SlidesScriber: An Efficient and Robust System for Automated Extraction of Presentation Slides from Teaching Videos

1st Anonymous Student
The School of Computer Science and Technology
Xi'an Jiaotong University
Xi'an, China
anonymous email address

***Abstract*—SlidesScriber presents a novel solution for the automated extraction of presentation slides from teaching videos in university recording platform. The system offers efficiency and robustness in identifying and extracting slides. Leveraging image processing and sequential outlier detection algorithms, SlidesScriber accurately samples video frames to detect and extract slides, facilitating efficient content retrieval. Through massive uses by students, the system demonstrates robust performance across various video recording environments, providing a valuable tool for learners in accessing and utilizing educational resources effectively.**

***Index Terms*—automated extraction, image processing, sequential outlier detection, content retrieval**

I. Introduction

In recent years, the use of digital learning platforms [1] has led to an exponential increase in the availability of educational resources, particularly in the form of teaching videos captured during university lectures. These videos often contain valuable content in the form of presentation slides [2], which serve as key reference materials for students. However, manually extracting these slides from lengthy video recordings before final examination can be time-consuming and labor-intensive, presenting a significant challenge for learners in university.

To address this challenge, we present SlidesScriber, a cutting-edge system designed for the automated extraction of presentation slides from teaching videos within university recording platforms. SlidesScriber offers efficiency and robustness, leveraging image processing techniques and sequential outlier detection algorithms to accurately identify and extract slides from video frames. By intelligently sampling frames and detecting sequential outliers indicative of slide transitions, SlidesScriber ensures precise extraction with minor computational overhead.

In this paper, we provide an overview of the SlidesScriber system, detailing its architecture, underlying algorithms, implementation, and case study evaluation. We demonstrate the system's effectiveness through massive uses across various video recording environments, showcasing its ability to reliably extract slides from diverse video content. Ultimately, SlidesScriber represents a valuable tool for learners, enabling seamless access to presentation slides within teaching videos and enhancing the efficiency of educational resource utilization.

II. Related Works

A. Computer vision and image processing

Computer vision [3] and image processing [4] are interdisciplinary fields at the intersection of computer science, artificial intelligence, and engineering, aiming to enable machines to interpret and understand visual information from the real world. These fields encompass a broad range of techniques and algorithms designed to extract meaningful insights from digital images and video streams.

At its core, computer vision involves the development of algorithms and systems that enable computers to perceive, analyze, and interpret visual data much like humans do. This includes tasks such as object detection, recognition, tracking, image segmentation, and scene understanding. By harnessing the power of computer vision, machines can autonomously extract valuable information from visual inputs, enabling applications in various domains. Image processing, on the other hand, focuses on the manipulation and enhancement of digital images to improve their quality, extract useful features, or facilitate subsequent analysis. Image processing techniques encompass operations such as filtering, transformation, feature extraction, and noise reduction. These operations are applied to raw image data to achieve specific objectives, such as enhancing image contrast, removing noise, detecting edges, or segmenting objects of interest.

In the context of automated slide extraction from teaching videos, computer vision and image processing techniques play a vital role in preprocessing to enhance the robustness. By analyzing video frames, these techniques enable automated systems to transform critic information to the following outlier detection pipeline, facilitating efficient content retrieval for educational purposes.

State of the art techniques in these domains involve utilizing deep learning architectures such as convolutional neural networks [5] (CNNs), which great at capturing semantic features within images. However, for our particular application context,

