LECTURE SUMMARIZATION FOR MOBLIE LEARNING

Eli Spencer Ellis
Georgia Institute of Technology
Atlanta, GA

Mainampati Novith Reddy
Georgia Institute of Technology
Atlanta, GA

ABSTRACT

Video summarization is an active field of study with uses in reducing video file sizes. This was applied to lecture videos to enhance studying on a mobile device without needing to stream video. Different frame extraction techniques need to be used on different types of video (e.g. news, sports, action movies, surveillance, or lectures). This paper explores largest contour removal and analyzing text on a lecture video. A program was developed and deployed to be able to run on any lecture video.

KEYWORDS

Video summarization, key frame extraction, mobile learning

1. INTRODUCTION

The OMSCS program at Georgia Tech is an accredited Master's program that is provided completely online. The video lectures produced by Georgia Tech faculty are available at Udacity.com and available for anyone with a Udacity.com account. These classes share the exact content Georgia Tech OMSCS students use however they exclude the projects and exams that are part of the program.

Online Video lectures provide many benefits over traditional classroom lectures. They provide the ability to view them when convenient but they could also be bandwidth intensive. These lectures can also be viewed on Mobile devices however some students could face issues watching them considering the bandwidth restrictions in developing countries or remote areas.

To assist OMSCS students and to help minimize the effort required to obtain class notes, a tool was created that derives Study Guides from Lecture videos. This tool creates PDFs which contains images of important events within the video lectures along with relevant text derived from subtitle files.

1.2 Motivation

OMSCS students are not just highly motivated but most are extremely busy with full time jobs and personal commitments. It is crucial for such students to get access to immediate and relevant study guides to help prepare for their classes. The PDFs created by Lecture2PDF provides these students the required material so they leverage their study time accordingly. The PDFs provides searchable study guides to gather and collect important notes so the students don't have to invest time in rewatching videos. Given the PDFs can be downloaded on mobile devices, it provides the students the capability to learn during travel while also not impacting their bandwidth usage.

1.1 Previous Work

Video summarization has many techniques, but there is not one specific technique that can be used for all video types. Some different video types include, news shows, landscape shots, action movies, surveillance videos, and class lectures. For surveillance video, rather than capturing all the frames, only the frames with motion detection are saved. This method results in creation of smaller cameras, such as the FLIR camera which uses a combination of motion and trajectory analysis (Liszewski, 2015). This will lead to optimal results and reduces storage requirements, whereas using the same methods on a News video will not perform well. News videos would use a combination of audio-visual, mosaic, and gestured based approaches. (Ajmal et al. 2012). No specific programs have been found that focuses directly on class lecture videos.

2. LECUTRE VIDEO SUMMARIZATION

To solve summarization of lecture videos into a document format that can be read on a mobile device, a tool called Lecture2PDF was created. A user supplies videos and subtitle files to the program and the output is a single PDF. The program was made to run on a computer since video processing is resource intensive. A website repository was also created for users to download/upload specific lectures for the OMSCS program at Georgia Institute of Technology. This allows mobile users without a computer to access these lecture documents. The upload ability on the website provides the functionality for any user to submit lecture PDFs generated for existing or new courses.

