Integrating Natural Language Processing and Computer Vision into an Interactive Learning Platform

Benson Liu
bensonliu@gmail.com

Rithesh Rajasekar
rithesh.rajasekar@gmail.com

Aditya Shukla
adityashukla2015@gmail.com

Adina Solomon
adina4solo@gmail.com

Lucy Xu
lucythexu@gmail.com

Jessica Kuleshov*
jessica.kuleshov@columbia.edu

New Jersey’s Governor’s School of Engineering and Technology
July 21, 2020

*Corresponding Author

Abstract—Virtual learning platforms are digital interfaces that allow educators and learners to communicate and share resources that facilitate education. Currently, most of these lack resources to increase engagement among students. Barriers such as language, digital literacy, and limited features often make platforms difficult to use. The purpose of this research is to create a comprehensive learning platform by incorporating concepts of Natural Language Processing (NLP) and machine learning. The platform was developed with Python and APIs such as socket programming for server connections, OpenCV for visual machine learning, and Google Speech Recognition for auditory NLP. After integrating components to the platform, the features were rigorously tested for accuracy to assess the success of a more comprehensive learning environment.

I. INTRODUCTION

The trend towards virtual learning has become increasingly popular. According to the National Center for Educational Statistics, 3.3 million students attended college exclusively online in 2018 [1]. Online platforms offer students access to resources that are often inaccessible due to geographic barriers or other socioeconomic factors. Additionally, online learning allows for a more tailored educational experience. Most studies on the efficacy of virtual learning have shown improvements in student understanding. A study conducted by Shifit E-Learning found that students engaged in virtual learning have a knowledge retention rate of 25-60%, whereas face-to-face learning is around 8-10% [2]. When used effectively, online learning allows students to improve their performance by readily providing resources that are unique to the student’s needs. The same study found that students were able to learn nearly five times more material than traditional learning without increasing time spent. More recently, the global Coronavirus pandemic has disrupted 1.2 billion learners globally, forcing many educational institutions to adopt virtual platforms such as Zoom, Google Classroom, WebEx, or Canvas to replicate traditional educational models [3] - [5]. For these reasons, online learning is now a serious educational option for both private and public institutions.

Although the demand for learning platforms continues to grow, several concerns with usability and low digital literacy have made online learning a challenge. Most online platforms are restrictive and students have difficulty following along with lectures, staying engaged, and facilitating discussions or asking questions. Furthermore, limited features make it hard to share resources and communicate on a single platform. For instance, Zoom & WebEx are popular platforms for communication, while Canvas and Google Classroom are used for resource sharing. By far, one of the largest barriers is combating distractions for younger students. Many studies have shown that online learning for this age group is not as effective because of their inability to mediate distractions. The lack of features to engage students on learning platforms does little to resolve this issue. A study published in the Journal of Child Psychology investigating dual-modality learning exposed students to visual and auditory sources while reading. The research suggested that this method helped improve students’ vocabulary [6]. Video conferencing platforms, if integrated with the right features, can take advantage of dual modality to create an enhanced virtual learning environment.
With rising demand in the short and long term, the online education industry is experimenting with adaptable learning platforms of the future. Despite the growing necessity for virtual learning, most contemporary products struggle to accommodate traditional teaching methods. This paper discusses how NLP and computer vision were implemented to tackle common issues with existing virtual learning platforms.

II. TECHNICAL BACKGROUND

Initial research was conducted on existing video conferencing features and the programming techniques used to implement them. After identifying features to enhance, research was conducted on possible implementations.

A. Socket Programming

Socket programming is frequently used to grant access to the World Wide Web with Hypertext Transfer Protocol (HTTP). It is also used with other communication platforms to request and receive information. This module gives multiple devices the ability to connect to a singular network. A socket is an endpoint to a two-way communication pathway. The model starts with at least two sockets, a server and a client, but can incorporate multiple clients simultaneously without the knowledge of each individual's address. The server acts as a central hub with the ability to receive and distribute data between multiple clients through different network protocols. The client communicates directly with the server, requiring information about the address of the server to initiate the stream of communication. The client and server can communicate through a Local Area Network (LAN) if they reside in the same local network or utilize a Wide Area Network (WAN) to communicate across multiple LANs as shown in Figures 1 and 2 [7].

B. Encryption

Although socket programming provides an optimal method for the exchange of large quantities of data, as communication becomes more accessible over networks and the amount of data transferred increases, the threat of data breaches also increases. From simple searches on Google to calls on Zoom, data is constantly encrypted to protect sensitive information.

