Frame-Skipping and Interpolation Algorithm for Efficient Data Processing

Author: Atishay Singh
Project Advisor: Dr. Carol Flannagan
Sponsors: DEVIATE, UMTRI, Federal Highway Institute

Date: May 2, 2021

Introduction

When processing image and video data for machine learning algorithms, a major obstacle that many research teams face is having to manually label the collected data before training can begin. This process can be incredibly slow, taking away valuable time and resources away from other necessary tasks, such as developing and tuning the machine learning algorithm or collecting a larger, more diverse set of data for testing. This, in turn, can lead to poor model performance during training and testing. In addition, much of this effort is often spent on labeling consecutive, nearly identical frames, which is redundant and adds little new information for the model to work with. The goal of this project is to expedite the labeling process by using key features within a video to detect and extract unique, high-information frames for manual labelling and propagating those labels to the rest of the dataset through interpolation. In this paper, the implementation, testing protocol, and observed results for this frame-skipping and interpolation algorithm are described, and further improvements and extensions are discussed.

Motivation

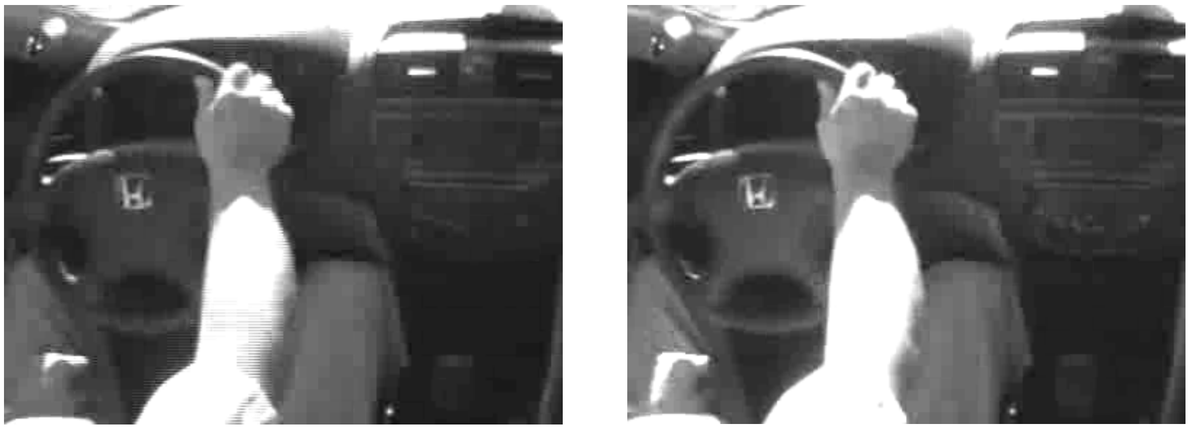
One major bottleneck that DEVIATE faces, as do a lot of small machine learning and computer vision research teams, is the process of data labelling. When working on a specific research task, especially one that requires collection of data rather than use of publicly available datasets, it is often necessary to label each frame or image in the image or video datasets by hand in order to ensure that the objects of interest are properly labelled throughout the dataset. Since most machine learning and computer vision models need to be trained on large datasets to achieve a reasonable degree of accuracy, this process can be time-consuming.

While large research teams can afford to hire dedicated labelers or crowdsource their labelling tasks, small research teams often do not have this option. As such, these teams often have to choose between labelling and processing additional data or working on refining some other aspect of the model. In the long run, this results in machine learning models that underperform, either during the training process or upon use in a real-world application, since the models may not have been trained on a wide variety of data.

The goal of this capstone project is to expedite the labelling process by automating it, allowing researchers to incorporate a wider variety of training data or work on other aspects of the model. The approach pursued for this project focused on *frame-skipping*, which involves detecting and skipping low-information frames.

