Lecture Video Info Retrieval Application

Yoke Kai Wen 叶凯雯
2020280598

Contents

1.Introduction

2.Demo

3.System Design

4.Evaluation

5.Conclusion and future work

1. Introduction

1. Introduction - motivation

- **Lectures are increasingly delivered online**
  - improved technology (recording equipment, video compression techniques, high-speed networks),
  - ongoing global pandemic
- **Lecture videos are long (>1hr)**
  - takes time to manually locate the segment that includes the requested information
- **Therefore, it is useful to create an application to allow users to conveniently locate the video segments relevant to their search query**

1. Introduction – background

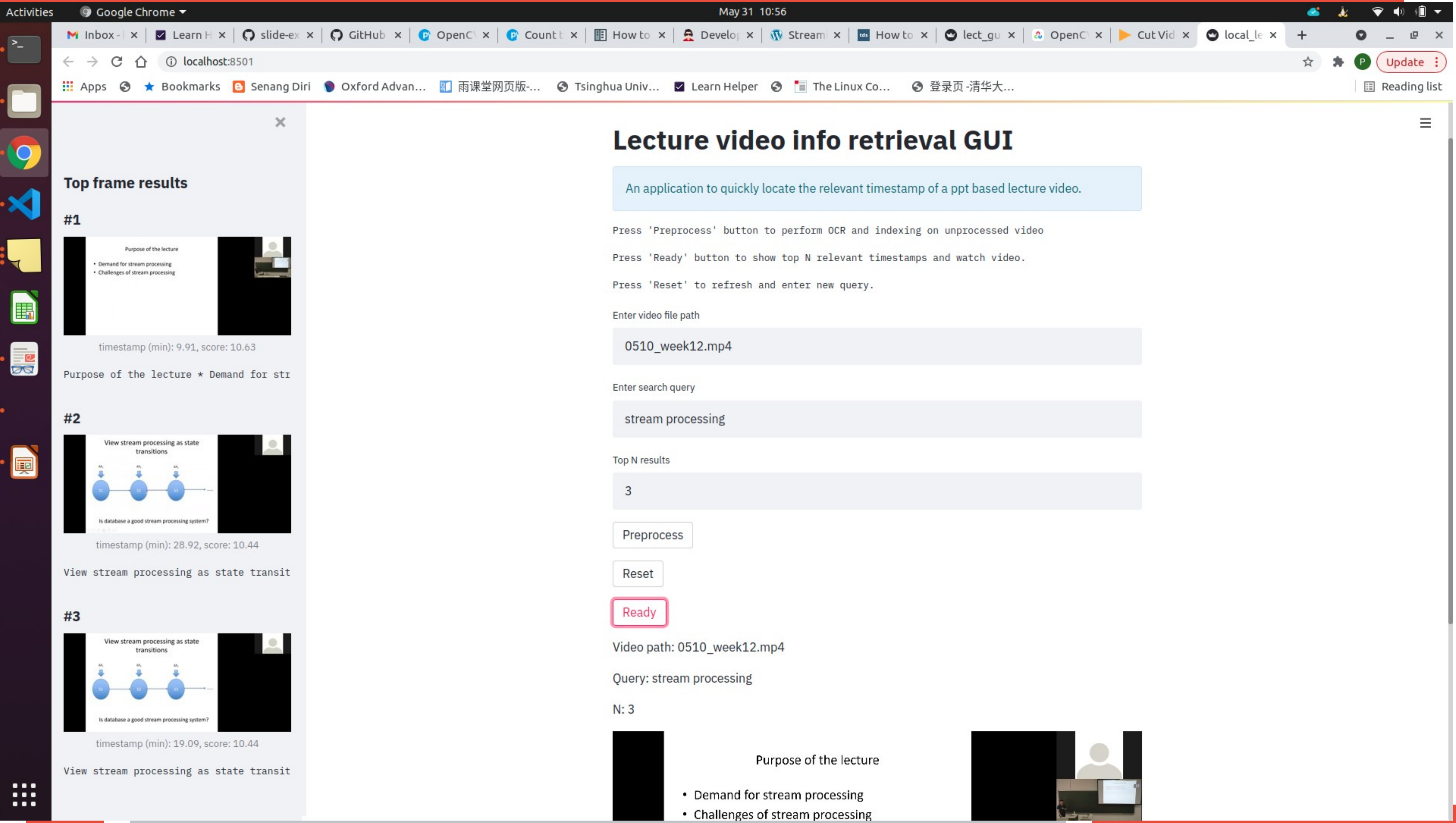
1. **Content based lecture video retrieval**
    1. Speech (ASR)
    2. Video text (OCR)
    3. Visual feature extraction
    4. Metadata: title, genre, brief description etc
2. **Different types of lecture recordings**
    1. PPT based screen sharing
    2. Whiteboard
    3. Multi-scene: screen + speaker + whiteboard

