# EVA: Providing High-Fidelity Real-Time Feedback for Weightlifting

Johann Pally
jpally2@illinois.edu
UIUC MCS '23

Cody Wang
yaohuiw2@illinois.edu
UIUC MSCS '24

Yuxuan Liu
yuxuan38@illinois.edu
UIUC MCS '23

## ABSTRACT

Our Erroneous Vector Analysis uses novel pose estimation software in conjunction with bar-mounted IMU sensor data to classify the form correctness of an athlete conducting an incline chest press motion. The linear motion of a chest press is generalizable to other common weightlifting exercises. We demonstrate higher degrees of detail for feedback for classifying incorrect motion as opposed to previous works in a contactless, real time, low power fashion using widely popular technology (smartphone/watch).

## KEYWORDS

pose estimation, human-computer interaction, gesture classification, real-time feedback

**ACM Reference Format:**
Johann Pally, Cody Wang, and Yuxuan Liu. 2023. EVA: Providing High-Fidelity Real-Time Feedback for Weightlifting. In *Proceedings of Smart-X '22: Fake ACM Symposium on Smart cities, homes, phones, and beyond (Smart-X '22).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

According to the National Safety Council (NSC), in 2021, equipment exercises accounted for about 409,000 workout injuries, the most in any category of sports and recreation. Further, according to the Global Health and Fitness Association, around 50% of Americans feel intimidated to start using the gym, commonly citing feeling worried they could hurt themselves or feel embarrassed doing exercises they don't know how to do.[9] Personal training is an expensive, yet common suggestion to learn weightlifting skills. Without a trainer, an individual must record themselves, conduct unsupervised form repetitions, and then personally judge whether or not their form is incorrect. This could be potentially dangerous, especially for beginners, when they lack the knowledge of correct forms. Another issue is that, even without hurting oneself with this training mode, one could be training the muscles in an incorrect way, and therefore suffer from poor training results, which could further damage self-confidence and increase gym anxiety.

Therefore, there exists a dire need for an inexpensive, objective, and effective solution to provide accurate feedback. Benefiting from the rise of apps, the average consumer's smartphone can track heart rate, and sleep patterns, and provides journal spaces to track exercise. However, most of these current commercial solutions are incapable of providing users with accurate real-time feedback for incorrect movements during exercises. Therefore, the goal of our work is to devise a system that can leverage the integration of sensor data and machine learning to reliably help users correct their movement.

Our approach, Erroneous Vector Analysis (EVA), fuses data from a smartphone (video) and a wrist mounted IMU on a computationally capable back end device to provide detailed feedback on user form for an incline bench press, a motion generalizable to several other weightlifting motions.


Fig. 1: Mockup of the proposed user setup

Challenges for our solution are standardization across variable body types and recording orientations, accurate remote localization of key body parts without calibration, and time synchronization across the distributed sensors. EVA applies Google's MediaPipe pose estimation software, using the shoulders as a point of origin and distance between shoulders as a standardized unit of measure to analyze the correctness of user flexion. Importantly, when incorrect motion is detected, EVA immediately cues audio feedback on the smartphone to provide 'during-exercise' correction. After experimenting with threading, naive sequential sampling functions at a faster rate, with generally low latency. Evaluated across several users, we successful segment activity with no calibration.

## 2 BACKGROUND AND RELATED WORK

| | Method | Uses Available Hardware | Standardization | Real Time | Contactless | Low Power | Low Cost |
|---|---|---|---|---|---|---|---|
| EVA | Camera & IMU | Yes | Yes | Yes | Yes | Yes | Yes |
| GymCam '18 | Camera | No | Yes | Yes | Yes | No | No |
| RecoFit '14 | Arm Worn IMU | No | No | No | No | Yes | Yes |
| MiLift '18 | Smartwatch | Yes | No | No | No | Yes | No |
| ERICA '20 | IMUs | No | No | Yes | No | Yes | Yes |

Table. 1: Comparing Methods

Currently, in this IoT era, we are all surrounded by huge amounts of smart devices with sensors embedded and this trend also occurs in the exercise tracking area. Commercial products like smartwatches could report very accurate heart rate tracking but it usually lacks the ability to do activity monitoring or more specifically wrong posture warning and real-time feedback. Generally, activity

*Smart-X '22, Jan 18–May 05, 2022, Urbana, IL*
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

monitoring could be divided into two major parts: CV-based [1] or inertial sensors-based [2, 3, 6] CV-based approaches have a long history.

## 2.1 CV-based approaches

With cameras deployed at different locations in the gym, people's 3D orientation and movements could be constructed in real-time and compared with labeled datasets in order to tell whether the movements are valid or not.[1] However, it would be too costly and difficult to carry out as the system is physically large and might violate nearby people's privacy. Also, multiple dimensions image processing requires relatively large processing power which needs to be done on the cloud. Moreover, when two individuals are exercising close to each other, they exhibit similar motion features such as phase and frequency, and maybe cause them to be recognized as a single motion cluster. Besides academia, commercial mobile applications have been a game changer in terms of personal weightlifting movement correction. For example, Barsense relies on computer vision algorithms to dynamically track and record bar displacement from the lateral side during exercises that involve heavy weights using the barbell. It does not provide feedback during the exercise, but users can use it as a recap of their barbell displacement and self-assess the movement. One major issue is that weightlifting can go wrong in a short span of time, and BarSense cannot provide auto analysis and warnings to the user as they lift. If the user made mistakes in posture and does not know how to bail the barbell, the results could be catastrophic. To provide a more generalized example, Onyx was a 2019 workout app promising form correction for a comprehensive list of popular exercise motions. The app also contains playlists of exercises with different emphases and difficulties that simulate actual gym sessions. From its AppStore reviews, users applaud the interface and the on-point arrangement of playlists. However, the app is performing poorly in terms of motion tracking rep counting. Even worse, there is a 70-dollar paywall to using its full functions, which significantly limits its usage by casual exercisers.