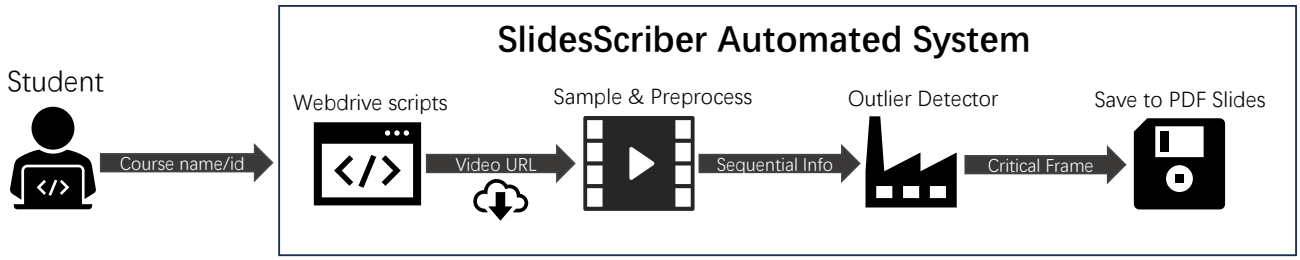


Fig. 1. **Architecture workflow overview** The architecture of our system for extracting slides from teaching videos from university recordings is designed to efficiently process student inputs, leverage WebDriver scripts to navigate through web pages, extract video URLs, sample frames, preprocess data to enhance robustness, detect sequential information, identify outliers as critical frames, and save them as PDF slides.

where semantic representation isn't essential, using these networks would introduce unnecessary computational overhead.

B. Sequential outlier detection

Sequential outlier detection algorithms [6] are a class of techniques used to identify anomalies or outliers within a sequence of data points ordered by time or another sequential attribute. These algorithms are particularly useful in various domains, including finance, manufacturing, cybersecurity, and environmental monitoring, where detecting unusual patterns or deviations from expected behavior is crucial for decision-making and anomaly detection.

The primary objective of sequential outlier detection algorithms is to distinguish abnormal data points from the normal or expected ones within a time-ordered sequence. Unlike traditional outlier detection methods that analyze static datasets, sequential outlier detection techniques consider the temporal ordering of data points, recognizing that anomalies may manifest as temporal deviations or irregularities.

Sequential outlier detection algorithms typically operate by analyzing the sequential relationships and temporal dependencies among data points. They may utilize statistical models, machine learning approaches, or rule-based techniques to identify outliers based on various factors such as trend analysis, seasonality, periodicity, or sudden deviations from historical patterns.

One common approach is based on time series analysis, where statistical measures such as mean, median, standard deviation, or autoregressive models are employed to characterize the underlying temporal structure of the data. Deviations from expected values or patterns are then flagged as potential outliers. Another approach involves the use of dynamic thresholding techniques, where thresholds for anomaly detection are adaptively adjusted based on the evolving characteristics of the data stream. This allows the algorithm to adapt to changes in data distribution or behavior over time, enhancing its sensitivity to outliers while minimizing false positives.

Such detection algorithm has been employed in our SlidesScriber system to detect slide transitions within video sequences. By analyzing temporal changes in visual frames, these algorithms can identify instances where new slides are displayed, enabling precise extraction of slide content. We made numerous adjustments to align with our application's scenario.

C. WebDriver

WebDriver is a powerful tool for automating web browsers [7]. It provides a programming interface for controlling and interacting with web browsers programmatically. Developed as an open-source project, WebDriver has gained widespread adoption in web testing, web scraping, and web automation tasks.

At its core, WebDriver allows developers to write code that simulates user interactions with web browsers. This includes actions such as clicking links, filling out forms, submitting data, and navigating through web pages. WebDriver supports various programming languages such as Python, Java, and JavaScript, making it accessible to developers across different platforms. One of the key features of WebDriver is its cross-browser compatibility. It provides drivers for popular web browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, and Opera, allowing developers to automate tasks across multiple browsers seamlessly.

WebDriver operates by sending commands to a browser's native automation engine using a protocol called the WebDriver protocol. This protocol defines a standardized way for interacting with web browsers, ensuring consistency and interoperability across different implementations. With WebDriver, developers can create automated tests to verify the functionality and behavior of web applications across different browsers and platforms. It also enables the creation of web scraping scripts to extract data from web pages and perform repetitive tasks efficiently.

In our system, tools such as WebDriver play a crucial role in automating the extraction of video URLs from digital learning platforms.