1. Introduction – project scope

1. Focus only on textual information in video frames
2. Assumes lecture videos are PPT screen sharings

2. Demo

2. Demo: http://54.169.99.59:8501/



Top frame results

#1

timestamp (min): 9.91, score: 10.63

```
Purpose of the lecture • Demand for str
```

#2

timestamp (min): 28.92, score: 10.44

```
View stream processing as state transit
```

#3

timestamp (min): 19.09, score: 10.44

```
View stream processing as state transit
```

Lecture video info retrieval GUI

An application to quickly locate the relevant timestamp of a ppt based lecture video.

```
Press 'Preprocess' button to perform OCR and indexing on unprocessed video

Press 'Ready' button to show top N relevant timestamps and watch video.

Press 'Reset' to refresh and enter new query.
```

Enter video file path: 0510_week12.mp4

Enter search query: stream processing

Top N results: 3

Video path: 0510_week12.mp4

Query: stream processing

N: 3

Purpose of the lecture

- Demand for stream processing
- Challenges of stream processing

2. Demo – key points

1. Enter search query and show results on preprocessed video
2. Play video at timestamps
3. Offer choices: BM25, tf-idf, word frequency etc
4. Preprocessing: OCR + Indexing (slow)
    1. Show demo video
    2. Show files saved in directory

3. System Design

3. System Design - Overview

1. **Video analysis**
    1. Extract video segment keyframes
    2. Perform OCR on extracted keyframes
2. **Indexing and ranking**
    1. Index keyframes based on extracted text
    2. Rank and score keyframes based on search query
3. **GUI design**
    1. Design: Streamlit
    2. Launching: AWS-EC2 instance

3. System Design – video analysis

1. **Video slide keyframe extraction**
    1. Typical lecture video: 25 fps, 1.5h --> 135k frames
    2. Segment video into representative keyframes
    3. Assume each lecture slide represent one video segment
2. **OCR**
    1. OpenCV
    2. Tesseract v4 supports deep-learning based OCR, very accurate
        1. Localises and recognises text

3. System Design – video keyframe extraction

1. **Focus on centre area of the frame**

   centreFrame = frame[int(h/4): int(3*h/4), int(w/4): int(3*w/4), :]

