International Journal on Document Analysis and Recognition (IJDAR)
https://doi.org/10.1007/s10032-019-00327-y

SPECIAL ISSUE PAPER

Generalized framework for summarization of fixed-camera lecture videos by detecting and binarizing handwritten content

Bhargava Urala Kota[1] · Kenny Davila[1] · Alexander Stone[1] · Srirangaraj Setlur[1] · Venu Govindaraju[1]

Received: 16 November 2018 / Revised: 3 April 2019 / Accepted: 30 May 2019

Abstract
We propose a framework to extract and binarize handwritten content in lecture videos. The extracted content could potentially be used to index video collections powering content-based search and navigation within lecture videos helping students and educators across the world. A deep learning pipeline is used to detect handwritten text, formulae and sketches and then binarize the extracted content. We exploit the spatio-temporal structure of our binarized detections to compute associativity information of content across all video frames. This information is later used to segment the video. Experiments are conducted to compare the performance of key components of our framework in isolation, as well as the impact on overall performance, with respect to existing methods. We evaluate our framework on the publicly available AccessMath lecture video dataset obtaining an *f*-measure of 94.32% for binary connected components. Code for the framework (including trained weights) and summarization will be released.

Keywords Lecture video summarization · Handwritten text detection · Binarization · Deep learning

1 Introduction

Lecture videos are a useful resource for students and educators across the world; however, they are still not properly indexed by most common search engines. The main reason is that, in many cases, search engines depend on existing text annotations in order to index video collections. If such annotations are unavailable, search engines do not have a principled way to retrieve them. Manually producing such annotations is a hard task given the scale of lecture video content available.

In this paper, we provide a pipeline for automated lecture summarization, which performs detection, binarization and temporal refinement of detected regions. We concentrate on videos where a single lecturer conducts a class while explaining and producing handwritten content on a whiteboard. Handwritten content in lecture material is often loosely structured and exhibits significant variance such as sentences, math expressions, matrices, sketches and plots. When combined with background noise, illumination changes, video compression artifacts and occlusions due to the lecturer, this presents a significant challenge for automatic extraction of handwritten content. Availability of limited annotated lecture video data further compounds the problem.

Our lecture summarization pipeline takes as input a video stream uniformly sampled at 1 frame per second. First, we detect bounding boxes enclosing handwritten content within a frame and binarize the full frame in parallel. After handwritten content is extracted from all frames of videos, the next task is to analyze the detected regions and identify association relationships of text regions across space and time—including cases where text regions are occluded due to motion of the lecturer. We propose a locality-based and lecturer detection-based algorithms to group and consolidate predicted bounding boxes *within* and *across* frames and use this to bring back binarized content occluded by the speaker.

These frames with recovered binary content are further refined by performing a connected component (CC)-level

✉ Bhargava Urala Kota
buralako@buffalo.edu

Kenny Davila
kennydav@buffalo.edu

Alexander Stone
awstone@buffalo.edu

Srirangaraj Setlur
setlur@buffalo.edu

Venu Govindaraju
govind@buffalo.edu

[1] Department of CSE, University at Buffalo, Buffalo, USA

Published online: 15 June 2019

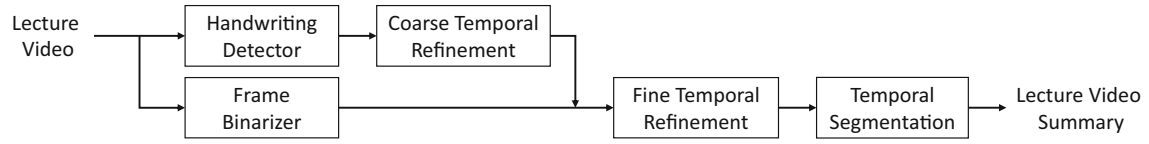


Fig. 1 Our generalized framework for summarization of still-camera whiteboard lecture videos

analysis. Stable CCs across frames are identified and missing CCs in frames due to inconsistency in detector outputs are recovered in this stage. In the next step, frames that contain CCs conflicting in spatial location during non-overlapping time intervals in the entire lecture video are identified. Finally, we segment the videos by greedily selecting frames that resolve the maximum number of spatio-temporal conflicts in content associations across frames [7], thus obtaining the *keyframes* of the lecture video. A keyframe of a segment ideally contains all unique content within that segment. Our pipeline is shown in Fig. 1.

The performance of our system is benchmarked against a publicly available dataset [6]. Evaluation metrics used are the number of keyframes produced as well as the recall, precision and *f*-measure of keyframe content (at the level of binarized connected components) with respect to all unique text content in the lecture video. Along with the corresponding temporal associativity index, keyframes extracted from a set of lecture videos can potentially be used to search and navigate to the frames of occurrence of handwritten queries within the collection [3,8]. The main research questions being investigated in our work include:

- How does the annotated lecture video data compare with transfer learning from scene text data for the purpose of training the handwritten content detector?
- How does full frame handwritten content binarizer perform for the overall task of summarization?
- What temporal refinement strategies of detected and binarized content are needed for efficient summarization?

2 Background

We view lecture video summarization as a special case of the general video summarization problem in computer vision. Prior work in this area can be can be categorized into domain-specific topics, i.e., the types of videos handled such as egocentric, sports, surveillance, lecture. This problem has also been viewed as a supervised learning problem, where human expert annotations are used to train and generate video summaries. On the other hand, unsupervised methods are used when annotation is too expensive to discover visual similarities within the video and select representative elements. Video summarization methods can also be broadly classified based on the kind of summary that is generated. Keyframes that contain the highlights of the video content is a typical form of summary found in the literature [17,18,22]. Broadly, the main approach for video summarization is to obtain a *representation* of every frame and use supervised or unsupervised methods to filter out the *relevant* subset of frames in order to produce a summary.

For the specific domain of whiteboard lecture videos, we seek to build a representation of frames by detecting and binarizing handwritten content within that frame. Unsupervised methods are used to obtain similarity information across frames by learning to extract features from detected areas of content. Then, the optimum frames to segment the video are found based on this similarity information to generate summary keyframes. We propose a deep neural network-based methodology for detecting and representing content inspired by recent work on detection of text in natural scenes. We then investigate methods to binarize and aggregate detected content across time. Finally, a space-time 'conflict' minimization approach is deployed to obtain the summary.

In the following subsections, we detail some of the existing work in the field of lecture video summarization, natural scene text detection methods and benchmark datasets in these fields.

2.1 Lecture summarization

We provide an overview of prior work on our main focus, lecture videos with handwritten content on a whiteboard. Most approaches in the literature follow the general pipeline of preprocessing, content extraction and summarization. Image processing and computer vision techniques are used extensively in each stage to obtain better performance.

At first, binarization techniques such as Otsu's [32] algorithm applied on every frame are used in simple videos without many challenges in illumination and background [40]. Segmentation of region of interest or some background subtraction followed by specialized binarization techniques is required for preprocessing when the lecture video poses illumination and background challenges [4,5,7,21].

After preprocessing, handwritten content is extracted or separated from background content. A common approach is to divide the video frame into a grid of cells followed by rule-based or statistical classification of each cell as content, background and noise [1,9,38,40]. Grouping and

refining of handwritten content using OCR-based methods [40] or temporal analysis [7,31] to handle noise and occlusion are commonly used as well. Some work uses contrast enhancement [9,40], while others use super-resolution-based methods [38] to improve readability of whiteboard content.