III. Architecture

Our SlidesScriber system, depicted in Figure 1, functions highly autonomously, requiring only the course name or identifiers specified by students. Upon input, the system generates PDF-format slides corresponding to the desired course.

A. Modules

- Input Module: Students provide the course name or ID, which serves as the only input to the automated system.
- WebDriver Scripts: WebDriver scripts are employed to automate web browsing tasks. These scripts navigate through digital learning platforms, locate the specified course, and extract the URLs of teaching videos(the slides part) associated with the course.
- Sample Frames: Video content is sampled to extract frames according to it's playing rates, which serve as the basis for slide extraction. A queue data structure facilitates the management of sampled frames as they traverse through the preprocessing and analysis pipeline. The queue operates within a multi-threaded producer-consumer model, ensuring efficient handling of frame extracting and processing tasks.
- Preprocessing Module: Feedback from earlier users indicates that certain recordings from the platform contain background noise, prompting the need for noise reduction techniques. Thus the sampled frames undergo preprocessing to enhance their robustness to noise.
- Sequential Information Analysis: Processed frames are analyzed to identify sequential information, such as temporal patterns within the video content.
- Outlier Detection: An efficient outlier detection algorithm is applied to the sequential information to identify anomalous frames or critical points as slide transitions within the sampled video stream.
- Slides Creation: The critical frames are compiled into pdf format slides, preserving the visual content and structure of the original presentation.

While designing an efficient outlier detection algorithm tailored to our specific task remains the most important, the core focus of our system development lies in the evolution of this algorithm. We will propose the foundational framework for its advancement.

IV. Outlier Detection Algorithm Design

A. Outliers and Two Types of Errors

In a teaching video, where most of frames are stationary, outliers identified through sequential outlier detection are primarily characterized as slide transitions. These transitions represent significant temporal shifts within the video content, marking the transition from one slide to another. There are two types of errors are encountered by the detection algorithm: false positives and false negatives.

- **False Positives**: False positives occur when the algorithm incorrectly identifies a normal data point as an outlier. In the context of our system for extracting slides from teaching videos, the algorithm mistakenly flags a frame as a transition frame, which lead to unnecessary alerts and duplicated pages in the resulting slides.
- **False Negatives**: False negatives occur when the algorithm fails to detect an actual outlier in the dataset. In other words, the output will miss certain pages compared with original slides.

Minimizing both false positives and false negatives is crucial for precise outlier detection, with a particular emphasis on mitigating **False Negatives** due to their significance in our task. Our algorithm is specifically designed to achieve a zero **False Negatives** rate while simultaneously striving to minimize false positives. This approach ensures that no important slide transitions or critical events are missed.

B. A Naive Approach

The most straightforward approach to identifying transition frames is by computing the Euclidean Distance between two consecutive sampled frames.

$$\begin{aligned}&\text{Distance}(Frame_i, Frame_{i+1})\\&= \sqrt{\sum_j (Frame_i[j] - Frame_{i+1}[j])^2}\end{aligned}$$

Algorithm 1 Distance Based Detection

```
procedure DetectTransitions(frames, threshold)
    transitions ← ∅
    for i ← 1 to len(frames) − 1 do
        distance ← Distance(frames[i], frames[i + 1])
        if distance > threshold then
            transitions.append(frames[i])
        end if
    end for
    return transitions
end procedure
```

However, this method presents two notable **drawbacks**. Firstly, it needs manually setting a threshold value to determine transitions, a parameter that can vary significantly across different videos. Secondly, when adjusting the threshold to meet our zero false negatives constraint, this method often results in a high number of false positives.

C. Bounded Metric and Critical Observation

We have turned to calculate the cosine similarity between two consecutive frames to circumvent the unpredictability of the distance threshold. Through observation, we have found that the angle associated with the cosine value is significantly more distinguishable between normal and transition frames.

$$\theta = \arccos \frac{Frame_i \cdot Frame_{i+1}}{\|Frame_i\| \|Frame_{i+1}\|}$$

We opt to compute the Euclidean angle directly instead of utilizing their visual representation through a vision encoder for several reasons:

1) With numerous videos per course and a considerable number of frames generated per video, the computational overhead grows dramatically upon introducing a vision encoder.