Frame-skipping

One major source of inefficiency in labelling comes from the labelling of low-information, nearly identical frames. Since cameras generally capture several frames per second, the change in the filmed environment between frames can be very small, even if there is motion in the environment. In addition, since DEVIATE works with video data of driver behavior, many of the videos feature stretches in which the driver and environment are almost perfectly still. Manually labelling these stretches takes time and adds very little new information when it comes to training a model. In other words, labelling these non-unique frames can be a major time sink, especially when taken across a dataset of hundreds of videos. As such, being able to detect and separate these redundant frames from the important frames could speed up the labelling process greatly. Figure 1 shows an example of two frames within a video, taken 20 frames apart, for which the environment is nearly identical. In fact, the environment was almost identical throughout the 20 frames.



Frame 160 Frame 180

Figure 1: Two frames, extracted from a single video from the IVBSS dataset.

The focus of this project is on designing and implementing an algorithm that can extract key features within images and track these features reliably throughout a video. These features do not necessarily have to be the objects of interest being labelled in each frame, but they must be relatively simple to detect and track from frame to frame. The motion of these features will be tracked throughout the video, and frames for which there is significant motion observed for a user-defined threshold proportion of the key features will be marked as unique and saved for manual labelling. Once the labels are generated for these frames, they will then be propagated to the rest of the frames through an interpolation process, allowing for use of the full dataset without having to spend the time to manually label each frame.

The main drawback to the process described above is that the accuracy of the propagated labels will likely be lower than if all of the frames were labelled by hand. However, if the error is small enough relative to the size of the image, it is likely that the loss in accuracy will not greatly affect the performance of a model trained on interpolated label data. In addition, should the algorithm succeed in significantly reducing the number of frames that need to be manually processed, the loss in accuracy would be offset by the ability to spend more time on incorporating a larger volume of data or on tuning the model itself. Much of the work done for this capstone project has been to assess the degree to which label interpolation affects accuracy and performance, and whether or not this change in performance is justified by the time saved through this process.

Implementation Details

The algorithm, as described above, is made up of two main components. The first, the *frame-skipping* component, processes a video or a set of videos, saving the unique frames to a specific directory and the list of unique frames to a file. These outputs are then handed to the labelling team, who will process and provide label information for the unique frames. Once the unique frames are labelled, the labels and the list of unique frames are used as inputs for the *interpolation* component, which propagates the labels throughout the rest of the video by interpolating the label coordinates to the previously unlabeled frames.

I tested several frame-skipping and interpolation combinations in order to determine which approach performs the best in terms of label accuracy. My experiments and final algorithm approach are described below, and the results are described in the Experimentation section.

Frame-skipping

At a high level, the frame-skipping algorithm takes as input a video file and three threshold parameters, the **match threshold**, the **distance threshold**, and the **similarity threshold**, used to determine whether any given frame is unique or redundant. First, the initial and final frames of the video are labeled as unique. Starting from the second frame, the algorithm iterates through each frame of the video, detecting and extracting key point information for the latest frame and the last detected unique frame. The algorithm then attempts to find matching key points between the two frames, computing the proportion of key points detected in the last unique frame for which there were matching key points in the latest frame. If this proportion is lower than the **match threshold**, the algorithm labels the latest frame as unique, since the environment has likely changed significantly enough that very few matches can be detected. Otherwise, the algorithm iterates through the matched key points in both frames, using the pixel coordinates of the matching key point pair computing the Euclidean distance between the positions of the key points. If the distance is larger than the **distance threshold**, then the key point is labeled as mobile, and is otherwise labeled as static. The proportion of static key points to the total number of matched points is computed and compared to the **similarity threshold**. If the proportion is lower than the similarity threshold, the frame is labeled as unique, and is otherwise labeled as redundant. The frame number of each unique frame is written to an output file, and the unique frames are saved to an output directory to make it easier for labelers to process the correct frames.