## 2.2 IMU-based approaches

On the other hand, IMU-based approaches are far more light-weighted and easier for deployments. We've seen arm-worn inertial sensors, with automatic activity prediction and feedback given upon activity recognition.[2] Although providing robust results in recognition, this approach lacks the ability to address form issues in real-time, and it requires the user to mount hardware onto the forearm, which in some cases may hinder shoulder movements. On the other hand, some similar approaches relied on commercial smartwatches to apply machine learning-based models on recorded cardio data to classify repetitive weight-lifting activities.[3] Also, state-of-the-art projects such as ERICA have tried combining inertial sensor readings with other sensors in order to perform a more accurate estimation and detection.[6] These approaches were limited in that they can't accurately detect mistakes comprehensively during exercises. Although ERICA can give feedback on certain mistakes such as range of motion, it cannot provide feedback regarding the angle of joints or the placement of limbs.

## 2.3 Other sensor-based approaches

Besides those two conventional approaches, there are also some other approaches like using Electromyogram(EMG) sensors to detect muscular activities in order to predict the movements[4], using RFID to do on-site weightlifting recognition[5] and using fine-grained CSI value on mobile phones as data to predict whether people are in exercise process[7]. Although these approaches are novel, the accuracy of activity recognition and anomaly detection is much lower than normal CV-based and inertial-sensor-based methods. Since no one has combined CV and inertial sensors together, we plan to leverage the advantages of both methods and use one to supplement another in EVA.

# 3 SYSTEM OVERVIEW

We propose to take advantage of the output of a smartphone camera and barbell-mounted IMU to classify a user's form as correct or incorrect (injury prone). The camera and IMU would offload skeleton data and IMU readings to a cloud device (external laptop) over WiFi, where classification takes place using our model. On the laptop, we conduct skeletonization, activity segmentation, and motion classification to label plotted motion, provide real-time feedback, and playback video of the athlete's motion. The typical user experience of this system is straightforward. Imagine the user attaching an IMU to the barbell and placing the smartphone in front of the bench to record, and when the user starts to perform the exercise, the system will output audio feedback in case of detected incorrect motion. After the exercise, the user can also view the playback on the laptop. The setup processes for both types of devices are fairly easy, and headphones can be paired with the smartphone to receive feedback discreetly. A further path of inquiry is pushing all processing onto the smartphone, however since our preliminary model runs on Python, we continue development for processing on the laptop device.

Our system is modeled after three research questions.

- **RQ1: No Stereo?** The typical user will go to the gym with a single camera on their smartphone. To achieve low cast and generalizability, we conduct skeletonization including depth estimation using Google's MediaPipe software. Further, smartwatches have become more popular. For further refinement, we include IMU sensor data from bar-mounted sensors. How effective is depth estimation complemented with IMU data?
- **RQ2: Variation?** Every person has a different figure and body proportions. How can we standardize our approach across different body types and angles of recording?
- **RQ3: Data Fusion?** We argue an advantage of combining IMU data with image processing. How can we fuse the data streams and handle time synchronization?

We correlate IMU data from sensors mounted to the bar as a supplemental extension for refinement. The camera is the main sensor, which will be responsible for activity segmentation, repetition counting, and feedback. We perform skeletonization of the camera video, and to increase their precision, we augment the results with IMU readings. Although the camera alone is able to provide some results, IMUs can help provide more insights into wrist positioning

and angle. We have also chosen these specific sensors because cameras and audio outputs are ubiquitous in current mobile devices, and IMUs are ever increasingly popular in smartwatches. Lastly, these two data streams are time synchronized by feature-matching activity segmentation of early packets of information.

Finally, our system has three evaluation metrics:

- Usability in different settings
- Accuracy of analysis
- Correction Prompt latency

To gather a usability score, we'll conduct user testing and gather feedback on the intuitiveness of our display and the feedback provided. Our benchmark will be existing user reviews for exercise-tracking apps such as BarSense. To establish accuracy, our system will analyze pre-recorded videos that have ground truth incorrect motions. Our metric will be the F1 score across all training data for our system. Note the variability of our margins of error depending on the tuned physics constraints for incorrect motions. Lastly, we will plot the time latency for all fragments of our system processes to observe bottlenecks and support our notion of "real-time". As rationalized above, the latency of feedback

## 3.1 Challenges

Time synchronization is the main issue of this project. Since a slightly incorrect weight-lifting posture can produce danger in a split second. All devices must be accurately synchronized in time and produce the least latency in each step.

Another issue is caused by the different environments and setups of each user. Visual noises in gyms and each user's different height and distance from the sensor create disturbances that could cause wrong judgments from the model. In order to correctly help the trainee, the model should be robust in dealing with noises and different setups of users.

Lastly, the hardware energy consumption should be optimized as our devices are not designed to be directly plugged in. We should intentionally make trade-offs between performance and energy consumption by choosing hardware configurations of IMU and the smartphone.

# 4 PROPOSED APPROACH

Below is the diagram highlighting the modularization of our repository. Upon launch, a connection layer manages connections between IMU device, the Android smartphone, and the laptop over 2.4 GHz Wifi to communicate data between the IMU and smartphone to the cloud device. With fused data, the backend device processes datapoints for the physics-bound analysis of skeletonization and motion. Lastly, a visualization displays the trainee's movement, labeling part of the motion which was injury-prone. The displayed results are paired with improvement instructions.

