AN INTELLIGENT VIDEO EDITING AUTOMATE FRAMEWORK USING AI AND COMPUTER VISION

Mei Yi Yang[1] and Yu Sun[2]

[1]Southpointe Academy, 1900 56 St, Delta, BC V4L 2B1
[2]California State Polytechnic University,
Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

The issue to be resolved was videos may be a difficult and time-consuming process to edit, specifically with cropping videos [4]. The solution that was implemented was a mobile application that was capable of cropping videos using a body tracking solution called MediaPipe Pose [5]. Using a landmark model that labeled the body parts of a person, the application can recognize where the person is located in the video frame by saving the minimum and maximum x- and y- coordinates. In the image array, the rows and columns not included within those coordinates are deleted, which leaves only the area with the person inside. To prove the effectiveness of the application in daily life, a survey was performed on fifteen participants. Each participant was shown the same video demonstration of the application being used, then the participants answered questions regarding how useful the application would likely be in daily life and how convenient the application would be to use. Results indicate that the general public would be willing to use the application as a long-term solution for video cropping [7].

KEYWORDS

Video Editor, Dancing, Video Cropping, Flutter.

1. INTRODUCTION

The idea of this application sparked when dancers faced the problem of not being able to edit their dance videos efficiently and aesthetically to post online. Users had to conduct a lot of research over a period of time to find all the methods available. Because they change positions when dancing, they have to film the frame fairly wide so it covers my entire range of motion. However, this makes the body really small in the video and very hard to see. When looking for apps that have the functions of having the frame follow the person in a video but there were no such products. Over the past few months, we have experimented with similar functioning applications. The popular video sharing platform TikTok has a body zoom filter which can be used when filming on the app [8]. However the flaw in this method is that the filter is exclusively used for filming in the application, meaning that a video cannot be uploaded to then apply the filter. Another flaw was that it zoomed very far in, which captured the full range of motion however is occasionally aesthetically displeasing. Another editing app that has a similar idea to what we were looking for was CapCut. This editing software allows users to upload videos and edit by hand. They have a feature that allows users to hand edit each frame they wish to zoom in or out and left or right. This allows the users creative freedom in how they wish the frame to track their body, however the major flaw in this is that it is very time consuming. The benefits of

David C. Wyld et al. (Eds): AI, AIMLNET, BIOS, BINLP, CSTY, MaVaS, SIGI - 2022
pp. 69-79, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121807

this app is that it is able to be used to film videos with the app that will automatically edit the video for the user as well as upload and process filmed videos. This product should be extremely relevant in solving many dancers' problems that are similar to mine.

Some existing video editors attempt to make video cropping easier. Video editors exist in which cropping is performed by typing in the minimum and maximum x- and y-coordinates of the video that should still be kept in [9]. However, this may be too much effort for an ordinary user, as the user will have to measure the resolution of the video, then estimate how many pixels they would have to crop out from each side of the video. By having a designated crop button and allowing the user to manually stretch and resize the region that should be kept in, the user may find this method of cropping videos more intuitive than the previous method.

One of the biggest issues with these general-purpose video editors, however, is that they are so feature-heavy that it can become difficult to navigate through them all. With so many buttons on the screen, these editors can be incredibly daunting for those who are new or inexperienced with video editing. Furthermore, more heavy-duty video editors generally require a strong machine to use, which can lead to those with older devices either running the video editor very slowly or not being able to run the video editor at all. The ideal video cropping solution is a lightweight application that is capable of video cropping with very little effort needed from the user [10].

A method that has been tested before is "smart" video cropping, which examines the video for what parts should be kept in the video before cropping automatically. Much of this was done to resolve the issue of unusual aspect ratios, which may have resulted in awkward cropping. For example, the video could be completely centered and cut off a person that is on the very side of the screen, but this new method could crop so the person shows up on the screen. However,many of these implementations were made more than ten years ago, and there may be newer technologies today that may be more accurate.

The tool that was created to tackle the issue of video cropping being a difficult and daunting process is a mobile application that automatically crops the video for the user. The mobile application was made using Flutter for the front-end code and Python for the back-end code. The main feature of this application is smart video cropping. The user of the application will simply be able to input a video and be returned a video that is cropped to have the person in the image be the focus of the video. This feature was intended for dancing videos, but it can also be applied to other videos that require the person of the video to be the primary focus. As some people can be too far away from the camera at the time of recording, such an application can refine videos and enhance the viewing experience of those who watch them. Compared to general-purpose video editors, this application is much more lightweight. However, the application lacks the multitude of features that come with full video editors. Since the application was designed for only one primary purpose, the lack of features is not too detrimental to the application. As this application was made with recent technologies, they may be more consistently accurate than implementations developed in the past.

In order to prove that the application would be effective and would provide benefits to the general public if it was widely released, a survey was conducted on Google Forms. Each participant would watch the same video demonstration of a person using the application and showing off each of the features. Then, the participants would answer questions that ask how useful the application is in daily life and how convenient the application would be to use. They will be provided with a scale from one to ten to answer each question, in which one is the worst rating and ten is the best rating. In case the participants have more thoughts they would like to share, they would be able to do so inside an optional free-response feedback section at the bottom of the survey.

The results of the survey could indicate if there are any particular aspects of the application that could use more work. They could also gauge how successful the application would be and how much of a positive impact it would potentially make on society. Convenience and usefulness are both very valuable qualities for an application to have. If the application is too inconvenient to use or requires too much time and effort to deliver the desired result, people will likely try to find alternatives. If the application provides no benefit or practical use in daily life, users may not be intrigued enough to download the application in the first place. Only when both qualities are done well in the application would users install the application and use it long-term.

The remainder of the paper will be divided into five sections, labeled from 2 through 6. Section 2 describes the hurdles that had to be overcome when coming up with ideas for the application and implementing the application. Section 3 offers an explanation for how the application as a whole, as well as specific sections of the application, was implemented. Section 4 analyzes how effective the application is using experiments that test the functionality of the application and the feedback from the general public, and Section 5 introduces several related works and how they compare to this work. The final section, Section 6, provides a conclusion that includes the summary of the application, some current limitations to the application, and what can be done in the future to resolve these limitations.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Coming up with an Idea for What the Application Should Do

The first challenge was coming up with an idea for what the application should do. The topic that was to be addressed was how dancing videos had the dancer too far away from the camera, which may detract from the quality or viewing experience of those who watch the video. At first, one idea that was considered was that the application should let the user choose a specific box in the video, and the application would crop out everything else in the video except the specified box. However, that would still take much effort from the users of the application, and they would likely not use the application again if it was too inconvenient for them to use. To address this, the video cropping idea was still utilized, but what was cropped out of the video could be decided based on an AI that detected where the person was in the video.

2.2. Implementing the Application itself

Another obstacle was implementing the application itself. While the ideas and concepts were already confirmed, applying those ideas and concepts to code and software was a much more challenging feat. While creating the front-end with Flutter took a relatively short amount of time, coding with Python was much more difficult, as it involved using a model that would be able to track the human body quickly. The model that was settled on was a detection model from MediaPipe Pose, which would use landmarks to predict where the specific body parts of a person were inside video frames. Using the detection model for the first time was confusing and frustrating, and there were not many examples of working Python code involving MediaPipe Pose that could be easily found online [11]. After much trial and error with the code, a working implementation was finally created. By retrieving the x- and y-coordinates of where the predicted landmarks were in the image array, a box could be created and the rest of the rows and columns in the image array could be deleted.

2.3. Coming up with an Experiment that could Effectively Test the Application for Relevant Information

A challenge that was also encountered was coming up with an experiment that could effectively test the application for relevant information. The first step in doing so was determining exactly what would be considered as "relevant information" [12]. Anything that would improve the chances of the application succeeding if it were to be widely released to the general public, such as practicality and ease of use, would be very important to gauge. In order to do so, a survey was used by gathering participants to view a video demo and asking them to complete a Google Form. The questions that the survey would have should ask for relevant information, which would allow the participants to share their thoughts regarding the convenience and usefulness of the application. However, there is the possibility that the participants have other feedback they would like to share that is not covered by the previous questions. Therefore, an optional free-response section was added to the survey as well.

3. SOLUTION

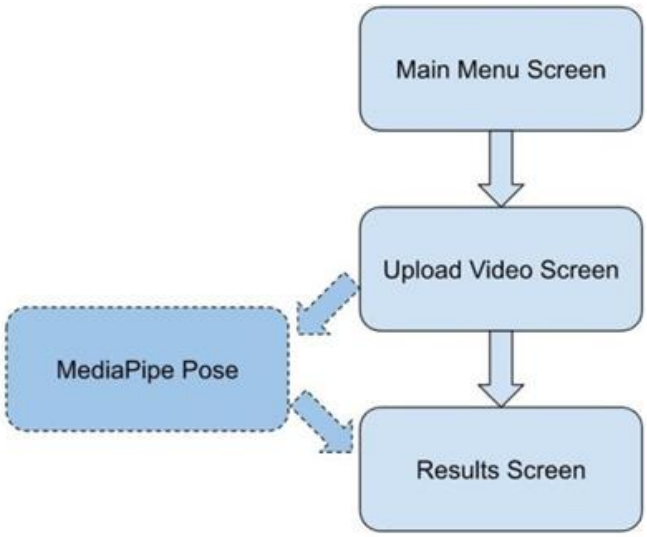


Figure 1. Overview of the solution

The application is comprised of three primary screens in Flutter. The first screen is the main menu screen, which is what the users of the application see when they first open up the application. On this screen, the words "Zoom in Dancing" are displayed as the title, and an image of a dancer is located below the title. Below the image is a single button labeled "Start". The second screen is the upload video screen, which can be accessed by tapping the start button on the main menu screen. On this screen, the users are provided with two options as to how they will input their video into the application. The first option is to select a video file from their library, which allows the user to choose a file that is saved in the device's storage. The second option is to access the camera from the application, in which users will have the opportunity to take a video of themselves dancing without having to exit the application and do so from the camera app. Once a video file has been selected or a video has been taken using the camera, the application reaches its third and final screen. This screen is responsible for displaying the resulting video, which will be cropped to place more emphasis on the person in the video. Such a feat is performed using Python back-end code, specifically with MediaPipe Pose, which is a body pose tracking solution that can detect human bodies in video frames.

The application was created using Flutter for the front-end code and Python for the back-end code. To access the different screens in the application, certain triggers were used [15]. To transition from the main menu screen to the upload file screen, the user would have to press the start button. The button is programmed so that when it is pressed, Flutter calls Navigator.push() which routes the application to a separate screen. In the case of transitioning from the upload video screen to the results screen, this same method is used, but it only triggers when the application has had a proper video file inputted into it, whether that be through selecting a file from the device or recording a video within the application using the device camera.

The functionality related to the cropping of the input video in the application comes from the Python back-end code. One of the most significant components of the application is MediaPipe Pose. MediaPipe Pose tracks body poses in videos using the landmark model. The landmark model inside MediaPipe Pose performs a prediction on thirty-three "landmarks" of the human body [6]. These landmarks range from areas on the limbs such as the elbows and knees to more specific body parts, including the pinkies and thumbs of each hand. Facial features are even included as some of the landmarks, such as the corners of the mouth, inner eyes, outer eyes, nose, and ears. Within the Python code, a minimum detecting and tracking confidence level of 0.5 is needed to determine the pose in the inputted video. Since MediaPipe Pose's models were designed with real-time detection in mind, the majority of modern-day smart devices possess the capabilities to utilize these models.

As for how the MediaPipe Pose is implemented into the application, the landmarks that the landmark model finds are processed and stored in a variable. Each of the landmarks is looped through. For the x-coordinate of each landmark, it is checked with the current minimum and maximum x-values and updated if the x-coordinate is lower than the current minimum or higher than the current maximum. The same procedure happens for the y-coordinates of the landmarks. This is mainly done to determine where the person is located in the video. With this information, the application will know what can safely be cropped out of the video. The minimum and maximum x- and y-coordinates that MediaPipe has detected are multiplied by the width and height of the screen to take ratios into account. Using built-in functions in MediaPipe, the landmarks that the landmark model has recognized are drawn out and indicated directly on the output video file. The outputted video file is created by taking each frame of the inputted file and saving the frame as an image array. When the time comes for the video to be cropped, the image arrays are retrieved from a list and the image arrays have only a portion of their rows and columns saved, based on the final screen height and width that MediaPipe has determined.

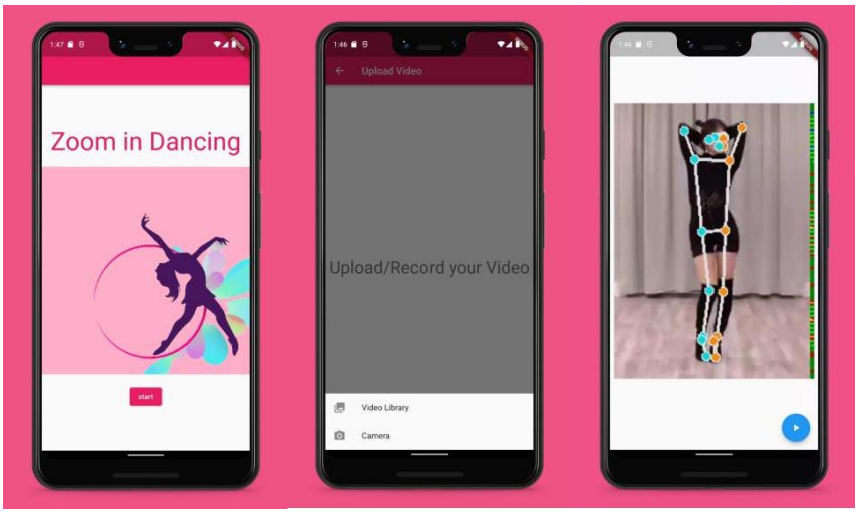
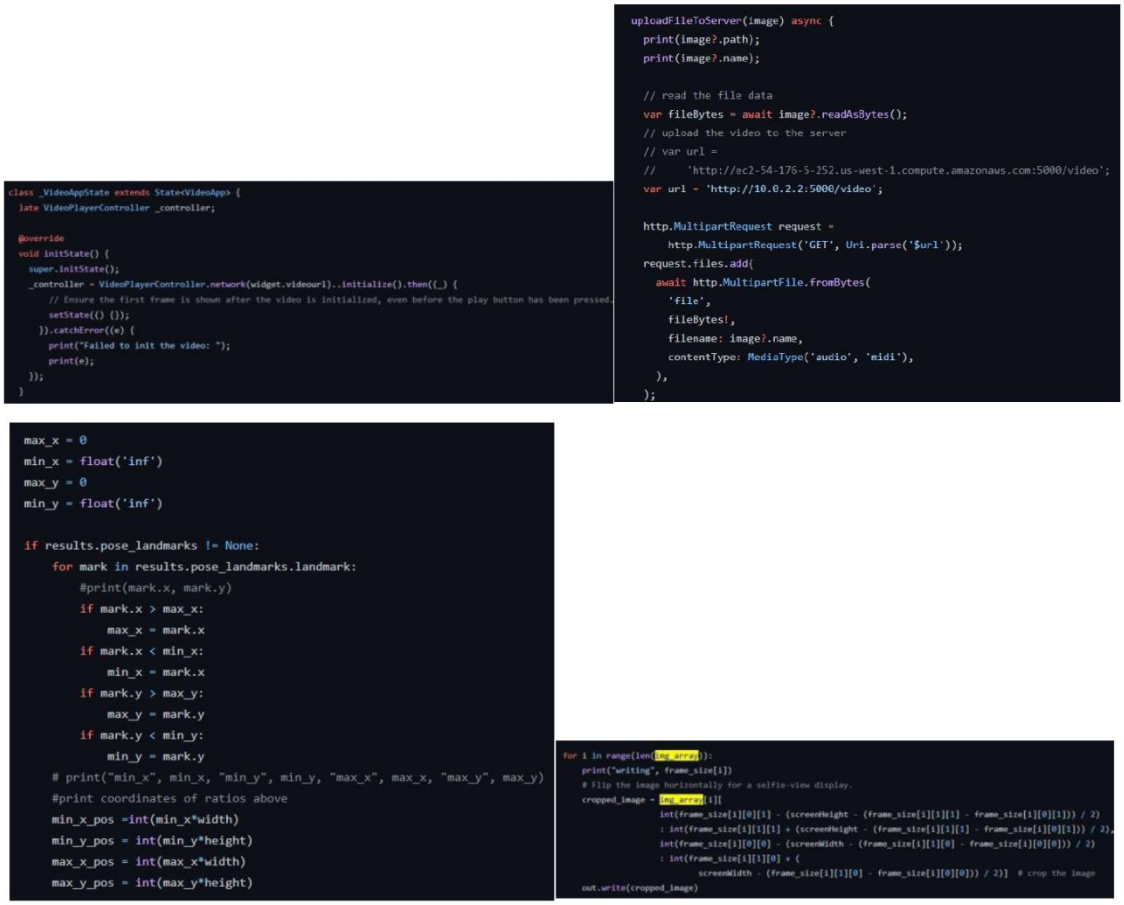


Figure 2. Screenshots from the video cropping application



```
class _VideoAppState extends State<VideoApp> {
  late VideoPlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = VideoPlayerController.network(widget.videourl)..initialize().then((_) {
        // Ensure the first frame is shown after the video is initialized, even before the play button has been pressed.
        setState(() {});
      }).catchError((e) {
        print("Failed to init the video: ");
        print(e);
      });
  }
```

```
uploadFileToServer(image) async {
  print(image?.path);
  print(image?.name);

  // read the file data
  var fileBytes = await image?.readAsBytes();
  // upload the video to the server
  // var url =
  //     'http://ec2-54-176-5-252.us-west-1.compute.amazonaws.com:5000/video';
  var url = 'http://10.0.2.2:5000/video';

  http.MultipartRequest request =
      http.MultipartRequest('GET', Uri.parse('$url'));
  request.files.add(
    await http.MultipartFile.fromBytes(
      'file',
      fileBytes!,
      filename: image?.name,
      contentType: MediaType('audio', 'midi'),
    ),
  );
```

```
max_x = 0
min_x = float('inf')
max_y = 0
min_y = float('inf')

if results.pose_landmarks != None:
    for mark in results.pose_landmarks.landmark:
        #print(mark.x, mark.y)
        if mark.x > max_x:
            max_x = mark.x
        if mark.x < min_x:
            min_x = mark.x
        if mark.y > max_y:
            max_y = mark.y
        if mark.y < min_y:
            min_y = mark.y
    # print("min_x", min_x, "min_y", min_y, "max_x", max_x, "max_y", max_y)
    #print coordinates of ratios above
    min_x_pos =int(min_x*width)
    min_y_pos = int(min_y*height)
    max_x_pos = int(max_x*width)
    max_y_pos = int(max_y*height)
```

```
for i in range(len(img_array)):
    print("writing", frame_size[i])
    # Flip the image horizontally for a selfie-view display.
    cropped_image = img_array[i][
                    int(frame_size[i][0][1] - (screenHeight - (frame_size[i][1][1] - frame_size[i][0][1])) / 2)
                    : int(frame_size[i][1][1] + (screenHeight - (frame_size[i][1][1] - frame_size[i][0][1])) / 2),
                    int(frame_size[i][0][0] - (screenWidth - (frame_size[i][1][0] - frame_size[i][0][0])) / 2)
                    : int(frame_size[i][1][0] + (
                            screenWidth - (frame_size[i][1][0] - frame_size[i][0][0])) / 2)]  # crop the image
    out.write(cropped_image)
```

Figure 3. Screenshots of the video cropping application code

4. EXPERIMENT

4.1. Experiment 1

The experiment to test the effectiveness of the video cropping application is a survey created in Google Forms. Fifteen participants were gathered to take the survey, which is a large enough sample size to mitigate the effects of any variability. First, each participant was asked to watch a video demo of the application and how it would affect an inputted video. Because every participant was given the same video demo, there are fewer confounding variables in the experiment. Then, the participants would be given the link to a Google Forms survey. The first question asks how practical and useful the application seemed on a scale from one to ten.

Participant Number	Usefulness Rating of Application
1	7
2	7
3	9
4	7
5	8
6	6
7	8
8	10
9	8
10	8
11	8
12	9
13	7
14	8
15	6
Average	7.73

Figure 4. Table of participant number and usefulness rating of applicant

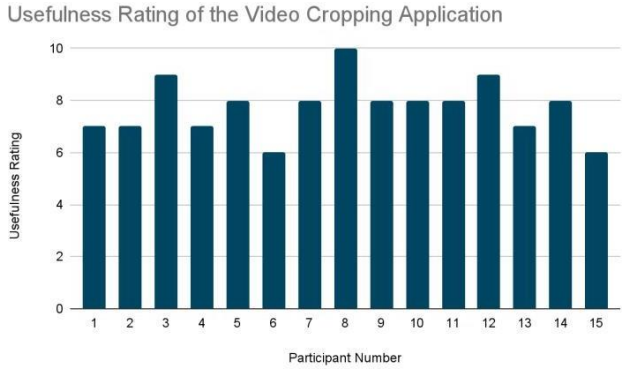


Figure 5. Graph of usefulness rating of the video cropping application

The responses from the first question were generally positive. All of the usefulness ratings from the participants were 6 or higher, and the average of the ratings was a comfortable 7.73. This could indicate that if the application were to be widely released, the general public may perceive the application as a tool that could reliably crop videos in a meaningful way that enhances the viewing experience of those who watch the refined videos. With the video demo that was . In the feedback that was provided by the participants, many said that this would be a helpful tool. Even those who claimed they were proficient and had experience in video editing admitted that this application could save a lot of time and effort in certain situations. However, one participant

expressed concern about how the application may not always be accurate when it comes to cropping the video in the right areas, and manually cropping the videos may be a more consistent choice.

4.2. Experiment 2

The second question from the Google Forms survey asked whether the application would be convenient and easy to use. Just like the previous question, this question would use a scale from one to ten for the participants to choose from. For both questions, a score of one indicates that the application completely lacked that specific quality, while a score of ten indicates that specific quality was done extremely well. Because an optional free-response section would be placed at the end of the survey, participants will have more freedom in expressing their thoughts and opinions, as the first two questions are quite limited in gathering the participants' full thoughts.

Participant Number	Convenience Rating of Application
1	6
2	7
3	10
4	8
5	6
6	7
7	8
8	10
9	9
10	7
11	8
12	7
13	8
14	8
15	6
Average	7.67

Figure 6. Table of participant number and convenience rating of applicant

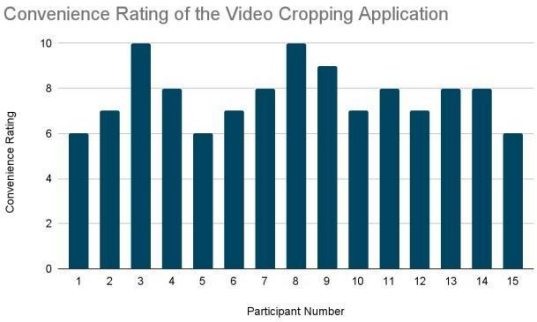


Figure 7. Graph of convenience rating of the video cropping applicant

From the results of the second question, participants were generally pleased with how intuitive the application was to use. The convenience ratings were similar to the usefulness ratings of the application, since all the ratings were 6 and above and the average was at a fairly high 7.67. However, the convenience of the application may have been reduced by the addition of too many features. The feature that emphasized the points where the application recognized the human body appeared to be particularly polarizing among the participants. While some stated in their feedback that they simply thought of such a feature as a nifty little bonus, others thought that it provided too much unnecessary information that the average user of the application simply had

no use for, as it only seemed to get in the way and people would normally already be able to tell where a person is in the video when watching it.

According to the results of the first experiment, the application can be a great help to many people around the world. The participants are regarded as a small sample of the general public, which means that the general public can make use of the application, particularly those that make dance videos or other similar videos that require cropping. A fairly high score on the usefulness of the application was to be expected, as the primary purpose of creating the application in the first place was to improve the quality of life for people who have recorded videos that could be improved through an automatic cropping system.

The results of the second experiment indicate that application will be able to be used by many without any steep learning curves and without much effort. This is significant for the success of the application, as users will likely not spend the time to learn how to use a complicated application and would prefer to use an application that requires minimal interaction and effort to reach a final product. This was also an expected result, as the general public was kept in mind during the development of the application.

5. RELATED WORK

One related work highlights how video editing can be a difficult process, and a new video editor named Silver was developed to address that. Research indicates that the use of metadata can help video editor tools become more helpful to the users, such as resolving inconsistencies in audio and video boundaries [1]. The related work is similar to this work in that tools were created in the hopes of making a complicated-seeming process like video editing become less daunting and more convenient to use. However, while the related work places more focus on video editing as a whole and dual channels of video and audio, this work focuses more on the idea of video cropping using a body detection model.

Another related work presents a way to convert video files to different aspect ratios. This would be particularly useful for devices that have screens with uncommon aspect ratios. A trained scoring algorithm is used to figure out what the screen should focus on most when cropping a video [2]. This related work is very similar to this work in that smart video cropping is performed using the help of advanced back-end code. The related work provides a large emphasis on varying aspect ratios fitting cleanly on a screen. On the other hand, this work presents an application that chooses to crop in order to focus on a person in the video.

A third related work also focuses on different aspect ratios and how to prevent the issue of awkward video crops. The solution that is presented is a video retargeting method that makes use of critical regions to determine areas of the video that should not be cropped out [3]. This related work also shares a similarity with this work in that a video cropping method is used to include important portions of the video. The related work focuses on critical regions to figure out what should stay inside the final video, while this work utilizes a person detection model involving landmarks.

6. CONCLUSIONS

To tackle the issue of people not being able to easily crop videos, an application was created that can automatically detect the person inside a video and crop out the rest of the video so that the person is the primary focus of the video. This application was mainly intended for dancing videos, but it can also be applied to other types of videos as well. The application was built using

Flutter and Python, and the application made use of a person detection model from MediaPipe Pose that tracks the “landmarks”, or body parts, of a person in video frames [13]. Using the coordinates of the landmarks, the application deletes the unneeded rows and columns of the image arrays to create the final video product.

The experiment that was performed on the application was a survey. Fifteen participants watched a video demonstration of someone using the application and showing off the features in the application. Then, the participants were asked to fill out the survey, which had two main questions. The first question asked how useful the application seems in daily life, and the second question asked how convenient using the application would be. According to the results, the participants seem to generally agree that the application would be useful and convenient based on the video demo, which can indicate that the application would likely be used by people long-term. This could also solve the issue of cropping videos in the future, and an application like this may potentially lead the way for more applications implementing a tool or feature similar to this in the future.

A major limitation in this program is that it has difficulty in processing videos containing more than one person. Running any video with multiple people can result in a lot of errors because the software is only built to acknowledge one body at a time. If multiple are present, the video product tends to jump from one person to another randomly. Another aspect that could be improved is video quality and processing speed. Additionally to improving the two, options for each could be implemented so the user could select the size of the file they wish to export.

With more development, the application could have more features to be implemented to appeal to the dance audience specifically such as special editing features, filters, effects and sharing features. For example, the user could choose to crop their video to an ideal size for TikTok, Instagram reels, Youtube, and other platforms [14].

REFERENCES

[1] Casares, J., Long, A. C., Myers, B. A., Bhatnagar, R., Stevens, S. M., Dabbish, L., Yocum, D., & Corbett, A. (2002). Simplifying video editing using metadata. Proceedings of the Conference on Designing Interactive Systems Processes, Practices, Methods, and Techniques - DIS '02, 157 – 166. https://doi.org/10.1145/778712.778737

[2] Deselaers, T., Dreuw, P., & Ney, H. (2008). Pan, Zoom, Scan - Time-coherent, trained automatic video cropping. 2008 IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2008.4587729

[3] Wang, Y.-S., Lin, H.-C., Sorkine, O., & Lee, T.-Y. (2010). Motion-based video retargeting with optimized crop-and-warp. SIGGRAPH '10, (90), 1–9. https://doi.org/10.1145/1833349.1778827

[4] Kopf, Stephan, et al. "Automatic scaling and cropping of videos for devices with limited screen resolution." Proceedings of the 14th ACM international conference on Multimedia. 2006.

[5] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." arXiv preprint arXiv:1906.08172 (2019).

[6] Zhang, Fan, et al. "Mediapipe hands: On-device real-time hand tracking." arXiv preprint arXiv:2006.10214 (2020).

[7] El-Alfy, Hazem, David Jacobs, and Larry Davis. "Multi-scale video cropping." Proceedings of the 15thACM international conference on Multimedia. 2007.

[8] Montag, Christian, Haibo Yang, and Jon D. Elhai. "On the psychology of TikTok use: A first glimpse from empirical findings." Frontiers in public health 9 (2021): 641673.

[9] Song, Hwa-Sun. "Job analysis of video editors based on the DACUM method." The Journal of the Korea Contents Association 7.12 (2007): 95-104.

[10] Deselaers, Thomas, Philippe Dreuw, and Hermann Ney. "Pan, zoom, scan— time-coherent, trained automatic video cropping." 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008.

[11] Denning, Dorothy E. "An intrusion-detection model." IEEE Transactions on software engineering 2 (1987): 222-232.
[12] Ehrlich, Danuta, et al. "Postdecision exposure to relevant information." The journal of abnormal and social psychology 54.1 (1957): 98.
[13] Michon, Pierre-Emmanuel, and Michel Denis. "When and why are visual landmarks used in giving directions?." International conference on spatial information theory. Springer, Berlin, Heidelberg, 2001.
[14] Che, Xianhui, Barry Ip, and Ling Lin. "A survey of current YouTube video characteristics." IEEE MultiMedia 22.2 (2015): 56-63.
[15] Morrison, Alison. "Entrepreneurship: what triggers it?." International Journal of Entrepreneurial Behavior & Research (2000).