Show Me Your Hands!
Hand Pose Estimation and Tracking with Deep Learning

David Whisler
Stanford University
dwhisler@stanford.edu

Nikka Mofid
Stanford University
nmofid@stanford.edu

Samuel Turchetta
Stanford University
samuelt@stanford.edu

Abstract

We create a hand tracking pipeline by using a Graph CNN model coupled with a hand bounding box model to perform hand pose estimation on single RGB images from a webcam. The thumb tip and index fingertip keypoints can be used to aid motor-impaired persons perform computer control tasks such as scrolling or zooming without the need for mouse and keyboard. We show that the performance of hand tracking is improved by cropping the image to the hand before running pose estimation, and achieve near-perfect accuracy. However, performance is limited by the inference time for pose estimation on a laptop CPU. This is mitigated by using a multi-threaded approach and various heuristics to interpolate predictions between available pose estimates.

1. Introduction

Since the advent of Computer Vision, hand pose estimation and tracking has been a long-standing and critical problem, highly relevant to applications like augmented reality and virtual reality. Hand pose estimation is the process of modeling a human hand as a skeleton composed of keypoints, usually at the finger joints and palm, and finding their positions either in 2D or 3D coordinates [1]. In this project, we developed a hand pose estimation and real-time keypoint tracking pipeline. We use a Convolutional Neural Network (CNN) based model to estimate the 21 keypoints of a hand given a RGB image and build a hand tracking pipeline to track the position of a few selected hand keypoints via a webcam. We mapped the movements of the detected hand keypoints to emulate a computer mouse or scrolling functions, with a potential application to help those with fine motor-control impairments to better interact with a computer.

1.1. Problem Statement

In this project, we build a real-time Hand Pose Estimation and tracking pipeline with the goal of mapping movements of hand keypoints to emulate a computer mouse or scrolling functions, to aid those suffering from fine motor-control impairments with computer interaction. To estimate hand pose, we utilized a Graph Convolutional Neural Network (Graph CNN) model from the recent CVPR paper "3D Handshape and Pose Estimation from a Single RGB Image"[2]. This model was trained on the paper's corresponding dataset of 375,000 synthetic hand RGB images, which also included 3D ground truth hand meshes and 3D ground truth joint locations. This model successfully estimates the 21 keypoints of the hand. For our application, we choose to we look at the predicted pixel coordinates in 2D space on just the thumb tip and index finger tip keypoints, as these will be most useful in computer control tasks like scrolling and zooming. To evaluate our real-time Hand Tracking Pipeline, we developed a novel test dataset of hand labeled images of our own hands performing different sequential motions. We explored different preprocessing steps to improve the performance of our hand pose estimation model, including segmenting and cropping the image on the hand as well as performing facial removal, and the efficacy of these preprocessing techniques are evaluated.

1.2. Dataset

For our hand pose estimation model, we utilized the Hand Graph Convolutional Neural Network model from the recent CVPR paper "3D Handshape and Pose Estimation from a Single RGB Image". As part of the paper, the authors generated a novel dataset of 375,000 synthetic hand RGB images and their 3D ground truth meshes and 3D ground truth joint locations which they trained the model on. Thus, for our project we utilize the paper's corresponding dataset of 375,00 synthetic hand RGB images and their 3D ground truth meshes and 3D ground truth joint locations. The Hand Graph Convolutional Neural Network model is trained on