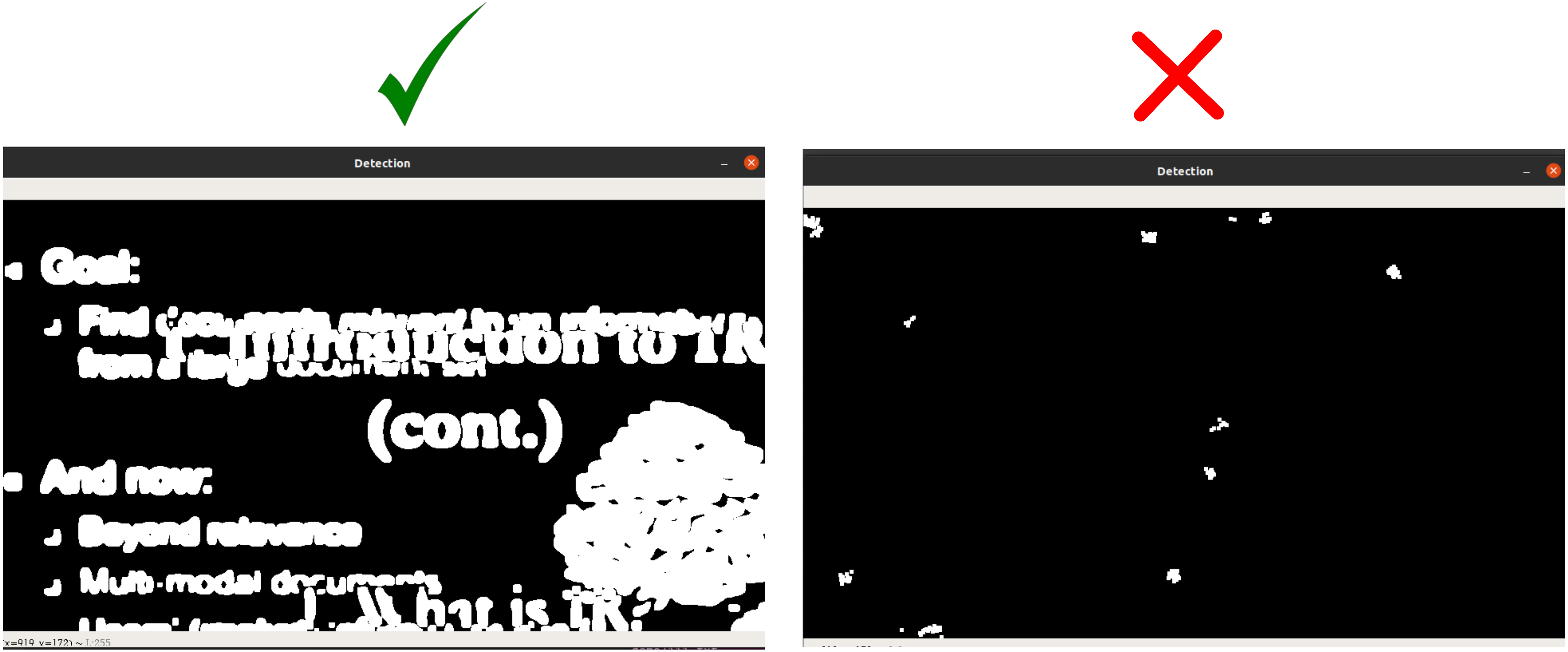
2. **Take frame pixel difference**

   frameDiff = currFrame - prevKeyFrame

3. **Apply blurring, thresholding, and contour detection on frameDiff**

   1. Only contours with area > minArea are counted
   2. If numContours > minContours && slide has not changed for 3s --> set currFrame as newKeyFrame