My initial frame-skipping implementation approach involved the use of OpenCV's ORB key point detection library. This library, described in further detail in [1], essentially detects corners, edges, and other easily identifiable features within an image by using a combination of image filtration and edge detection techniques. While there is very little control over what kinds of points are identified, the library does have built-in support for finding matching key points between sets of images, which does help in tracking key points throughout a video. In addition, for each key point detected, the ORB library saves a large descriptor vector based on the pixel location and different visual characteristics of the key point and the surrounding image. Given these capabilities, using the ORB library was a good starting point for the frame-skipping algorithm. Figure 2 shows a few examples of images processed using the ORB detection library.

After implementing the ORB key point based frame-skipping algorithm, I performed some research into alternate key point detection methods, ultimately implementing an alternate frame-skipping algorithm using the Shi-Tomasi key point extraction library. Like the ORB detection library, the Shi-Tomasi library, described in [2], extracts and marks corner points in images. However, while the algorithm is often used in object tracking algorithms, the library does not have built-in point matching algorithms. Most of the research I did seems to indicate that points are generally matched by computing Euclidean distance or other complex distance calculations using the pixel locations in subsequent frames. Unlike the ORB key point descriptors, descriptors for Shi-Tomasi key points only save the pixel locations of the detected points. This approach could result in inaccurate key point assignments, especially when dealing with a large number of key points.

Interpolation

Once the unique frames detected using the frame-skipping algorithm are labeled, the list of unique frames, their corresponding labels, and the desired degree of interpolation are accepted as inputs to the interpolation algorithm. The algorithm loops through the unique frames, computing a curve of best fit based on a subset of unique frames and using that curve to estimate the label positions for all of the redundant frames between the unique frames used. The size of the subset used depends on the degree of the interpolating curve. For example, computing a linear curve requires two unique frames, while a quadratic or cubic curve would need three or nine unique frames, respectively.

The interpolation algorithm was designed to propagate labels corresponding to single points and bounding boxes. The testing performed for this project involved single point labels, but the interpolation algorithm is also capable of interpolating bounding box positions by estimating the positions of the upper left and lower right corners of a given bounding box.