Encryption is a process in which a message is encoded and decoded so that only certain parties can access the data in a readable format. The decrypted data is referred to as the plaintext, while the encrypted data is referred to as the ciphertext. A message is encrypted and decrypted through the use of cryptographic keys that translate between plaintext and ciphertext. There are two types of encryption: symmetric and asymmetric. Symmetric encryption uses a single key for both encryption and decryption which both parties must have access to [8]. On the other hand, asymmetric encryption uses public and private keys for encryption and decryption. Both methods include encryption algorithms to mathematically use a key to convert data into ciphertext. Although there are many different encryption algorithms, Advanced Encryption Standard (AES) is commonly used with symmetric encryption. AES revolves around the idea of a substitution-permutation network where certain outputs are based on direct substitution while others are formed by rearranging inputs. The number of rounds in AES encryption is based on the length of the keys. For a general 128-bit key, there are ten rounds with a unique 128-bit encryption round key. Each round for a 128-bit key includes four steps. First, the 128 bits are consolidated as 16 bytes, or a 4 by 4 matrix that undergoes substitution with a fixed table. Then, each row in the matrix is shifted to the left and each column undergoes a mathematical computation. Lastly, all 128 bits are XORed (refer to figure 5) before an output is generated. In order to decrypt the data, the rounds and individual sub-sections are performed in the reverse order with the same keys [9].
Fig. 3. AES encryption architecture with different sized keys. Source: Adapted from [9].

Fig. 4. Stages of AES encryption in a single round with 128 bits cipher keys. Source: Adapted from [9].

Fig. 5. The logic behind XOR gates with 2 inputs. Source: Adapted from [10].

C. Computer Vision & Image Processing

For the purpose of this project, machine learning will refer to an application of Artificial Intelligence geared towards training systems to autonomously master and refine their ability to execute certain tasks. Computer vision draws heavily from machine learning to process visual (image and video) data. As humans, we have the ability to recognize and draw conclusions from our surroundings using vision and perception [11]. Computers replicate this phenomenon by employing algorithms for object identification, feature classification, and image segmentation. Using these trained models, computers can locate, track, and analyze objects [12].

To aid computer vision tasks, image processing applies certain transformations to refine raw input images so that they can be used for further analysis [13]. Image manipulation techniques such as filtering, feature sharpening, edge and color detection, and noise removal are performed to prepare data for the computer vision algorithms.

1) OpenCV: OpenCV is an open source machine learning software library designed to optimize real-time computer vision processes. OpenCV was originally developed in C++ but is utilized as a cross-platform software, compatible with popular languages including Python, C#, and Java. It has more than 2500 built-in algorithms that lay out a comprehensive infrastructure for developers to use [14]. Specifically, in its applications with Python, users are able to perform operations and analysis on image arrays with greater efficiency using modules such as NumPy and Matplotlib.

2) Images & Image Convolution: A digital image is an arrangement of pixels in a two-dimensional grid with a specified height and width. Each pixel stores color information as numerical data. Different color models such as RGB or CMYK require varying amounts of bits per pixel to represent the proper color. Grayscale images are typically 8 bits, while RGB pixels are typically 24 bits (3 bytes) or 48 bits (6 bytes). RGB pixels store separate red, green, and blue values which are combined to form a single color [15], [16]. To store more information about the pixel, with each additional byte, an additional value is added into the pixel or the range of pre-existing values is expanded. The figure above illustrate that more bits per pixel allow for greater specificity and a wider range of represented colors. These color values can be scaled down or converted with convolution techniques. Convolution is the mathematical operation of combining two arrays to create a resulting array that is used to apply filters on images to alter their appearance. The convolution procedure incorporates three main components: the input image, kernel, and output image. A kernel is a smaller square matrix that acts as a filtering mask when layered over a specific region of an input image [18]. Each entry in the kernel is assigned a weight that is applied to the pixel under it. The new pixel values are generated by multiplying the kernel weights to the pixels in the corresponding positions below them, and then summing the totals together to return an output for the pixel directly below the origin, or center entry of the kernel. Convolution is a linear form of image processing that relies on the weighted sum of neighboring pixels to generate output values.
The general equation of convolution is represented as:

\[ g(x, y) = w * f(x, y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} w(dx, dy) f(x+dx, y+dy) \]

In this equation, \( g(x, y) \) is the filtered output image, \( f(x, y) \) is the input image, and \( w \) is the filter kernel. The total output value can also be normalized by dividing the output by the total kernel entry count to ensure that the pixel preserves its original brightness and its value stays within range. The convolution filter cannot be applied to edge pixels unless a padding is applied, in which a set of fixed values, usually all 0’s or 1’s, creates a border around the image to retain the image’s original dimensions [20]. The kernel shifts across the image, and the convolution process is repeated until all the pixel values in the input image have been converted into a separate output image.