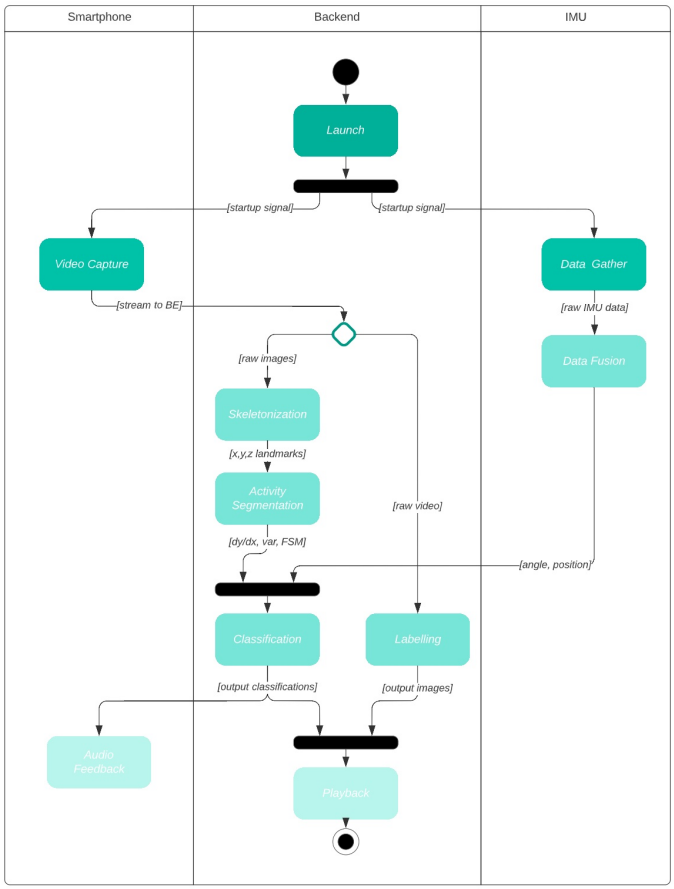



Fig. 2: Module Structure

## 4.1 Data Sampling

We data sample either in a naive sequential, or asynchronous threaded manner. The sequential approach sends and waits on a query request from both the smartphone and IMU, applies all data analysis steps, sends appropriate audio feedback to the smartphone, then repeats all steps for the next frame. The asynchronous method maintains three parallel threads. The first sends query requests to the smartphone and IMU at a constant rate (30 fps),filling a dynamic queue, pairing a tuple of IMU and image data points by associated time stamp of request. A second thread pops the head data tuple and applies all data analysis steps, saving the results to a separate global reference stack for final display. If data analysis results in an injury-prone classification for a frame, the analyzer hands off to third parallel thread which sends an error code to the smartphone, which emits an audio feedback message. In our evaluation, we illustrate the advantages and disadvantages of either approach.

## 4.2 Skeletonization

**With a lack of stereo vision, is it still possible to conduct localization for key body part movement for 3D motions?** The first step for image processing is the skeletonization of input images. Using the MediaPipe Pose estimation software from Google, we can gather information on key landmarks of a person estimated

on an input image. Below is an image displaying the segmentation overlaid on a video input. For further reference, following is a link to a sample video for the expected types of video inputs our method was tested on (link).

Once parsed, the skeletonization returns points in 3D of the shoulders, elbows, wrists, etc. over time across the input video (link). Below are plots of the x, y, and z over time. L Wrist (Blue), L Elbow (Orange), L Shoulder (Green), R Shoulder (Red), R Elbow (Purple), R Wrist (Maroon). Upon first glance, there are clear outlines of starting the video (noisy start), unracking the bar (small incline in y for wrists and elbows above shoulder), three reps (rise and file in y and two-fold break in x), reracking the bar (slow decline in y), and finally ending the video (noisy end).

**Provided noise, how can we smooth data to be interpretable for the classification of injury-prone motion?** Below are graphs, decoupling the x, y, and z data for key landmarks across a sample video. Note the obviously apparent localization in both x and y motion. It is intuitive to gather the up and down motions of the wrist with respect to the shoulders, reflecting the bench press motion of the user.

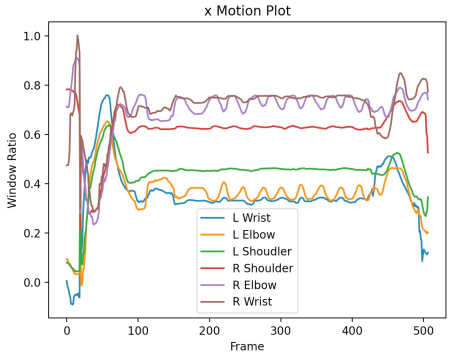



Fig. 3: Plotting landmarks of interest (time vs. x). Note the stability in shoulder data and the motion in elbow data across repetitions.

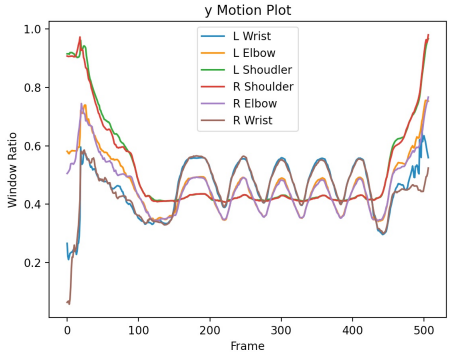



Fig. 4: Plotting landmarks of interest (time vs. y). Note the stability in shoulder data and the motion in wrist data across repetitions.