Several methods exploit computer vision techniques to take advantage of the characteristics of lecture videos. Explicitly modeling the speaker allows better handling of occluded content [4,9,21,31,36,38], while detecting erasure events is useful for segmentation and content extraction [4].

The final stage after preprocessing and content extraction is the summarization of the video. Video summaries in general could be of the 'keyframe' variety (described in Sect. 1) or a 'video skim' which is a shorter version of the input video containing the highlights of the entire video. We concentrate on keyframe-based summaries for our work. Keyframes are typically decided by analyzing content peaks in frames over time and segmenting the video when the content drops from a maxima, which generally corresponds to erasure events in the lecture [4,21]. A recursive algorithm to find the correct frames to segment based on spatial conflicts of extracted content has been proposed along with the public release of the AccessMath dataset [7]. Other forms of summaries of lecture videos include recognized text lines extracted from the video [38,40] and production of composite images that contain all content [4,21].

In our previous work [20], a handwritten content detector adapted from a scene text word detector model is used in lieu of the preprocessing and content extraction stages and the recursive conflict resolution approach is used to generate keyframes. In our current work, we train a content detector only on the AccessMath dataset.

2.2 Scene text detection

In order to detect handwritten content from video frames directly, we focus on some relevant prior work carried out in the domain of scene text detection in images and videos. A comprehensive survey of methodologies and evaluation strategies for text detection, tracking and recognition in video images is presented by Yin et al. [43]. In general, evaluation of detecting text content in video is done by treating text regions as objects and using multiple object tracking (MOT) metrics [2]. Specifically detecting handwritten content as a specialized case of scene text detection in video is covered in a survey by Ye and Doermann [42].

Earlier methods extract pixel or component level features to identify the text candidates which are then postprocessed using statistical learning models [10,29]. A list of prior methods can be found in the survey by Zhu et al. [45]. Of late, deep learning-based methods have been adopted due to their effectiveness in taking advantage of large annotated datasets [12,41], and most of these treat text detection as a specialized case of object detection.

Popular strategies in deep object detection use a fixed set of predefined anchor windows that slide across convolutional feature maps in a deep neural network. At each location in the feature map, a detector network makes a prediction of whether an anchor window contains an object and if so it regresses the offsets to the anchor window dimensions to fit the object in a tight bounding box. This is the basic principle in state-of-the-art object detectors like Faster-RCNN [35], YOLO [34] and SSD [25]. SSD, in particular, carries out detection on feature maps at multiple depths and combines the predictions using non-maximum suppression (NMS). Text detection neural networks generally adapt object detection networks by making modifications to size and aspect ratio of anchor windows and search locations across feature maps [12,23,39,44] to suit text localization in scene images.

Recently, deep learning-based detectors have also been used to detect handwritten content on whiteboards. Our prior work [20] shows summarization of lecture videos by detection of text, formulae and plots using domain adaptation from natural scene text datasets such as SynthText [12] and ICDAR2015 Robust Reading Competition Dataset [16] while Jia et al. concentrate purely on detection of whiteboard handwritten text [15].

2.3 Dataset and evaluation

AccessMath is the largest, publicly available, benchmarked dataset for this purpose. It was created from a collection of linear algebra lecture videos [7]. These HD videos (1920×1080 pixels) were recorded using a single, still camera covering the entire whiteboard with no zooming, tilting or panning. It consists of 12 lecture videos—5 for training and 7 for testing. The average length of all videos is about 49 min.

AccessMath uses the 'keyframe' method of lecture summarization and evaluation is carried out by measuring the average number of keyframes produced by the summarization methodology. Apart from this, the average recall and precision of all 'matching' binary connected components (CC) are measured across the entire video ('global') as well as per frame. The AccessMath dataset is annotated at the binary level in order to facilitate this evaluation scheme and the benchmarking methodology at the CC level [7].

To determine if one or more ground truth CCs correspond(s) to one or more predicted CCs, the predicted summary frames are translated and aligned with the ground truth frames for the corresponding temporal segment, such that overall pixel-wise recall is maximized. Then, overlapping CCs are selected and appropriately grouped using one-to-many, many-to-one and many-to-many matching groups [7]. The pixel-wise recall and precision is computed, and only

the groups with more than 50% on each of these metrics are accepted as valid matches. These measures are designed to compensate for variations in thickness and focus on readability of extracted summary CCs.

3 Detection of handwritten content

We treat the problem of detecting handwritten content as a specialized case of text detection in scene images. The problem of finding text is compounded by the variety of possible handwritten content which is elaborated in Sect. 1. We modify and adapt EAST [44] for this task, mainly because it is anchor-free and produces dense per-pixel outputs which are well-suited for a detection task with high variance in text content and layout.

3.1 Structure

The general EAST detector consists of a pyramidal feature extraction block, with downsampling convolutional layers and upsampling transpose convolutional layers and a convolutional region proposal network consisting of two sub-networks. One sub-network predicts text/non-text for every pixel in the upsampled feature map, whereas the other predicts four real values that correspond to displacements from top, right, bottom and left edges, respectively, which can be used to reconstruct a rectangle enclosing text around a particular point. It also predicts a fifth value that corresponds to an angle of rotation for the predicted rectangle, giving this model the capacity to detect oriented text. In the original article, the authors use a PVANet backbone [19], where the upsampled features are concatenated with features from the downsampling path as shown in Fig. 2.

However, in our implementation, we found that a feature pyramid network (FPN) converged better during training for both natural scene text detection and for detection of whiteboard handwritten content. FPN has shown state-of-the-art performance for object detection [24] and comprises of a ResNet [13] backbone with residual connections between features of corresponding scales in the upscale and downscale paths. We use unactivated upsampling layers and residual connections across scales as originally prescribed for FPN [24]. The region proposal sub-network, which acts on the last upsampled featured map, consists of a 3×3 convolution layer with one output channel for text/non-text predictions and a similar layer with five output channels for bounding box displacements and angle of rotation. This layer has a sigmoid activation function. Since the annotations in the AccessMath dataset have been released at the binary and at the axis-aligned bounding box level, we suppress the angle prediction output of the EAST region proposal sub-network. The final structure of the network is illustrated in Fig. 3.