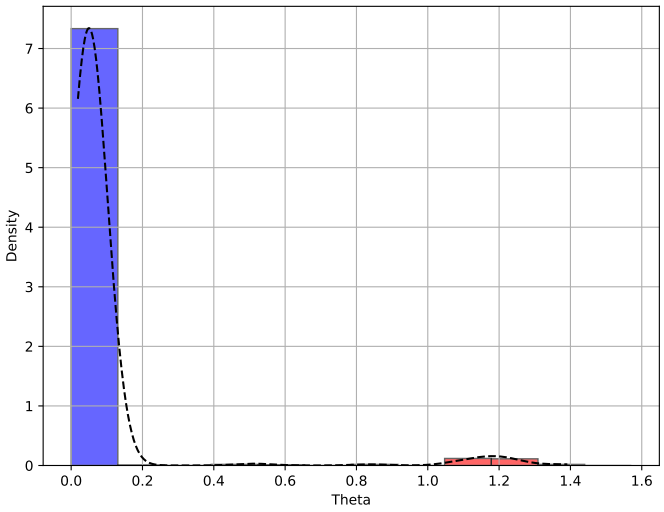


Fig. 2. Similarity angle θ distribution in a case study

2) Teaching slides primarily change pixel-level rather than semantic understanding. Consequently, visual representations derived from a vision encoder do not align with our requirements.
3) As illustrated in our case study depicted in Figure 2, our straightforward metric proves to be highly effective and efficient for our task.

This approach not only addresses computational concerns but also ensures alignment with the specific objectives of our task, ultimately enhancing efficiency and effectiveness in transition frames detection.

Here we show a case study. By examining the distribution of these theta values in Figure 2, we gain a deeper understanding of the variability in frame-to-frame similarity within the video content. This analysis highlights the effectiveness of cosine similarity as a metric for distinguishing between normal frames and transition frames in teaching videos for three key reasons:

1) **Theta bounded between** 0 **and** $\frac{\pi}{2}$: The nature range of angle provides a clear normed boundary that doesn't change significantly among different video input.
2) **The observation of distinct distributions**: The case study reveals the presence of two obvious peaks in the distribution of theta values. Normal frames predominantly cluster in the blue region, corresponding to smaller theta values, indicating higher similarity between consecutive frames. In contrast, transition frames are concentrated in the red region, characterized by larger theta values, signifying a greater deviation or change between consecutive frames. This clear boundary in the distributions underscores the efficacy of cosine similarity in distinguishing between normal and transition frames within teaching videos.
3) **Strong universality** While this case study provides valuable insights, its applicability extends universally to various teaching videos despite potential differences in content or context. More specifically, the two peaks distinguishability remains consistent across diverse video datasets in practice.

D. Temporal Information Adaption

This task extends beyond outlier detection within stationary distributions. We must also leverage temporal information to address scenario changes, a phenomenon observed in certain videos that the previous case study did not reveal. The previous case study retains its value as stationary properties remain within a time window. Therefore, we can apply a low-pass filter to capture the temporal scope information effectively. The Kalman filter [8] is a recursive mathematical algorithm used to estimate the state of a dynamic system from a series of noisy measurements over time. We simiplify this algorithm to estimate the dynamic angle value and its variability.

Subsequently, we utilize the estimated temporal angle mean and variability values to implement the Dynamic Boundary Estimation Based Detection Algorithm (see Algorithm:2) to find the boundary between normal frames and transition frames along with temporal scenario dynamically. Transition frames are flagged when the angle value deviate beyond a predefined number of variability from the mean.

In order to mitigate the sharp impact of outlier angle values on our estimation process, we apply a damping mechanism to control the influence of exceptionally large outlier angle values during the update phase. This damping is achieved through the utilization of our introduced *f-average* method.