3) Image Filtering Techniques: Depending on the arrangements of weights within a kernel, the effect that the mask has on the output image will vary. Different kernel convolutions have the ability to blur, sharpen, emboss, and outline features in images. Figure 8 shows several common kernel operations that can be applied to an image using the convolution method mentioned in the previous section.

Gaussian blur kernels are applied in order to smooth out noise impurities during the preprocessing phase. It performs better than other blurring techniques because it yields finer results by using a normal distribution function represented as:

\[ G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

\( X \) and \( Y \) are the respective horizontal and vertical distances from the origin, and \( \sigma \) is the standard deviation. Rather than applying a uniform spread, gaussian blurs put more emphasis on center weights in the kernel [22]. As shown in Figure 9, this effect can be modelled using a 3D-bell-curve to illustrate the distribution of the values in a gaussian blur kernel matrix. The values are more elevated towards the center, indicating they have a higher weight than the surrounding ones [23]. Additionally, larger kernels will have a greater blurring effect on the underlying image.

D. Natural Language Processing

NLP is a field of machine learning focused on converting human languages into data that a computer can recognize and extract information from. NLP allows a computer to understand human languages, like English and Spanish, by converting letters into numbers using advanced algorithms [24]. Speech recognition, the subset of NLP used in this project, includes voice search and context recognition [25].

Current speech recognition systems have three sets of conversions: sound to electrical signals, analog electrical signals
to digital signals, and digital signals of audio to text. To convert the digital signal into text, most modern systems use the Hidden Markov Model (HMM) which assumes that a speech signal, when interpreted on a small timescale (like ten milliseconds), can be viewed as having constant properties. These small fragments of speech are converted into vectors known as cepstral coefficients and then matched to a fundamental unit of speech called phonemes. An algorithm is then applied to determine the closest word from the specific set of phonemes. Since phonemes vary based on speaker, these programs require intensive training in order to improve accuracy. Modern systems use neural networks to refine the accuracy. Modern systems use neural networks to refine the accuracy.

This project specifically uses two Python APIs for NLP: PyAudio and Speech Recognition. PyAudio is an open source library based on the PortAudio library in C/C++ that allows audio to be recorded and played easily [27], [28]. The Speech Recognition library consists of multiple APIs including Google Speech Recognition and GoogleTrans which were used in this project. This provides multiple features such as voice-to-text and text-to-text translations.

III. Experimental Procedure

This section documents the different challenges that arose while implementing the following features: speech-to-text subtitles, translation in English and Spanish, voice-driven polling using OpenCV, a screen sharing feature, and a web search feature. Motivations for choosing certain implementations over others are also discussed.

A. Socket Programming Implementation

Socket programming provides an efficient method to transmit data between a server and a number of clients. In order to create a virtual learning platform, the transmission of data needs to be fast (in terms of the processing rate) and reliable. The two primary types of protocols are Transmissions Control Protocol (TCP) and User Datagram Protocol (UDP). TCP establishes a connection-oriented protocol used in TCP/IP networks, while UDP is a connectionless protocol. Therefore, TCP provides additional in-built error checking in order to confirm that the packets of data were sent appropriately. This method allows TCP to have a specific flow control mechanism that dictates the specific amount of data flowing from clients to servers and servers to clients. The data is sent through a send-and-receive buffer to prevent an overwhelming amount of data from transmitting at a given moment. As a result, TCP is generally utilized in applications that require high reliability, such as the web (HTTP) and email. Therefore, TCP provides the optimal type of communication for a steady flow of data and ensures that the connection is maintained at all times during a call [29].