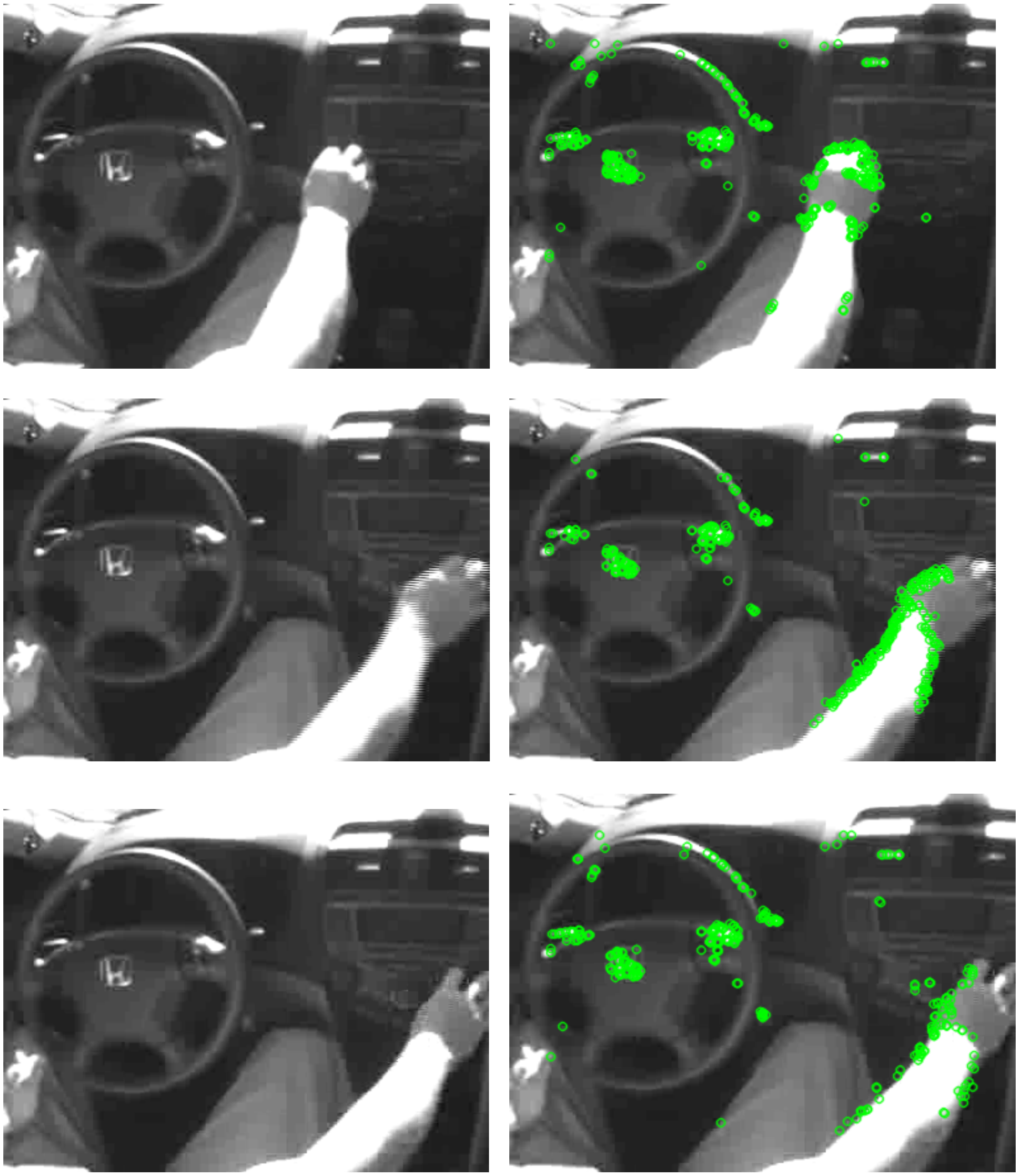


Figure 2: Frames from a cabin view video, before (left) and after (right) ORB key point processing. All ORB key points are marked in green.

To implement the interpolation algorithm, I used the SciPy interpolation library. The library is capable of handling linear, quadratic, and cubic interpolation, so I tested the algorithm with all three to see which worked the best. The results of this testing are described in the next section.

Experimentation

In order to test the frame-skipping and interpolation algorithms described above, I processed 51 videos from the IVBSS dataset, collected from face cam views of a series of volunteer drivers. The objects of interest in these videos, all of which were labelled for each frame using a pretrained neural network, are five facial features, including the eyes and the corners of the mouth. The frames in each video have dimensions of 300x448 and are in black and white. Figures 1 and 2 show frames from cabin view videos in the IVBSS dataset.

Testing Protocol and Evaluation

The setup for the algorithm is largely the same as described in the Implementation section. The key difference is that upon processing a video and extracting the unique frames using the frame-skipping algorithm, the corresponding labels for that video are used instead of manually generated labels. Upon interpolating these labels to the rest of the frames in the video, the results are compared to the corresponding neural-network generated labels, which are treated as the ground truth for testing.

As discussed previously, the frame-skipping algorithm accepts three threshold parameters corresponding to the proportion of matched points to total points required to process a frame, the distance threshold between matching points that determines whether an individual point is considered moving, and the minimum proportion of mobile matched points to total matched points required to consider a frame as unique. Since the values of these thresholds could differ from video to video due to a number of factors, such as lighting and testing environment, I set up a grid search algorithm that searches over a range of values for each parameter, testing all possible combinations over that range given a step size for each variable. The first threshold parameter mentioned above was hard-coded at 0.40, to make the testing process quicker. Aside from the defined range, the only constraint on the threshold values was that the proportion of ground-truth frames used could not exceed 80% of the total number of frames, to prevent the algorithm from considering low-error solutions which do not skip many frames.

