Slidecast - Low Bandwidth Lecture Delivery Platform

by

Harshit Gupta(Roll No.: 190050048)
Pradipta Parag Bora(Roll No.: 190050089)

Supervisors:
Prof. Varsha Apte

Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

2019

Abstract

Due to the pandemic, a lot of new technology has emerged to allow interaction between people in remote ways. One of them is the use of online platforms for lectures, presentations and talks. A major part of these just include slides and the speaker's audio. Currently, to share such things, the only way possible is to create a video recording or to distribute the slides early on. Both of these suffer from disadvantages. Sharing the slides early on poses the problem of synchronisation between the audio and the slides. Video recording suffers from the huge amount of extra memory it uses.

We present a solution for both of these problems. Slidecast allows us to record presentations in much the same way as a video but at a very low memory requirement. It accomplishes this by storing the slides as pdf, the audio and some metadata about transitions and cursor movements. Since there is no visual information stored, the quality of the video is not compromised. The audio quality can be set by the user depending on his satisfaction but even a high quality audio results in a size much less than the video.

Contents

Chapter 1

Introduction

1.1 Background

There has been a lot of changes in the world during the year 2020. A major change came in the way people communicate with each other which has led to the invent of new technologies. A significant portion of these interactions are lectures, presentations, talks, seminars etc. These are important sources of information for a lot of people. But everyone does not have enough bandwidth to be able to download a whole video since such videos can go as high as in gigabytes. So, there is a need to reduce the size as much as possible without affecting the quality much. We try to provide a solution for this issue.

Chapter 2

Design and Implementation

This chapter covers the various ideas and implementation details that were considered during the course of this project. First, the basic algorithm to compress is discussed and then the various implementations are discussed one by one.

2.1 Idea

The basic idea is to remove the unnecessary visual data about the slides from the files as they are repeated over multiple frames. The slides are stored separately as a pdf. The mouse events and key strokes are stored separately to get to know about the transitions. The audio is stored separately as an mp3 file. Having this much data, we can re-create the video easily. The current frame is created using the slide as background and mouse pointer over it. The transitions are identified using the keystrokes. This is the whole visual data in the video. Along with the audio, this completes the whole video.

2.2 Implementation - Recorder

We built two different recorders. One was python based and the other one is JavaScript based.