3. System Design – video keyframe extraction

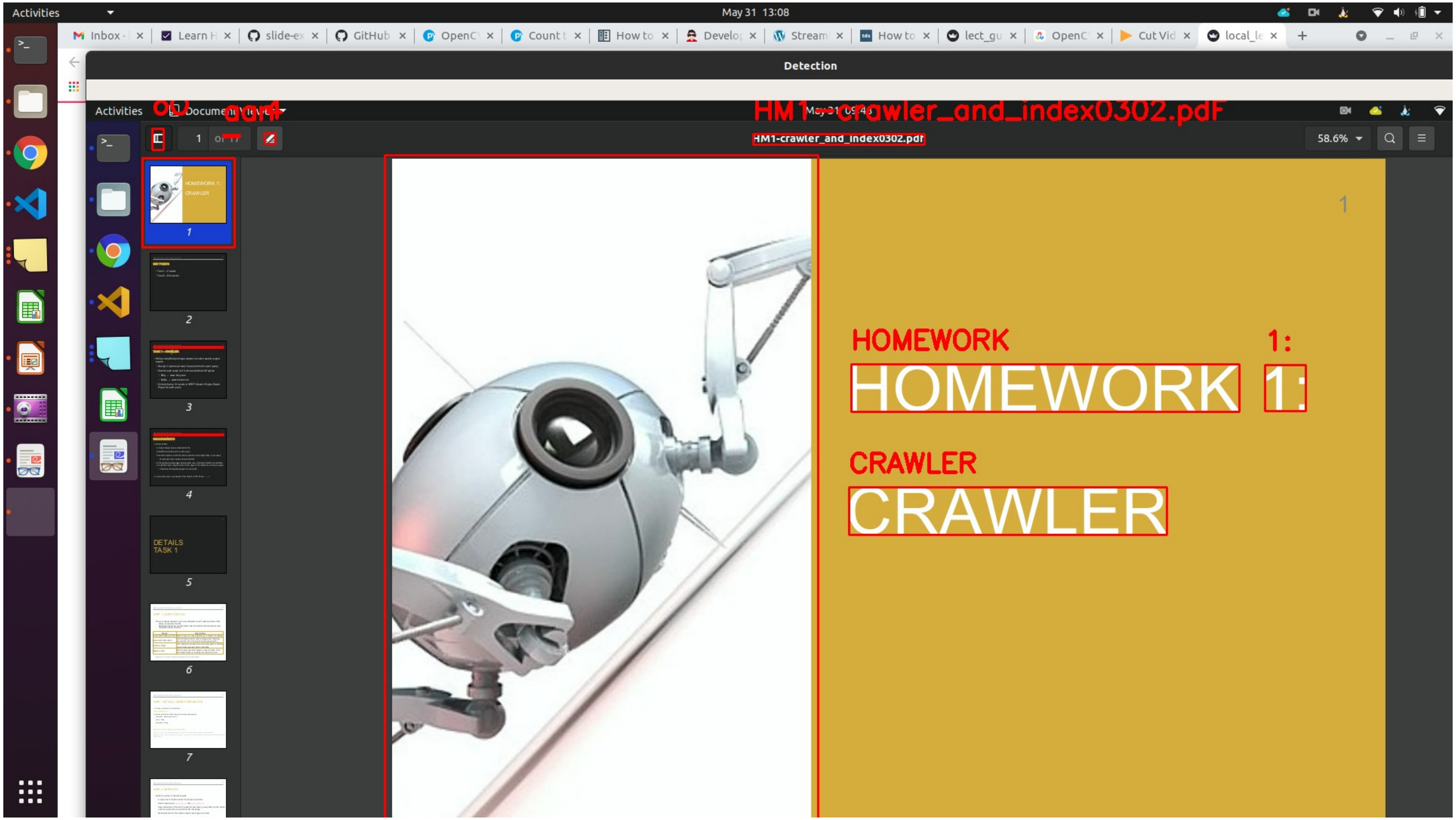


Example frameDiff after blurring and thresholding

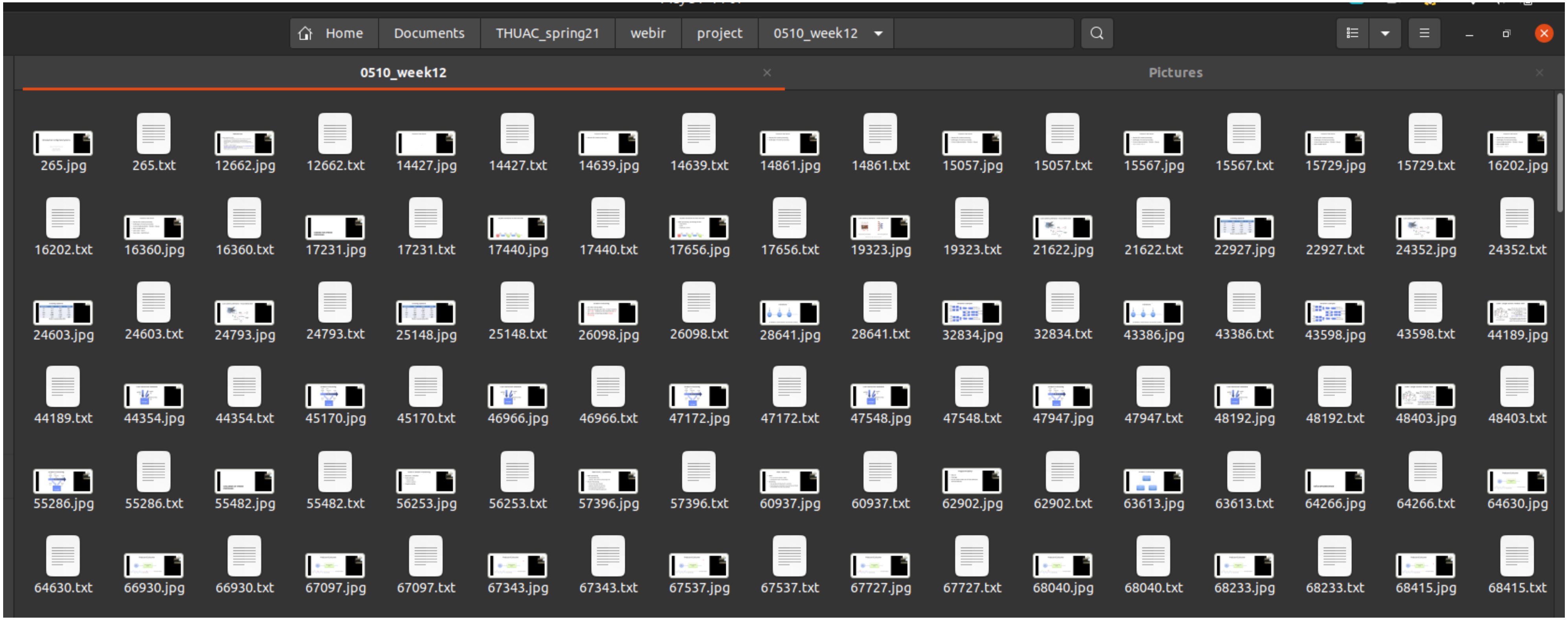
3. System Design – OCR

Tesseract v4 pretrained to detect and recognise text

3. System Design – OCR



3. System Design – Keyframe OCR output

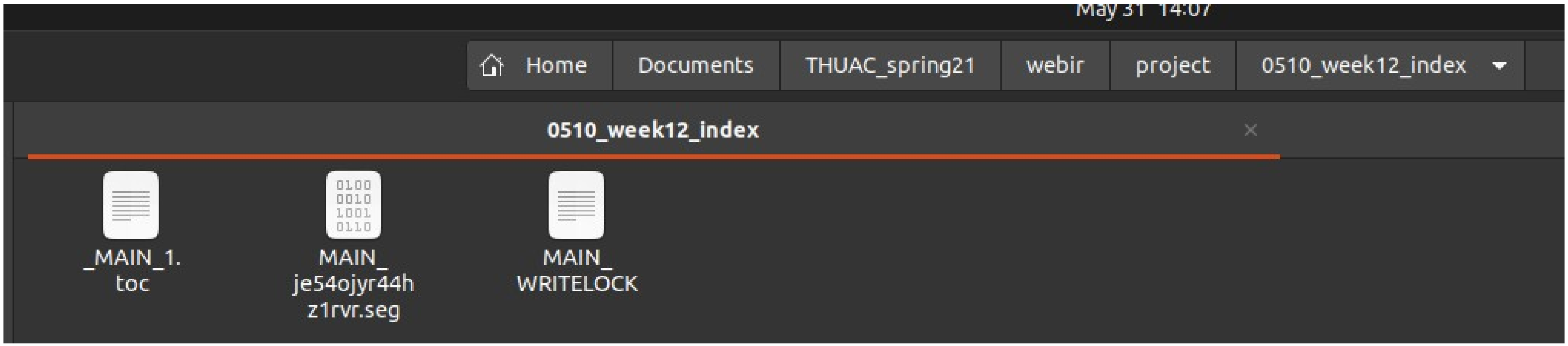