In general, the goal of these experiments was to evaluate the algorithm described here in terms of accuracy and time taken. Since the labels for each object of interest are given here in the form of pixel coordinates, the interpolation error is measured by computing the Euclidean distance between the ground truth labels and the interpolated labels for the frames marked as redundant. For each video, I recorded the average error per frame, including and excluding the unique frames, the ideal threshold parameters for the video, and the proportion of frames labelled as unique. This information allows me to assess both how accurate the interpolation is and the time and effort saved by processing the videos through the frame-skipping algorithm rather than labelling everything by hand.

In order to assess the time taken per video, I recorded the number of frames and the time taken, in seconds, to process each video. This allows me to explore the relationship between video length and time taken to process the video, and to see if the algorithm can run quickly enough to work for a real-world application.

Ideal Setup

The first round of experiments for this algorithm involved testing different combinations of frame-skipping and interpolation approaches to see which combination worked the best. As discussed earlier, when deciding which frame-skipping implementation to use, I found that the ORB detection library handles detecting and tracking matching key points far better than the Shi-Tomasi library, since the Shi-Tomasi corner detector only returns information about the position of the corner and does not have an implementation for point matching. As such, I ultimately decided to use the ORB key point detection approach.

For the interpolation algorithm, I tested linear, quadratic, and cubic interpolation methods, using the SciPy library, across a set of 20 videos. In general, I found that the linear interpolation approach had a much lower average error than the other approaches. When attempting quadratic or cubic interpolation, I found that the derived fitting functions often gave unrealistic estimations for label locations, including coordinates that were far beyond the bounds of the image. One possible explanation for this is that the fitting algorithm is unable to properly handle sudden head turns or other motions, fitting incredibly steep quadratic or cubic functions to account for these movements. In any case, the linear interpolator worked far better than the alternatives.

Time Performance

Through my experimentation, I found that the frame-skipping and interpolation algorithms are capable of processing videos very quickly. Generally speaking, I found that the interpolation algorithm took under a second to run, regardless of video length, making the frame-skipping algorithm the bottleneck as far as time is concerned. Figure 3 shows a plot of the time taken to run the frame-skipping algorithm on each video, as a function of the number of frames.

As seen in Figure 3, the relationship between the video length, in frames, and the time taken to process the video is linear. As seen in the line of best fit provided in Figure 3, the algorithm is capable of processing roughly 78 frames per second, regardless of the overall length of the video. To put that in perspective, the longest videos tested, at roughly 13 minutes, took under two minutes to process. Therefore, in terms of time, the algorithm performs incredibly well.

Despite this performance, however, there are two important caveats to note. The first is that while an individual run takes very little time, the parameter grid search described above requires multiple runs of the algorithm per video in order to find the optimal set of parameters for each video. For example, since we tested three different values for the distance threshold and five different values for the mobile point threshold, there are a total of 15 possible combinations of parameters per video, meaning each video could take anywhere between a few seconds and half an hour to fully process with the optimal parameter set. Unfortunately, despite testing extensively, I was unable to find a set of parameters that works reasonably well universally, nor was I able to find any pattern that would make it easier to determine the optimal parameter values more quickly for each video.