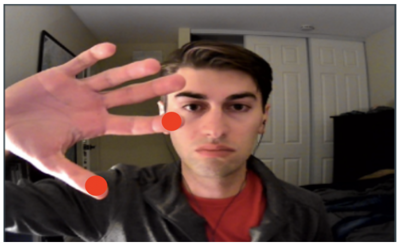


Figure 1: Example of hand-labeled dataset

Figure 2: Example of hand-labeled dataset

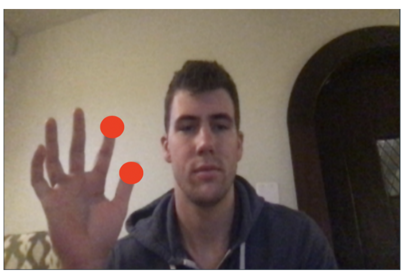


Figure 3: Example of hand-labeled dataset

315,000 of these images and validated on 60,000 of these images. To fully explore the pose estimation space, the images in the dataset are made up of 500 common hand gestures and 1000 unique camera viewpoints. The dataset also simulates real word diversity with 30 different lightings and five skin colors. In addition, we created three novel hand-labeled datasets of images of our own hands to test and fine-tune our model. We created a novel evaluation dataset of 50 images of our own hands in different poses in order to evaluation the performance of our hand pose estimation pipeline. In addition, we created a second hand-labeled small training set of 50 images of our own hands in order to explore fine-tuning our model. To evaluate the smoothing of our Kalman filter, we created a third hand-labeled dataset of 100 sequential images of one of our hands performing a sequential fluid motions. We refer to this dataset as our "Sequential Motion" hand-labeled evaluation dataset.

1.3. Evaluation Method

We evaluate the performance of our Hand Pose Estimation model by utilizing the the average and median squared pixel error between the predicted and actual coordinates on a our novel hand-labeled test dataset. We evaluated the ability of the model to perform in real-time by measuring the amount of time it takes to run an inference step on an image. It is important to note that our Kalman filter was evaluated on a hand-labeled dataset of sequential images of one of our teammates hands performing a series of fluid motions to gain more insight into its noise reduction performance. In addition, the sequential image dataset was used to evaluate the real-time performance of the full tracking pipeline at different simulated framerates.

1.4. Expected Results

We expect to see improvements in the average and median squared pixel error of the Hand Graph CNN pose estimation by preprocessing real-time image inputs by segmenting and cropping on a subject's hands. In turn, we expect to see improvements in the Hand Graph CNN model after fine-tuning it by training on images of our own hands. We anticipate that utilizing a Kalman filter will reduce noise in hand detection in our end-to-end real time hand tracking pipeline, as well as help fill in the inference time gaps to improve tracking speed. We expect our hand pose estimation model and tracking pipeline to struggle when hands are occluded, small relative to the background, or other objects distract the model such as human faces.

2. Related Work

One useful reference was the recent CVPR paper "Hand Pose Estimation: A Survey" [1]. This paper explains the hand pose estimation problem in detail as well as a number of approaches to solving this problem. The paper introduces several hand pose estimation methods we are interested in using as a baseline including detection and regression, as well as RGB based methods without depth maps. We also plan to examine papers like "Moving Object Tracking Using Kalman Filter" [3] which details methods for object tracking using a Kalman filter, which may help to reduce noise and outliers in our hand tracking pipeline. Finally, we also learned from the "GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB" paper [4] in which hand pose models which performed well on this problem.

3. Technical Approach Hand Pose Estimation

3.1. Hand Pose Detection

To perform pose detection of the hand, we utilized an existing Convolutional Neural Network (CNN) based model [2] to do hand pose and shape detection from single RGB

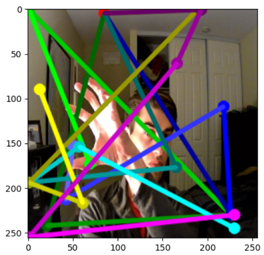


Figure 4: Example of failed pose detection

images like we would be receiving from a webcam. This method uses a Graph Convolutional Neural Network (Graph CNN) that detects 21 identified keypoints of a hand, including the wrist, palm, and all of the finger joints. It generates a probability-based heatmap of the hand keypoints, and then from this generates an estimated 3D hand mesh and the corresponding pixel coordinates of the keypoints in both 2D and 3D. For our project, we use the 2D pixel coordinates rather than the 3D world-space coordinates, since in our application of using pose to do computer control tasks such as scrolling and zooming, only relative 2D position is important. The pose detector model takes as input images with a resolution of 256x256, so prior to estimation, the raw webcam images were rescaled.

3.2. Preprocessing Hand Crop

One failure mode we saw with the hand pose detection was that when the hand was small relative to features in the background of the scene, the pose detection performed poorly after being distracted. This can be seen in Figure 8.

In order to remedy this, we helped focus the attention of the pose detector model by cropping the received image to only the hand. This was achieved by using another existing CNN optimized to detect bounding boxes of hands in real-time [5]. After detecting the bounding box, the estimate was inflated by a 25% margin to accomodate the whole hand, and rectified so that the dimensions were square. The square rectification helped prevent stretching distortions when rescaling to the 256x256 dimensions necessary for the pose estimation model. A full example of the hand cropping pipeline can be seen in Figure 5.