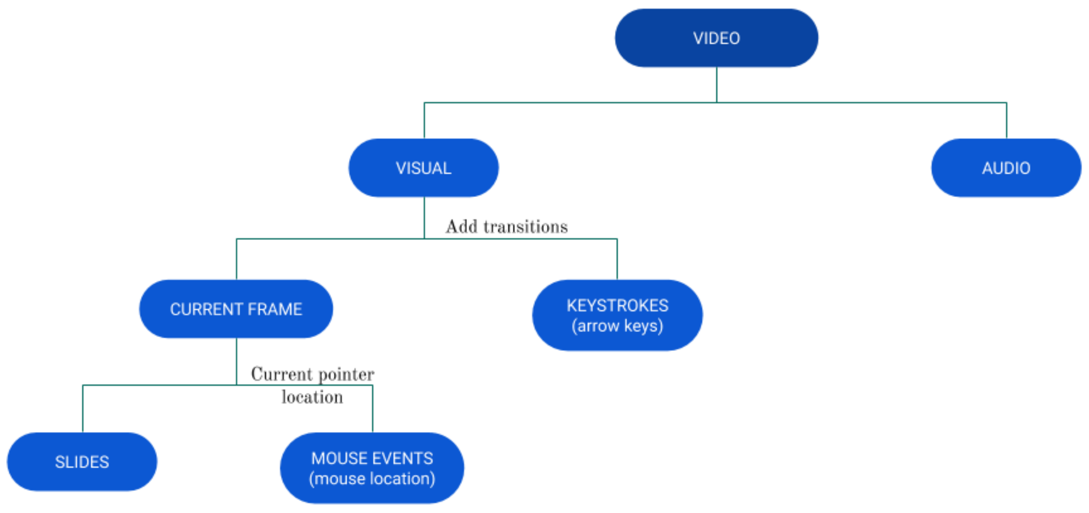


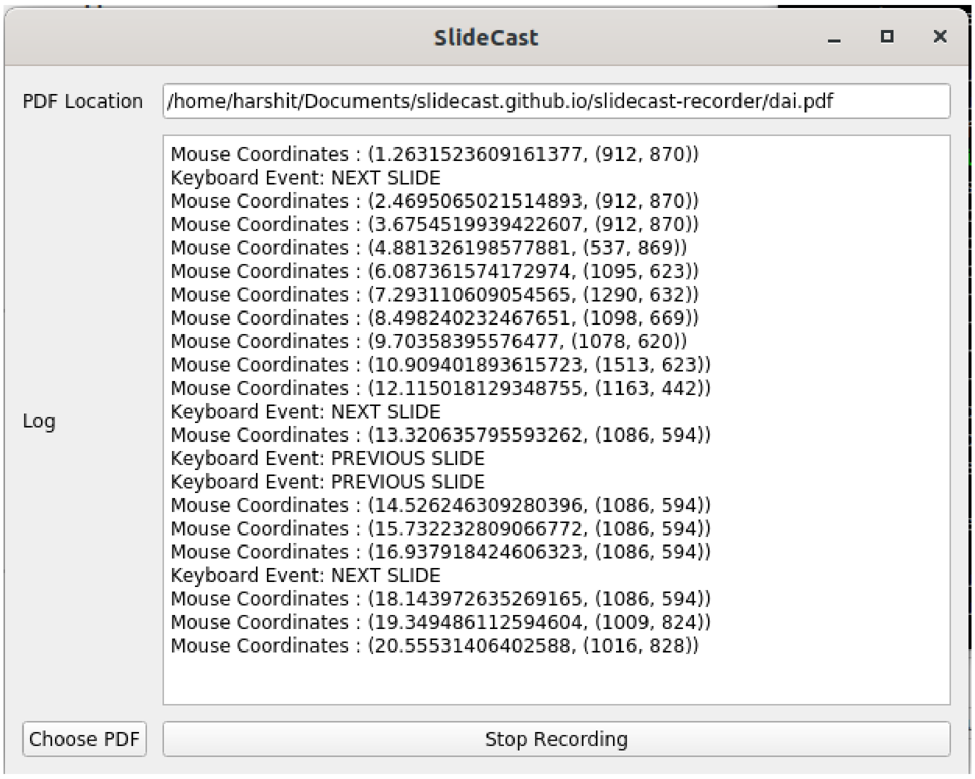
Figure 2.1: Structure of data

Python based recorder

In the python based recorder, the pdf renderer is not a part of recorder. So, the user can use any pdf viewer of his/her choice. Since each user uses different pdf viewer depending on their preferences, this has an advantage over a complete system. This recorder has a command line as well as graphic user interface.

Although this recorder gave a lot of freedom to the user, it suffered from drawbacks due to differences between viewers. Since the recorder has no access to the window size of the pdf viewer, it is assumed that the pdf is occupying the full screen. But this assumption can not be enforced on the window as the recorder has no control over the window size of the pdf viewer. Further, the user may want to open some other document besides the pdf like speaker notes which is not possible due to this assumption. Further, even if the user uses full screen mode for the presentation, most of the viewers have an extra boundary resulting in deviations from the actual video which may be fatal in some cases.