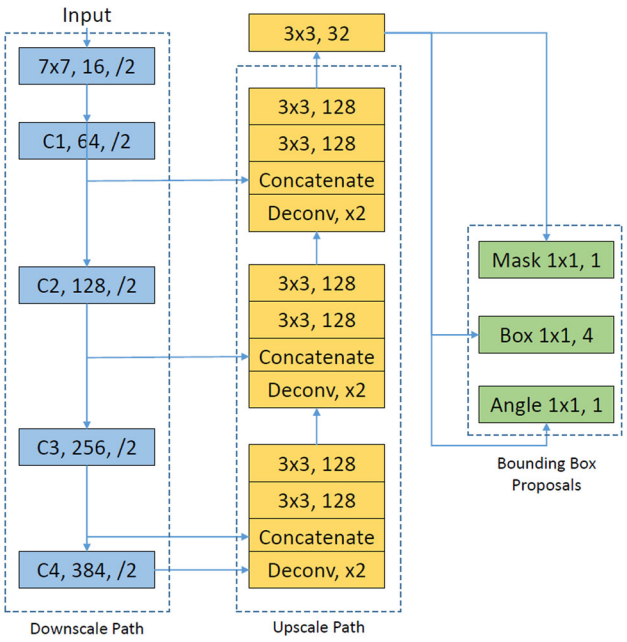


Fig. 2 The structure proposed by Zhou et al. for EAST Detector. Here, $k \times k, c$ indicates kernel size k and number of channels of a convolutional layer $/n$ or $\times n$ indicates downsampling or upsampling by factor of n. C stands for block of multiple convolutional layers according to the PVANet [19] structure

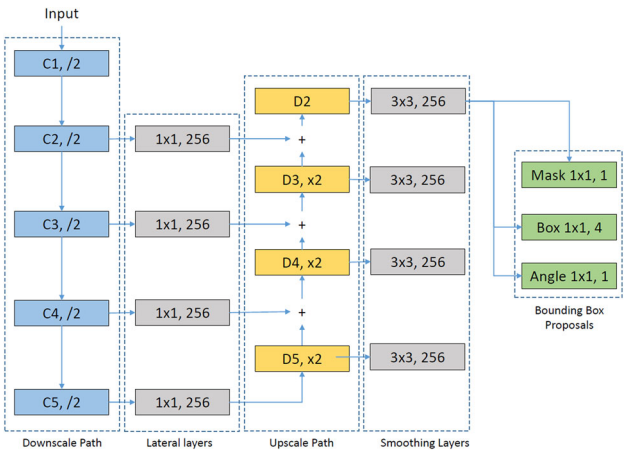


Fig. 3 Our modified structure for EAST Detector. Here, $k \times k, c$ indicates kernel size k and number of channels of a convolutional layer $/n$ or $\times n$ indicates downsampling or upsampling by factor of n. C and D stands for block of multiple convolutional layers and transpose convolutional layers, respectively, according to the feature pyramid network [24] structure

3.2 Training

Lecturer bounding box for every frame of the training videos are extracted at first. It should be noted that these are not part of the original annotations, and we use a SSD [25] detector trained on the PASCAL VOC object detection dataset [11], to obtain the lecturer bounding boxes in the training videos. We

then eliminate all ground truth text boxes which overlap with the bounding box of the lecturer if their area of intersection is a greater fraction of the text box area than 25%. To train the EAST region proposal layer, we need to generate ground truths corresponding to a text/non-text mask, and four ground truth maps with bounding box regression targets in each edge direction.

Text pixel prediction A mask image is produced, where every pixel inside a shrunken version of every text bounding box has a label of 1 (denoting foreground) and every other pixel has a label of 0 (denoting background). The shrinking procedure is as follows:

- For every edge vertex v_i in a bounding box, we find r_i which is the minimum of the lengths of either edges emanating from v_i
- For every pair of vertices v_i, $v_{j=i+1 \bmod 4}$ that form an edge of the box, we shift the vertices inwards by $0.3 \times r_i$ and $0.3 \times r_j$, respectively,

where $i \in \{0, 1, 2, 3\}$ corresponding to each vertex of the bounding box in clockwise order starting from top-left. The idea of training against a shrunken version of the original bounding box for dense predictions has been seen in DenseBox [14] and is mainly used to ensure tight bounding boxes during inference. Further, since the foreground pixels also have bounding box proposal targets corresponding to bounding box displacements, this procedure also ensures targets are nonzero.

Text box prediction For every foreground pixel, based on the ground truth bounding box we compute the ideal displacements to the top, right, bottom and left edges from the pixel. There is no loss computed for these targets on background pixels. An illustration of the ground truth masks, effect of shrinking and each displacement target is shown in Fig. 4.

Loss functions A generalized DICE coefficient loss is computed between text pixel predictions and ground truth targets for every iteration using the equation below:

$$L_{\text{dice}} = 1 - 2 \times \frac{\sum_{l=1}^{2} w_l \sum_n r_{\ln} p_{\ln}}{\sum_{l=1}^{2} w_l \sum_n r_{\ln} + p_{\ln}}; \; w_l = \left(\sum_{n=1}^{N} r_{\ln} \right)^{-2} \tag{1}$$

where r and p are the targets and predictions respectively, $n \in [1, N]$ are the pixels in the target and $l \in [0, L-1]$ are the set of pixel labels.

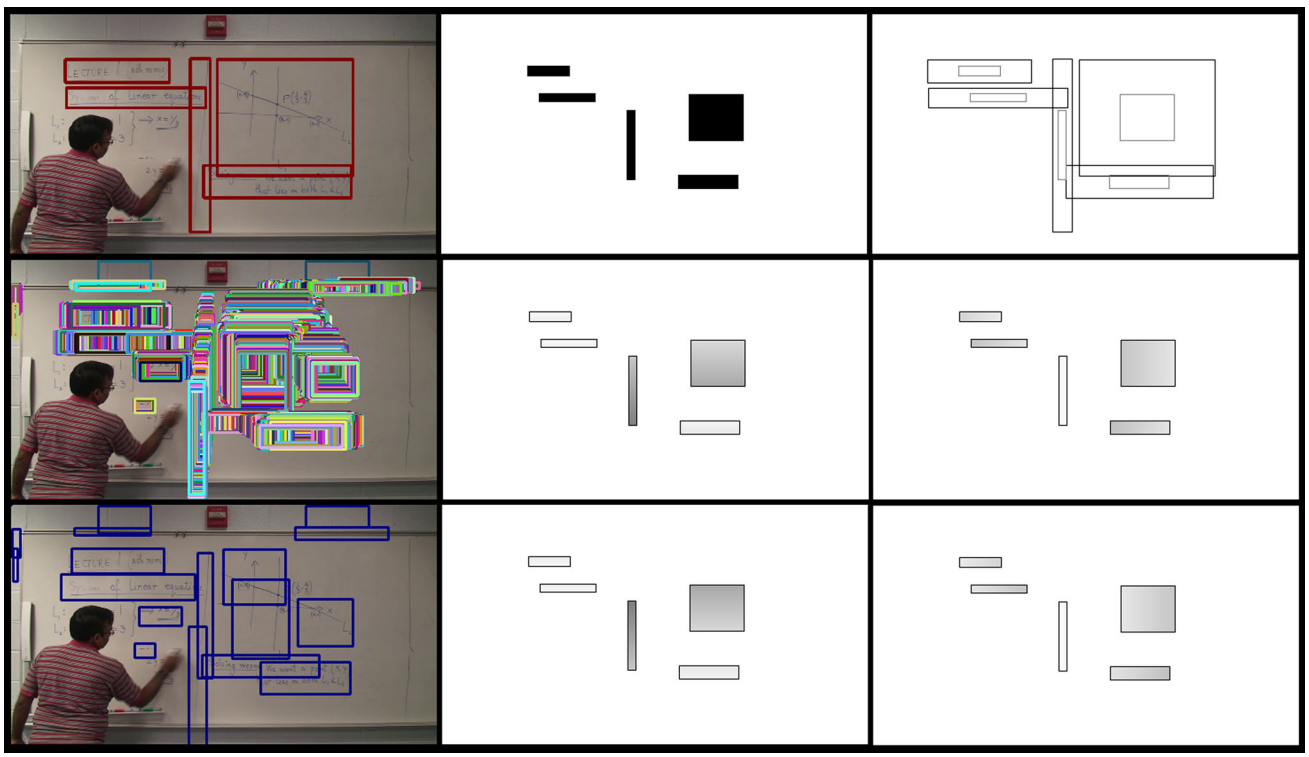


Fig. 4 The leftmost column shows a sample frame with ground truth annotations, dense bounding box predictions and bounding boxes after our clustering and non-max suppression. The two rightmost images in the top row show the pixel prediction targets and effect of box shrinking, respectively. The two rightmost images in the middle and bottom row show the box regression targets for top, right, bottom and left edges, respectively. Note that the colors are inverted, so darker pixels denote numerically greater values

DICE losses have previously been used in segmentation of bio-medical images and have been shown to perform well under unbalanced classes [37]. In our implementation, we found that the DICE loss was numerically more stable during training than weighted binary cross-entropy (BCE) loss for our data. Invalid numbers were observed during training with BCE loss possibly due to exponential and log computations for BCE along with random initialization.