3.3. Preprocessing Face Removal

Another failure we noticed was that our Hand Pose Estimation model would often confuse faces with a subjects' hands. Thus, we explored preprocessing our images with OpenCV Haar Cascade Facial Classifier in order to detect a face and then blur it out in hopes that our model would focus on the hand rather than on the face itself.

3.4. Model Finetuning

Because the Hand Graph CNN model we utilized was trained on 375,000 synthetic hand RGB images, we were interested in seeing if we could improve its accuracy on our hand-labeled test set and also its performance in tracking by fine-tuning the model on a fine-tuning dataset of images of our own hands. Thus, we created a second hand-labeled finetuning dataset with images of all three of our hands and wrote a fine-tuning script to fine-tune our trained model on this dataset. Our fine-tuning dataset had approximately 50 images and we fine-tuned for 1 epoch with a learning rate of 0.0001.

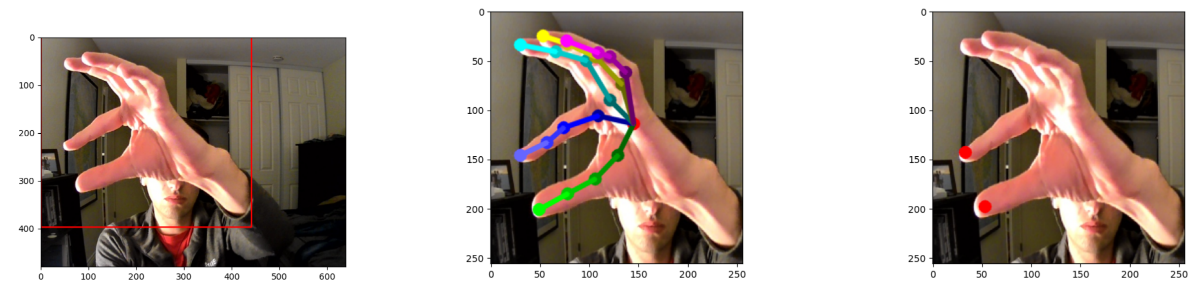
4. Technical Approach Hand Tracking

4.1. Bounding Box based Interpolation

Because of the relatively slow inference time of the hand pose estimation CNN model (approximately 0.5 seconds on CPU), this is only fast enough to support a 2 Hz frame rate (Table 3), which is well below the threshold needed for real-time performance. In order to mitigate this, we used heuristics to interpolate pose estimates between available CNN predictions, effectively boosting the pipeline's frame rate. This was achieved by switching to a multi-threaded approach, with the pose estimation CNN running in its own thread. When new CNN estimates were available, the pipeline used those, but in between inferences, the pipeline used a bounding box based heuristic. Since the hand crop bounding box inference time was much faster (approximately 23 Hz on CPU, Table 3), these estimates were used to create an approximate motion vector by calculating the difference between hand bounding box centroids on subsequent frames. This motion vector was added to the previous CNN pose estimate to effectively interpolate between subsequent predictions.

4.2. Kalman Filter

Given our CNN is not perfect, applying a Kalman filter here is a useful exercise to increase the confidence we have in given predictions as hand movements transition between each frame. When designing it we had to make a number of assumptions for the transition, covariance, noise, jacobian, and error matrices. We defined the expected transition as the movement from the bounding box. We decided to model our expected noise and error matrices after the kalman filter defined in class. We assume our CNN is essentially noise, so the jacobian matrix becomes the identity matrix. Lastly, our covariance model we decided to simplify and have it be the mean squared error for x and y. Because of these assumptions, we do not expect our Kalman filter to work perfectly in rejecting outliers, but we anticipate to see some improvement utilizing the filter over the baseline in reduc-



(a) Detected hand crop (b) Detected pose skeleton on hand crop (c) Detected fingertips on hand crop

Figure 5: Example pose detection using bounding box crops

ing noise which is particularly important in real time settings like this one.

