguarding against unintended exposure of sensitive text, including profanity, personal names, addresses, and other private information. At its core, this initiative is driven by the fundamental need to enhance user experience and uphold community standards.

The automatic and precise removal of sensitive text from each frame before videos go online not only aligns with privacy norms but also streamlines content moderation processes for platforms. The ensuing sections delve into the intricacies of the implemented technologies, shedding light on how they collaboratively work to achieve the removal of sensitive texts in videos.

II. Related Work

In video censoring task the main goal is to censor malicious content, particularly profane words, scenes or audio. A recent work [1] introduces an innovative approach to automate video moderation, which involves silencing the audio and pixelating, or blurring, the lips in video segments containing profanity. However, this approach does not censor any text appearing on the screen. Extensive research has also been made in text censoring. A recent method [2] introduces a decision system that employs unsupervised learning, leveraging skip-gram and cosine similarity, to detect obfuscated abusive language. The system incorporates various efficient mechanisms, including blacklists, n-grams, edit-distance metrics, mixed languages (words from two different languages in the same sentence), abbreviations, punctuation, and special characters, to identify the intentional obfuscation of abusive words. Notably, the integrated decision system demonstrates a high accuracy for malicious word detection in news article comments, and similar strong performance for online community comments and Twitter tweets. However, this system is designed to work on text documents such as newspaper articles, tweets, social media comments, etc. This means that their system works on pre-extracted sentences and paragraphs. Extracting sentences from each frame of the video is not in the scope of this project and thus, a system for word-level filtering is needed. Another work [3] developed a machine learning-based approach to detect hate speech in user comments from various domains. They also create a unique corpus of annotated user comments for abusive language. This system also needs pre-extracted sentences for its processing, and thus, for the above-mentioned reasons is not suitable.

Scene text removal involves removing text regions from scenes while preserving the background of the text. One recent work introduces FETNet [4], that performs this by erasing text features and using an attention module to generate feature similarity guidance. Further, it uses a Feature Transferring Module to transfer features to different layers based on attention guidance. This resulting end-to-end network performs impressively against other state-of-the-art methods in SCUT-Synth [5] and SCUT-EnsText [6] datasets, especially showcasing its excellent ability to reconstruct background. Another work introduces MTRNet++ [7], a novel one-stage mask-based text inpainting network. Its architecture uses feature mask refinement, coarse inpainting, fine inpainting branches, and attention blocks, which enables it to remove text with and without external masks. This multi-branch architecture is also easily trainable in an end-to-end manner. MTRNet++ achieves superior performance on the SCUT-Synth [5] and SCUT [8] datasets, compared to state-of-the-art methods, without relying on external ground-truth masks. However, these models, on their own, cannot identify what the textual transcription is. This would result in an approach to remove all text in the scene. Thus, these models are not suited for selective text removal, where the task is to remove profanity or sensitive text, while preserving other text in the scene. Therefore, using text detection and recognition models along with a blur filter is more suitable.

Scene text detection is the task of detecting/segmenting regions where text appears on the scene. The field of scene text detection has seen significant advancements in recent years, with various methods aiming to enhance accuracy and efficiency. EAST [9], offers a pipeline that directly predicts words or text lines with arbitrary orientations and quadrilateral shapes from full images using a single neural network, eliminating the need for intermediary processing steps such as candidate aggregation and word partitioning. Experimental results on ICDAR '15 [10], COCO-Text [11], show a good accuracy, although with a speed trade-off. Choosing a text detection model with good accuracy is needed when processing videos. Another work, SRFormer [12] introduces a Mask-informed Query Enhancement module, which uses segmentation results for soft region-of-interest extraction to enhance instance queries. Empirical analysis showcases the performance of SRFormer across benchmarks such as [13] achieving high accuracy.

Scene text recognition involves identifying and transcribing the text present in natural images or scenes. There are many methods proposed to tackle this task. One such method introduces Aster [14], an end-to-end neural network model with two key components, a rectificaton network and a recognition network. The rectification network adaptively transforms input images to rectify text within them. The recognition network in ASTER is a sequence-to-sequence model with attention mechanisms. It directly predicts a character sequence from the rectified image. Evaluation on text recognition datasets show that Aster performs well in IIIT 5k-work [15] and ICDAR 2013 [16] datasets, but struggles with accuracy on ICDAR 2015 [10] dataset. Another work introduces Multi-modal Text Recognition Network (MATRN) [17], a novel approach that improves recognition performance by using interactions between visual (pixel-level information of the text) and semantic (context-rich information associated with the text captured using language models) features. It encourages the fusion of semantic features into visual features during training by masking visual clues associated with characters. MATRN shows excellent results in IIIT 5k-word [15], ICDAR 2013 [16], and ICDAR 2015 [10] datasets.