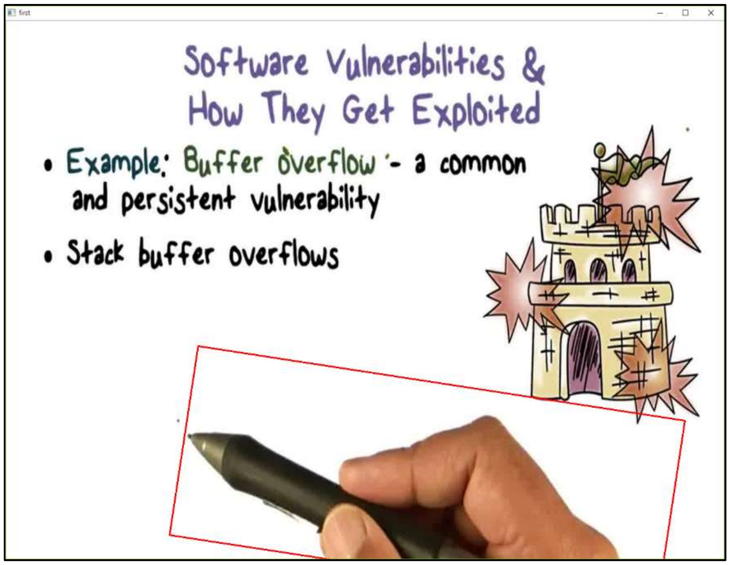
2.1 Techniques Used

As mentioned before, certain techniques work for specific video types and a combination of techniques provide better results. 90% of the lecture videos contains a whiteboard as the background, text and pictures are added to the whiteboard, and the professor’s hand. Lecture2PDF focuses on such lecture videos. The other 10% include the professor speaking in front of the camera.

Lecture2PDF first splits the video into I-frames, then uses contour tracking and optical character recognition (OCR) to save only the frames that are considered important. Important frames in a lecture video are when new data is presented on a whiteboard. Skipping frames with new information would confuse the reader and having too many frames would cause the document to be too large and difficult to read.

I-frames are used in video compression which represent frames that are standalone images that do not require other frames to decode. If an I-frame exists, then there is a scene change or movement; it also depends on how the video is encoded and compressed. This method alone generates too many frames. Each time the professor’s hands moves, a frame is captured. Other cases, nothing changes on the video and a frame was captured.

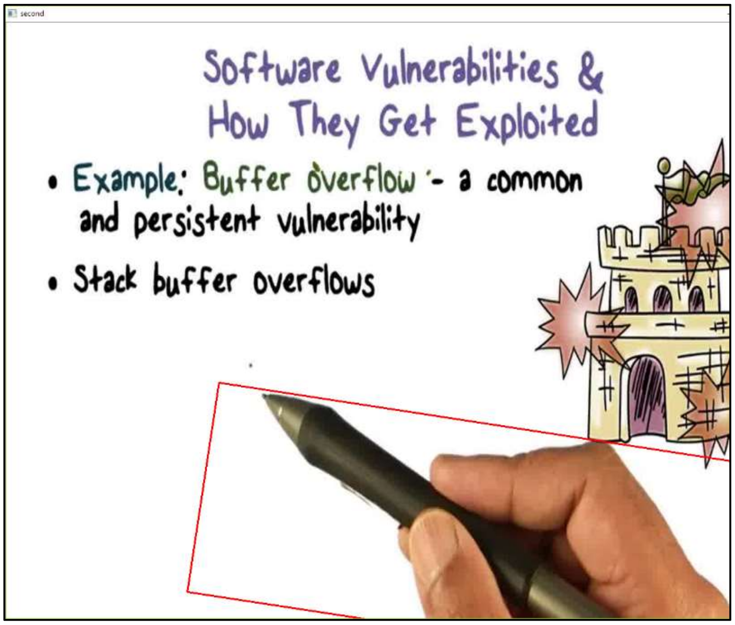
With the I-frames, the program then reduces it further by contour tracking and OCR. Contour tracking finds the boundary of objects that have moved or changed. In majority of the videos, it is the professor’s hand that is moving and not the actual content on the slide. Also, the professor’s hand is usually the largest contour. The program finds the largest contour and changes it to the background’s color. Once the hand is removed, the program can compare the images using the Tversky Index. If the images are different to a certain threshold, then the program saves the frame. If the frames are similar, then it does not save the frame. Figure 1 and 2 shows how a largest contour is detected between two sequential I-frames. The space inside of the box is turned white on both frames and then the images are compared. If the hand was not removed, the Tversky Index would calculate these as different images due to the hand movement.