4.3. Outlier Rejection

Another method of smoothing out the noisy CNN pose estimates was to apply a heuristic to reject obvious outliers. Occasionally, the CNN pose estimate was wildly outside of where it should be. To detect these cases, the centroid of the current CNN pose estimate keypoints was calculated, and checked to make sure that it is within the latest hand crop bounding box. If not, this most recent pose estimate was thrown out, and the next pose was calculating by performing another step of the previously mentioned bounding box interpolation from the previous good CNN pose estimate.

4.4. Hand Labeled Evaluation Dataset

To quantitatively evaluate our methods, we created a small (approx. 50 images) hand-labeled dataset of images of our own hands from a webcam. This images were labeled with the pixel coordinates of the thumb tip and index finger tip, which are two of the 21 keypoints detected by the hand pose CNN model and are particularly relevant to scrolling and zooming motions. Evaluation was done by computing the squared pixel error between the predicted thumb and index finger tip locations and the ground truth labels. To evaluate the smoothing of our Kalman filter, we created a second hand-labeled evaluation dataset (approx. 100 images) of sequential images of one of our hands performing fluid motions. We refer to this dataset as our "Sequential Motion" hand-labeled evaluation dataset.

5. Demonstration Video

The following video shows a demonstration of our Hand Pose Estimation and Tracking pipeline with mapped mouse movements. As you can see in the video, the user is using her hand to guide the mouse to the "games" folder than the "analysis" folder. The 21 predicted coordinates of our Hand Pose Estimation model are mapped directly onto the hand in the webcam forming a skeleton overlay on the hand.
Link: https://www.youtube.com/watch?v=LMQqcNgktmA

6. Quantitative Results

The following tables below detail our results.

In addition to the hand pose estimation model error, we also evaluated the performance of the multi-threaded interpolation techniques. To do this, we recorded a sequence

Table 1: Hand Pose Estimation Model Squared Pixel Error

Thumb	Mean SE	Median SE
Hand Graph CNN	4155.05	721.33
CNN + Face Removal	10372.14	6417.32
CNN + Hand Crop	**1222.33**	**12.38**
CNN + Hand Crop + Finetune	6715.368	**53.42**
Index Finger		
Hand Graph CNN	3596.83	1395.97
CNN + Face Removal	9168.30	6206.55
CNN + Hand Crop	**1686.97**	**13.46**
CNN + Hand Crop + Finetune	6777.71	**28.39**

Table 2: Hand Pose Estimation Model Squared Pixel Error for Sequential Motion Hand-labeled Evaluation Set

Thumb	Mean SE	Median SE
Hand Graph CNN	18295.56	27185.41
CNN + Hand Crop	2110.13	**22.47**
CNN + Kalman	**1921.71**	145.561
Index Finger		
Hand Graph CNN	19094.28	11341.70
CNN + Hand Crop	**1265.46**	**36.43**
CNN + Kalman	6530.20	3587.03

of 100 frames in which the index fingertip keypoint location was hand annotated. Then, we ran this image sequence through the image pipeline at different simulated frame rates (3, 10, and 30 Hz) to evaluate the tradeoff between accuracy and real-time performance. These frame rates were simulated for the pose estimation CNN only, the pose estimation with bounding box interpolation, and the pose estimation with bounding box interpolation and outlier rejection via the pose validity check. This can be seen in Figures 9, 10, and 11.

6.1. Quantitative Analysis

The model accuracy results can be seen in Table 1. This showed that the CNN + Hand Cropping outperformed both the vanilla Hand Graph CNN and the CNN + Face Removal methods, achieving near-perfect accuracy in the majority of cases. However, there were a few outliers that had very large errors, as can be seen in the difference between the median and mean error over the evaluation dataset. The presence of these outliers made the outlier rejection technique useful in the full hand tracking pipleine. It makes sense that the Hand Graph CNN with Hand segmentation and cropping performed best because it provided the pose estimation CNN with a more close up image of the subject's hand with minimal background interference.

Although one would expect that removing subject's face might help the pose estimation CNN model focus more on the hand and hence lead to improved performance of our tracking pipeline, we found that it actually performed the worse. We suspect this is because in many instances the hand was either covering or very close to a subject's face leading the face removal process to remove part of the hand with it thus leading to reduced performance.