3. System Design – indexing and ranking

Whoosh Python library

1. Fast, featureful full-text indexing and searching library implemented in pure Python
2. Performs indexing based on configured schema – fields to be indexed for each document
    1. For this project, I simply used all the text identified by OCR as one field
    2. Can potentially separate into title, content, captions etc
3. Supports several ranking algorithms: Frequency, TF-IDF, BM25, cosine scoring
    1. BM25 used as default in this project

3. System Design – indexing and ranking



3. System Design - GUI

1.Streamlit library

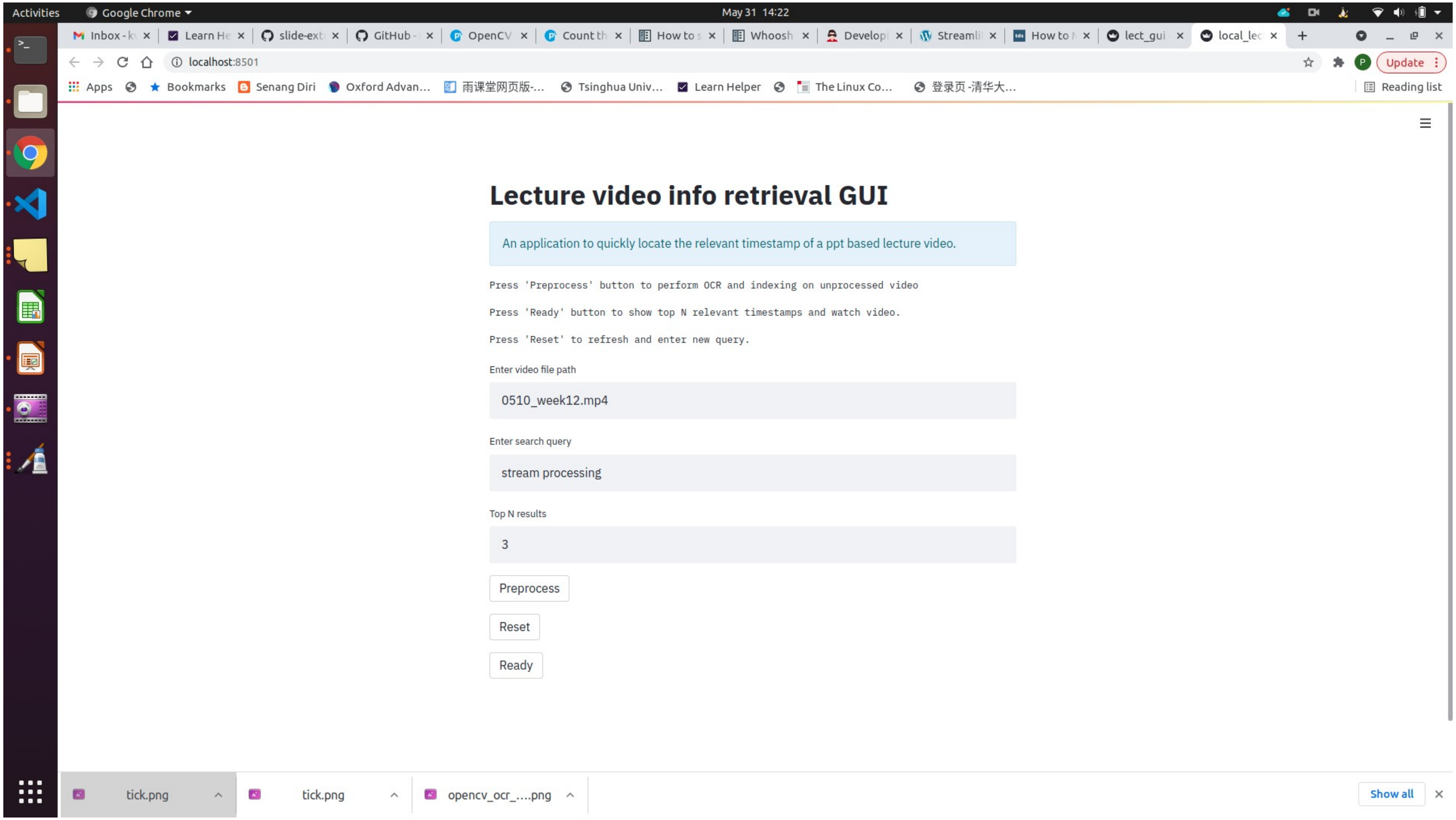
1.Converts python code into beautiful and interactive UI with very few lines of code (only 70 lines for the GUI in this project)