Both, text removal and text recognition models need text detectors/segmenters to find where text regions appear in a given image or scene. Selective text removal requires identifying or transcribing what text appears in the segmented text regions, and then removing or blurring the text that matches the sensitive word vocabulary. While a pipeline of text detector, text recognizer and text removal models could be used, it is very heavy on processing to run all three models together. Moreover, a simple blur filter can serve the purpose of censoring the sensitive text and achieve the goal of content moderation. Therefore, a combination of text detector and text recognition models is sufficient for this task.

III. Methodology

Figure 1 shows the system design. The frames from the video are extracted by OpenCV [18] and each frame is processed individually. To identify where textual profanity occurs in a video, the system needs to detect exactly what words appear in each frame, and then selectively remove any words that have a match in the sensitive word vocabulary. A scene text detection model, FAST [19], performs the required text segmentation, and was chosen due to its inference speed and accuracy. A text recognition model, PARSeq [20], is used to recognize text, chosen for its robustness and high accuracy. Flagging the text for removal is done by using string matching using algorithms such as Aho-Corasick [21] to find matches in the sensitive word vocabulary. Moreover, spelling and detection errors are handled by using Levenshtein Distance [22]. Using OpenCV [18], a blur filter is used to remove the flagged text from the scene.

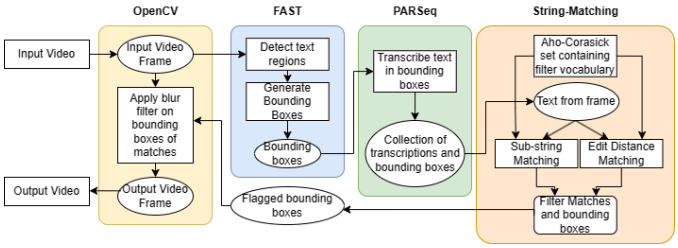

Fig. 1. Figure showing the system design

A. Text Detection

The system uses FAST [19] text detection model to generate bounding-boxes of all texts detected in the frame. It uses a minimalist kernel representation (MKR), where unnecessary features and functionalities are removed and only the fundamental components used for basic system operation are kept, resulting in reduced size and complexity of the kernel. Moreover, a GPU-parallel post-processing step called text dilation, is used, in which complete text lines are rebuilt from predicted text kernels, to efficiently assemble text lines with reduced time overhead. FAST achieves a notable trade-off between detection performance and inference speed, achieving comparable detection accuracy to SRFormer [12] while also being faster than EAST [9]. In this system, the FAST-640 configuration is used which is trained on the TotalText dataset [13]. The frame is first processed by image transforms and then sent to the FAST model for inference and text detection. The configuration of the model uses a filter to only return detections that have a confidence score higher than 70% . The model returns a list containing bounding-box coordinates of all detected texts in the frame. this list is passed on to the text recognition module.

B. Text Recognition

The text recognition module uses PARSeq [20] text recognition model, which employs Permutation Language Modelling to train autoregressive language models with shared weights. Autoregressive models are designed to predict the next element in a sequence given the previous elements. Sharing weights means using the same set of parameters (weights and biases)

for multiple parts of the network. This approach unifies context-free, where each token is generated independently of the others, and context-aware decoding, where previously generated tokens are considered when predicting the next token in the sequence. This contributes to robustness, particularly in handling vertical and rotated text commonly found in real-world images. PARSeq delivers high accuracy on benchmarks such as COCO-Text [11] and delivers comparable performance to MATRN [17]. Moreover, PARSeq performs particularly well in situations where the parts of the text in the images is obscured. Its ability to predict what the obscured letters be, is the reason it was chosen for this system. Videos can often contain text overlaps or partially visible text as seen in video game interfaces or user-added effects. In this system, the 'tiny' configuration of PARSeq was used, which is trained on the [23] dataset. The list of bounding-boxes are used to crop out the text regions from the frame. Next, all the bounding-box and text region pairs are then put in a batch and send to the model. The model performs the inference and transcribes the text in the text regions. It returns a dictionary containing the transcribed texts and a list of corresponding confidence scores and bounding-boxes.

C. String Matching

The list of transcribed words along with the corresponding bounding-boxes are sent to the string matching module. The detections and transcriptions are not always accurate. There may be cases where the detection contains multiple words and as such the transcription will be a concatenation of these words. In such cases, finding a word-to-word match in the sensitive word vocabulary will not work.