Software Vulnerabilities &
How They Get Exploited

- Example: Buffer overflow - a common and persistent vulnerability
- Stack buffer overflows

Figure 1



Software Vulnerabilities &
How They Get Exploited

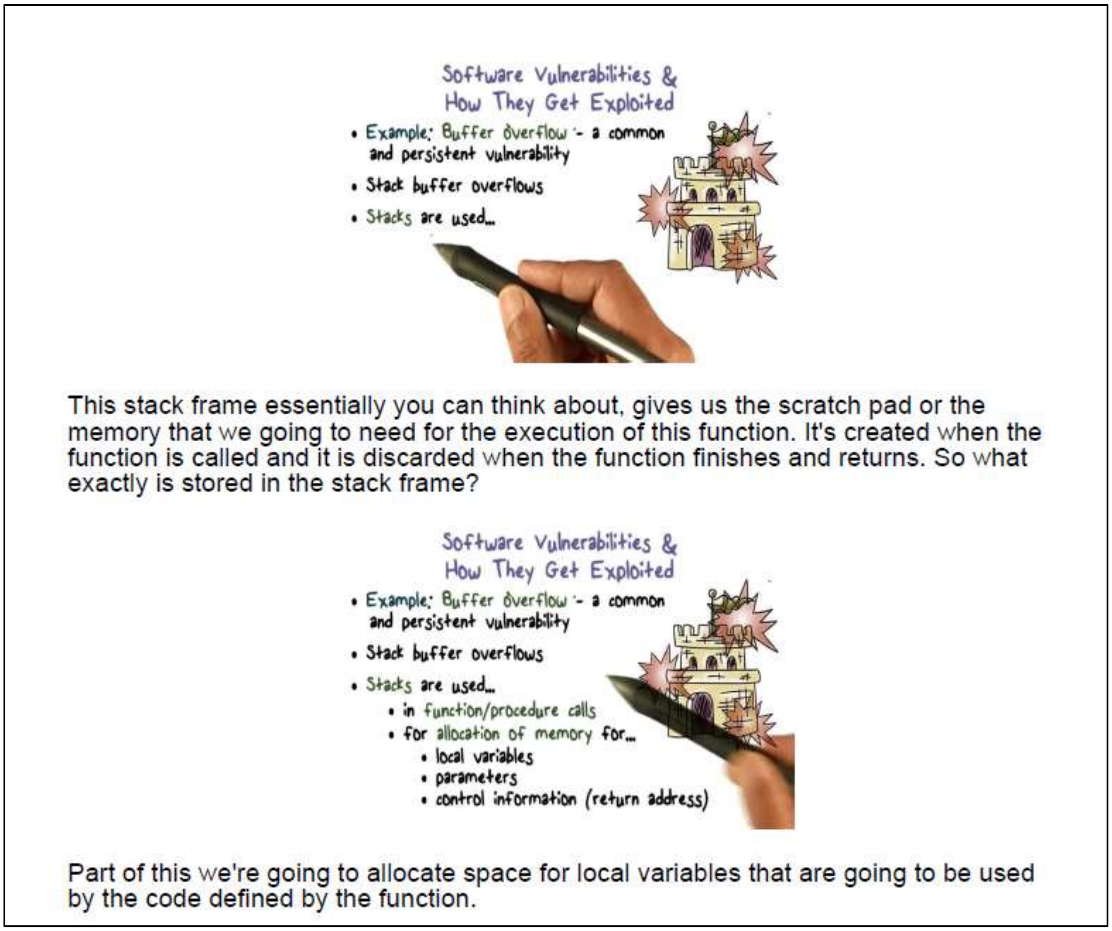
- Example: Buffer overflow - a common and persistent vulnerability
- Stack buffer overflows

Figure 2

Removing the hand is not enough when the added text or object is not enough to flag a difference in the images. To combat this, OCR is used on the images. The program uses Tesseract which is open sourced and sponsored by Google. The images are preprocessed to give definition to the text. OCR is run on the images to count the lines on the images. In Figure 1 and 2, OCR would count 5 lines. If the number of lines changed between two frames, then the program would save the frame.

2.2 Implementation

The program runs on all videos and subtitles in specified folders. The program requires a subtitle file for each video. Timestamps are saved for each frame when the I-frames are extracted and the subtitle files have timestamps for the text. With the extracted frames, the program compares using contour detection and OCR. If either technique flags a change in the frames, the frame is saved. It also saves the very first and last frames. Once all the important frames are saved, the subtitle files are parsed and grouped by the timestamps of the saved frames. The program then loops through the saved images and timestamps to write to a PDF. This repeats for the rest of the videos in the specified folder. After all PDFs are generated for the videos, the program merges all the PDFs into a single file called "Lesson-Final.pdf". Finally, the program cleans up temporary files. An example output of two saved frames written to a PDF are shown in Figure 3.