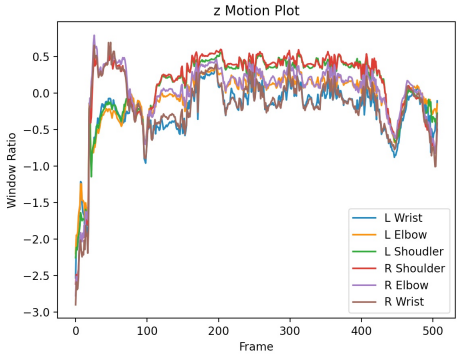



Fig. 5: Plotting landmarks of interest (time vs. z)

Figure 5 demonstrates how z data from skeletonization is far too noisy for detailed feedback analysis, further reiterating that without stereo video, IMU sampling provides ideal supplemental data for trends along the z axis (as well as smoothing or reinforcing results across all axes).

**Given different body types and styles for recording, how do we standardize analysis?** Edmond et. al. published a finding [13] demonstrating the correlation between the vector across shoulders in relationship to the length of extremities across gender, age, and body fat distribution. We apply this claim, using shoulders as an origin location, dividing the difference in x and y between elbows

and wrists with relation to the shoulders to standardize analysis across videos.

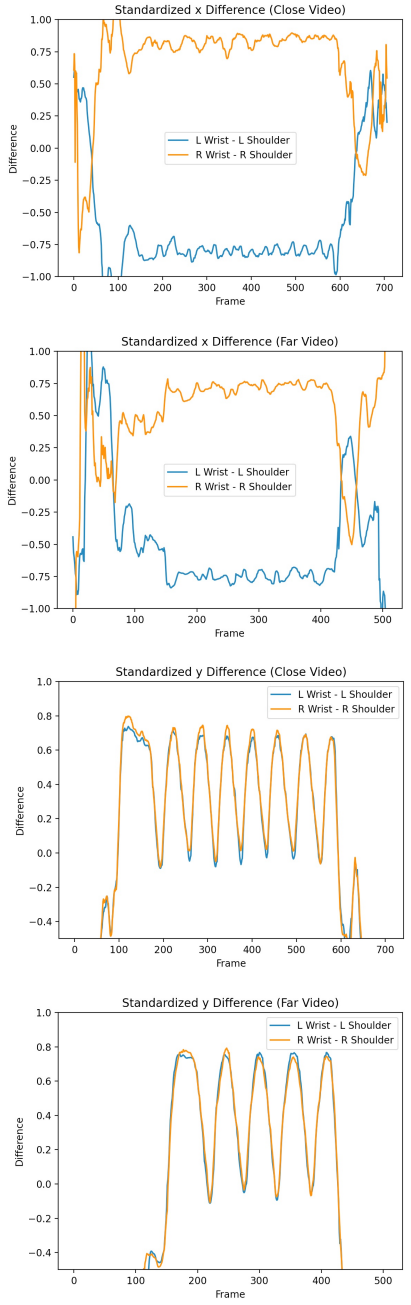



Fig 7: Plotting standardized values in x and y. Note the constrained y axes across plots.

The plots above demonstrate standardization across two disparate users with two recording angles (first video taken with a stand close to the user. Second video taken with the phone placed on the ground, further away from the user). The constrained y axes demonstrate that despite differences in body type and recording style, we are able to standardize x and y value trends for analysis further down the pipeline. Using this finding, we can conduct activity segmentation and analysis on flexion and pace from image data alone.

## 4.3 Activity Segmentation

For initial activity segmentation, we conduct a sliding window variance analysis. Plotted below is the sliding window over 30 frames (1/4 of a second for our sample video) for the left and right shoulder. This reveals when the participant is sitting at the bench as opposed to moving.

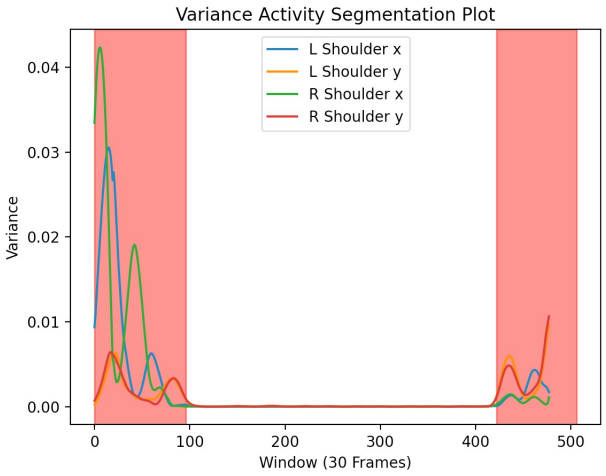



Fig. 8: Plotting moving (highlighted) or sitting (time vs. var)

Highlighting sections of the data over the variance data, we illustrate successful initial activity segmentation for when a person is sitting at the bench or moving using the variance of motion of the shoulder.

Next, we apply the finite state machine logic (displayed below) to declare frame windows for holding the bar at the top of a repetition, moving the bar down during a repetition, holding the bar at the bottom of a repetition, and moving the bar up during a repetition.

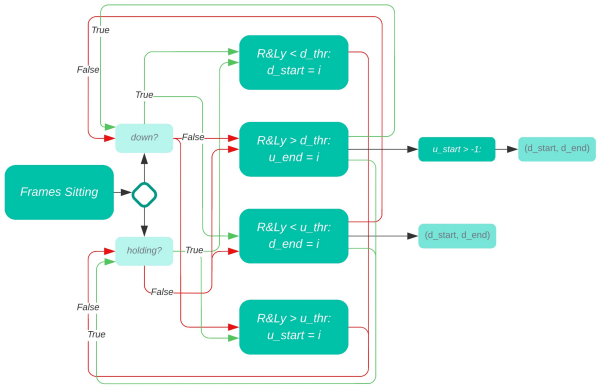



Fig 9: FSM threshold logic for subdividing activity segmentation of exercise motion.

Plotted below is the superposition of the extracted frame windows over the standardized y data, we confirm our claim that the

FSM properly segments activity in relationship to the up and down sub-motions of the incline bench press exercise motion.

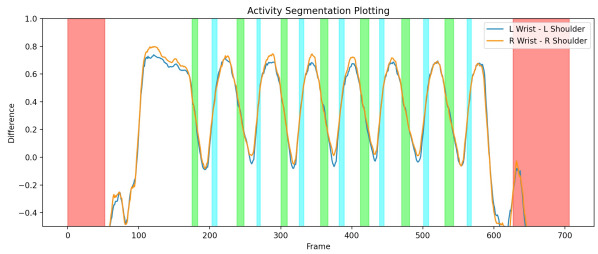



Fig 10: Superimposed highlighting frame windows for subdividing motion. Green is down, blue is up.

## 4.4 Classification

From these frame windows, we can further analyze the data to assess incorrect form. Our program analyzes the following errors. Note the delegation between image and IMU data analysis which conjointly forms high fidelity feedback which would otherwise, indepently, be unachievable.

#### 4.4.1 *FLEXION_BOTTOM_ERROR_CODE (1).*

#### 4.4.2 *FLEXION_TOP_ERROR_CODE (2).*

These error codes compute the 3D vectors for the forearm (wrist - elbow) and upper arm (elbow - shoulder) by applying $\theta = cos^{-1}\frac{a \cdot b}{\|a\|\|b\|}$ to analyze the flexion angle at the top and bottom of a repetition (plotted below).

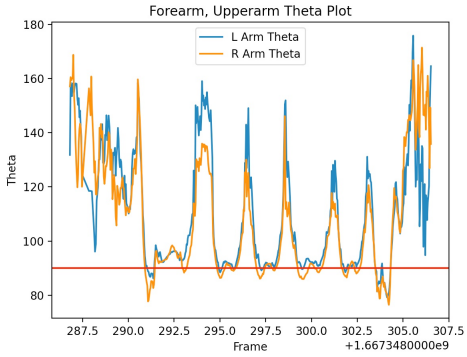



Fig. 11: Plotting angle between the forearm and upper arm (time vs. theta)

Flexion is considered erroneous during hold_bottom frame windows if not averaged below 95 degrees and during hold_top frame windows if not averaged above 135 degrees.

#### 4.4.3 *TILT_DOWN_ERROR_CODE (3).*

#### 4.4.4 *TILT_UP_ERROR_CODE (4).*

These error codes average IMU gyroscope and accelerometer data to determine whether the bar is not horizontal during up or down window frames.

#### 4.4.5 *INSTABILITY_ERROR_CODE (5).*

The instability error code takes the variance of gyroscopic, accelerometer, and magnetometer data to determine struggle or jitter at any point during the repetition.

#### 4.4.6 *ROTATION_ERROR_CODE (6).*

The rotation error code takes the sliding window average of gyroscopic and magnetometer data to determine whether the user has rotated their wrist forwards or backwards at any point during the repetition.

# 5 IMPLEMENTATION

Our hardware system consists of an Android smartphone for image capturing and real-time audio feedback, an IMU edge device, and a cloud device (laptop) for higher-level data classification.

**What unique benefit does IMU data bring to skeletonization?** IMU data is mostly with accelerometer, gyroscope, and magnetometer readings. By fusing accelerometer and gyroscope data together, we can have a clear understanding of what current position athletes are in and whether they are carrying out any motions. As we mentioned before in the introduction part, we will attach both STM32-based IMU edge devices to both ends of the barbell, so we can keep track of the relative linear and angular displacement in order to tell whether athletes are prone to get injured.

Since the images captured might not fulfill the requirements to do skeletonization perfectly, like there will be large noise in the depth direction, IMU data can then act as a supplementary part as it can easily detect movements in the z-axis that the camera lacks. For example, there might be wrist movement both back and forth along the z-axis, basically rotations on the x-axis or y-axis, when athletes are not paying enough attention to correct postures. In this case, edge devices at both ends of the bar could detect and calculate how the orientation of the bar changes according to its initial orientation, thus providing a relatively reliable first-stage estimation of the current condition of the athlete and sending the data to the cloud for further data analysis.

Also, it's easy for IMU-based edge devices to detect fast linear displacement along all 3 dimensions. After zero-calibrating at the initial position, we can calculate the relative displacement of the bar and add onto the initial position to get the current position, based on the linear acceleration value that we gathered along the time. These two are just examples picked from all categories of anomalies that athletes could run into, later we will get into depth about how IMU data reporting varies under each specific possible anomaly that we come up with.

**How do we fuse data from both data sources?** Timestamps will be attached to both the images and IMU fused reading when they are done on edges, data will be classified and grouped together on the cloud depending on its specific corresponding time tags.

First, the correlation between the two datasets will be tested. Features of both datasets on different segments of the exercise will be extracted. In-phase image information and IMU readings will be paired up and fused together into a virtually integrated multi-dimensional dataset used for later testing.