1) Aho-Corasick: In order to address this issue, the Aho-Corasick [21] python library is used to find substrings of the transcriptions that have a match in the vocabulary. The vocabulary is a set of words that the Aho-Corasick algorithm uses to make an Aho-Corasick automaton. This automaton uses a trie (a tree-like data structure) augmented with additional links to efficiently handle multiple pattern searches. Each node in the trie represents a character in one or more patterns. The root of the trie represents an empty string. For each pattern in the set, it traverses the trie, adding nodes and edges as needed. In addition to regular transitions (edges) between nodes representing characters, the Aho-Corasick algorithm introduces "failure links". If a match fails at a certain state, the failure link guides the automaton to the longest proper suffix of the current pattern that matches a prefix of another pattern. By following these links, it can quickly skip ahead in the text when there is a mismatch. To start matching the text against the trie, it follows the edges corresponding to the characters in the text. If a mismatch occurs, it follows the failure link to skip ahead in the text and continue matching from there. If it reaches a node corresponding to the end of a pattern, it is a match.

Using Aho-Corasick automaton to hold the vocabulary, substrings of each transcription are checked against the patterns appearing in the automaton. The substrings are generated iteratively and checked against the automaton for a match. However, if the automaton returns a "fail state" flag, the current substring is completely dropped and new substrings are generated from the character after the end of the current substring. For example, if the transcription is "classroom", and the substring "class" returns a "fail state" flag, then the next substrings will be generated from "r", such as "ro", "roo", etc. This method helps avoid unnecessary processing for redundant non-matches. Moreover, as soon as a single match is found, the transcription that was currently being tested is flagged for removal. The system does not check if there are multiple matches found in the different substrings of the transcription, as the goal is to flag the entire transcription regardless. Fig. 2 shows an example of how Aho-Corasick is used in the system. In the example, the vocabulary contains the word "classroom". However, the text detected is "THECLASSROOM". Leveraging Aho-Corasick the system finds substrings of the detected text that match the vocabulary. The system is able to find a match with substring "CLASSROOM" in the vocabulary and thus, the detected word "THECLASSROOM" is flagged for removal.

2) Levenshtein Distance: In some cases the text appearing in the frame or the transcription generated may have spelling errors. In such a case, Aho-Corasick will not be able to find a substring that has a match in the vocabulary, as the misspelled word will not appear in the pattern. Therefore, to address this case, the Levenshtein Distance [22] between the transcription and the vocabulary is calculated. Levenshtein Distance, also known as the edit distance, is a metric used to measure the similarity between two strings by counting the minimum number of single-character edits required to transform one string into the other. These single-character edits can be insertions, deletions, or substitutions. If Aho-Corasick found no matches, then, the system calculates the Levenshtein distance between the transcriptions and the vocabulary. If the vocabulary contains a word that is 1 edit distance away from the transcription, then the transcription is flagged for removal. Fig. 3 shows an example of how Levenshtein Distance is used in the system. In this example, the vocabulary contains the word "coinswatch". However, the text in the frame is "coinswitch". Since the Levenshtein Distance between "coinswitch" and "coinswatch" is 1, the word "coinswitch" is flagged for removal.

Table I shows the various cases of string matching that the system encounters. In the "Direct Match" scenario, the detected text exactly matches a word in the vocabulary. For instance, the detected text "panda" corresponds directly to the vocabulary entry "panda". The "Sub-string Match" case involves the detected text containing a sub-string that directly matches a word in the vocabulary. An example is the detection of "scrapbook", where the sub-string "crap" aligns with a vocabulary entry. The "ED-1 Match" situation occurs when the detected text can be modified by substituting, adding, or

Fig. 2. Example of Aho-Corasick helping in identifying matches. Here the vocabulary contains the word "classroom". However, the transcription detected is "THECLASSROOM". In this case, Aho-Corasick is able to match the substring "classroom" to the vocabulary.

Fig. 3. Example of Levenshtein Distance helping in identifying a match. Here the vocabulary contains the word "coinswatch". However, the text appearing in the frame is "coinswitch". Levenshtein Distance between the two is 1 and so, "coinswitch" is flagged

deleting a single character to achieve a direct match in the vocabulary, as demonstrated by "bass" transforming into "ass". Lastly, the "No Match" category denotes instances where the detected text fails to match any vocabulary entry under the defined criteria, such as "crepe" having no corresponding match with "crap" in the vocabulary.

D. Blurring and Ouptput

After the string matching process, the flagged bounding-boxes are sent to the blurring module. Using OpenCV [18], the Gaussian Blur filter is applied to each of the bounding-box regions. The kernel size parameter is set to "(45, 45)", where the tuple holds the height and width of the filter matrix that is used for the convolution for Gaussian Blur. The standard deviation parameter is kept as "0". This means that the standard deviation value, "sigma", is calculated using the kernel size, given by equation (1), where "ksize" is the kernel size. Once the blur filter is applied to all the flagged bounding-box regions of the video frame, the modified video frame is written to the output video file using OpenCV.

$$sigma = 0.3 \times ((ksize - 1) \times 0.5 - 1) + 0.8 \qquad (1)$$

IV. Evaluation

Evaluation of the system is done in four parts. First three parts are evaluating the performance of the detection and recognition capabilities of the system. The fourth part of evaluation entails conducting an informal usability study of the system.