Software Vulnerabilities & How They Get Exploited

- Example: Buffer overflow - a common and persistent vulnerability
- Stack buffer overflows
- Stacks are used...

This stack frame essentially you can think about, gives us the scratch pad or the memory that we going to need for the execution of this function. It's created when the function is called and it is discarded when the function finishes and returns. So what exactly is stored in the stack frame?

Software Vulnerabilities & How They Get Exploited

- Example: Buffer overflow - a common and persistent vulnerability
- Stack buffer overflows
- Stacks are used...
  - in function/procedure calls
  - for allocation of memory for...
    - local variables
    - parameters
    - control information (return address)

Part of this we're going to allocate space for local variables that are going to be used by the code defined by the function.

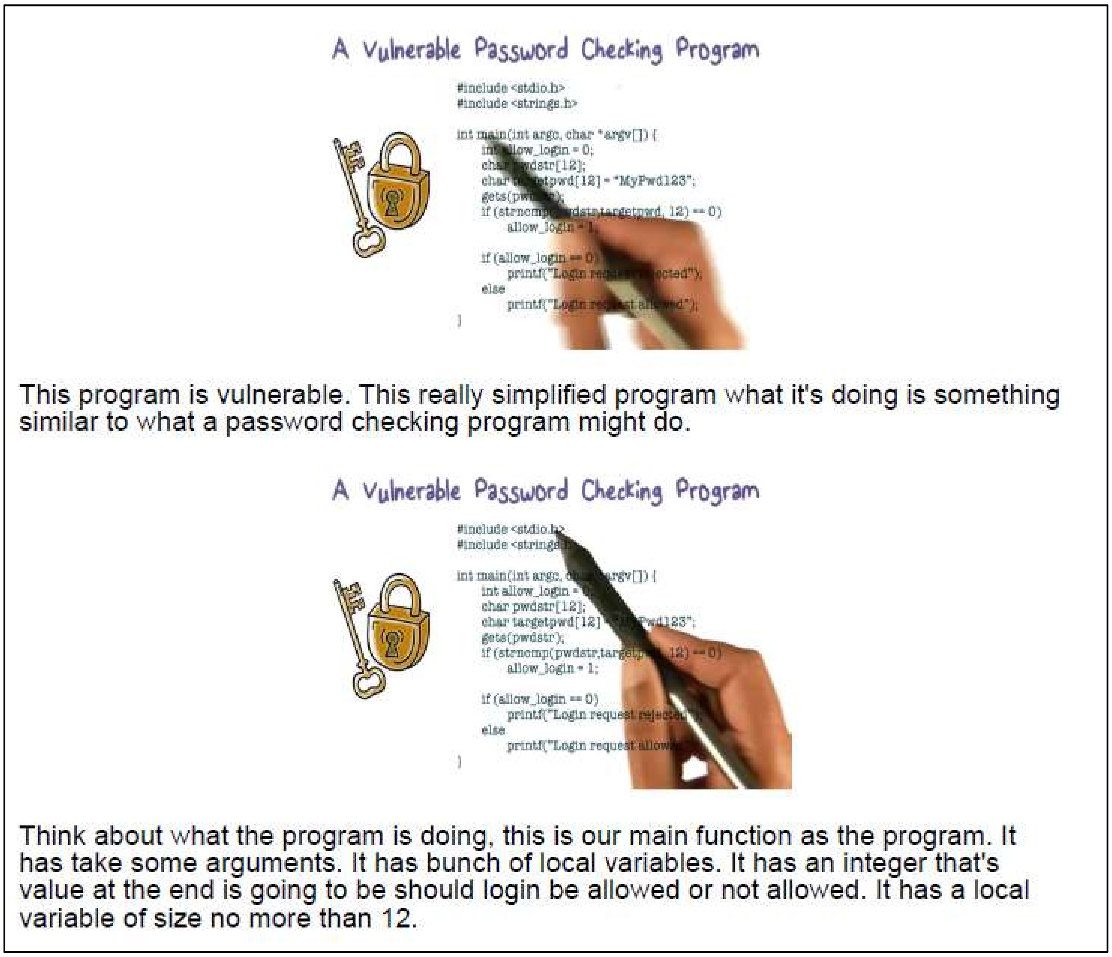
Figure 3

2.3 Results

The program functions correctly for all OMSCS videos. To measure the success of the document, it requires manual analysis. The program works best for what it is designed for, white background, with text/objects added as the video progresses. It can reduce a video into only frames where information changes, but some videos included images with nothing new added. The overall size of the PDF is reduced by 10 times compared to the videos. A group of videos was 115MB and the produced document was 10M. The program runs in about a quarter of the time of the videos. For 26mins lesson with 23 videos, it took 8mins to complete. For 94min lesson with 40 videos, it took 20mins to complete.

There are some limitations to removing the largest contour which could result in duplicate slides or not saving an important slide. Sometimes the largest contour was not the professor's hand, but a new image added to the whiteboard. The program would remove the new image and flag the two frames as the same image. Another case, the largest contour was the hand, but the hand was over new information. Removing the box around the hand also removed the new information making the images seem similar.

The OCR also has limitations and does not accurately find the text on some images. It would sometimes count the background images as text, or any pen marks on the whiteboard as a character. The OCR did not perform well with code as text. See Figure 4, the lines of text do not change, but the OCR would count a different number of lines for the two frames. That lesson should have just 2 frames saved, but the program generated 14.



A Vulnerable Password Checking Program

```
#include <stdio.h>
#include <strings.h>

int main(int argc, char *argv[]) {
    int allow_login = 0;
    char pwdstr[12];
    char targetpwd[12] = "MyPwd123";
    gets(pwdstr);
    if (strncmp(pwdstr,targetpwd, 12) == 0)
        allow_login = 1;

    if (allow_login == 0)
        printf("Login request rejected");
    else
        printf("Login request allowed");
}
```

This program is vulnerable. This really simplified program what it's doing is something similar to what a password checking program might do.

A Vulnerable Password Checking Program

```
#include <stdio.h>
#include <strings.h>

int main(int argc, char *argv[]) {
    int allow_login = 0;
    char pwdstr[12];
    char targetpwd[12] = "MyPwd123";
    gets(pwdstr);
    if (strncmp(pwdstr,targetpwd, 12) == 0)
        allow_login = 1;

    if (allow_login == 0)
        printf("Login request rejected");
    else
        printf("Login request allowed");
}
```

Think about what the program is doing, this is our main function as the program. It has take some arguments. It has bunch of local variables. It has an integer that's value at the end is going to be should login be allowed or not allowed. It has a local variable of size no more than 12.

Figure 4

For the video types with the professor speaking in front of the camera, the program did not behave optimally. Largest contour and OCR calculated that all I-frames were different. OCR would still report characters found on an image when none were present. These types of videos only account for 10% of the OMSCS videos. More development time is needed to find techniques that work for this type of video.

Finally, the document is limited to the files provided by Udacity. Some videos did not have a subtitle file or vice versa. Also, the video file names never matched 100% to the subtitle file names. The program matches video file with subtitle file by the order in the folders. Some of the subtitle files have typos or wrong translations. To resolves these issues, Udacity would need to be contacted.

3. CONCLUSION

Overall this program is useful and produces documents are small enough for mobile usage. Peers reviewed the outputted documents and stated the information is accurately portrayed. The program focused more on ensuring no information was missed instead of having the least number of slides per document. It is more detrimental to have missing information than a few duplicated images. Some information cannot be translated from video to document, such as the professor pointing to a specific object and hand gestures or the way the professor is speaking. The documents are better staged to be used as study guides that are searchable and were not designed to replace the lecture videos.

ACKNOWLEDGEMENT

We would like to thank our professor for Education Technology, Dr. David Joyner and our mentor, Ray Reese for their direction and help.

REFERENCES

Ajmal, M. et al., 2012. Video Summarization: Techniques and Classification. In *Lecture Notes in Computer Science*. pp. 1–13.

Liszewski, A., 2015. New FLIR Security Camera Turns Hours Of Footage Into Bite-Sized Clips. *Gizmodo*. Available at: http://gizmodo.com/flirs-new-security-camera-turns-hours-of-footage-into-b-1695991190 [Accessed July 30, 2017].