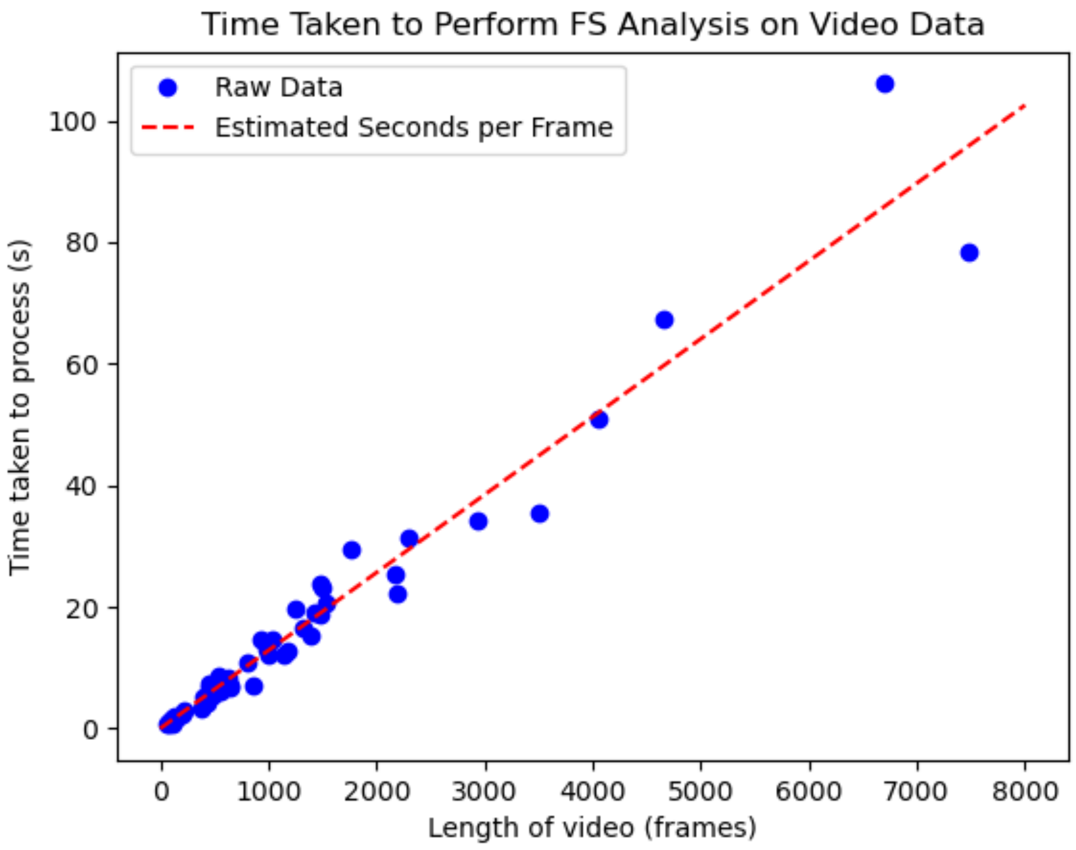
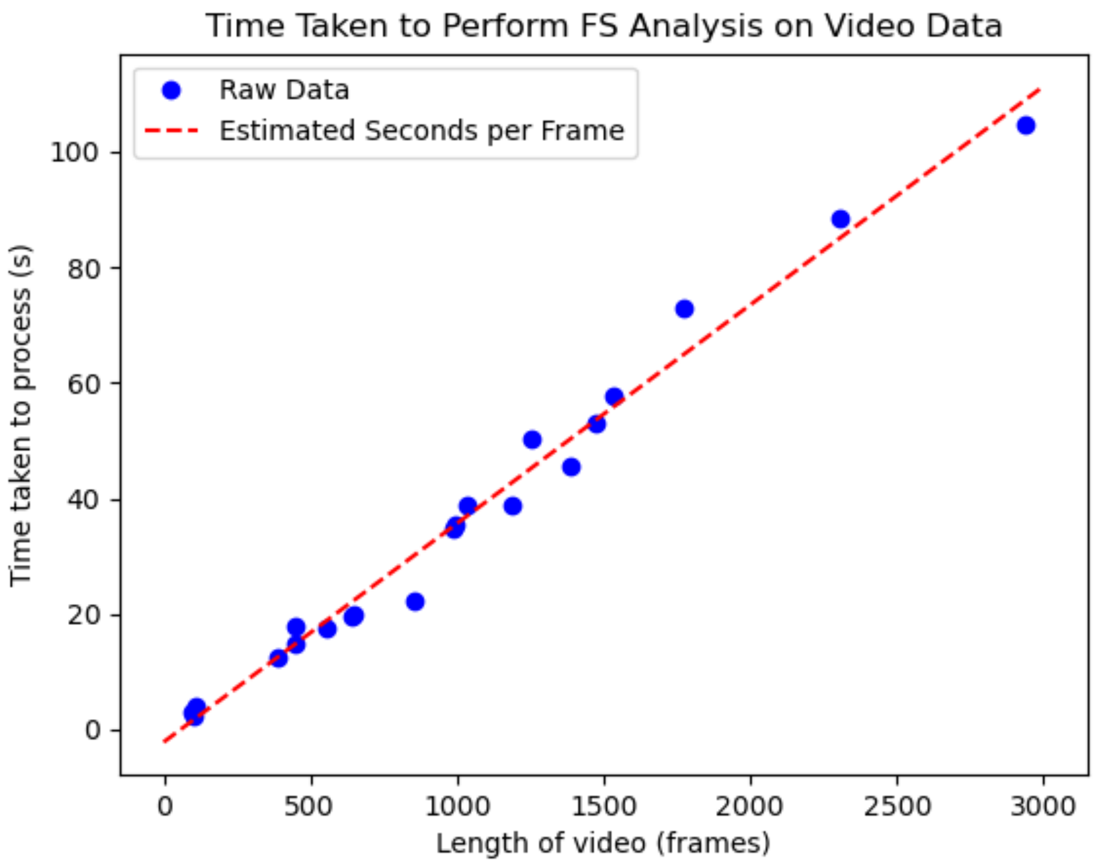