A. Evaluation on ICDAR ArT Dataset

The ICDAR '19 ArT (arbitrarily shaped text) [23] dataset is known for its diversity in text shapes, encompassing curved, slanted, and irregularly shaped text regions. It consists of three tasks: text detection, text recognition and text spotting (detection + recognition). The dataset contains over 10,000 samples, with both real and synthetic images. Fig. 4 shows an overview of the ArT dataset.

TABLE I
EXAMPLES OF THE DIFFERENT CASES OF STRING-MATCHING ENCOUNTERED BY THE SYSTEM

Cases	**Description**	**Example Detection**	**Example Vocabulary**
Direct Match	The detected text has a match in vocabulary with identical character sequence	"panda"	"panda"
Sub-string Match	The detected text contains a sub-string which has a direct match in the vocabulary	"scrapbook"	"crap"
ED-1 Match (edit distance of 1)	The detected text can be modified by substitution, addition or deletion of one character and the resulting text has a direct match in the vocabulary	"ship"	"shit"
No Match (edit distance greater than or equal to 2)	The detected text has none of the above matches in the vocabulary. No direct or sub-string match found and the edit distance is greater than or equal to 2.	"crepe"	"crap"

Video streams contain a wide variety of text with varying orientations, sizes and shapes. The ArT dataset allows to assess the system's capability to accurately detect and recognize text within such scenarios. The system is evaluated on Task 1: Text Detection, Task 2: Text Recognition and Task 3: Text Spotting.

1) Evaluating Text Detection: Task 1 of ArT dataset is used to benchmark the text detection capabilities of the system. The task contains images with multiple text regions in different sizes and orientations. The input images are given to system's FAST [19] module to generate bounding-boxes for all detected text regions in the images. For each input image, the generated

Fig. 4. Overview of the ICDAR '19 ArT dataset [23]

bounding-boxes are compared against the ground truth bounding coordinates. This is done by calculating the intersection-over-union (IOU) between the two. The task requires that the threshold for a correct detection is an IOU of over 0.5. For the evaluation of the model on the task, its precision, recall and accuracy are computed. The accuracy is measured as the harmonic mean. These metrics are given by:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP) + False Positives (FP)}} \quad (2)$$

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP) + False Negatives (FN)}} \quad (3)$$

$$\text{Accuracy} = \frac{\text{2*Precision*Recall}}{\text{Precision + Recall}} \quad (4)$$

2) Evaluating Text Recognition: Task 2 of ArT dataset is used to evaluate the text recognition capabilities of the system. Task 2 comprises of cropped images containing a text region with just one word of text in latin characters, in varying orientations and shapes. The input images are given to the system's PARSeq [20] module for text recognition. The model generates the transcription of the text. This transcription is then compared to the corresponding ground truth for evaluation. The transcription is considered correct only if it is identical to the ground truth. Moreover, Normalized Edit Distance (1-NED) is calculated between the transcriptions and ground truth. This measures the similarity between the two strings. Edit Distance (ED) is the minimum number of edit operations (additions, removals, substitutions) required to transform one string to another. Normalized Edit Distance (NED) is then calculated by normalizing the Edit Distance by the length of the reference string. This normalization accounts for the fact that longer strings are expected to have higher raw edit distances. An NED of 0 represents a perfect match and an NED of 1 represents no similarity. The task requires to calculate the value "1 - NED" which represents that a perfect match would have a value of 1 and strings having no similarity will be 0. The accuracy is calculated similar to equation (4).

$$\text{1-NED} = 1 - NED = 1 - \frac{\text{Edit Distance (ED)}}{\text{Length of the reference string}} \quad (5)$$

3) Evaluating Text Spotting: In this task the performance is evaluated on detection and recognition of every text instance in the provided image in an end-to-end manner [23]. However, the system is not an end-to-end approach as it uses two different models sequentially. Due to this, it cannot be compared against the state-of-the-art methods in this task. Just like the task 2, the input images are sent to FAST for segmentation and generating the bounding-boxes for all the text detected in the image. These bounding-boxes are sent to PARSeq where the input image is cropped based on the bounding-boxes and transcriptions are generated for text in each bounding-box. The performance of the system is evaluated by comparing the IOU (intersection-over-union) between the bounding-boxes of the output and the ground truth. The task specifies that if the IOU between the output and one of the ground truth boxes for the image is over 0.5 then the corresponding transcription is compared against the ground truth for the evaluation. In the case of multiple matches of the output and the ground truth, only the ground truth bounding-box with the highest IOU is considered, and the rest are taken as false positives. The accuracy is calculated similar to equation (4).