Image info will be analyzed first to get a coarse estimation of the athlete's situation. If it's reporting an anomaly, IMU readings will be utilized to validate or give a more precise depiction of what's

actually the anomaly(eg. wrist rotations, unsteady barbell movements). In addition, since the single-camera approach to 3d posture recognition is not very accurate, IMU data could be used to raise the accuracy of our detection scheme, and increase the true positive rate and true negative rate.

**How is network performance influencing the image processing workflow?** Our initial implementation of the image transmission layer of the system averages 20 FPS, with considerable delays received on the processing side (laptop). To enhance the latency and throughput of the transmission layer. We employed several image pre-processing techniques, which boosted the performance to 40FPS for processing and lower latency. (1) Compression is done to reduce image size. (2) Image color space properties are converted to RGB format, removing the added computation overhead to perform the conversion on the laptop (3) Decreased the socket buffer size, which mitigates the congestion of socket buffers, which causes spikes in transmission latency. (4) All transmission tasks are multi-threaded, which leverages the performance of multi-core CPUs on some smartphones. This also increased the throughput as it no longer requires each task to finish before processing the next.

## 5.1 Front End (Android)

For image-capturing, a smartphone is mounted on the wall in front of the trainee. Generally, the smartphone's image-capturing speed could reach around 60 fps with a resolution of 640 by 480 pixels. Prior to transmission, we perform data pre-processing on the smartphone, such as frame compression and format conversion. In brief, our first implementation of the image transfer algorithm is to initiate an image transmission on the Android side when an image in the camera buffer becomes available. However, the preliminary evaluation of this strategy shows space for improvement. The frame rate image transfer stabilizes at around 30 to 40 Fps. This performance is sufficiently high compared to our expected frame granularity and could satisfy the design requirements. But the latency of transmission is unacceptably high. From there, we attempted other solutions, mainly regarding using multithreading to alleviate the latency. However, our trial runs reveal that Java multithreading for this specific task introduces more complexity than optimizes performance. It causes frame rate drops, and the synchronization of different threads drags the run time further. On the other hand, the solution that finally proved to be both functional and efficient involves changing the design to make the smartphone behave like an API server. Initially, we implemented the cloud end as a passive receiver of the images. The latency of the image transmission is in the magnitude of hundreds of seconds. After careful consideration of our design for the system pattern, we discovered the main issue with the previous approaches was socket congestion, where images are consumed much slower than they are created. Therefore, the intuition of our new algorithm is to follow a request and response pattern. This pattern matches the rates of both the producer and the consumer. Images will be requested synchronously at a controllable interval, dramatically reducing congestion and image-receiving latency. The new algorithm maintains an image buffer on Android, which is updated continuously using the result of camera fetch API calls. With this, the cloud server code can send requests to the Android side to harvest this buffer, which the cloud server then performs further processing and skeletonization analysis.

The communication between android end and server end is achieved utilizing several macros. For example, with encoded byte b'0 indicating the server is requesting image data. So, the phone will send a meta-data (size of the image) first to the server indicating bytes for receiving and consequently transmitting the whole image as a byte array. Also, right after the detection logic finishes running on the server, it will send audio feedback requests like b'1 throughout b'5 indicating status codes for the android to play audio reminder to trainee. The most recent audio feedback has the highest priority to be executed, so it's reasonable that we sometimes will witness former feedback is shut down in the middle and the newer one is played.

## 5.2 Edge Device End (IMU)

Finally, for edge device, we used ST's nucleo-wb55 dev board as the platform and mount a Adafruit LSM6DSOX and LIS3MDL 9-DoF IMU module onto it. LSM6DSOX is an IMU sensor, incorporating both accelerometer and gyroscope, with advanced Digital Function and Finite State Machine. LIS3MDL is a popular magnetometer produced by ST for pointing magnetic field strength within the current location. The sampling frequencies of all types of sensors that were employed in our system excluding the camera are all high and could even reach like 48kHz, so the temporary storage and transmission of data is truly a huge challenge for our hardware setup. That's one of the reasons why we choose this MCU as it is a dual-core microcontroller consisting of an ARM Cortex-M4 and another Cortex-M0 processor which will satisfy our requirements for processing speed. Also, it has 1 Mbyte of Flash and 256 Kbytes of SRAM, which ensures enough space for code and pre-transmission data storage. The data acquisition is achieved by I2C protocol between MCU and sensors, while the connection between MCU and server is achieved by UART protocol.

Prior to data acquisition, we run a set of calibration functions to automatically get the offsets for accelerometer and gyroscopes, and soft and hard iron offset for magnetometer finding the true North. After the raw data being gathered continuously in the main while loop, the MCU UART feedback keeps listening for requests from the laptop and break from data acquisition to sending if requested. Mutexes are used to prevent interrupt calls during data buffer updates. Simultaneously with acquisition, we applied our IMU calibration coefficients to achieve a set of IMU reading with higher accuracy. Also, Madgwick's algorithm is implemented on edge to compute the current relative orientation and location update compared to the former moment. So, the data output format is a list of 15 entries, including raw accelerometer(3-axis), gyrometer(3-axis), magnetometer(3-axis), processed relative orientation(3-axis), and processed relative position(3-axis).

The UART communication between edge end and server end is achieved with server sending specific bytes. In each data acquisition round, server should send a byte b'r, indicating request to avoid noise captured by the edge. Edge will in turn reply with a set of 15-element IMU reading array to server. Bluetooth 5.0 is not used as the transmission protocol both for the technical difficulties to achieve and its instabilities for data transmission. While Bluetooth is easy

to use on platforms like Arduino or Raspberry which are with well wrapped functional libraries, Bluetooth on industrial level MCUs involve more lower-level set-ups like protocol domain knowledge (GAP, GATT, profile, event, characteristic, etc.), inter-processor communication, MCU real-time scheduling, and more.