Figure 3: Time taken to perform the frame-skipping processing per video

The second caveat is that while the testing gave consistent time results most of the time, there were some significant anomalies in which the performance was far below average. During one early round of testing on a subset of 20 videos, I found that the algorithm was only able to process 25 frames per second, indicating that the frame-skipping process was taking over three times as much time per video as reported above. While the slower value was reported during the capstone presentation, prior and subsequent testing has consistently shown that the frame-skipping algorithm performance is generally far closer to the faster performance described above. Since all of the tests conducted include these 20 videos, and since the algorithm had not changed significantly between these tests, it is unlikely that the slow performance described here is anything other than an anomaly. It is possible that the testing server was in use by other researchers during this round of testing, which would result in slower performance since the algorithm would not have had access to the full computing power of the server in this case. For reference, Figure 4 shows the time performance observed during earlier tests.



Accuracy and Benefits

As expected, I did see some loss in accuracy when comparing the interpolated ground truth labels to the original labels. The interpolation error for each video, on average, is a distance of 30 pixels per frame, and the median error is a distance of 25 pixels per frame. Given the dimensions of the frames, this is an error of roughly 6%. This is an acceptable level of error for the benefits that the procedure provides in terms of reduction of the labeling effort.

On average, the frame-skipping algorithm labels roughly half of the frames as unique, meaning that labelers would only have to label half of the frames for each video tested, on average. This is a significant reduction in the amount of effort required to label each video. In addition, it seems that the length of a video does not have a strong correlation with the proportion of frames needed to label a video, meaning that the results described here hold true even for larger videos. As such, this algorithm could potentially reduce the time needed to label to half of the time it would have taken to manually label each frame.

The performance of the algorithm can vary significantly from video to video. For example, while the mean error per video is a distance of 30 pixels between the target and the interpolated label, the maximum error is a distance of over 100 pixels, and the standard deviation is relatively large. By analyzing the videos for which the algorithm performed poorly, I found that these videos often feature a lot of sudden movements or changes in lighting, which may explain the poor interpolation performance.

Metric	Interpolation Error (excluding ground truth frames)	Interpolation Error (including ground truth frames)	Percent of ground truth labels used
Mean	31.00	16.51	51.21%
Median	25.09	10.29	59.76%
Range	7.17 - 105.93	1.80 - 58.67	8.84% - 79.28%
Standard Deviation	20.46	14.93	21.36%

Table 1: Interpolation Error. Error for a single video is calculated by computing Euclidean distance between ground truth and interpolated label for each frame and object of interest and then taking the average of these distances. The values above are derived by computing the corresponding metric across the testing dataset of 51 videos.

Extensions

While the results described above look promising, there are some concerns left to be addressed regarding the interpolation accuracy, the large variance in performance from video to video, and how the observed results could affect machine learning models trained on data processed using the frame-skipping and interpolation algorithm, rather than being manually labeled. The extensions described below are intended to address these concerns.