B. Informal Usability Test

The primary goal of this usability study is to evaluate the effectiveness of detecting and blurring specified words in videos. The insights gathered from this study help identify areas for improvement and inform further developments to enhance the overall user experience. In this study 5 videos, each about 10 seconds long, were processed by the system. There were two videos from news category, two videos from video game captures, and one video clip from a live-stream which contained on-screen captions and viewer chat. The system was given each video along with corresponding vocabulary words for the video. The system generated two output videos for each input video. One video contains detection visualizations of the flagged words in the video, as seen in 5. In the other video, these flagged words are blurred out, as seen in 6.

There were four evaluators, all of whom were students of Golisano College of Computing and Information Sciences. I was not an evaluator. The evaluators were asked to fill a questionnaire for each video. The questions asked were:

- Were there any instances where the system incorrectly blurred text that did not match the pre-defined list of words? (Evaluators were also asked to list the words if they answered 'yes')
- Were there any instances where the system missed blurring text that should have been detected? (Evaluators were also asked to list the words if they answered 'yes')

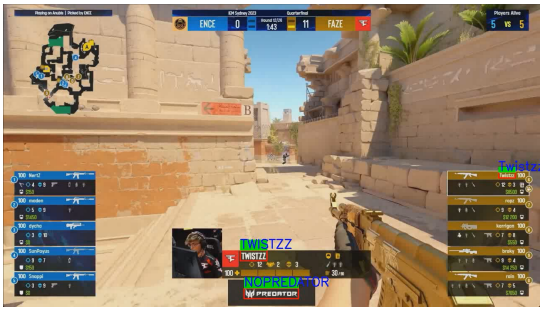

Fig. 5. This figure shows a frame from a visualization video used for the informal usability study.

Fig. 6. The figure shows a frame from a blur video used for the informal usability study.

- Overall, did the system perform well in blurring specified words in the video? (Scale of 1-5, where 1 is poor and 5 is excellent)
- In your opinion, how accurate was the system in detecting specified words? (Scale of 1-5, where 1 is not accurate and 5 is highly accurate)

The evaluators were also asked to provide any additional comments they may have for each video.

V. Results

This section discusses the results obtained from the evaluations on ArT dataset on text detection, text recognition and text spotting. The system is compared against results of end-to-end systems for the respective tasks. Moreover, the results of the informal usability study are analyzed along with the qualitative analysis discussion of various cases showing the strengths and weaknesses of the system.

A. Text Detection

Table II shows the results of text detection evaluation. The system achieved an accuracy of 80.27%, precision of 83.42%, and recall of 77.36%. For comparison, the performance of state-of-the-art systems on this dataset are also provided. the AntFin-Cascade Mask R-CNN [24] method demonstrated higher performance with an accuracy of 85.18%, precision of 87.08%, and recall of 83.36%. Whereas, the I3CL [25] method also showed competitive results, achieving an accuracy of 84.03%, precision of 87.26%, and recall of 81.03%.

B. Text Recognition

In Table III the results of text recognition are presented, highlighting the accuracy and the Average 1-NED (Normalized Edit Distance). The system achieved an accuracy of 82.65% in recognizing texts. For the recognition quality using the Average 1-NED metric the system performs at 93.96%. To contextualize the system's performance, it was compared to two state-of-the-art methods in text recognition. The CLIP4STR [26] approach achieved an accuracy of 85.9% and the MGP-STR [27] method achieved an accuracy of 84.9%. Unfortunately, data for the Average 1-NED was not available for comparison.

C. Text Spotting

In Table IV the results of the text spotting (detection + recognition) task are presented, highlighting the accuracy, precision, and recall of the system. The text spotting system achieved an accuracy of 78.32%, a precision of 82.8% and recall of 74.2%. This task requires evaluation on end-to-end models and the system does not fit this criteria. However, to provide context, evaluation performance of Tencent TEG OCR [28] and Sogou OCR [29], on the task are provided. Tencent TEG OCR achieved an accuracy of 68.40%, while Sogou OCR achieved an accuracy of 61.78%.