An initial implementation of socket programming with video conferencing was analyzed based on two-person and four-person video conferencing platform. The program included a traditional client and server model in which each client sends their data to the central server before the server transmits the data to each individual client. The program also dealt with unique ports to transmit different types of data such as the audio and the video [30]. However, this approach included an unnecessary step when dealing with only two clients as socket communication can occur directly through the server (teacher) and the client (student). Moreover, using multiple ports increases the overall load on the computer which causes lag, reducing the quality of transmission. The following code shows the basic binding of a server socket to a port and a client socket to the server socket.

```
sock.bind((host, port))
self.sock.connect((self.addr, self.port))
```

In search of a more efficient method of communication between a single server and client, a video/audio chat project created by DBC projects was adapted to create a base platform with direct communication between the server and client sockets. With this logic in the platform, the host or the server socket waits for a connection to itself while the client seeks to connect to the server socket. The transmission of the data through sockets for this project was centered around the idea of using only one port to send and receive data [31]. This resulted in a better overall performance during testing. After building an initial framework for audio and video transmission, the audio contained noticeable lag. In order to mitigate the issue, experimentation with a lower sample rate per second from 22050 to 8000 and a lower number of frames per buffer from 2048 to 1024 significantly reduced lag in the audio. Although this reduced the audio quality itself, the change was unnoticeable and it allowed for a less intensive transmission. The code below packages all the information including the frame size, image frame, audio frame, caption, and poll information into byte format before it is sent through the socket.

```
frame_size = (4 + len(img_str) + len(self.audio_buff) + captionLength + pollLength).to_bytes(4, byteorder='big')
send_frame = frame_size + img_size + img_str + self.audio_buff + captionBytes + pollBytes
```

Two devices were initially connected through LAN with the IPv4. This allowed for a simple setup by passing in the other device’s IPv4 address. However, this was limited to within a single local network. In order to connect remotely within a WAN, the concept of port forwarding was utilized to allow data to flow to a certain port on a device from an outside source. This eliminated the need to take down all the firewalls which protect a device by blocking unauthorized connections from a private network.

B. AES Encryption

In order to keep the information of students and teachers safe, encryption was used in the educational platform.
Especially with minors, the connection between the teacher and the student needs to be secure at all times. Even if the data is intercepted, a third party should not be able to read it. Symmetric encryption was the best option for a voice conference so multiple users could encrypt and decrypt the data with a common key. Therefore, AES Encryption was utilized with 128 bit keys, using the Crypto Cipher package. In order for the server and client to accurately encrypt the data on the sending side and decrypt the data on the receiving side, the encryption key must be identical on both ends. It is generated through a parameter, in this case, the meeting ID. If the server and client both use the same meeting ID, they will have the same encryption and decryption key and can view the data received. However, if the meeting IDs do not match, then the encryption keys generated on the client and server side will be different. Similarly, if a third party attempts to access the data transferred between the sockets, they will only have the ciphertext which can not be read without the correct encryption key. The code below shows how the encryption and decryption key is created based on the meeting ID entered by the user. The encryption and decryption object was used before sending and after receiving the message.

```python
self.encryptionObject =
AES.new(self.meetingid.encode("utf-8"),
AES.MODE_CFB,'This is an IV456'
.encode("utf-8"))

self.decryptionObject=
AES.new(self.meetingid.encode("utf-8"),
AES.MODE_CFB,'This is an IV456'
.encode("utf-8"))

send_frame_encryption =
Gui.encryptionObject.encrypt(send_frame)
self.sock.sendall(send_frame_encryption)

d = self.conn.recv(recv_size)
d = Gui.decryptionObject.decrypt(d)
```

This platform provides end-to-end encryption that most video conferencing platforms do not have. For example, Zoom stores all the keys for data encryption in its cloud server. Therefore, they could theoretically decrypt your personal data at any point [32]. However, this educational platform does not store encryption keys, so the user’s personal data is only accessible during the call by the intended parties. This means students and teachers can be confident in using a platform where their security is a high priority.

C. Translations/Subtitles

In order to use voice commands, NLP was used to convert the user’s speech into data the computer can interpret and return as a string of text. Speech Recognition, a Python library equipped with the Google Web Speech API, was used for this project. Since accuracy is the chief concern of a NLP program, a method to adjust for ambient noise from the Recognizer class of Speech Recognition was used to set a baseline for the background noise. The PyAudio library was utilized to easily connect to the user’s default microphone. After preliminary tests, the speech recognition feature was integrated into the main server code. Using the recognize_google method of the Recognizer class, the user’s words were transcribed into a text string which was then used for the subtitles as shown in the code below.

```python
r.adjust_for_ambient_noise(source, duration =0.5)
self.transText = r.recognize_google(audio) .lower()
```