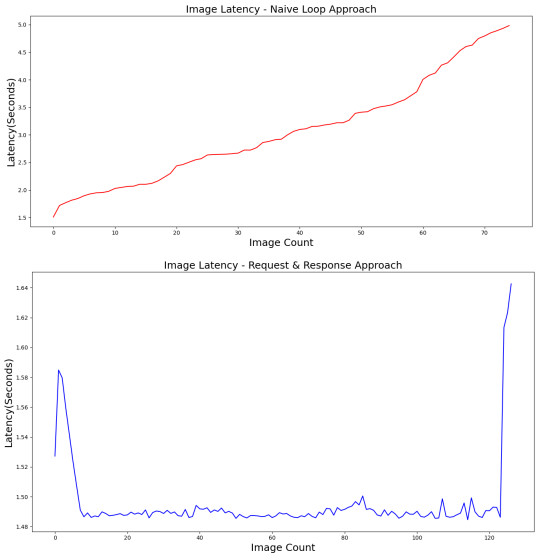
### 5.3 Back End (Python Server)

At the end, the synchronization between IMU data and images is achieved in a callback fashion. Initially, we planned to do it in a broadcast fashion and run through synchronization algorithms on the server while capturing metadata with a time tag. But later, for simplicity and reliability of the system, we move onto callback setups on both android end and edge device end. Each time the main while loop runs, we collect an image and a corresponding chunk of IMU readings through both WiFi and wired connections. Throughout this method, we are certain that readings taken from each part are in pace and at the same moment as the other reading. Similar to what we faced on the android end, python threading is not stable to apply on data request and reception and will cause fps slowing down to about 10, which is tried and discarded. In total, image request will cost 5ms, IMU data request will cost about 16ms each round, so the sampling frequency combined could reach $1000/(5+16) \approx 47$Hz theoretically, which indeed could satisfy our real-time data processing and feedback requirement.

Received images are combined into an on-disk video for running a skeletonization API with. With the current image, we can gather sets of approximated 3-d locations of trainees' all body joints, specifically the shoulder, arm, and wrist which we are interested in the most. Feeding back those position vectors into our control flow for detecting joint angles, we can do vector norms to calculate the approximated angle and positions of arms, for doing repetition counting and activity segmentation (going up/down). Comparing computed output with our predefined constants, we can know what they are doing right now. Corresponding IMU data are compiled together and utilized for furthering the details of reporting: using the fused orientation result, we can know whether the bar is tilting and trainees' wrist movements; with positions, we can trace all movements of the bar.

In the whole while loop running on the server, after the laptop starts its WiFi server ports on IllinoisNet, Android end also needs to connect to Illinois Net and requires a click on the application for connection trials. After a connection is established, the server end will request image, IMU data concurrently, and then run the processing code for gathering the estimated user situation, sending feedback, and starting again. Together at data reception, we provide an optional video playing functionality drawing out the real-time captured image sequences using canvas. After all trials are finished, we will draw out trainee's joints movements using plt.

## 6 EVALUATION




As the main functionality of our system is to provide real-time feedback to trainees, the latency and the accuracy of feedback are our two primary evaluation metrics. The above two figures show comparatively how parallelism solved the issue of socket congestion, with an image latency reduction up to 4 seconds overall. We observe that sequential processing not only has a high baseline latency, it's trending upwards almost linearly as the number of queries increase. This is the result of more and more images queuing in front of the socket. On the other hand, parallelism features a below 1 second average latency, which indicates that socket queuing is menial for this approach. However, threading did cause rendering issues with our GUI, which only impacted the final feedback view. However, printed data reflects the same results as sequential processing.

In order to evaluate the user experience , we conducted a handful of gym testing sessions. The activity consists of having individuals perform incline bench press with the system set up running and giving prompts. Evaluated metrics include the latency of feedback, types of feedback given and their accuracy, with an emphasis on discovering false positives and false negatives. To set up the testing hardware, we positioned the bench directly under the barbell of the squat rack facing outward. The smartphone is fixed on a phone holder on the ground in front of the user, with the back cameras facing the user. The IMU sensors were installed onto the center of the barbell with electrical tape, and are powered by a wire connection to a MacBook, which is running the cloud server script. Across our 5 tested individuals, the activity segmentation was accurate with marginal tuning, reflecting successful standardization. According to a confusion matrix across all tests, our model had an F1-score of 0.87. One challenge is that our error analysis follows a if else boolean structure, which may overshadow later error codes with earlier. For example, the model may indicate that there is jitter when the 'ideal' feedback would have been a tilt error. Reprioritizing the order of feedback checks solves this issue to some extent, however we propose implementing likelihood estimations for each error and argmaxing over the list for better feedback decision making.

During our presentation session, 6 additional users tested our system. All 11 of our participants were asked the following questions.

- Does EVA operate with acceptable latency?
- Was the feedback provided intuitive and helpful?
- Is this system practical in your exercise regimen?

Question 1 received 10 positive responses. Question 2 received 11 positive responses. Question 3 received 8 positive responses. Notable negative responses were as follows: user does not own a smartwatch, user would not like to record themselves in the gym, user thought the feedback was distracting during the middle of their next repetition. In the future, we would like to expand our user testing and conduct a Pareto analysis to better understand improvements to the usability of our system.