D. Results of Informal Usability Study

This section discuses the results of the informal usability study. The ratings given by the evaluators for blurring and detections were collected. The average rating was calculated for each video for both categories. The average of all ratings given, for blurring, was 4.30 with a standard deviation of 0.71. Figure 7 shows the average ratings for each video for blurring. The evaluators rated the system highest for "News 1" video, giving an average rating of 4.75 out of 5. Whereas, the lowest average rating was given to "Game 2", which was 3.75 out of 5. The average of all ratings given, for detection, was 4.55 with a standard deviation of 0.73. Figure 8 shows the average ratings for detections. The evaluators rated the system highest for "News 1" and "Captioned" videos, giving an average rating of 5 out of 5. Whereas, "Game 2" shows the lowest rating once again, getting an average rating of 3.75 out of 5. The evaluators expressed that the system struggled with "Game 2" video, missing detections and blurring on several instances.

E. Qualitative Analysis

In this section, various cases handled by the system are discussed. The system shows robustness in detecting and transcribing text that have some obscurity in its characters or orientation. Figure 9 shows a cropped video frame where the word "VETERANS" appears and the first few characters are appear slightly obscured due to its orientation. The system is able to accurately detect and transcribe this word. This is a strength of the PARSeq [20] model used in the system. Figure 10 shows that the system has trouble detecting numbers that are next to graphical items or symbols. Moreover, is is unable to detect "DHL" either. This limitation of the FAST [19] text detection model could be due to the datasets it was trained on.

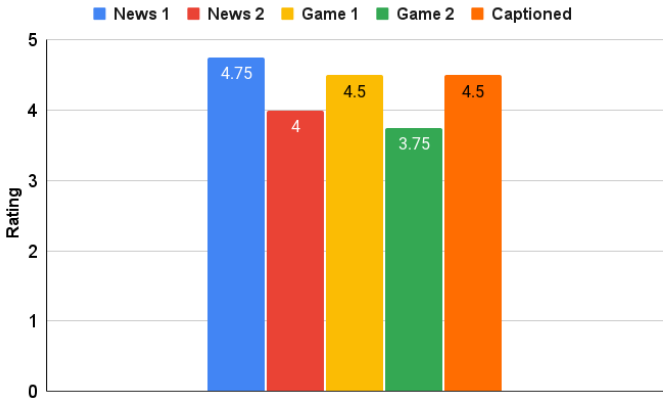

Fig. 7. This figure shows the average ratings for each video in blurring performance, in the informal usability study.

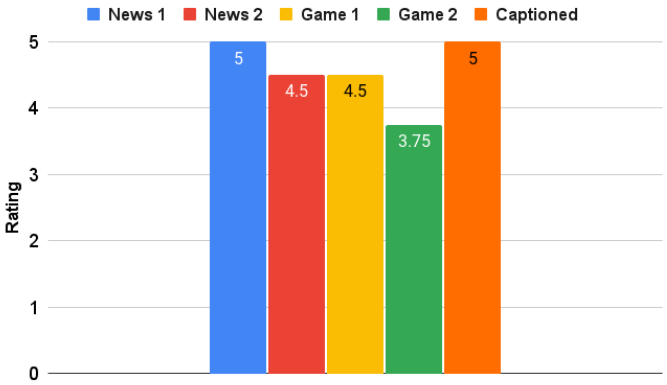

Fig. 8. This figure shows the average ratings for each video in detection performance, in the informal usability study.

The system needs to be able to blur text to the point that it is illegible. It is able to do so for most texts that have a scale much smaller than the video, as seen in Figure 11. However, the system struggled with text that is large and bold. It was unable to make the text illegible by blurring, as seen in figure 12. This was due to the constant kernel size parameter in the Gaussian Blur function. This was fixed after the study, by implementing a "dynamic blur" approach, where the kernel size of the Gaussian Blur function was adjusted based on the height of the bounding-box. Larger kernel size provides more distortion which helps blurring large and bold text. The kernel size in dynamic blur is calculated by taking the value of the bounding-box height and scaling it by a factor of "1.5". This value is then used for both, the height and the width of the kernel. Figure 13 shows the result of dynamic blur on the frame. It is able to render big, as well as, small text illegible.

VI. Conclusion

This paper introduces a system for censoring sensitive text from videos in order to improve content moderation for online video platforms. The system utilizes text detection and recognition models along with string matching techniques to

TABLE II
EVALUATION OF PERFORMANCE OF THE SYSTEM ON ART TASK 1: TEXT DETECTION

Method	Accuracy	Precision	Recall
This System	80.27%	83.42%	77.36%
AntFin-Cascade Mask R-CNN [24]	85.18%	87.08%	83.36%
I3CL [25]	84.03%	87.26%	81.03%

TABLE III
EVALUTION OF PERFORMANCE OF THE SYSTEM ON ART TASK 2: TEXT RECOGNITION

Method	Accuracy	Avg. 1 - NED
This System	82.65%	93.96%
CLIP4STR [26]	85.9%	-
MGP-STR [27]	84.9%	-

TABLE IV
EVALUATION OF PERFORMANCE OF THE SYSTEM ON ART TASK 3: TEXT SPOTTING

Method	Accuracy	Precision	Recall
This System	78.32%	82.83%	74.21%
Tencent TEG OCR [28]	68.40%	70.30%	66.61%
Sogou OCR [29]	61.78%	65.68%	66.61%

Fig. 9. This figure shows that the system is able to correctly identify the word "VETERANS" from the video frame.