For foreign languages, a parameter in the recognize_google method can be used to specify the language the user is speaking in. This was made possible with the Google Speech Recognition API, which has language translation capabilities for over a hundred languages. If the user has specified Spanish as their primary language, the program then translates the Spanish text string into English and stores that information in the subtitle variable. This ensures that the other user will be receiving translated English subtitles. On the receiving end, English subtitles will be translated into Spanish if the user has specified that Spanish is their primary language. All translations occur on the Spanish speaker’s side in order to decrease lag time. This prevents unnecessary translations if there are only English speakers on the video call. The code below translates Spanish audio into Spanish text, before translating the Spanish text into English text.

```python
real_audio =
r.recognize_google(audio, language= "es-ES")

string = str(real_audio)

self.transText =
trans.translate(string, scr='es',
dest='en').text
```

D. Polling Feature

This section discusses the implementation of a completely voice-driven polling function through the use of voice commands and OpenCV for finger detection.

1) Video Commands: Since the polling feature is enabled through voice commands, the program constantly searches the captions for certain keywords. Multiple keywords were chosen for their relevance to the topic of polling. However, the program did not always pick up the keywords. By including homophones like “done” and “dunn,” mistranslations of the user’s audio were accounted for in the voice commands. After ensuring that the program accurately recognized certain keywords, various methods were considered to obtain the required information from the user for the poll. Using the Google Text-To-Speech (gTTS) Python library, the computer
reads prompts to the user. Using state functions, the program can identify different stages of the poll including initiating the poll, creating the question, generating the options, and seeking an answer. The user’s responses were edited to remove the keyword “done” that signaled completion. Finally, the responses for each part of the poll were graphically displayed on the UI for the student to view. The code below prompts the user for a question which is later repeated with the answer options.

```python
tts = gTTS(text="Say your question and use done to end", lang='en')
tts.save("speech.mp3")
playsound("speech.mp3")
```

2) Voting with OpenCV: After the poll is created, the program uses OpenCV to recognize the number of fingers a user holds up as an indicator for which option they are selecting. As the program processes the image frames streamed by the user’s video-camera, it is able to locate the hand’s position within the frame and identify fingers using basic trigonometry and built-in OpenCV methods.

   a) Experimentation of Hand detection: One of the main focuses for hand detection was not only the implementation, but also the optimization of the hand identification accuracy. The three main methods of hand detection that were developed were basic color filtering, facial detection-assisted color filtering, and background subtraction. To determine the optimal method, each one was assessed for reliability.

   The first iteration relied solely on basic color filtering in which color thresholding was used to differentiate the hand from the image background [33]. To improve the program’s accuracy response to different lighting, the image colorspace was first changed from OpenCV’s standard BGR protocol to HSV formatting instead. HSV pixel values are split into three separate channels for hue, saturation and value, where color occupies a single channel unlike BGR, where color information is contained inside all three channels. This conversion greatly simplifies the color range specification since HSV is able to adapt to mixed lighting environments with varying saturation and detect the hand as long as it is within the correct hue and color range.

   The minimum and maximum boundaries for the threshold range were defined to be [0,20,70] and [20, 255, 255] respectively. Then, the OpenCV inRange() function converted the image array from HSV to a binary (black and white) image. The regions within the frame that did not fall inside the specified HSV skin color threshold were assigned a value of 0 while those that did were assigned a value of 255 [35]. This binary mask that was created was able to display the filtered silhouette of the hand.

   For the next iteration, experimentation with a new method involved incorporating facial detection within the program, which would ideally be able to identify the square region of the image containing the user’s face, and then extract the color values from within the region to create a unique color range that is applied to the rest of the image to search for hands. To perform face detection, a pre-trained Haar Cascade Frontal face dataset model was accessed from the official OpenCV repository and passed into a cascade classifier function. This Haar feature-based approach was trained using positive and negative images to distinguish important edge and line features for facial tracking [36].

   7

Fig. 10. 3D cylindrical representation of the HSV colorspace. Source: Adapted from [34].

Fig. 11. Haar features being applied to a female face to identify various facial structures. Source: Adapted from [37].

   After locating the face, an image segmentation technique known as histogram back projection was used. The program generated a histogram that mapped out the range of hue and saturation values found in the detected face which was then projected onto the target region where the user placed their hand to search for similar skin-color pixel values [38]. The shape of the resulting matches were displayed in binary format and the colorspace was converted into grayscale so that the computational processing was less intensive for the rest of the procedure.