E. f-average design

Definition 1. *f-average: Let $f : \mathbb{D} \to X$ be a monotonic function that maps a subset of real numbers to another space X. It is easily known that f is reversible within its monotonic domain. The f-average of a set of real numbers $a_1, a_2, \ldots, a_n$ is defined as:*

$$\bar{a}_f = f^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} f(a_i)\right)$$

where f^{-1} is the inverse function of f, and n is the number of elements in the set.

Example: Let $f(x) = x^2$ be a function that maps positive real numbers to their squares. The f-average of the numbers $1, 2, 3$ is calculated as:

$$\bar{a}_f = f^{-1}\left(\frac{1}{3}\sum_{i=1}^{3} f(a_i)\right) = \sqrt{\frac{1}{3}\cdot(1^2+2^2+3^2)}$$

Theorem 1. *If f is twice differentiable, the f-average is greater than the arithmetic average if and only if sign$(f'(x))\cdot f''(x) > 0$.*

Proof. First, the f-average is greater than the arithmetic average means:

$$\bar{a}_f > \bar{a}$$

which means

$$f^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} f(a_i)\right) > \frac{1}{n}\sum_{i=1}^{n} a_i$$

by their definition.

If $\text{sign}(f'(x)) > 0$, indicating that f is increasing, then we obtain

$$\frac{1}{n}\sum_{i=1}^{n} f(a_i) > f\left(\frac{1}{n}\sum_{i=1}^{n} a_i\right)$$

This demonstrates that f is convex. Given that f is twice differentiable, $f''(x) > 0$.

Conversely and similarly, we can get when $\text{sign}(f'(x)) < 0$, $f''(x) < 0$.

Hence, we conclude that the f-average is greater than the arithmetic average if and only if $\text{sign}(f'(x)) \cdot f''(x) > 0$. □

Theorem 2. *If f_1 and f_2 are both twice differentiable, then the f_1-average is greater than the f_2-average if and only if $sign(f_1'(x)) \cdot sign(h'(x)) \cdot h''(x) < 0$, where $h(x) = f_2 \circ f_1^{-1}$.*

Proof.

$$\bar{a}_{f_1} > \bar{a}_{f_2}$$

means

$$f_1^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} f_1(a_i)\right) > f_2^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} f_2(a_i)\right)$$

by definition.

If $\text{sign}(f_1'(x)) > 0$, indicating that f_1 is increasing, then we obtain

$$\frac{1}{n}\sum_{i=1}^{n} f_1(a_i) > f_1 \circ f_2^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} f_2 \circ f_1^{-1}(f_1(a_i))\right)$$

We have

$$\frac{1}{n}\sum_{i=1}^{n} b_i > h^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} h(b_i)\right)$$

where $b_i = f_1(a_i)$ and $h(x) = f_2 \circ f_1^{-1}$, this shows $\text{sign}(h'(x)) \cdot h''(x) < 0$ by Theorem 1.

Conversely and similarly, we can get when $\text{sign}(f_1'(x)) < 0$, $\text{sign}(h'(x)) \cdot h''(x) > 0$.

Hence, we conclude that the f_1-average is greater than the f_2-average if and only if $\text{sign}(f_1'(x)) \cdot \text{sign}(h'(x)) \cdot h''(x) < 0$. □

In our damping mechanism, we choose $q(x) = ln(x+1)$, which means $\bar{a}_q = \prod_{i=1}^{n}(a_i+1)^{\frac{1}{n}} - 1$ by definition. Additionally, we observe that if $g(x) = ln(x)$, then the *g-average* represents the geometric mean.

Theorem 3. *$\bar{a}_q$ is between geometric mean and arithmetic average.*

Proof. $q'(x) = \frac{1}{x+1}$, $q''(x) = -\frac{1}{(x+1)^2}$, so $\text{sign}(q'(x)) \cdot q''(x) < 0$, which means $\bar{a}_q$ is less than arithmetic average by Theorem 1.