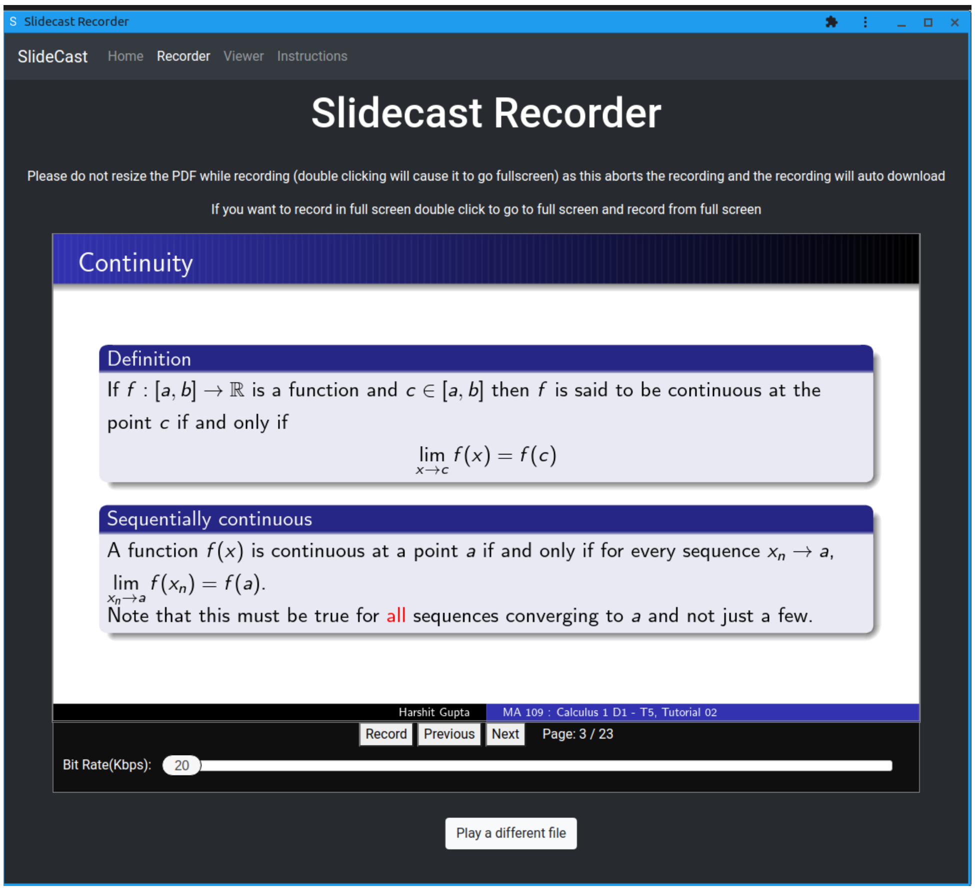
Further, this recorder requires superuser access to the computer which may not be deemed correct by many users. This is because we are capturing both the mouse and keyboard events across the entire workspace. This can only be done with superuser access since we are monitoring all the events of the user. We used python mouse and keyboard modules along with PyQt for the GUI framework to create this standalone local recorder.



/home/harshit/Documents/slidecast.github.io/slidecast-recorder/dai.pdf

```
Mouse Coordinates : (1.2631523609161377, (912, 870))
Keyboard Event: NEXT SLIDE
Mouse Coordinates : (2.4695065021514893, (912, 870))
Mouse Coordinates : (3.6754519939422607, (912, 870))
Mouse Coordinates : (4.881326198577881, (537, 869))
Mouse Coordinates : (6.087361574172974, (1095, 623))
Mouse Coordinates : (7.293110609054565, (1290, 632))
Mouse Coordinates : (8.498240232467651, (1098, 669))
Mouse Coordinates : (9.70358395576477, (1078, 620))
Mouse Coordinates : (10.909401893615723, (1513, 623))
Mouse Coordinates : (12.115018129348755, (1163, 442))
Keyboard Event: NEXT SLIDE
Mouse Coordinates : (13.320635795593262, (1086, 594))
Keyboard Event: PREVIOUS SLIDE
Keyboard Event: PREVIOUS SLIDE
Mouse Coordinates : (14.526246309280396, (1086, 594))
Mouse Coordinates : (15.732232809066772, (1086, 594))
Mouse Coordinates : (16.937918424606323, (1086, 594))
Keyboard Event: NEXT SLIDE
Mouse Coordinates : (18.143972635269165, (1086, 594))
Mouse Coordinates : (19.349486112594604, (1009, 824))
Mouse Coordinates : (20.55531406402588, (1016, 828))
```

(a) GUI for python recorder



Slidecast Recorder

Please do not resize the PDF while recording (double clicking will cause it to go fullscreen) as this aborts the recording and the recording will auto download

If you want to record in full screen double click to go to full screen and record from full screen

Continuity

Definition

If $f : [a, b] \to \mathbb{R}$ is a function and $c \in [a, b]$ then f is said to be continuous at the point c if and only if

$$\lim_{x \to c} f(x) = f(c)$$

Sequentially continuous

A function $f(x)$ is continuous at a point a if and only if for every sequence $x_n \to a$, $\lim_{x_n \to a} f(x_n) = f(a)$.
Note that this must be true for all sequences converging to a and not just a few.