We were also interested in seeing if we could achieve further improvements by fine-tuning our best model on a small hand-labelled fine-tuning set of images of our own hands. We found that while fine-tuning also lead to a comparatively low median error to our best model, it did not lead to further improvement in the mean SE. We believe this is due to the noise in the labels, as we labelled our small training set ourselves and the labels that we put on the thumb and index finger may not have been perfectly accurate leading to the model not learning the best information during the fine-tuning process. We believe with better labelled images fine-tuning could be a very promising avenue to further improve model performance.

Table 3: Mean Inference Time (CPU)

	Avg Time (s)
Hand Graph CNN	0.539
Hand Cropping	0.043
Face Removal	0.071

In order to evaluate the noise reduction and smoothing performance of our Kalman filter, we ran it on our Sequential Motion Hand-labelled Evaluation set (our second evaluation dataset). Unlike the evaluation dataset used in Table 1, the images in this dataset are sequential frames of fluid hand motions. To provide comparison points, we re-ran our Hand Graph CNN baseline and our best model CNN + Hand Crop on this sequential motion dataset in order to get comparison points to our Kalman filter. It is important to note that for Mean and Median SE values for all our models are higher when run on this dataset due to lower camera quality than our original evaluation dataset (images taken on a different camera than our other evaluation set). Looking at Table 2, we see that the Kalman filter appears to be able to successfully reduce outliers when predicting the thumb coordinate. We see this as the utilization of the Kalman filter results in the lowest Mean SE of the three models for the thumb on the sequential motion evaluation set. For the index finger, the Kalman Filter is still able to lower the Mean SE and the Median SE over the baseline Hand Graph CNN, however, the CNN Model with the Hand Crop still performs better. It makes sense that our Kalman filter did not perform perfectly as in order to implement it we had to make several assumptions about the noise and state. Overall, it is exciting to see that despite these assumptions we were able to see a general decrease in noise over the baseline.

Additionally, the inference time was measured, as seen in Table 3. This was important, since the goal is to run this model in near real-time to perform scrolling and zooming tasks on a computer with a minimum of lag. The inference time for the individual models can be seen in Table 3. As can be seen, the majority of the time is taken by the pose detection CNN model, which takes an average of approximately 0.5 seconds to run on an image frame. This translates to a maximum of 2 Hz possible, which is slower than needed to perform in real time. One of the mitigating techniques for this included moving to a multi-threaded approach with the bounding box interpolation heuristic, as described in the technical approach.

Finally, the index finger keypoint error versus time was plotted for several simulated framerates (3, 10, and 30 Hz) to evaluate the performance of the full multi-threaded pipeline. These can be seen in Figures 9, 10, and 11. These plots demonstrate that the error increases with increasing simulated frame rate, which is expected as the inference time of the CNN pose estimation (2 Hz) and the hand bounding box interpolation (23 Hz) start to fall behind the frame rate. These plots also quantitatively demonstrate the efficacy of the hand bounding box interpolation heuristic, as it helps to "fill in the gaps" pose estimates needed between available hand pose CNN predictions. In addition, it shows that the outlier rejection heuristic also had a small

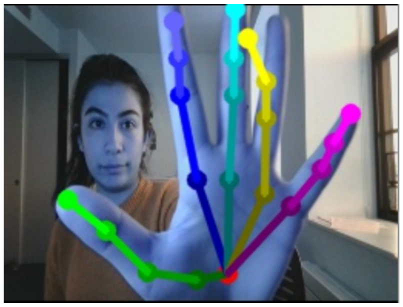


Figure 6: Example of good pose estimation

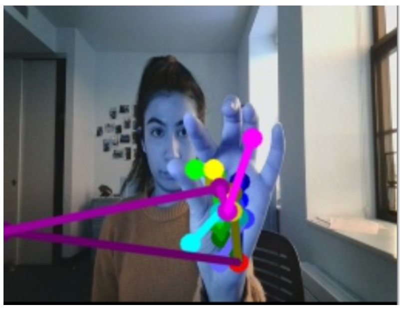


Figure 7: Example of fail on pinch

but notable effect in reducing the error, seen most clearly in the 10 Hz plot. The effect of this outlier rejection also tends to be sequence dependent, since if the lighting conditions produce especially bad CNN pose estimates, it will have a stronger effect.