Training Using Interpolated Data

Ultimately, the goal of the project is to make image and video labeling more efficient to allow researchers more time to work on training or improving their machine learning models. As such, in order to fully assess the feasibility of using the frame-skipping algorithm in place of manual labeling, I am currently working on training a machine learning algorithm using the interpolated labels generated during the experimentation phase and comparing its performance to the same model trained on the ground truth data. While I was unable to complete this phase of experimentation this semester, I am currently working on setting up and performing testing with machine learning algorithms.

One area of application that DEVIATE focuses on is assessing driver behavior and potential distractions while driving. In order to study the use and effects of cell phones while driving, the team has developed a machine learning model that accepts the positions of facial features as inputs and outputs an activity label corresponding to whether or not the driver is calling someone or otherwise interacting with a cellphone. I am currently working on adapting this model to accept my interpolated facial feature labels as inputs during the training process in order to assess how this affects the prediction accuracy of the machine learning model. The performance will be compared to that of the same model trained on the ground truth data. If I find that the accuracy of the model trained on interpolated label data is comparable to that of the model trained on ground truth data, I would then conclude that the frame-skipping algorithm, in its current state, could be used in place of manual labelling. Even if the performance is slightly worse, the time saved on labeling could be used to improve the model or incorporate larger datasets during the training process, potentially offsetting any loss in prediction accuracy.

Background Point Detection

When looking at Figure 2, one trend that becomes apparent is the consistent detection of static background points. For example, in the frames depicted in Figure 2, there are a significant number of key points detected on the steering wheel and dashboard, objects which are not of interest and will see very little movement throughout the video. Since the frame-skipping algorithm identifies unique frames by assessing the proportion of key points that are matching and the proportion of these matches that are static, having a large number of static points can artificially inflate these proportions and make it harder to detect whether there is motion between frames. While this effect can be partially offset by choosing stricter threshold parameters, it is possible that these stricter thresholds result in failing to identify unique frames, resulting in lower interpolation accuracy.

In order to address this issue, I attempted to preprocess the videos by finding key points in all of the frames of a given video and then saving the key points which are consistently marked static throughout the majority of the video. However, this approach does significantly increase the runtime of the frame-skipping algorithm and, as of now, is not able to consistently detect background points. Based on the distance threshold set, the algorithm generally only detects one or two points, not significant enough to noticeably improve the performance of the frame-skipping algorithm.

After concluding testing with the machine learning model, as described above, I plan to focus on more consistently detecting and pruning out background points in a given video. This is likely to improve the performance of the frame-skipping algorithm, since it allows the algorithm to focus only on the points we would expect to move. Should the approach above not work, I have also considered pruning points based on location. Under this approach, I would disregard points that are present within areas of the frame in which I would not expect motion. This requires some manual preprocessing for each video to be labeled and could significantly slow down the labeling process, although the frame-skipping algorithm runs fast enough and can reduce the time needed to label enough that this would still be an improvement over manual labelling.

Conclusion

Over the course of this semester, a frame-skipping and interpolation algorithm has been designed and implemented. The algorithm is capable of extracting unique frames from a given video using key features within each frame of the video and interpolating the labels for these unique frames to the rest of the frames. Extensive testing of this algorithm has shown that the algorithm runs very quickly and can reduce the average number of frames to be manually labeled per video by half, while only marginally reducing the accuracy of the labels as compared to labeling all of the frames by hand. As such, the algorithm described in this paper is suitable for use in place of manual labeling, given the immense time reduction observed through use of the algorithm over manual data processing.

Further work on this frame-skipping algorithm is planned in two areas. First, there is still work to be done in assessing the effects of using interpolated labels to train machine learning algorithms. In addition, the accuracy of the interpolation method can also be increased by ignoring background points during the frame-skipping process.

References

[1] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.

[2] J. Shi and C. Tomasi, "Good features to track", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 1994 IEEE Computer Society Conference on.*, pp. 593-600, 1994.