## 7 CONCLUSION

In this paper, we present a novel system - Erroneous Vector Analysis (EVA), that aims to perform real-time exercise form correction and help users prevent injury. By using computer vision based pose estimation and an inertial measurement unit (IMU), EVA is capable of detecting incorrect form, and providing feedback audio prompts that are highly accurate. Both are tailored to the specific proportions of the user's body without additional user intervention. EVA is intended for gym trainees at all levels. EVA works with readily available hardware - a smartphone and IMU (common in smartwatches). In addition, EVA is non-intrusive, low power, and low cost.

Our contribution consists of three main components: 1. Inertial data sensing and its corresponding data fusion algorithm 2. Data analyzer which standardizes input data across users, segments activity, classifies incorrect motions, and creates audio and visual feedback presentation 3. Client Android app and cloud server scripts that enables low latency image transmission and communication between different components

From our evaluation, we conclude that EVA runs at sufficient latency and provides helpful feedback. However, the system still faces a handful of limitations. Firstly, our analysis only scrutinizes incline bench press motions. Pose detection and IMU data, in its current state, are insufficient to overcome challenges such as occlusion and analyzing the curvature of the back for other weightlifting activities such as the squat or deadlift motion. Future involves changing IMU analysis from bar-mounted IMU to wrist mounted IMU data. Lastly, EVA requires a fairly high network bandwidth. Future work involves load balancing, dynamic vision model retraining, and smart movement interpolation under extreme network conditions.

Beyond its technical contributions, we hope that our research inspires others to help remove barriers to weight training exercises.

## REFERENCES


[1] Rushil Khurana, Karan Ahuja, Zac Yu, Jennifer Mankoff, Chris Harrison, and Mayank Goel. 2018. GymCam: Detecting, Recognizing and Tracking Simultaneous Exercises in Unconstrained Scenes. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 185 (December 2018), 17 pages. https://doi.org/10.1145/3287063

[2] Dan Morris, T. Scott Saponas, Andrew Guillory, and Ilya Kelner. 2014. RecoFit: Using a wearable sensor to find, recognize, and count repetitive exercises. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14).* Association for Computing Machinery, New York, NY, USA, 3225–3234. https://doi.org/10.1145/2556288.2557116

[3] Shen, Chenguang, Bo-Jhang Ho, and Mani Srivastava. Milift: Efficient smartwatch-based workout tracking using automatic segmentation. *IEEE Transactions on Mobile Computing* 17.7 (2017): 1609-1622. https://doi.org/10.1109/TMC.2017.2775641

[4] Bo-Jhang Ho, Renju Liu, Hsiao-Yun Tseng, and Mani Srivastava. 2017. MyoBuddy: Detecting Barbell Weight Using Electromyogram Sensors. In *Proceedings of the 1st Workshop on Digital Biomarkers (DigitalBiomarkers '17).* Association for Computing Machinery, New York, NY, USA, 27–32. https://doi.org/10.1145/3089341.3089346

[5] Han Ding, Longfei Shangguan, Zheng Yang, Jinsong Han, Zimu Zhou, Panlong Yang, Wei Xi, and Jizhong Zhao. 2015. FEMO: A Platform for Free-weight Exercise Monitoring with RFIDs. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15).* Association for Computing Machinery, New York, NY, USA, 141–154. https://doi.org/10.1145/2809695.2809708

[6] Meera Radhakrishnan, Darshana Rathnayake, Ong Koon Han, Inseok Hwang, and Archan Misra. 2020. ERICA: enabling real-time mistake detection & corrective feedback for free-weights exercises. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems.* Association for Computing Machinery, New York, NY, USA, 558–571. https://doi.org/10.1145/3384419.3430732

[7] Xiaonan Guo, Jian Liu, Cong Shi, Hongbo Liu, Yingying Chen, and Mooi Choo Chuah. 2018. Device-free Personalized Fitness Assistant Using WiFi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 165 (December 2018), 23 pages. https://doi.org/10.1145/3287043

[8] Misha Patel and Aisling Ann O'Kane. 2015. Contextual Influences on the Use and Non-Use of Digital Technology While Exercising at the Gym. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15).* Association for Computing Machinery, New York, NY, USA, 2923–2932. https://doi.org/10.1145/2702123.2702384

[9] Rachel Valerio. 2019. Fitness Industry Roundup: 'Gym-timidation' Is Real. The Global Health and Fitness Association, https://www.ihrsa.org/improve-your-club/industry-news/fitness-industry-roundup-gym-timidation-is-real

[10] Gray SE, Finch CF. The causes of injuries sustained at fitness facilities presenting to Victorian emergency departments - identifying the main culprits. *Inj Epidemiol.* 2015;2(1):6. doi:https://doi.org/10.1186%2Fs40621-015-0037-4

[11] Novatchkov H, Baca A. Artificial intelligence in sports on the example of weight training. *J Sports Sci Med.* 2013;12(1):27-37. Published 2013 Mar 1.

[12] National Safety Council. 2021. Sports and Recreational Injuries Resulting In Emergency Visits, 2013-2021. https://injuryfacts.nsc.org/home-and-community/safety-topics/sports-and-recreational-injuries

[13] Edmond T, Laps A, Case AL, O'Hara N, Abzug JM. Normal Ranges of Upper Extremity Length, Circumference, and Rate of Growth in the Pediatric Population. Hand (N Y). 2020 Sep;15(5):713-721. doi: 10.1177/1558944718824706. Epub 2019 Feb 1. PMID: 30709325; PMCID: PMC7543216.