2.Launched on AWS-EC2 instance

1.Stored a single preprocessed video in the server for demo

2.Demonstrate preprocessing and searching function

3. System Design - GUI



Lecture video info retrieval GUI

An application to quickly locate the relevant timestamp of a ppt based lecture video.

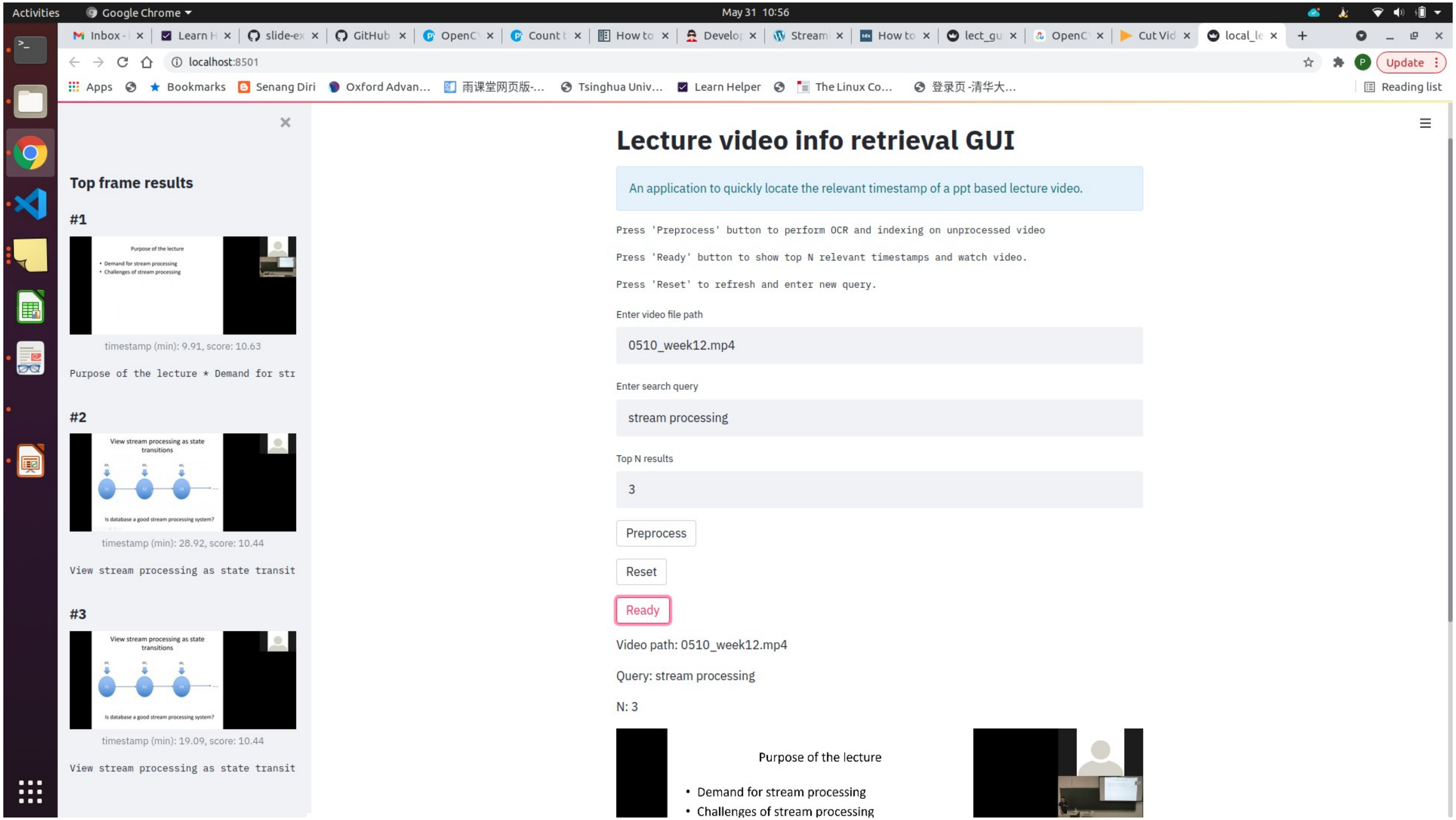
```
Press 'Preprocess' button to perform OCR and indexing on unprocessed video
Press 'Ready' button to show top N relevant timestamps and watch video.
Press 'Reset' to refresh and enter new query.
```