$h(x) = g \circ q^{-1} = ln(e^x - 1)$, $h'(x) = \frac{e^x}{e^x-1} = 1 + \frac{1}{e^x-1}$, $h''(x) = -\frac{e^x}{(e^x-1)^2}$, so $\text{sign}(q'(x)) \cdot \text{sign}(h'(x)) \cdot h''(x) < 0$, which means $\bar{a}_q$ is great than geometric average by Theorem 2.

Hence, we conclude that $\bar{a}_q$ is between geometric mean and arithmetic average. □

We select $\bar{a}_q$ as the damping mechanism for mitigating the sharp impact of outlier angle values on our estimation process, guided by the following considerations:

1) The use of $\bar{a}_q$ provides a smooth and gradual adjustment, which helps to prevent abrupt fluctuations in our estimated angle values and its variability, resulting in smoother and more consistent estimates that better reflect the underlying boundary of the two peaks distribution.
2) Upon examining Figure 2, it becomes evident that the arithmetic average is too large and is prone to cause **false negatives** which we desire none, while the geometric average experiences a sharp decline particularly when one of the elements approaches zero. We attribute this behavior to the domain of $g^{-1}(x)$, which spans $(0, +\infty)$. In contrast, the function $q(x)$ lies between the arithmetic and geometric means by Theorem 3, with its inverse function $q^{-1}(x)$ having a domain of $(-1, +\infty)$. Consequently, the angle data tends to be avoid from the lower extreme, thereby giving a more stable update process.

F. The Improved Algorithm

By combining all the aforementioned considerations, we propose the following algorithm for efficient and robust temporal outlier detection.

Algorithm 2 Dynamic Angle Boundary Estimation Based Detection

procedure DETECTTRANSITIONS
  transitions $\leftarrow \emptyset$
  Initialize angle estimate $\hat{\theta}_0$ and variability estimate $\hat{d}_0$
  Initialize constant α_0, α_1 and n
  Get sampled frame F_0 from producer Queue
  for $t \leftarrow 1$ **to** N **do**
    Get sampled frame F_t from producer Queue
    Compute new angle: $\theta_t \leftarrow \Theta(F_{t-1}, F_t)$
    Compute new variability: $d_t \leftarrow \|\theta_t - \hat{\theta}_{t-1}\|$
    if $\theta_t > \hat{\theta}_{t-1} + n \cdot \hat{d}_{t-1}$ **then**
      transitions.append(F_t)
      Damping outlier's angle: $\theta_t \leftarrow \bar{a}_q(\theta_t, \hat{\theta}_{t-1})$
      Damping outlier's variability: $d_t \leftarrow \bar{a}_q(d_t, \hat{d}_{t-1})$
    end if
    Update angle estimate: $\hat{\theta}_t \leftarrow \hat{\theta}_{t-1} + \alpha_0(\theta_t - \hat{\theta}_{t-1})$
    Update variability estimate: $\hat{d}_t \leftarrow \hat{d}_{t-1} + \alpha_1(d_t - \hat{d}_{t-1})$
  end for
  return transitions
end procedure

While there are parameters yet to be specified during initialization, they can be hardcoded, and the algorithm is adaptive enough to videos encountered in practice.

We apply both algorithms to the case study video, and the Table I presents the respective numbers of false positives generated by each.

TABLE I
ALGORITHM EVALUATION ON CASE STUDY VIDEO

Algorithm	**# false positives**
Distance	9
Dynamic Angle Boundary Estimation	0

In this case study there are 47 transitions in the ground truth slides, Algorithm 2 showcases remarkable effectiveness by providing entirely accurate results, with notably fewer false positives compared to Algorithm 1 when operating under the constraint of zero false negatives.

G. Preprocessing & Robustness

Our improved algorithm demonstrates practical utility already; however, user feedback has highlighted instances of background color noise in certain videos, often stemming from camera hardware damage. To further improve our system be more robust against such challenges, we have refined our preprocessing pipeline.