Fig. 10. The figure shows that the system is unable to detect "DHL" and numbers next to graphical items.

Fig. 11. The figure shows that the system is able to blur the texts to the point that they are illegible.

Fig. 12. The figure shows that the system struggles with censoring large text as the text is still legible after blurring.

Fig. 13. The figure shows how dynamic blur is able to blur both large text and small text, rendering them illegible.

flag the text, that matches the filter vocabulary, in various cases and situations. The system was evaluated on text detection, text recognition and text spotting tasks. An informal usability study was conducted to analyze the user experience of the system. Moreover, the strengths and weaknesses of the system were discussed in various cases. Overall, the system is able to achieve the goal of automating censorship of sensitive words from the videos which is a step forward in the domain of content moderation.

A. Future Work

Looking at the evaluation of the system, its design can be improved.

1) Improving Throughput: The average processing framerate of the system is about 10 frames per second, which is very low considering that the test videos had a framerate of 30 frames per second. The bottle-neck in performance comes from the sequential processing, i.e., the processing by the detection model followed by the recognition model. This can be overcome by implementing parallelization of the frame processing module of the system. This process would buffer 30 frames from the video and distribute them in same-sized batches amongst multiple process threads running on distributed hardware. Frames processed by process threads are collected and put in an output buffer while also preserving the order of the frames as they were in the input video. If the throughput higher than 30 frames per second can be achieved, then this system could also be used to filter sensitive words on live video streams, in "real time".

2) Context-sensitive Filtering: The string-matching module does a good job at matching exact as well as mispelled words. However, it lacks any consideration of the context. For example, if the vocabulary included the word "ass" and the detected word in the frame was "classroom", the string-matching would flag this word as it includes "ass" substring. Context-sensitive filtering would leverage natural language processing (NLP) to capture the context of the flagged word along with the words in its vicinity (words in closest bounding coordinates). If the context of the flagged word is not classified as profanity, based on the NLP model's training, then this flagged word is not removed from the frame. This type of filtering would only work on profanity and not on sensitive information such as names, addresses, etc. in the vocabulary.

ACKNOWLEDGMENT

I would like to thank Dr. Richard Zanibbi and Dr. Leonik Reznik for their guidance. I would also like to thank Parashuram Ramesh, Prakhar Gupta, Divyank Kulshastra and Rakesh Puri for their evaluations in the informal usability study.

REFERENCES

[1] A. Chaudhari, P. Davda, M. Dand, and S. Dholay, "Profanity detection and removal in videos using machine learning," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, 2021, pp. 572–576.

[2] H.-S. Lee, H.-R. Lee, J.-U. Park, and Y.-S. Han, "An abusive text detection system based on enhanced abusive and non-abusive word lists," *Decision Support Systems*, vol. 113, pp. 22–31, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923618301064

[3] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, p. 145–153. [Online]. Available: https://doi-org.ezproxy.rit.edu/10.1145/2872427.2883062

[4] G. Lyu, K. Liu, A. Zhu, S. Uchida, and B. K. Iwana, "Fetnet: Feature erasing and transferring network for scene text removal," *Pattern Recognit.*, vol. 140, p. 109531, 2023. [Online]. Available: https://doi.org/10.1016/j.patcog.2023.109531

[5] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] C. Liu, Y. Liu, L. Jin, S. Zhang, C. Luo, and Y. Wang, "Erasenet: End-to-end text removal in the wild," *IEEE Transactions on Image Processing*, vol. 29, pp. 8760–8775, 2020.

[7] O. Tursun, S. Denman, R. Zeng, S. Sivapalan, S. Sridharan, and C. Fookes, "Mtrnet++: One-stage mask-based scene text eraser," *Comput. Vis. Image Underst.*, vol. 201, p. 103066, 2020. [Online]. Available: https://doi.org/10.1016/j.cviu.2020.103066

[8] S. Zhang, Y. Liu, L. Jin, Y. Huang, and S. Lai, "Ensnet: Ensconce text in the wild," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 801–808.

[9] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: An efficient and accurate scene text detector," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2642–2651.