Since the ideal bounding box displacements per pixel are known, we compute the intersection over union (IOU) loss [44] as described by Zhou et al. This loss is found to work better for text prediction in natural scene images than L_2 or smooth L_1 losses which are generally preferred for continuous value predictions [44]. If the angle predictions are to be used (which is not the case in our work), then loss for angles is computed as one minus cosine of the difference of predicted angle and ground truth angle of rotation. The overall loss for the network is computed as the sum of DICE loss and IOU loss.

3.3 Inference

Training procedure and hyperparameters are described in detail in Sect. 6. After trained weights are obtained, the neural network predicts a mask of text regions and for each foreground mask pixel it also predicts a corresponding text bounding box. In order to reduce the number of redundant predicted boxes, we use a combination of localilty-based clustering and non-maximum suppression. The set of predicted bounding boxes is traversed in a row-wise scanning pattern and if the IOU between two successive bounding boxes are greater than a threshold (θ_{loc}) then it is marked for clustering. This procedure is carried on greedily until a bounding box with IOU lesser than θ_{loc} is obtained. From this point, we continue the algorithm while maintaining a new 'cluster' until all predicted boxes have been visited. All bounding boxes marked for clustering are collapsed into a single bounding box that barely contains all of them. We take the surviving bounding boxes and perform a standard non-maximum suppression, with a separate threshold (θ_{nms}), to obtain the final set of predicted boxes. The effect of our post-processing procedure can be seen in Fig. 4. In Table 2, we compare performance of isolated text detection on the testing frames of the AccessMath dataset with existing methods. For the summarization metrics, we compare with a model adapted from natural scene text used in our prior work [20] in Table 1.

4 Binarization of handwritten content

A fully convolutional neural network with encoder–decoder architecture is used for binarization of lecture video frames. Such architectures have been successfully used for various image segmentation tasks [26]. Recently, Meng et al. [27] used a similar architecture to binarize degrade document images and obtained state-of-the-art results on the DIBCO competition dataset [33]. Encouraged by this performance, we use the same network for our binarization task.

4.1 Structure

The encoder part of the binarization neural network consists of four blocks of convolutional layers. The first two blocks contain two layers, and the last two blocks contain three layers. Each layer has 3×3 kernel sizes, batch normalization and leaky ReLU activation ($p = 0.2$). The output of each block undergoes max-pooling with a stride of 2×2. All layers in the first block have 32 filters, and subsequent blocks have double the number of filters per layer as the last block. The decoder part of the network consists of four blocks, each of which consists of transpose convolutional layer, batch normalization, leaky ReLU activation ($p = 0.2$) and dropout regularization ($p = 0.2$). Number of filters in each layer is equal to the number of filters in the corresponding encoder block. At each decoder block, the output of the corresponding encoding block (at same scale) is concatenated before feeding it to the next decoder block. At the final

Table 1 Summarization metrics comparing the content detector from our prior work and current work

Method	AVG	AVG global			AVG per frame		
	N_{f}	R	P	F	R	P	F
Without TR							
Current work	24	92.50	92.59	92.54	92.07	91.45	91.76
Prior work [20]	22	90.98	94.77	92.83	89.89	93.93	91.86
With TR							
Current work	28	93.44	85.47	89.28	92.89	84.30	88.39
Prior work [20]	20	92.33	94.16	93.23	91.69	93.45	92.56

Otsu's Binarization is used in both methods for fair comparison. Results with and without temporal refinement are shown

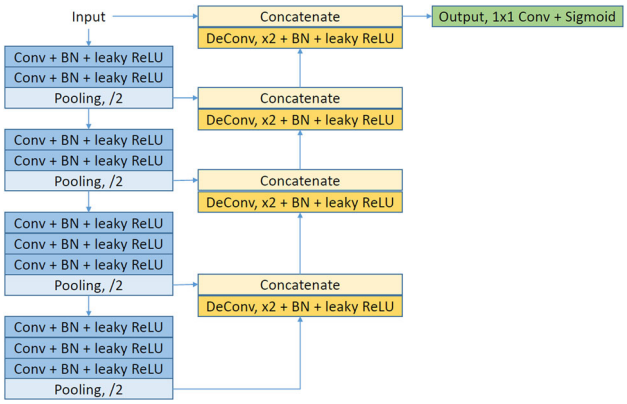


Fig. 5 Structure used for Binarization in our pipeline

block, the input image itself is used and a final convolutional layer with sigmoid activation is used as the prediction layer, l_{bin}. Figure 5 shows the structure of our binarizer neural network.

4.2 Training

The AccessMath dataset consists of binary ground truth for keyframes in each lecture video. The total number of full training frames amounts to roughly 80 frames. 128×128 patches are extracted randomly around the annotated ground truth boxes for each frame for data augmentation. Background patches are sampled such that they constitute roughly 10% of every training mini-batch. Both training and test images are preprocessed by bilateral filtering ($\sigma_{\text{space}} = 4.0$, $\sigma_{\text{color}} = 13.5$), followed by subtracting the median filter (33×33) response of that patch. We found that doing so improved binarization performance especially given lack of sufficient training data.

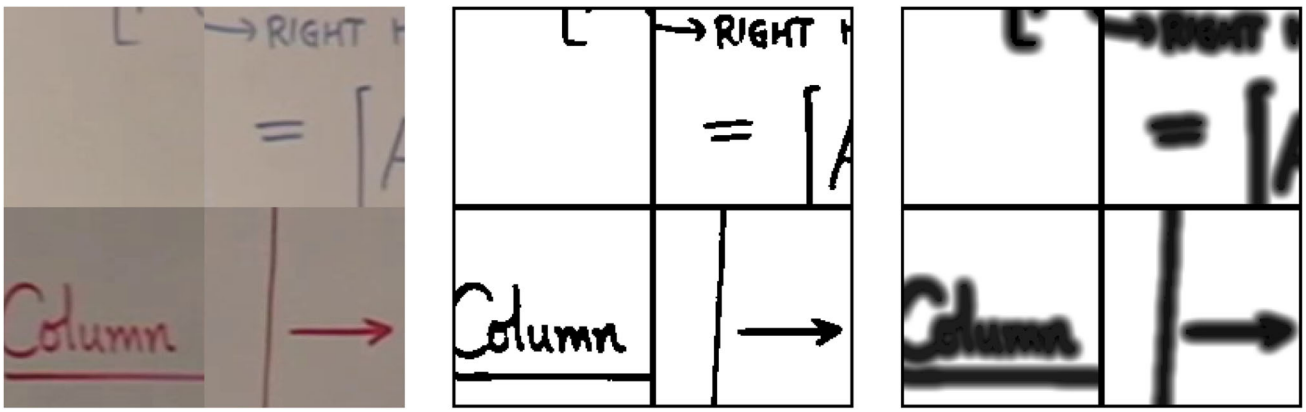
Losses A reconstruction loss is measured in each block of the decoder section of the binarizer. The output of each block in decreasing order of scale is upsampled by a transpose convolution layer (with strides 1, 2, 4 and 8), activated with a sigmoid layer and their losses are weighted by factors of 1.0, 0.5, 0.25, and 0.125, respectively. They are then added to the binarization loss computed at the final output layer l_{bin} of the decoder. All losses are computed using the binary cross-entropy function.

Pixel-wise weights A pixel weighting scheme is used to modify the importance of foreground pixels and neighboring background pixels while computing layer losses:

$$f(x, y) = t(x, y) * g(x, y) \tag{2}$$

$$w(x, y) = \log\left(\frac{f(x, y)}{\min(f(x, y))}\right) + 1 \tag{3}$$

where g is the Gaussian filter, t is the binary target mask, $*$ denotes convolution and $w(x, y)$ are the pixel-wise weights. The pixel-wise weight function is thus constructed in order to avoid numerically unviable scales of convolution outputs. We found that this weight scheme ensures that all pixels have at least a weight of one with the maximum pixel weight being 6–8 depending on sparseness of text in the random patch. This is a modified version of the scheme used by Davila and Zanibbi [7], adapted for neural network loss computation. In their work, dilation was used instead of Gaussian blurring, and a global re-scaling of weights was done such that the total weight of all pixels added up to 1. Further, their scheme was used to bias sampling locations for a patch-based binarizer



(a) Input Batch of Images **(b)** Binarization Targets (inverted) **(c)** Pixel-wise Weights (inverted)

Fig. 6 A sample training mini-batch with image crops and corresponding ground truth targets and pixel-wise weights are shown here. Note that in subfigures **b** and **c**, the foreground and background colors are inverted for display purposes and black pixel border is artificially added (color figure online)

using a random forest. Figure 6 shows inputs, targets and weights for a training mini-batch.

4.3 Inference

The output of the binarizer network is noisy especially in areas of background where there was minimal pixel weight. Therefore we used Otsu binarization [32] on the preprocessed image and combined the two responses in a hysteresis fashion to obtain the final binarized output image similar to the process described by Davila and Zanibbi [7]. We use the keyframes and binary ground truth provided in the test set of the AccessMath dataset to compare isolated binarization performance. We use the metrics proposed [30] for the degraded document binarization competition dataset [33] for this comparison, and the results are presented in Table 4. For the summarization metrics, we compare with the random forest + hysteresis-based binarizer used by Davila and Zanibbi [7]. Figure 8 shows the performance of our binarizer for a sample input frame.

5 Lecture video summarization

The detected handwritten content regions obtained from detector (described in Sect. 3) are used to generate a foreground mask which is applied on the binarized full frames (see Sect. 4). The evaluation scheme of the AccessMath dataset requires the prediction of binary 'keyframes' which are compared with ground truth binary keyframes using the evaluation described in Sect. 2.3.

In order to summarize, we first need to perform temporal refinement of predictions in order to find similarity relationship of content across video frames. Broadly, if content is similar enough across frames it is considered as a set of instances of the same content. Dissimilar content occupying the same spatial region but at different timestamps indicates *conflict*, i.e., content has been erased and new content has appeared; therefore, all frames in between are candidates for points of video segmentation. We use a greedy approach to segment the video at frames that resolve most of the conflicts per cut. Video frames that contain most content within a segment are then designated as keyframes.

Since we have two levels of predictions—bounding box of content regions and connected components (CC) from binarized full frames we explore coarse-grained and fine-grained temporal refinement in our pipeline. The general refinement algorithm first proposed by Davila and Zanibbi [7] is described below, followed by individual specifications of matching criteria for bounding box and CCs, respectively, which was proposed in our prior work [20]. In this work, we propose new methods to refine predicted bounding boxes.

5.1 Temporal refinement algorithm

During the first pass, for a given matching criteria and every sampled frame in sequential order, each element is tested for a match against every other element present in all previous frames that are within time t before the current frame's timestamp ($t = 85$ s). Matched elements are treated as instances of unique objects. After finding all the unique elements from the input frames, only those that appear in more than n frames ($n = 3$) are marked as stable and are kept for further processing.

In the second pass, elements that have an overlap in space and with duration intervals that overlap or have an intermediate gap of at most 5 s are grouped into larger 'temporally stable' groups. These can be used as the final units for content summarization and further temporal processing. Other elements that only overlap in space but do not belong to the same temporal group are marked to be in *conflict*, and this information is used for summarization in order to produce temporal splits of the video that minimize the number of conflicts present on each video segment.

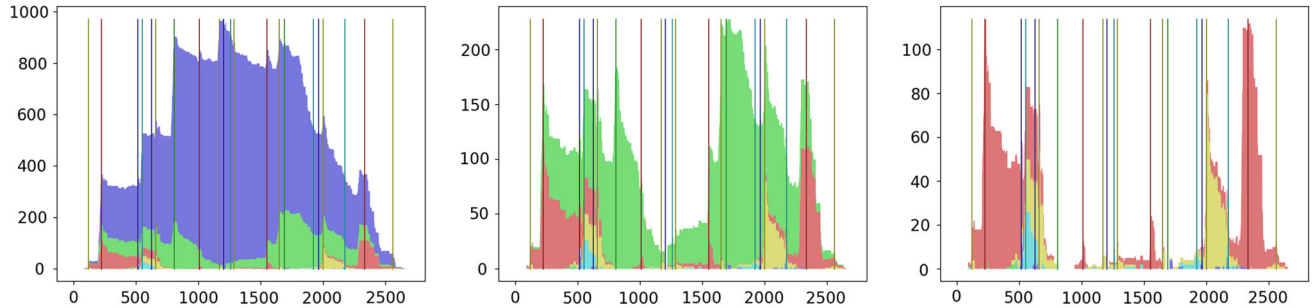
5.2 Coarse-grained temporal refinement

The output of our handwritten content detector is a set of bounding boxes. Due to false positives, variations in illumination and occlusion by the speaker, these bounding boxes might change for contiguous frames. We explore three strategies for coarse-grained temporal refinement for the detected text bounding boxes based on the application of the temporal refinement algorithm described above. The three strategies are compared in Table 5.

1. Two bounding boxes are accepted as a match if they have an IOU value above 0.5. They are merged by finding a bounding box that barely fits both of them.
2. Locality-based clustering followed by non-maximum suppression as described in Sect. 3.3 is used across both space and time to produce a set of refined bounding boxes
3. Person detection bounding boxes are used to clip or filter content boxes before performing coarse-grained refinement using strategy 2.