We employ a simple yet effective technique: initially, we compute the histogram of frames and identify the predominant color as the background value. Subsequently, we reassign these background pixels by averaging their values, thereby filtered noise. Additionally, alternative approaches, such as specific designed histogram refinement [9] to shrink background neighbourhood noise, offer both efficiency and efficacy in noise reduction.

FUTURE WORK

We profile the time usage of our system and find the majority of time is used in network transformation, which shows the efficiency of our detection algorithm, while we can still optimize SlidesScriber system in network part, such as Caching and Memoization [10] for users in the same network.

ACKNOWLEDGMENT

Thanks to users of SlidesScriber System, feedbacks from them provide valuable information to iterate the system to be more robust and usable.

REFERENCES

[1] xjtuClass, "xjtuclass," http://class.xjtu.edu.cn/, 2024, accessed: March 14, 2024.
[2] Microsoft Corporation, "Microsoft PowerPoint," Software, 2024, accessed: March 14, 2024. [Online]. Available: https://www.microsoft.com/en-us/microsoft-365/powerpoint
[3] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.
[4] K. R. Castleman, *Digital image processing*. Prentice Hall Press, 1996.
[5] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, pp. 611–629, 2018.
[6] W. Lu, Y. Cheng, C. Xiao, S. Chang, S. Huang, B. Liang, and T. Huang, "Unsupervised sequential outlier detection with deep architectures," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4321–4330, 2017.
[7] E. Vila, G. Novakova, and D. Todorova, "Automation testing framework for web applications with selenium webdriver: Opportunities and threats," in *Proceedings of the International Conference on Advances in Image Processing*, 2017, pp. 144–150.
[8] M. Khodarahmi and V. Maihami, "A review on kalman filter models," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 727–747, 2023.
[9] G. Pass and R. Zabih, "Histogram refinement for content-based image retrieval," in *Proceedings third IEEE workshop on applications of computer vision. WACV'96*. IEEE, 1996, pp. 96–102.
[10] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*, 2012, pp. 55–60.

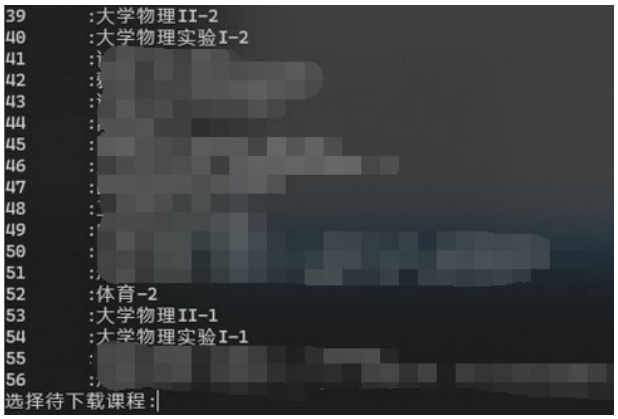
V. APPENDIX

A. Part of the result of Algorithm 2 on the case study

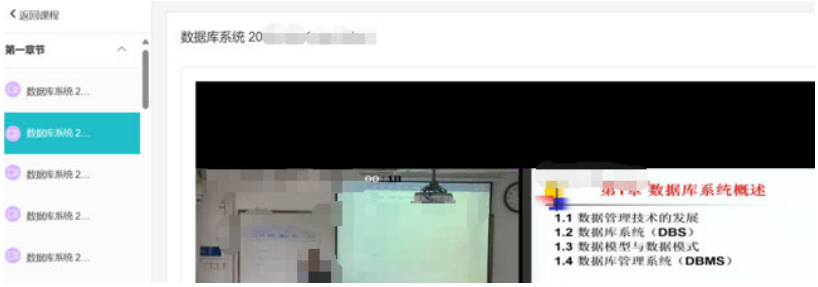
We show the user input interface, WebDriver display and part of the result of Algorithm 2 on the case study in appendix. The data of appendix is from one of our anonymous user.

Appendix

User input:



WebDriver:



Part of the result slides: