Falcon: the Pro Gym Assistant

Author: Vishal Baskar, Albert Chen, Venkata Vivek Thallam: Electrical and Computer Engineering, Carnegie Mellon University

Abstract—**A system capable of providing real-time feedback to users as they exercise. The system provides users with customized workouts based on their target areas and as they work out, they will get periodic feedback regarding their posture performing the appropriate exercise. The system involves a display and a side camera that takes images and periodically relays the images to an FPGA that does the image processing to identify the joints, which are then post-processed and analyzed to provide feedback on the users' screen.**

Index Terms— **Computer Vision, Fitness Assistant, FPGA, HLS, OpenCV, Posture Tracking, PyGame, Real Time Analysis, Spotify, SQLite, UART, Vivado**

I. Introduction

THE global spread of the novel coronavirus (COVID-19), has truly changed the way that we live. In order to mitigate the spread of the virus, we have transitioned to a remote environment where contact with one another is limited. This has impacted not only the way we work but also our ability to stay in shape. This has led to a rise in the popularity of at home workout options ranging from free fitness applications such as the Nike Training App to high-end workout systems such as Tonal and Mirror. The workout system Mirror provides users with personalized workouts, a video of a trainer performing the corresponding exercises, and biometric information such as heart rate and calories burned.

Our workout system strives to build upon this system by providing the same functionality as well as the capability to receive real-time feedback pertaining to the posture of the current exercise being performed. Falcon will be able to detect the users' joints at an accuracy of 90%. It will then parse this information and generate feedback that matches 100% to our designed models and ability to provide feedback to the user in less than 1.5 seconds, which corresponds to the average time it takes to perform 1 rep of a particular exercise.

II. Design Requirements

A. Joint Tracking

The main requirements of the joint tracking algorithm is split into pre-processing the image and pinpointing the trackers. We will have a software testbench to ensure that the pixel is downscaled and converted into a 160x120 pixel image in an HSV format. There should be a 100% size and format match because we are calling library functions. For pinpointing the trackers, there will be a lot of noise and the color of the lighting might affect the accuracy of the algorithm. Our software testbench will ensure that we have a 90% accuracy rate. We chose this so that an average set of 10 reps will have 1 rep misclassified at most. The inputs to our testbench will be a random image of a user wearing the dark suit, and we will compare the joint location to the expected joint location determined manually by tracing the image.

B. Transfer Protocol

The main requirements for the transfer protocol pertain to latency and accuracy. The latency to transmit the data from the CPU to the FPGA and back from the FPGA to the CPU has to be under 1 second to be able to provide the feedback at an appropriate rate. Likewise, the data that is transferred via UART has to be 100% accurate to ensure that we are able to extract the appropriate information after processing the image. Both of these requirements will be analyzed with the assistance of a hardware testbench where we analyze various packets of data sent between the computer and the FPGA to determine the accuracy and latency of the system.

C. Posture Analysis

For the posture analysis, we want to ensure that the algorithm performs 100% based on our model. Our model will have predetermined thresholds for what we want our lines and angles to connect the joints to be. Our software testbench will take in predetermined joint locations to ensure the feedback will be what we determine our model to be.

D. User Interface

For the user interface we want an application that is easy to navigate, gives feedback effectively and is overall user-friendly. More specifically there should be three main capabilities of the application: choosing and doing a workout, modifying their weight/age as well as being able to look through past workout session details. The user should be able to navigate these different sections through a combination of mouse and keyboard. During the workout the user should be able to see themselves as well as a model performing the workout and also receive live time feedback. Testing will be done through a visual inspection to make sure all requirements are met.

III. Architecture and/or Principle of Operation

A. *FPGA/HLS*

Our workout system involves a Kintex-7 XC7K325T FPGA that is responsible for the image processing. An explanation regarding why this particular FPGA was chosen is elaborated further in the Design Trade Studies section. The implementation for the image processing is done with the assistance of Vivado HLS.

B. OpenCV

OpenCV is a library implemented in python that allows for the processing of the images/videos. The library interfaces with our Logitech C270 webcam and allows for a live feed of the video in our application. It also captures images periodically, dependent on the workout routine, which will be sent to the FPGA for processing after being downscaled.

C. Pygame

The majority of the user interface will be written with the assistance of the Pygame library. This framework will interact with both OpenCV as well as the Pyserial library to send and receive data from the FPGA through the UART protocol.

D. SQLite

SQLite will be used to store data regarding the user in a robust relational database. There is a SQLite Python library which will be used to store relational data such as user profiles as well as past workout data.

E. Image Processing

The application will capture images at a fixed interval dependent on the workout routine. The image processing portion will be in charge of downscaling the image and converting it to the HSV format. The user will be wearing a dark suit with 3M colored bandages taped around the joints. The joint tracking algorithm would extract the joints by creating a binary mask based on the bandage color, then do morphological transformations to reduce noise, and finally pinpoint the largest concentration of pixels asserted to track the joint locations.

F. UART Communication Protocol

After the computer downscales the image that is captured from the computer's webcam, it is transmitted to the FPGA with the UART protocol. This transmitted image is then stored in the FPGA's RAM for further processing and after the FPGA is able to extract the joints from the image, this information is transmitted back to the CPU at the same baud rate.

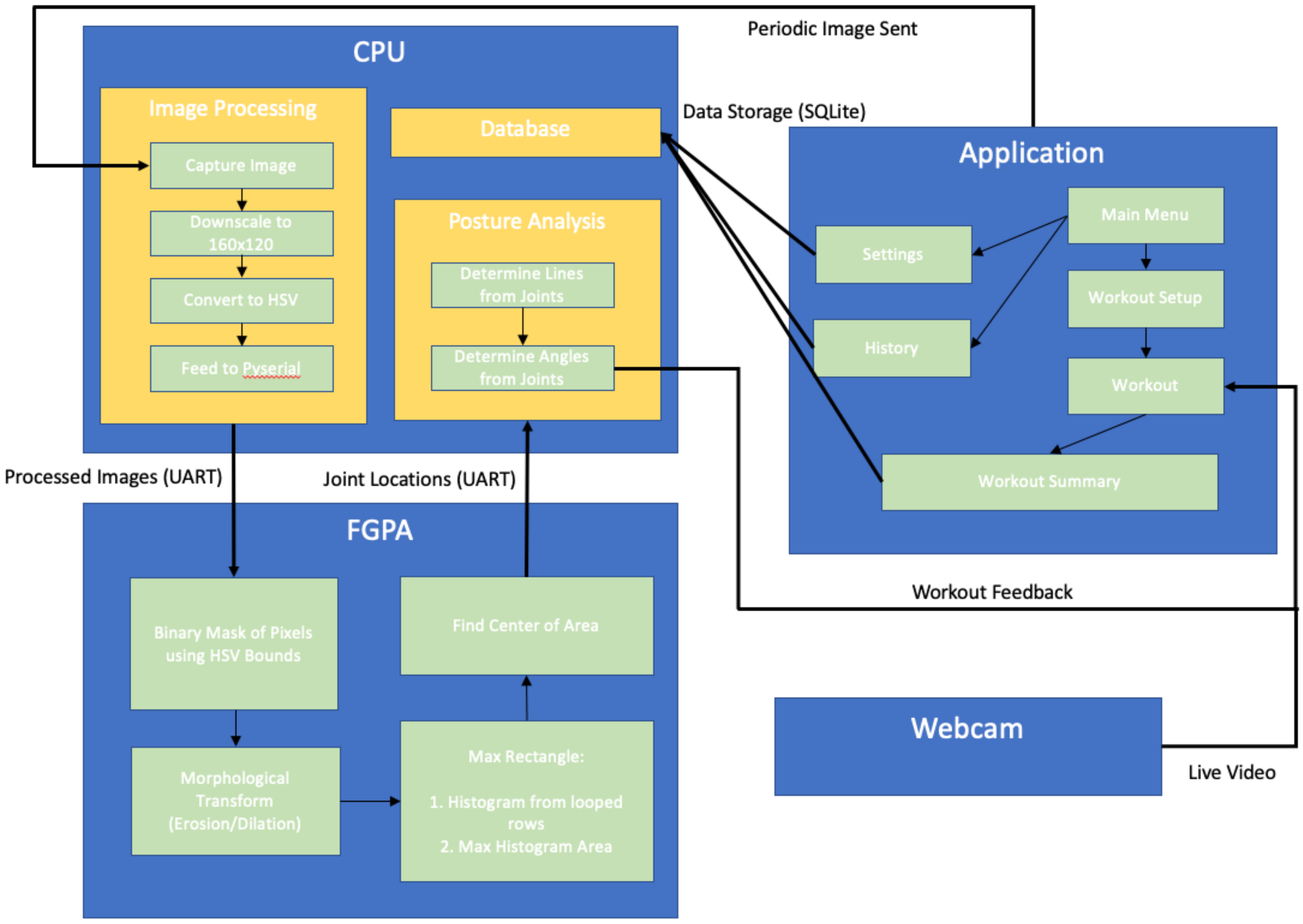


Figure 1: System Diagram

G. Posture Analysis

The posture analysis will receive joint locations from the FPGA and it will create lines from the joints. With these lines, angles can be calculated at the joint locations. We will be comparing slopes and angles to our predetermined models. There will be thresholds for each to account for the different human anatomy and angles of the webcam. If checks aren't met, feedback will be sent to the application to output to the user to change their posture.

H. User Interface

The user interface will be navigable through a combination of mouse and keyboard. A user will be able to choose between their workout focus (legs, core, upper) and have the option to save their profile and biometric information. Workouts will have feedback pertaining to each rep's form and history/statistics can be viewed at a later time.

IV. Design Trade Studies

A. Why FPGA?

We decided to use an FPGA to function as a coprocessor and process the image on the FPGA since we wanted to reduce as much of the computation on the CPU as possible. We chose to perform the computation on an FPGA rather than upload the images to a cloud service where we could have alternatively performed the computation since we wanted the images of users to stay on their local machine and not shared to an external platform for privacy and security.

B. Why Kintex-7?

While doing the appropriate research for our project, we came across the *Identity Checker on FPGA* project from the Spring 2019 semester. The project had a similar implementation to our current implementation, and we observed that they used the Kintex-7 FPGA. Since this FPGA is available in the course inventory and provides us with HLS capabilities and the appropriate baud rate for our use case, we decided to use this FPGA for our project.

C. Why UART Protocol?

After looking into various protocols, we decided to use the UART protocol for a combination of reasons. Previous capstone projects we looked into (such as the *Identity Checker on FPGA* project mentioned in the previous section), used the UART protocol, and it worked well for their specific use case. Since the UART protocol worked effectively for the other projects and it is a fairly simple protocol that is well-documented, we decided to do some analysis as to whether or not it would be effective for our use case. After performing some rudimentary calculations, we estimated that the time it would take to transmit the data from the CPU to the FPGA via UART - our current bottleneck - would be approximately 0.63 sec (the derivation for this number is provided in Equation 1 in the following section) which is enough for us to provide feedback. As a result, we decided to use UART for our communication protocol.

D. Why HLS?

To implement the image processing in HDL, we contemplated two options: using Vivado HLS and hand-writing RTL. We initially contemplated hand-writing RTL since it provides us with more capabilities to optimize the RTL for our project and because we had not used HLS in the past. However, after performing further research, we realized that we would exceed the number of registers available in the FPGA unless we used block RAM. Since interfacing with the block RAM requires the implementation of non-trivial handshaking logic with the block RAM and we already have an implementation of our image processing in a high-level language, we decided that it would be time-efficient to use HLS. Though not optimal, it allows us to stay within our schedule and focus on the integration and testing portions of our project.

E. Why Color Tracking and not RFID or HumanPose?

We decided to do color tracking instead of using RFID tags or HumanPose mainly because of its simplicity. The HumanPose implementation consists of a lot of library functions that will be hard to implement because we would have to convert it to RTL. There are not many resources that we can refer to for hardware accelerator implementation. RFID tags may be more accurate than the color tracking mechanism we are using, but the RFID radio frequency signal may lose its accuracy in contact with liquids. Since the user will sweat during the workout routine, it may not be the best option.

F. Why Pygame?

In order to create a cohesive application for working out we needed to choose a user interface framework. Due to the availability of many essential libraries such as Pyserial and OpenCV we decided that the best language to code the UI would also be Python. There were a few different options including tkinter, PyGame, and PyQT. Researching through these we found that tkinter was a bit too basic and barebones since it lacks widgets and many builtins. PyQT was a very comparable choice but was chosen over at the end of the day since needed features needed to be paid for. Pygame is more optimized for games but since we want a high refresh rate for the camera pygame will be the fastest, it will be more ideal even though this is not actually a game.

G. Why SQLite?

SQLite was chosen as the way to store persistent data due to its robust and relational nature. We originally contemplated using a simple .csv file to store data but since both user biometrics and past workout data will be stored it will be much more convenient to use the relational features. In addition, lookup and parsing will be much easier now instead of manually parsing the .csv. We can quickly look up all the workouts for a specific user using their user ID and simply calling a function provided by the SQLite API.

V. SYSTEM DESCRIPTION

A. *Image Processing*

The pre-processing of the image will be first to downscale the image to a 160x120 pixel image to reduce the latency of the overall system because we will be sending it serially to the FPGA. Then, the image will have to be converted into the HSV color space, because this format is easier to work with and more precise for image processing algorithms.

The image processing algorithm will consist of multiple existing OpenCV library functions. Since we will be using the FPGA as our hardware accelerator, we implemented the library functions from scratch and made modifications to the algorithm to suit our interests. First, the user will be wearing a dark suit taped with 3M colored bandages to provide a good contrast despite the downscaling of the image. The algorithm will create a binary mask of the pixels that land within the HSV bounds for each specific color of the bandages. It is extremely hard to be extremely precise with the HSV bounds because the trackers will easily be affected by noise and lighting. The fine-tuning is done by finding the upper and lower bounds of all HSV color space around the joint across multiple images.

After the binary mask is created, it will be put through morphological transforms, erosion and dilation, to reduce noise and to preserve the max area where the pixels are asserted. Here, we use the top, right, bottom, and left adjacent pixels as the transform mask for erosion and dilation. Finally, we would have to find the largest concentration of pixels asserted through the whole image. We modified an existing algorithm that finds the max area of a rectangle in a binary field. The algorithm loops through the rows of the binary matrix to create histograms of the pixels asserted. Then, every row will be put through another function that gets the max area under a histogram. It utilizes a stack to keep track of the previous indices that form areas under the rectangle. After finding the max area under the histogram and throughout the rows, the algorithm would output the center of the rectangle which will be the reference for our joint positions.

B. *UART Communication Protocol*

After the image that is captured from the webcam is downscaled to a size of 160x120 pixels by the CPU, this image is then transferred pixel by pixel to the FPGA at the baud rate of 921600 bits/sec with the assistance of the PySerial Python library that connects to the appropriate COM port. As explained in Equation 1, it takes 0.63 s to transmit the information from the CPU to FPGA.

$$\boldsymbol{Assumption}: 8\,\frac{bits}{byte} + 1\;start\;bit + 1\;stop\;bit = 10\,\frac{bits}{byte\;to\;send}$$

$$\#\;of\;Bits = (160 \text{ x } 120)\,\frac{pixels}{image} * 3\,\frac{bytes}{pixel} * 10\,\frac{bits}{byte} = 576000 \text{ bits}$$

$$Time\;to\;Transmit = 576000\;bits\;to\;transfer * \frac{1}{921600}\,\frac{sec}{bits} = 0.63\;secs$$

Equation 1. Time to Transmit Image to FPGA

This image is then received by the FPGA with the AXI UARTLite IP block at the same baud rate and then stored into the BRAM, which is used for image processing. The positions of the joints extracted are then transmitted back from the FPGA to the CPU via the same interactions at the baud rate of 921600 bits/sec. Since the amount of information is transmitted back is significantly less than the amount of information transmitted from the CPU to the FPGA, it takes 0.2 ms (Equation 2) for this step.

$\boldsymbol{Assumption}$:
Row: 8 $bit\;representation$
$Column$: 7 $bit\;representation$
$Identifier$: 3 $bit\;representation$

$$\#\;of\;Bits = 8 \text{ joints} * \left(8\,\frac{bit}{row} + 7\,\frac{bit}{col} + 3\,\frac{bit}{identifier}\right) = 180 \text{ bits}$$

$$Time\;to\;Transmit = 180\;bits\;to\;transfer * \frac{1}{921600}\,\frac{sec}{bits} = 0.2\;msecs$$

Equation 2. Time to Transmit Joint Locations to CPU

C. *Joint Extraction from the FPGA*

The data transfer from the UART port to the BRAM is handled by the MicroBlaze soft IP core provided by Xilinx. The block diagram of the design in the FPGA can be seen in Figure 3.

After the information is stored in the BRAM, the MicroBlaze core then performs the various morphological transformations mentioned in the Image Processing section. A representation of what the BRAM looks like at any given moment can be seen in Figure 2.