5.3 Fine-grained temporal refinement

Two CCs are assumed to be the same content if they have overlap of more than 92.5%. Since we currently work under the assumption that only one still camera is used, without zooming or panning, we also assume that missing CCs from intermediate frames, between their first and last known location are caused by occlusion due to the lecturer. Temporal Refinement Algorithm is then used to recover occluded content. The expected output is a set of reconstructed binary



(a) Before first pass **(b)** Before second pass **(c)** Before third pass

Fig. 7 Graphs showing the number of detected conflicts (Y-axis) against the frame index (X-axis) for one lecture video. Each sub-figure from left to right, shows a different recursion starting from first to third. The vertical lines indicate points of segmentation, chosen to greedily reduce the number of conflicts at each stage. Color coding indicates the conflicts resolved by a particular cut. Keyframes are selected as the frames with maximum content within each segment

Table 2 Quantitative evaluation of isolated handwritten content detector in our prior work and current work

Method	Framewise AVG	Pixel-wise AVG		
	BBoxes	Rec.	Prec.	*F*-meas.
Prior work [20]	12.25	81.87	76.20	76.48
Current work	12.35	88.43	68.39	75.27

video frames as though the lecturer was never occluding any of the handwritten content.

5.4 Segmentation by conflict minimization

The reconstructed video has redundant frames where text is seen being written gradually until the whiteboard is filled, erased and rewritten. In our work, we use an algorithm, originally proposed by Davila and Zanibbi [7] to produce lecture summaries. The goal is to find those 'keyframes' that completely contain all unique CCs from the previous erasure event up to the time new text appears in the same spot. We identify all conflicting CC pairs during a video segment and deploy *conflict minimization*. This greedy algorithm segments a video lecture so as to maximize the number of conflicts resolved per cut, recursively on the reconstructed video. Finally, keyframes are selected by maximizing content CCs within a segment.

A graphical depiction of the segmentation algorithm can be seen in Fig. 7, the *Y*-axis depicts number of conflicts and *X*-axis depicts frame index increasing with time. After the first pass, the video is segmented at frames indicated by dark blue vertical lines in Fig. 7a, resolving the conflicts marked by dark blue coloring, leaving behind the conflicts shown in Fig. 7b. This process is repeated until convergence, and each vertical line depicts a point of segmentation with the color indicating which conflicts are resolved by that cut. The final results can be found in Table 5.

6 Experiments

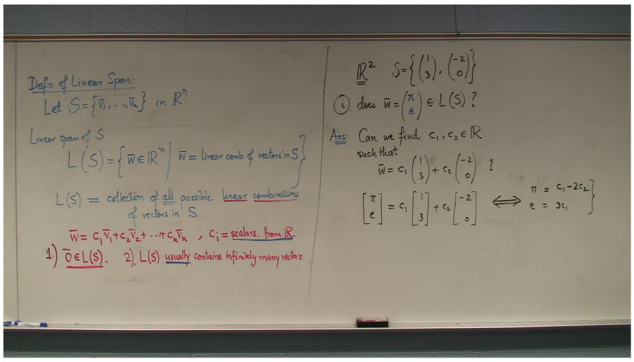
The AccessMath dataset was used to train and test our summarization methodology. Evaluation metrics used are the number of keyframes produced as well as the recall, precision and *f*-measure of keyframe content (at the level of binarized connected components) as described in Sect. 2.3.

6.1 Handwritten content detection

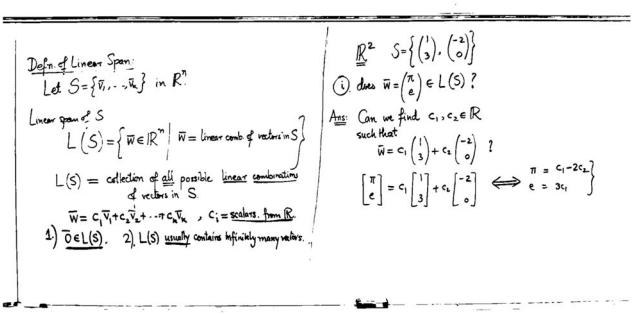
The training and test videos are annotated to provide bounding boxes [20] and binarized ground truth for keyframes [7]. We randomly split the annotated training frames into train and validation sets in the ratio 4:1. This results in about 10,000 training frames and about 2500 validation frames to train both the handwritten content detector (Sect. 3) and the binarizer (Sect. 4). We eliminate all text boxes which overlap with the bounding box of the lecturer if their area of intersection is a greater fraction of the text box area than 25%, in order to mitigate the effect of occlusion due to movement of the lecturer.

The handwritten content detector is trained for 50 epochs with a batch size of 16 using a stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.001. Each sample consists of random crops of size 512×512 with ground truth boxes adjusted accordingly. The learning rate is reduced at a constant rate of $\gamma = 0.7943$ per epoch. This ensures that the learning rate drops by a factor of approximately 0.1 for every 10 epochs. We initialize the ResNet portion of the detector with pre-trained ImageNet weights and use Kaiming-normal initialization for all other layers. For this work, we only train with the AccessMath dataset instead of pre-training on natural scene text data unlike our previous work [20]. Dense bounding box clustering thresholds θ_{loc} and θ_{nms} are both set to 0.2

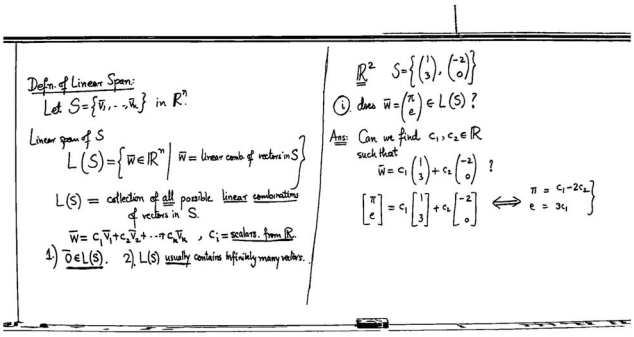
Table 2 shows the per-pixel recall, precision and *f*-measure evaluated using the ground truth bounding boxes and the



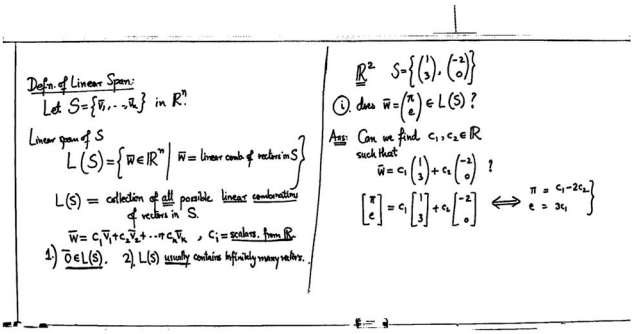
(a) Input Frame



(b) Otsu binarization after Median Background Subtraction



(c) Random Forest binarizer + Otsu using Hysteresis



(d) Our binarizer + Otsu using Hysteresis

Fig. 8 Comparison between different binarization strategies. **a** Raw frame to binarize. After computing and subtracting the background estimated with a median filtered version of the raw frame, **b** binarization performed using Otsu alone; hysteresis between Otsu output and either **c** random forest-based binarizer or **d** our proposed Deep Binarizer

predicted bounding boxes on the AccessMath test lectures. We can observe that the recall is higher, precision is lower and *f*-measure is comparable with the model in our previous work [20] which was trained on natural scene images and fine-tuned to the lecture video task. Our current model is based on dense pixel-wise predictions, whereas the previous model was restricted to predict bounding boxes that varied with respect to a fixed number of bounding boxes.

This increased flexibility of predictions allows us to capture more variety of handwritten content. However when coupled with illumination changes induced by the camera and lecturer motion across consecutive frames, the detected boxes display higher variation resulting in lower precision. Table 1 compares our current content detector with the one used in our prior work [20] on the summarization metrics keeping all other pipeline components same. It can be seen that even in the overall summarization pipeline, our current model shows the same trend of higher recall and lower precision when all other components are fixed.

The drop in precision could also be due to more false positives induced by insufficient training data. Previously many of the vertical line column separators on the whiteboard were missed which we are currently able to detect. However, the frame of the board which has similar structure is detected as text more often. False positives are seen on light backgrounds with strong darker edges like block walls. Examples of these cases can be seen in Fig. 4. Given our summarization pipeline, especially with the assumption of occlusion for reconstruction of content that disappears and reappears, a false positive is costly and we end up with oversegmentation and loss in precision in the final summaries.

6.2 Handwritten content binarization

Our binarizer is trained for 10 epochs with a batch size of 8 also using stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.01. The learning rate is reduced by a factor $\gamma = 0.1$ whenever the validation losses start to plateau. Random crops of 128×128 are used in each sample with corresponding binary targets and pixel weights computed using the procedure described in Sect. 4.2. All layers are initialized with Kaiming-normal initialization.

Figure 8 and Table 3 show qualitative and quantitative comparison of the random forest (RF)-based whiteboard binarizer [7] and our current work, respectively. Table 4 shows the comparison of the two binarizers in various configurations when run on the AccessMath test lecture keyframes, compared using the degraded document binarization com-

petition (DIBCO) [33] metrics. In particular, pseudo recall, precision and *f*-measure were introduced by the authors such that variations in thickness, while preserving connected component shapes, receive lower penalty when compared to noisy pixels that change the connected component shapes or introduce false merges/splits [30]. We can see that our binarizer performs better in terms of pseudo *F*-measure than previously used methods. The drop in precision for all methods is due to binarization of the background objects.

However, when we compare against the metrics used to evaluate summarization results, our binarization method suffers large penalties in precision and recall, as seen in Table 3. This might be due to the inherent challenges with binarization ground truth generation and evaluation. There is no agreement among human experts as to what is the ideal stroke thickness for each connected component and competitions such as DIBCO use machine generated ground truth in order to circumvent this challenge [33]. Using machine generated ground truth is debatable as well, since it encourages competitor systems to merely match the prediction of a combination of known binarization algorithms.

When the strictness of the connected component matching criteria (described in Sect. 2.3) was reduced to be more tolerant of predicted thickness (from 50 to 33%), our binarization began to display performances comparable to the random forest-based binarizer. This shows us that the drop in performance can chiefly be attributed to the thickness of predicted connected components.

6.3 Summarization methodology

Summarization metrics on the test lectures are compared in Table 5 by using the current detector and RF-based binarizer [7] to study the impact of the three coarse-grained refinement strategies (see Sect. 5.2). Table 5 also contains average summarization metrics for the entire pipeline choosing the best

Table 3 Average summarization metrics showing impact of proposed DNN-based binarizer on all test lectures in the AccessMath dataset

Method	AVG	AVG global			AVG per frame		
	N_f	*R*	*P*	*F*	*R*	*P*	*F*
Binarized							
Random Forest + Otsu [7]	296	98.96	63.83	77.60	98.71	63.1	76.99
Our Binarizer + Otsu	296	91.76	38.10	53.84	90.73	38.08	53.64
Summarized							
Random Forest + Otsu	22	95.81	92.90	94.33	95.40	92.28	93.81
Our Binarizer + Otsu	18	87.70	80.20	83.78	86.68	79.79	83.09

In all cases, hysteresis function is used to combine with Otsu's algorithm

Table 4 Comparison of random forest-based and DNN-based binarization methods with DIBCO metrics averaged over AccessMath testing keyframes

Method	PSNR	DRD	Standard			Pseudo		
			Rec.	Prec.	*F*-meas.	Rec.	Prec.	*F*-meas.
Otsu	16.36	15.88	85.46	55.93	67.02	91.89	54.54	67.95
Random forest	13.25	33.72	95.74	37.48	53.40	99.32	36.21	52.69
Random forest + Otsu	14.08	27.65	94.89	42.05	57.79	98.15	40.51	56.96
Our binarizer*	10.31	59.87	38.69	12.64	18.99	36.13	10.19	15.84
Our binarizer	13.95	25.59	98.01	41.36	57.94	98.40	37.73	54.37
Our binarizer + Otsu	15.81	15.82	95.84	52.08	67.30	96.86	48.14	64.15

Results for Otsu binarization included for reference. Background subtraction using procedure described in Sect. 4.2 is carried out in all cases except where marked with *

Table 5 Comparison of different methods of lecture video summarization by measuring recall (*R*), precision (*P*), *f*-score (*F*) and number of frames (N_f)

Method	AVG	AVG global			AVG per frame		
	N_f	*R*	*P*	*F*	*R*	*P*	*F*
AccessMath [7]	18	96.28	93.56	94.90	95.73	92.21	93.93
Prior work [20]	20	92.33	94.16	93.23	91.69	93.45	92.56
Maximum merge	21	95.80	92.88	94.32	95.40	92.44	93.90
Clustering	22	96.35	89.71	92.91	96.06	88.43	92.09
Using person	29	95.40	85.20	90.01	94.55	84.43	89.20

The second half of the table presents results for different temporal refinement strategies described in Sect. 5.2

temporal refinement strategy (Maximum Merge) and binarization technique (Random Forest + Otsu) and compared to existing results [7,20].

Variation in predicted bounding boxes makes the problem of matching content regions across frames challenging. This is amplified by the limited amount of training data. We found that the maximum merge strategy, i.e., finding largest bounding box to fit overlapping boxes gave us the best results in terms of summarization metrics. Clustering and using lecturer detections to filter content detection show high recall but are sensitive to false positives which gets propagated over time thus resulting in a loss of precision.

7 Future directions and conclusion

In the future, we would like to collect and annotate more lecture data in order to train the detector. Generating simulated whiteboard data are also a feasible step forward as this would give us a quick way of generating labeled training images. More training data will allow us to use stronger regularization, loss constraints and techniques such as hard negative mining to mitigate false positives and improve binarization. Generative Adversial models could also be explored to generate handwritten content on whiteboards.

Looking at the discrepancies between DIBCO metrics and pipeline metrics encourages a revisit of the AccessMath evaluation methodology. The current evaluation procedure matches two CCs if they have $\geq 50\%$ recall and $\geq 50\%$ precision when aligned. This punishes binarizations that may differ at a pixel scale from human provided annotations but are still visually similar without producing extra noise. For example, as seen in Fig. 8 with our binarization method, we obtained thicker traces which did not affect readability, but were penalized. Pixel-level IOU-based measures would be more suited to accommodate general binarizers especially since the training data is limited in this dataset.

Combining handwritten content CCs with fine-grained pixel mask segmentation of lecturer could lead to a more robust fine-grained temporal refinement. In order to improve coarse-grained refinement, we need a robust representation of detected area which is sensitive to degree of overlap between content and not just exact similarity. Use of long short-term memory layers and supervised neural network-based feature matching techniques such as Siamese networks and Triplet Loss Embeddings may also be considered when larger amount of training data is available in the future.

Given that one of the goals of lecture video summarization is to power content search in videos [8], annotations at a consistent text granularity should be considered. Currently equations, words, etc., maybe grouped together in the same box. Application of object tracking metrics on all unique content objects could serve as a potential evaluation scheme. Summarization in terms of 'key objects' rather than keyframes [28] may also be considered.

We have explored the potential of a generalized pipeline for still-camera whiteboard lecture video summarization. Our results using a general framework[1] are promising and are comparable to the benchmark which uses a highly specialized pipeline tailor-made for the AccessMath dataset. We analyzed the performance of modern computer vision techniques for this task which could be easily extended to more general cases of lecture videos summarization.

Acknowledgements This material was partially supported by the National Science Foundation under Grant No. 1640867 (OAC/DMR).

References

1. Banerjee, P., Bhattacharya, U., Chaudhuri, B.B.: Automatic detection of handwritten texts from video frames of lectures. In: 2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 627–632. IEEE (2014)
2. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. J. Image Video Process. **2008**, 1 (2008)
3. Castellanos, K.D.: Symbolic and Visual Retrieval of Mathematical Notation Using Formula Graph Symbol Pair Matching and Structural Alignment. Rochester Institute of Technology, Rochester (2017)
4. Choudary, C., Liu, T.: Summarization of visual content in instructional videos. IEEE Trans. Multimed. **9**(7), 1443–1455 (2007)
5. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
6. Davila, K., Agarwal, A., Gaborski, R., Zanibbi, R., Ludi, S.: Accessmath: indexing and retrieving video segments containing math expressions based on visual similarity. In: Image processing workshop (WNYIPW), 2013 IEEE Western New York, pp. 14–17. IEEE (2013)
7. Davila, K., Zanibbi, R.: Whiteboard video summarization via spatio-temporal conflict minimization. In: International Conference on Document Analysis and Recognition (ICDAR) (2017)
8. Davila, K., Zanibbi, R.: Visual search engine for handwritten and typeset math in lecture videos and latex notes. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE (2018)
9. Dickson, P.E., Adrion, W.R., Hanson, A.R.: Whiteboard content extraction and analysis for the classroom environment. In: 10th IEEE International Symposium on Multimedia, 2008. ISM 2008, pp. 702–707. IEEE (2008)
10. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2963–2970. IEEE (2010)
11. Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal visual object classes challenge: a retrospective. Int. J. Comput. Vis. **111**(1), 98–136 (2015)
12. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2315–2324 (2016)

[1] https://github.com/bhargavaurala/accessmath-ijdar

13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
14. Huang, L., Yang, Y., Deng, Y., Yu, Y.: Densebox: unifying landmark localization with end to end object detection (2015). arXiv preprint arXiv:1509.04874
15. Jia, W., Sun, L., Zhong, Z., Huo, Q.: A CNN-based approach to detecting text from images of whiteboards and handwritten notes. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE (2018)
16. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., et al.: ICDAR 2015 competition on robust reading. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 1156–1160. IEEE (2015)
17. Khosla, A., Hamid, R., Lin, C.J., Sundaresan, N.: Large-scale video summarization using web-image priors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2698–2705 (2013)
18. Kim, G., Sigal, L., Xing, E.P.: Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4225–4232 (2014)
19. Kim, K.H., Hong, S., Roh, B., Cheon, Y., Park, M.: PVANet: deep but lightweight neural networks for real-time object detection (2016). arXiv preprint arXiv:1608.08021
20. Kota, B.U., Davila, K., Stone, A., Setlur, S., Govindaraju, V.: Automated detection of handwritten whiteboard content in lecture videos for summarization. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 19–24. IEEE (2018)
21. Lee, G.C., Yeh, F.H., Chen, Y.J., Chang, T.K.: Robust handwriting extraction and lecture video summarization. Multimed. Tools Appl. **76**(5), 7067–7085 (2017)
22. Lee, Y.J., Ghosh, J., Grauman, K.: Discovering important people and objects for egocentric video summarization. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1346–1353. IEEE (2012)
23. Liao, M., Shi, B., Bai, X., Wang, X., Liu, W.: Textboxes: a fast text detector with a single deep neural network. In: AAAI, pp. 4161–4167 (2017)
24. Lin, T.Y., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR, vol. 1, p. 4 (2017)
25. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37. Springer (2016)
26. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
27. Meng, G., Yuan, K., Wu, Y., Xiang, S., Pan, C.: Deep networks for degraded document image binarization through pyramid reconstruction. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 727–732. IEEE (2017)
28. Meng, J., Wang, H., Yuan, J., Tan, Y.P.: From keyframes to key objects: video summarization by representative object proposal selection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1039–1048 (2016)
29. Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: Asian Conference on Computer Vision, pp. 770–783. Springer (2010)
30. Ntirogiannis, K., Gatos, B., Pratikakis, I.: Performance evaluation methodology for historical document image binarization. IEEE Trans. Image Process. **22**(2), 595–609 (2013)
31. Onishi, M., Izumi, M., Fukunaga, K.: Blackboard segmentation using video image of lecture and its applications. In: Proceedings of 15th International Conference on Pattern Recognition, 2000, vol. 4, pp. 615–618. IEEE (2000)
32. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)
33. Pratikakis, I., Zagoris, K., Barlas, G., Gatos, B.: ICFHR2016 handwritten document image binarization contest (H-DIBCO 2016). In: 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 619–623. IEEE (2016)
34. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
35. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
36. Shah, R.R., Yu, Y., Shaikh, A.D., Tang, S., Zimmermann, R.: Atlas: automatic temporal segmentation and annotation of lecture videos based on modelling transition time. In: Proceedings of the 22nd ACM International Conference on Multimedia, pp. 209–212. ACM (2014)
37. Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Cardoso, M.J.: Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp. 240–248. Springer (2017)
38. Tang, L., Kender, J.R.: A unified text extraction method for instructional videos. In: IEEE International Conference on Image Processing, 2005. ICIP 2005, vol. 3, pp. III–1216. IEEE (2005)
39. Tian, Z., Huang, W., He, T., He, P., Qiao, Y.: Detecting text in natural image with connectionist text proposal network. In: European Conference on Computer Vision, pp. 56–72. Springer (2016)
40. Vajda, S., Rothacker, L., Fink, G.A.: A method for camera-based interactive whiteboard reading. In: International Workshop on Camera-Based Document Analysis and Recognition, pp. 112–125. Springer (2011)
41. Veit, A., Matera, T., Neumann, L., Matas, J., Belongie, S.: Coco-text: dataset and benchmark for text detection and recognition in natural images (2016). arXiv preprint arXiv:1601.07140
42. Ye, Q., Doermann, D.: Text detection and recognition in imagery: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **37**(7), 1480–1500 (2015)
43. Yin, X.C., Zuo, Z.Y., Tian, S., Liu, C.L.: Text detection, tracking and recognition in video: a comprehensive survey. IEEE Trans. Image Process. **25**(6), 2752–2773 (2016)
44. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: an efficient and accurate scene text detector. In: Proceedings of CVPR, pp. 2642–2651 (2017)
45. Zhu, Y., Yao, C., Bai, X.: Scene text detection and recognition: recent advances and future trends. Front. Comput. Sci. **10**(1), 19–36 (2016)