Harshit Gupta MA 109 : Calculus 1 D1 - T5, Tutorial 02

(b) Javascript Recorder

Figure 2.2: User interfaces for the two recorders

JavaScript based recorder

This recorder includes a pdf viewer along with the functionalities provided by the python recorder. It uses pdfjs by Mozilla for this purpose.

In pdfjs, the current slide is shown on the screen using a canvas element. After getting the pdf file from the user, it is passed to pdfjs which creates a PDF object from it. This object can be interacted with to get the current slide(specified by the slide number) as an (re scaled) image.

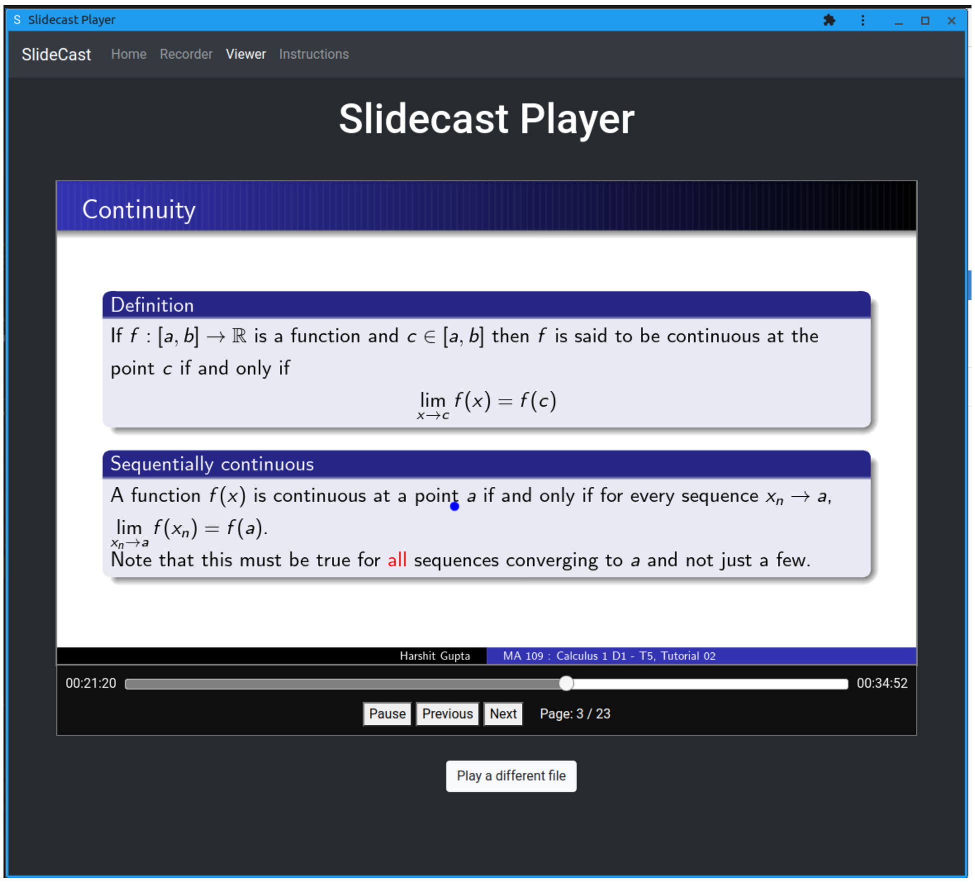
The key strokes are detected using simple inbuilt JavaScript functions. For the mouse pointer, p5js is used. "p5.js is a JavaScript library for creative coding and graphic rendering. It first creates a new transparent canvas above the canvas created by pdf.js and then renders a small circle on the mouse location which serves the purpose of pointer. It has inbuilt function for detecting the mouse location.

For recording the audio, it uses a MediaRecorder object. Finally, after the recording is done, all the data is dumped into files which is zipped using the jszip library. This creates a zip file containing all the data.