7. Qualitative Results and Analysis

Looking at our quantiative results, we were interested in seeing which images and poses influenced our mean and median square error. Analyzing our outputs by drawing the skeleton of the 21 predicted joints directly on our evaluation set, we found that our models generally performed best on clear images of the hand, close to the camera as shown in Figure 6. This makes sense as it gives the model a clear view of the hand that is most similar to the training dataset. In addition, we found that almost all our models struggled with complex hand movements such as pinching and pointing at an angle as shown in Figure 7. We believe that we could further improve the performance of model on complex hand movements with further fine-tuning on a larger hand-labelled dataset of complex hand movements.

When utilizing the Kalman filter, we found that though the predicted positions were often slightly off for most images, the transition between each state became significantly smoother. This is seen in Figure 8, where the green line on the image is the live kalman filter prediction of the thumb's path over several frames, and the blue line is the best prediction from the CNN + Hand Crop model of the

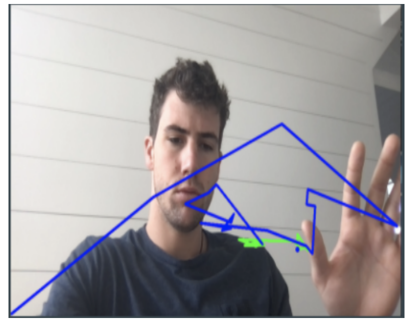


Figure 8: Kalman filter vs CNN + Hand Crop

thumb's path in real time over several frames. During this sequence of frames, the user makes a small, short wave at the camera moving his hand from left to right and back without moving up or down. The blue line which does not use the Kalman filter shows a path that is significantly more noisy. Predicting the thumb's path as going up, down,out of the frame, and back. The green path produced by the Kalman filter is significantly less noisy, showing the thumb's movement more accurately moving across a small distance from left to right. Thus, the Kalman filter's resulting smoother state transition makes it very useful for our use case of utilizing our Hand Pose Estimation and Tracking pipeline to move a mouse for those with fine motor control difficulties.

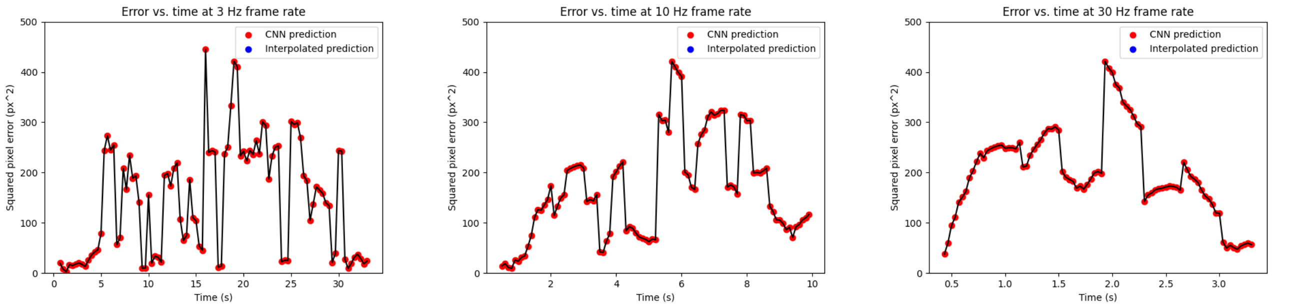
8. Conclusion & Future Work

We have successfully built and end-to-end Hand Pose Estimation and Tracking pipeline with mouse movement emulation while experimenting with different preprocessing, fine-tuning, interpolation, and noise reduction tehcniques. Our hand tracking pipeline performs well on real-time RGB webcam data, up to a speed of approximately 15 Hz which is suitable for most applications. The accuracy of the hand pose estimation was significantly improved by using a separate hand bounding box CNN model as a preprocessing crop step. Running the pipeline on a laptop CPU causes the inference time for the hand pose CNN to be too slow on its own, but using the multi-threaded approach with interpolation heuristics such as hand bounding box movement significantly improved the pipeline's real-time performance. Noisy hand pose CNN estimates continued to be an issue, though, but outliers were suppressed using basic outlier suppression heuristics using a bounding box consistency check, and the Kalman filter also showed some promise in this regard. For future work, improvement could be made in the Kalman filter tuning, especially in the model assumptions for the transition matrix and covariances. Finally, it would also be interesting to assess the performance of the model on a powerful, cloud-based GPU, as this may make the pose estimation CNN viable on its own in a real-time environment even without the bounding box interpolation heuristics.