   The last iteration used a Gaussian blur and background segmentation algorithm sourced from a Github repository. From the videostream, a static background frame was captured and then subtracted from each proceeding frame in order to extract the distinct foreground features while eliminating unnecessary background elements [39], [40]. Using this technique as a form of motion analysis enabled the program to detect when a hand was placed in the foreground of the frame. To adjust the sensitivity of this detection, the background thresholding value was increased to 90 which filtered out more noise.
b) Convolution and Morphology Filtering: Once the hands were detected, the image was converted into binary format. Both convolutions and a form of non-linear filtering called morphological operators were applied to remove noise from the image and polish the hand silhouette to eliminate holes within the mask’s boundaries [41]. Several iterative layers of either an erosion filter, dilation filter, or both were applied to the frame [42]. These type of non-linear filters depend on the pixels’ relative ordering rather than their numerical values. The erosion operation trims the boundary and slightly reduces the size, while the dilation operation has the opposite effect and expands it. After the white noise was removed, a basic gaussian blur kernel was applied over the mask to refine it even further.

![Fig. 12. Side-by-side comparison of hand mask with no filters vs. with erosion, dilation and gaussian applied](image)

C) Finger Tracking: To improve the overall accuracy of the readings, a small, cropped section of the frame was defined to be the region-of-interest for detection. Reducing the area of the frame that the program was processing mitigated the potential noise interference from other parts of the background and specified a consistent region where the user could place their hand. Thus, the dimensions were reduced to a 50x50 pixel frame on the right side of the screen. To determine the number of fingers a user was holding up, a contour outline was created along boundaries of the hand, and a convex hull outline was drawn to capture the general shape of the hand [43]. Next, the convexityDefects() function was used to identify the defects, which were points along the contour with large distance deviations in relation to the hull, representative of the gaps in between fingers [44]. For each defect, start and end point, representative of adjacent fingertips, were defined along the hull, and the actual defect coordinate was chosen by finding the local minimum along the contour concavity.

Then, the distances between the points were calculated using the distance formula. The Law of Cosine was used to find the approximate angle opening for each gap to gauge whether it could be considered a valid defect or not, since the angle must be less than 90 degrees. The area within the hull and contour was also calculated separately to generate an area ratio that was used as a threshold when there was no detected defect to determine whether or not zero or one finger was held up. Using this process, the number of fingers, other than when voting for one, is always one more than the number of defects.

![Fig. 13. The figure above shows the convexity defects of the hands and the respective formulas used to calculate the angle between the fingers](image)

E. Polling Integration

Once the hand detection portion was fully functional, the next step was integrating it into the polling feature. Initially, the image frames of the student were processed on the receiving side (the teacher) to determine the number of fingers the student held up while the poll was active. When the teacher sent a message with the poll questions and options, the student’s side activated the necessary steps to prompt the user for an answer. This method did not involve sending the data of the result from the student to the teacher through sockets. However, processing the image on the receiving end led to serious issues with the accuracy and reliability of the finger detection as it was using individual frames that could have lag. As a result, the image detection was transferred to the sending side (the student). Although this required additional data to be sent from the student to the teacher to convey the results of the poll, the image detection was using the webcam of the student which led to higher reliability. Once the teacher received an answer to the poll from the student, a popup appeared with the results of the poll.

F. Web Search

Most learning platforms do not include features to easily access data from the web related to lessons taught in class. However, this learning platform integrates the voice-to-text subtitles with the ability to search certain terms or ideas. By selecting a certain phrase or word in the subtitles at the bottom of the interface, the program uses the selected text to automatically open up a new browser window with the specific search parameter previously determined by the user. This allows the user to quickly search up important ideas throughout the lecture while remaining attentive. This easy-
to-use feature eliminates confusion during the lecture, so the student can feel truly comfortable about the material covered.

```python
search = 'https://www.google.com/search?q=' + searchPhrase
webbrowser.open(search, new=1)
```

**G. Screen Sharing**

Screen sharing is a crucial feature that allows the teacher to easily communicate with their students in class. This program allows the student to see the teacher’s screen. This feature is enabled through capturing and sending the frames from the teacher’s device rather than the frames from the webcam when the screen sharing feature is enabled. Teachers can display lecture notes or share websites, which creates a more interactive lecture.

```python
screenCapture = array(PIL.ImageGrab.grab())
screenCapture2 = cv2.resize(screenCapture, dsize=(160, 120), interpolation=cv2.INTER_AREA)
preview = cv2.cvtColor(screenCapture2, cv2.COLOR_BGR2RGBA)
```

The above code uses the Pillow ImageGrab library in order to screen share through capturing the frames on an individual device.

**IV. RESULTS**

The outcome of this research was an interactive video conferencing platform. The following section demonstrates the usage of several features and presents the results of accuracy tests performed for speech and image recognition.

**A. Remote Server Connection**

Through the socket programming model, the application was able to connect both remotely through a LAN network and through a WAN network with port forwarding. The image below shows two devices connecting remotely outside the same LAN network. The application was converted to an executable file using PyInstaller which could be distributed and run without manually downloading the required Python packages.

**B. Web Searching**

The web search feature was seamlessly integrated into the subtitle feature within the overall program. Highlighting a section of text in the subtitles leads to a Google Search of the specific phrase.

![Fig. 14. Two users connecting remotely](image1)

![Fig. 15. Web search feature](image2)

**C. Screen Sharing**

The screen sharing feature is activated when either user clicks a button to toggle between displaying their screen and their webcam. The screen sharing feature can display on-screen notes and promote an engaging lecture-style class.

**D. Accuracy Testing of Speech Recognition**

To verify the functionality of the speech-to-text features, several accuracy tests were completed involving a diverse group of test subjects. After opening the application, the trial participants were given pre-written phrases that were phonetically diverse and were asked to speak into the microphone (either their personal headphones or the computer audio) to allow the program to transcribe the phrase using the NLP speech-to-text feature. The phrases transcribed by the feature were compared to the pre-written phrases. If the phrase was transcribed correctly, then the trial was recorded as a success. If there was any deviation in the transcribed phrase’s wording, then the trial was denoted as a failure and the deviation was
Fig. 16. Screen share feature with notes

recorded. This process was repeated not only for a subset of Native English speakers, but also for native and non-native Spanish speakers to verify the compatibility with the translation feature. Seven different test subjects spoke in one of two languages (English and Spanish) and over 100 phrases were tested throughout the trials.

The data was processed with a point system. All correct answers received 1 point. The averages of this converted data set provided an accuracy for different subsets of test subjects. The results for each group can be seen in Figures 17, 18, and 19.

After combining all the trial data, the tests suggested that the usage of the audio features were successful. The accuracy for native English speakers was over 90%. However, the incorrect results were failures of the NLP program to differentiate between certain homophones and correctly place apostrophes for plural possessives. Additionally, microphone usage increased the accuracy of results by almost twenty percent since the audio was passed directly into the microphone’s receiver in the headset. For speech recognition in Spanish, the accuracy rate was lower than English, but still well over fifty percent. This discrepancy may be due to the tendency of Spanish speakers to talk fast which can manifest as unclear speech. The results did not differ significantly between native and non-native Spanish speakers. Due to the way Spanish voice recognition was conducted, some words were exchanged with synonyms during the recognition process, but this did not change the meaning of the sentence. Additionally, tests showed that continuous speech was better recognized than fragments because the adjustment for background noise caused the program to miss the first 0.5 seconds of recording as it does this noise adjustment.

Finally, two of the test subjects used a version of the voice recognition software that did not include an adjustment for ambient noise. All other conditions were kept the same as the previous trials and all background noises were natural. Without an adjustment for background noise, the accuracy rate was dropped to 75% and sometimes, the program did not return any text transcription of the audio. When this occurred, the program could not accurately recognize any words due to the level of background noise. It is notable that this phenomenon did not occur a single time when using the voice recognition program with the adjustment for ambient noise.

E. Accuracy Testing of Hand Detection Methods

In order to assess the accuracy of detection, several factors had to be taken into consideration when evaluating the consistency of detection across different user environments. Some of the variables including background and lighting affected the overall accuracy. The detection capabilities were tested on individuals with different skin tones to verify its uniformity. For each detection method, several trial participants were instructed to hold up either one, two or three fingers for
three seconds, record the value that the program returned, and indicate whether it fluctuated or not. This was performed at different times of day to observe how each method responded to different lighting. The results were compared to determine which method to permanently implement into the polling sequence.

The collected data was then scored using a point system. All correct finger detection attempts earned one point while all incorrect, or fluctuating readings were given no points. The averages were taken of this converted data set to obtain the accuracy for different subsets of test subjects. The results for each of these methods can be seen in Figures 21 & 22.

<table>
<thead>
<tr>
<th>Method</th>
<th>Daytime</th>
<th>Nighttime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color filtering</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
</tr>
<tr>
<td>Facial-detection-assisted color filtering</td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td>Background subtraction</td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
</tr>
</tbody>
</table>

Fig. 21. Daytime vs Nighttime

![Image](image7)

Fig. 22. The figures above show the image and graphical comparisons of accuracy of all three methods in two different time instances of a day.

During the trials, the biggest issue with detection using simple color filtering was its inconsistency in picking up hand pixels in different lightings. A user working in a room with a background similar to their own skin tone would experience inaccuracies in readings since the values of the background would also pass through the mask and interfere with finger detection. Detection on users with skin tones entirely outside the predetermined range would not work at all since the HSV threshold range was only representative of a limited selection of skin complexities. For instance, this method worked relatively well in broad daylight, but when applied alongside an artificial light source in the evening, the program was unable to accurately detect hands without picking up excess noise from the background and the accuracy rate fell to 13.29%.

Similar to the first method, facial detection-assisted color filtering also had difficulty sensing a hand when the user’s hand and their background shared similar colors. Although it was able to discern the hand easily, since the color threshold was adjusted based on the face’s skin tone, significant quantities of noise was picked up from the background which caused fluctuations in the accuracy and made detection imprecise. The results were more consistent than basic color filtering, but detection worked correctly less than \( \frac{3}{4} \) of the time.

For the background subtraction method, users were asked to leave their hand up initially (to capture the background frames before detection occurred) and then remove it when prompted by the program for detection on the negative silhouette. The results for the background subtraction had the highest rates of accuracy in both lighting settings. Attempting hand detection using this method worked consistently and had a near perfect accuracy as long as no other features in the background moved. This method was chosen for final integration into the program.

In order to maximize the background subtraction method’s effectiveness within the program, another test was conducted to determine the timing of hand placement. The two options considered were having the user put their hand up during background capture and remove it before the detection period so that its negative static silhouette is processed, or having them leave their hand down initially and then raise it after the background image was captured. To verify that it worked well in different background environments, the process was repeated by placing the hand against a plain background, and then against a background with clutter. The user was asked to hold up either 1, 2, or 3 fingers multiple times and record whether the program returned the right amount of fingers. Raising the hand before background capture was 10% more accurate than raising the hand after capture. Additionally, the background clutter did not significantly affect the accuracy, although detection in backgrounds with less clutter were slightly more accurate.

![Image](image8)

Fig. 23. Accuracy for hand raised after
A. Summary

This platform strives to improve upon video conferencing platforms that are available for public use and address issues that are currently neglected. The final outcome was a telecommunications software targeted towards the education industry. The code for the full program is available at https://github.com/joosthemoos/gset2020vdlp. The implementation of an intuitive user interface allows for students of all ages to utilize this product. This is of incredible importance as students are less receptive to learning over a virtual platform. This is remedied through the application of voice-driven technology. Many people (including groups with certain learning disabilities) have trouble navigating commonly used video conferencing platforms, despite simplistic UI. However, verbal communication with the server enables these groups to still use features which would normally be a challenge to interact with. Moreover, the inclusion of object recognition allows students to simulate the classroom experience by involving physical interaction. Additionally, the live subtitles and ability to display a transcription of the lecture enables all students to have a better understanding of class information. This also appeals to the ESL (English as a Second Language) Community, with the option to translate English speech to Spanish. This could be easily adapted to multiple languages to address a wider audience.

B. Future Plans

This project was mainly used as a proof-of-concept for multiple novel features including a completely voice-driven polling system. The platform could be expanded in the future. Most notably, the server currently handles two users due to the computing power of a single device. More users could be added if this program was hosted on a cloud-based server with a larger computational capacity. For the screen sharing feature, the presenter could transmit both the video and screen contents at the same time through increasing the amount of data sent. Lastly, emotion tracking would be an effective feature that could allow the professor to gauge students’ understanding of the material, closely resembling the environment of an in-person class. This would require training a neural network to recognize both emotions and mental states like boredom and confusion. The aforementioned advancements would help create a more innovative educational platform.

ACKNOWLEDGMENTS

The authors of this paper gratefully thank the following: project mentor Jessica Kuleshov for her extensive experience; Residential Teaching Assistant Akila Saravanan for her invaluable support and efforts in coordination and communication; Research Coordinator Benjamin Lee and Head Counselor Rajas Karajigikar for their guidance and feedback; Program Director Ilene Rosen, Ed. D. and Associate Program Director Jean Patrick Antoine, without whom this project would not have been possible. Finally, the authors appreciate Rutgers University, Rutgers School of Engineering and The State of New Jersey as well as the generosity of the sponsors of the New Jersey Governor’s School of Engineering and Technology: Lockheed Martin, the NJ Space Grant Consortium, and NJ GSET Alumni.

REFERENCES