[10] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. K. Ghosh, A. D. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny, "ICDAR 2015 competition on robust reading," in *13th International Conference on Document Analysis and Recognition, ICDAR 2015, Nancy, France, August 23-26, 2015*. IEEE Computer Society, 2015, pp. 1156–1160. [Online]. Available: https://doi.org/10.1109/ICDAR.2015.7333942

[11] A. Veit, T. Matera, L. Neumann, J. Matas, and S. J. Belongie, "Coco-text: Dataset and benchmark for text detection and recognition in natural images," *CoRR*, vol. abs/1601.07140, 2016. [Online]. Available: http://arxiv.org/abs/1601.07140

[12] Q. Bu, S. Park, M. Khang, and Y. Cheng, "Srformer: Empowering regression-based text detection transformer with segmentation," *CoRR*, vol. abs/2308.10531, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.10531

[13] C. Chng, C. S. Chan, and C. Liu, "Total-text: toward orientation robustness in scene text detection," *Int. J. Document Anal. Recognit.*, vol. 23, no. 1, pp. 31–52, 2020. [Online]. Available: https://doi.org/10.1007/s10032-019-00334-z

[14] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, "ASTER: an attentional scene text recognizer with flexible rectification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2035–2048, 2019. [Online]. Available: https://doi.org/10.1109/TPAMI.2018.2848939

[15] A. Mishra, K. Alahari, and C. V. Jawahar, "Scene text recognition using higher order language priors," in *British Machine Vision Conference, BMVC 2012, Surrey, UK, September 3-7, 2012*, R. Bowden, J. P. Collomosse, and K. Mikolajczyk, Eds. BMVA Press, 2012, pp. 1–11. [Online]. Available: https://doi.org/10.5244/C.26.127

[16] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. Almazán, and L. de las Heras, "ICDAR 2013 robust reading competition," in *12th International Conference on Document Analysis and Recognition, ICDAR 2013, Washington, DC, USA, August 25-28, 2013*. IEEE Computer Society, 2013, pp. 1484–1493. [Online]. Available: https://doi.org/10.1109/ICDAR.2013.221

[17] B. Na, Y. Kim, and S. Park, "Multi-modal text recognition networks: Interactive enhancements between visual and semantic features," in *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXVIII*, ser. Lecture Notes in Computer Science, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., vol. 13688. Springer, 2022, pp. 446–463. [Online]. Available: https://doi.org/10.1007/978-3-031-19815-1_26

[18] opencv, "opencv: Open source computer vision library," 2015. [Online]. Available: https://github.com/opencv/opencv

[19] Z. Chen, J. Wang, W. Wang, G. Chen, E. Xie, P. Luo, and T. Lu, "Fast: Faster arbitrarily-shaped text detector with minimalist kernel representation," 2021.

[20] D. Bautista and R. Atienza, "Scene text recognition with permuted autoregressive sequence models," in *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXVIII*, ser. Lecture Notes in Computer Science, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., vol. 13688. Springer, 2022, pp. 178–196. [Online]. Available: https://doi.org/10.1007/978-3-031-19815-1_11

[21] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, p. 333–340, jun 1975. [Online]. Available: https://doi.org/10.1145/360825.360855

[22] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals." *Sov. Phys. Dokl.*, vol. 10, p. 707–710, 1966.

[23] C. K. Chng, E. Ding, J. Liu, D. Karatzas, C. S. Chan, L. Jin, Y. Liu, Y. Sun, C. C. Ng, C. Luo, Z. Ni, C. Fang, S. Zhang, and J. Han, "ICDAR2019 robust reading challenge on arbitrary-shaped text - rrc-art," in *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*. IEEE, 2019, pp. 1571–1576. [Online]. Available: https://doi.org/10.1109/ICDAR.2019.00252

[24] Y. Lin, T. Xu, and A. Group, "Antfin-cascade mask r-cnn," 2023.

[25] J. Ye, J. Zhang, J. Liu, B. Du, and D. Tao, "I3CL: intra- and inter-instance collaborative learning for arbitrary-shaped scene text detection," *CoRR*, vol. abs/2108.01343, 2021. [Online]. Available: https://arxiv.org/abs/2108.01343

[26] S. Zhao, X. Wang, L. Zhu, and Y. Yang, "CLIP4STR: A simple baseline for scene text recognition with pre-trained vision-language model," *CoRR*, vol. abs/2305.14014, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.14014

[27] C. Da, P. Wang, and C. Yao, "Multi-granularity prediction with learnable fusion for scene text recognition," *CoRR*, vol. abs/2307.13244, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2307.13244

[28] M. Liao, P. Lyu, M. He, C. Yao, W. Wu, and X. Bai, "Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 532–548, 2021.

[29] J. Xu, L. Wu, M. Wang, H. Wang, L. Xu, L. Ma, and X. Su, "Sogou ocr," 2019.