Since this recorder has its own viewer, it has full control over the window size and does not have any deviation from the original. Further it allows the user to set the bitrate for the audio allowing the user to select a suitable one depending on his/her purpose.

2.3 Implementation - Viewer

The viewer is also JavaScript based and has a similar design as the Recorder. The rendering of elements is the same. jszip is used to extract the data from the zip file. Howler.js is used for audio. Although their are built in functions for audio playing too but Howler.js provides better interface and error handling. A simple code is used to synchronise various elements with time and the seeker.



Slidecast Player

Continuity

Definition

If $f : [a, b] \to \mathbb{R}$ is a function and $c \in [a, b]$ then f is said to be continuous at the point c if and only if

$$\lim_{x \to c} f(x) = f(c)$$

Sequentially continuous

A function $f(x)$ is continuous at a point a if and only if for every sequence $x_n \to a$, $\lim_{x_n \to a} f(x_n) = f(a)$.
Note that this must be true for all sequences converging to a and not just a few.

Harshit Gupta MA 109 : Calculus 1 D1 - T5, Tutorial 02

00:21:20 00:34:52

Page: 3 / 23

Figure 2.3: The Player for the recorded files

2.4 Packaging

Finally we had to figure out a simple yet effective mechanism to dispatch the software to users. Initially the Python based recorder required a lot of low level modules that had to be bundled into a .deb file for Debian based systems and a .exe for Windows. Even though the code in principle was cross platform, the different methods needed for creating an installer proved to be a big hassle as we would have to maintain three different build instructions for the three major platforms.

The JavaScript based recorder + viewer however can be run simply from any Modern Web Browser. This provided an alternative approach for creating an installer using Web Technologies that in principle can be distributed cross platform. Keeping this in mind we decided to bundle the webapp as a Chrome Application that can be installed using Google Chrome.

However Google was phasing out Chrome applications in favor of Progressive Web Applications and the final application is a fully progressive web application. We are using Service Workers to cache the resources for the application including the CSS and the Javascript which allows the application to run offline even if the internet connection is down. Plus installation is a one click process, all the user has to do is visit the website using Chrome/Firefox and click on the Install button on the toolbar. The application can now be started from the Start Menu even when the internet connection is down providing all the features of a native application while also being incredibly easy to install. We therefore recommend the Javascript approach for this task since it is very user friendly and easy to work with.

Chapter 3

Results and Discussions

The size of recordings for various softwares is as follows:

Time	Slidecast (20kbps bitrate)	Zoom	Kazam	Chrome Recorder
5 min.	1.6 mb	3.8 mb	12.4 mb	19.8 mb
10 min.	2.2 mb	12.4 mb	31.2 mb	50.0 mb
20 min.	3.7 mb	20.3 mb	76.6 mb	87.4 mb
45 min.	7.7 mb	57.8 mb	248.8 mb	245.8 mb
1 hr.	10.5 mb	91.2 mb	370.4 mb	333.8 mb

Table 3.1: Comparison of size of recordings for various softwares

From the data in Table 3.1, we can see that slidecast can achieve a high compression as