9. GitHub Code Repository

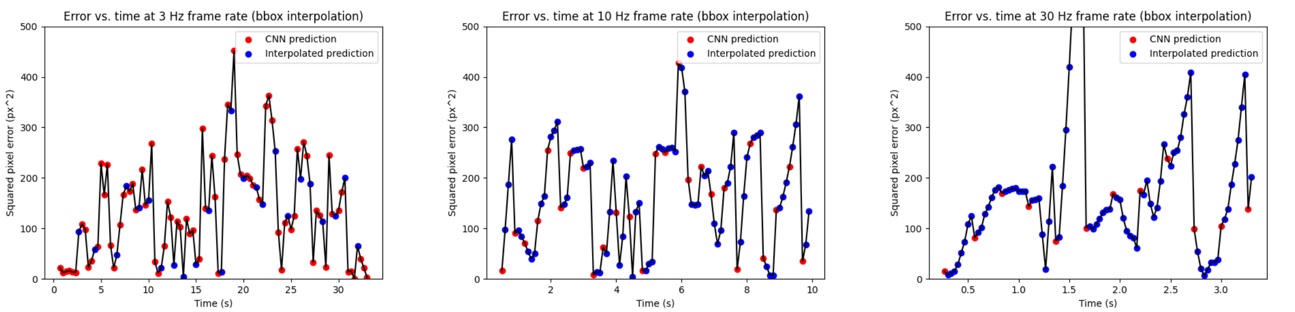
Our project code can be found at the following GitHub repository:

`https://github.com/nmofid1996/CS231aFinalProject`



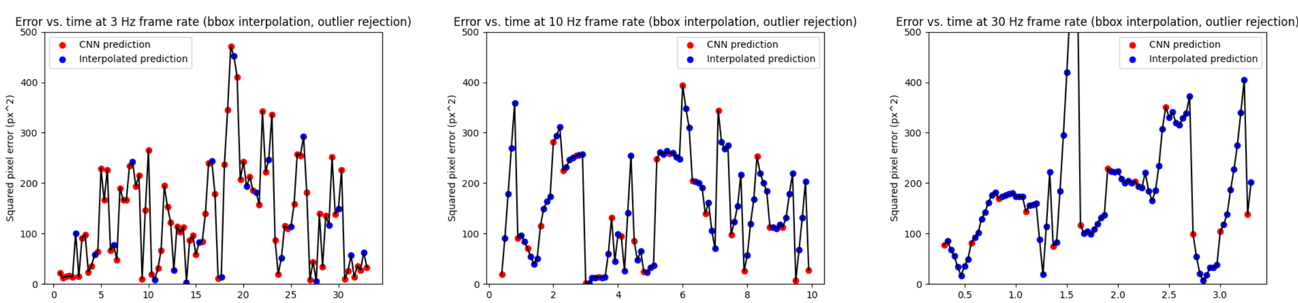
(a) Frame rate = 3 Hz (b) Frame rate = 10 Hz (c) Frame rate = 30 Hz

Figure 9: Index finger error vs. time for pose estimation CNN only (no interpolation)



(a) Frame rate = 3 Hz (b) Frame rate = 10 Hz (c) Frame rate = 30 Hz

Figure 10: Index finger error for pose estimation CNN + hand bounding box interpolation



(a) Frame rate = 3 Hz (b) Frame rate = 10 Hz (c) Frame rate = 30 Hz

Figure 11: Index finger error for pose estimation CNN + hand bounding box interpolation + outlier rejection

References

[1] B. Doosti. Hand pose estimation: A survey. *CoRR*, abs/1903.01013, 2019.

[2] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan. 3d hand shape and pose estimation from a single rgb image. *CVPR*, 2019.

[3] P. R. Gunjal, B. R. Gunjal, H. A. Shinde, S. M. Vanam, and S. S. Aher. Moving object tracking using kalman filter. In *2018 International Conference On Advances in Communication and Computing Technology (ICACCT)*, pages 544–547, 2018.

[4] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] D. Victor. Handtrack: A library for prototyping real-time hand tracking interfaces using convolutional neural networks. *GitHub repository*, 2017.