Enter video file path: 0510_week12.mp4

Enter search query: stream processing

Top N results: 3

3. System Design - GUI



Top frame results

#1

Purpose of the lecture
• Demand for stream processing
• Challenges of stream processing

timestamp (min): 9.91, score: 10.63

```
Purpose of the lecture • Demand for str
```

#2

View stream processing as state transitions
Is database a good stream processing system?

timestamp (min): 28.92, score: 10.44

```
View stream processing as state transit
```

#3

View stream processing as state transitions
Is database a good stream processing system?

timestamp (min): 19.09, score: 10.44

```
View stream processing as state transit
```

Lecture video info retrieval GUI

An application to quickly locate the relevant timestamp of a ppt based lecture video.

```
Press 'Preprocess' button to perform OCR and indexing on unprocessed video
Press 'Ready' button to show top N relevant timestamps and watch video.
Press 'Reset' to refresh and enter new query.
```

Enter video file path: 0510_week12.mp4

Enter search query: stream processing

Top N results: 3

Video path: 0510_week12.mp4

Query: stream processing

N: 3

Purpose of the lecture
• Demand for stream processing
• Challenges of stream processing

4. Evaluation

4. Evaluation

1. **Video analysis**
    1. Preprocessing slow (~60fps, i.e. 30min to process 1.5hr video)
    2. Video segmentation can be improved, currently has significant false positives, manual parameter tuning (minArea, minContours etc)
    3. Relatively accurate OCR, but can be improved with detection of slide location
2. **Indexing and ranking**
    1. Too reliant on accurate OCR
    2. Not flexible, query must be exactly the same as OCR output

5. Conclusion and future work

5. Conclusion and future work

1. **Implemented a simple application to search for relevant segments of a lecture video**
2. **Future work:**
    1. Leverage speech and other data modes for keyframe indexing
    2. Generalise to other lecture video types
        1. Handwriting recognition
        2. Identify ppt location in multi-scene videos
    3. NLP-based ranking and indexing algorithm for more flexible querying
    4. Extend to retrieval of video from a video database

THANK YOU