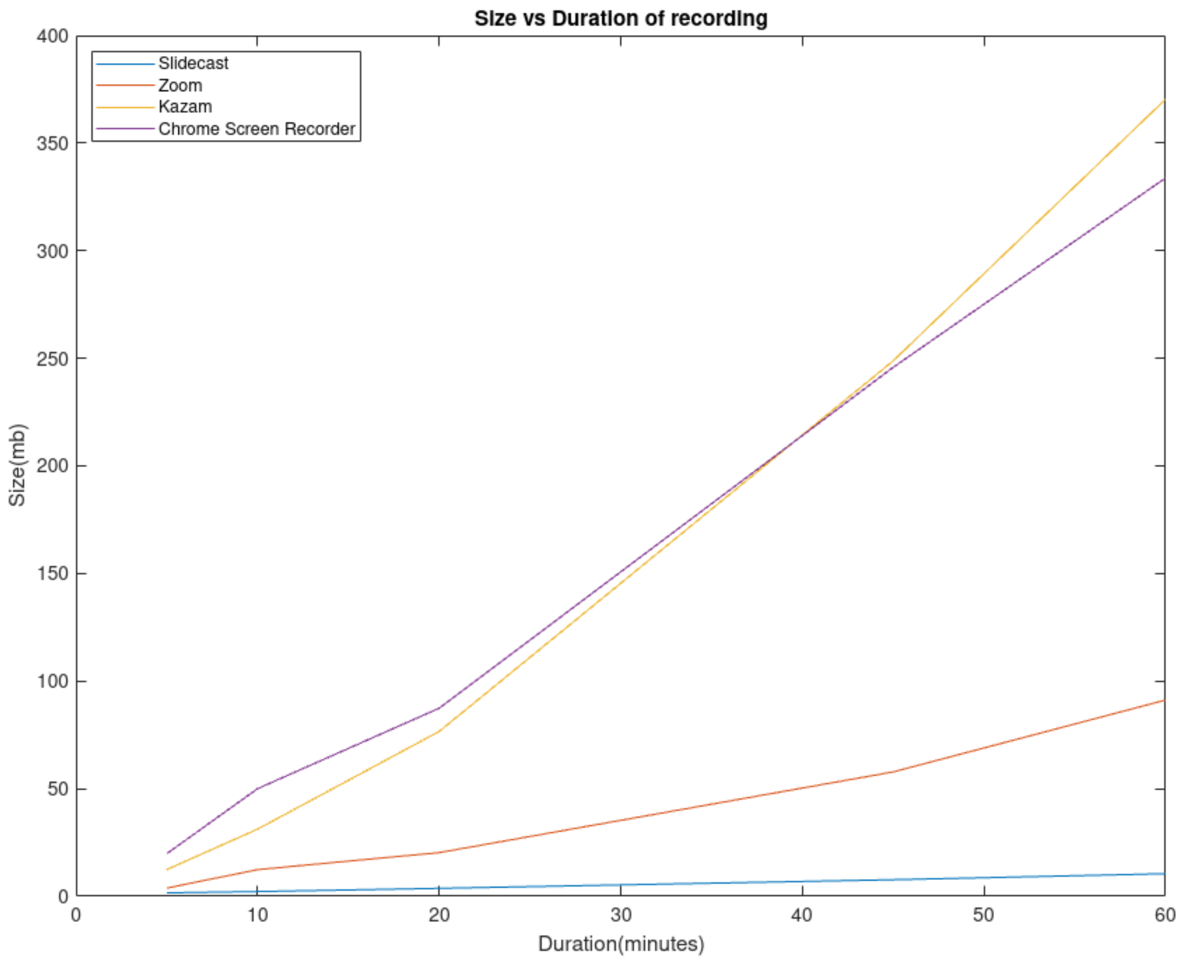


Figure 3.1: Comparison of sizes

compared to other softwares.

For small recordings upto 5 minutes, the size of pdf of slides dominates the size of recordings. So, it remains almost the same till 5 minutes. So, the compression is not very good(it still achieves a compression of 2X as compared to Zoom and about 7X as compared to the other 2 softwares). But as the duration of recording increases, the effect of compression becomes significant and we see that it achieves a compression of about 9X as compared to Zoom and 35X as compared to Kazam and Chrome Recorder.

Chapter 4

Summary and Conclusions

Through this RnD project we provide a simple yet effective means of delivering online lectures that rely heavily on slide based presentations by reducing the redundant data carried in video recordings of the presentations. The software package allows for easy recording and viewing through an offline progressive web application that is easy to install. We are seeing massive gains over normal screen recording with only 10 MB of space being needed for an hour of recording in the default bit rate. This size can be further changed by tweaking the bit rate as per the quality requirements.

Further improvement to the software stack can be done, right now the application can be installed on smartphones as well due to it being a progressive web application although the design can be improved to suit mobile users for viewing. Further annotation support can be added to the recorder where a mouse hold down event can be used to write on the slides and the written text can be rendered on the viewer.

We have used a lot of open source packages in this project including but not limited to Mozilla PDFJs, HowlerJS, jQuery and Bootstrap. We therefore believe that all the work that we have done in this project should be made open source and the code has been published for open access on the organistion: `https://github.com/SlideCast`. The application can be used and installed right now on the hosted domain: `https://slidecast.github.io`.