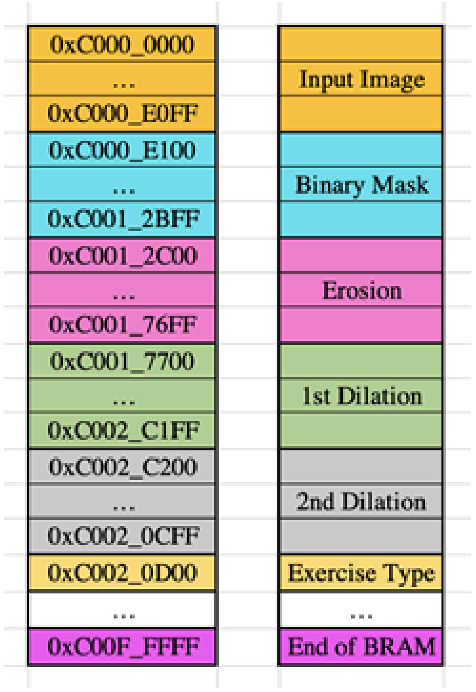


Figure 2: BRAM Layout

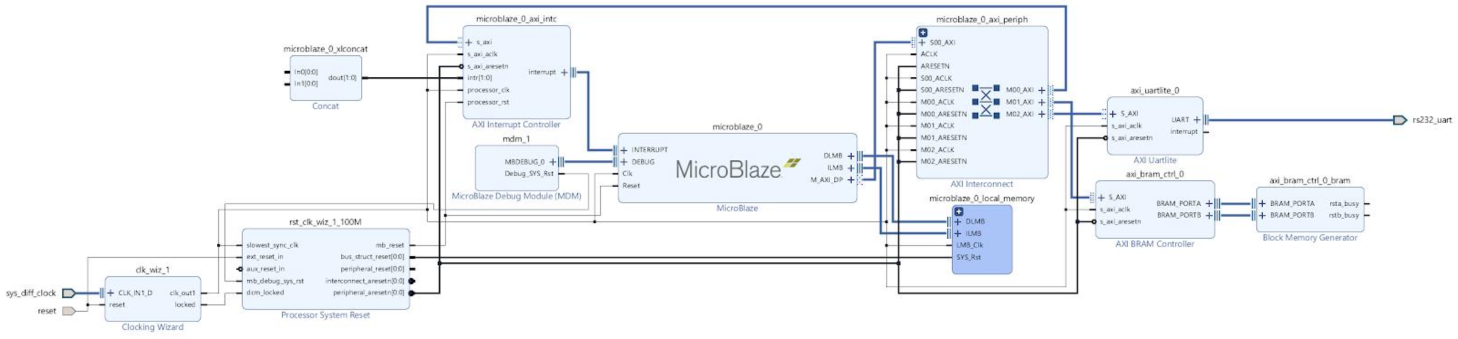


Figure 3: FPGA Block Diagram

In addition to sending the image from the CPU to the FPGA, we also send a byte that encodes the particular exercise that we are analyzing (stored at the address 0xC002_0D00). With the assistance of this byte, we know which of the joints we need to locate - by performing the various morphological transformations. In the case that we do not need to locate a particular joint, we simply return a location of (0, 0) since we know that we will not be using this joint in our posture analysis section. By encoding the byte for the exercise being analyzed, we were able to reduce the computation required to allow us to process 5 joints rather than all 8 joints, allowing us to meet our feedback requirement.

D. Posture Analysis

The output of the FPGA will feed the joint locations to the posture analysis component of the computer. Once the computer receives the joints from the FPGA, it will first perform an error handling to check for invalid joints, or determine if trackers are not detected properly. It will append an "Invalid Joints Detected" to the feedback list.

With a list of joint positions, the algorithm will create lines to connect adjacent joints. As seen in the first move of the leg raise in Figure 4, the shoulder and hip joints will form Line 1. With these lines, angles and slope can be calculated and compared to our predetermined models.

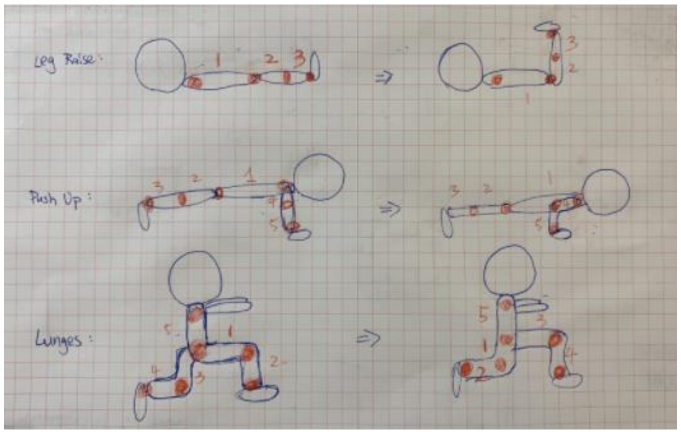


Figure 4: Hand Drawn Skeletons

In the pictures shown, the green line represents the perfect form. The red, blue, and orange lines represent the thresholds or the worst case positioning before a feedback is triggered. In the leg raise (Figure 5) the angles at the knees will have to be around 180 degrees. There will be a lenient threshold of 15 degrees to account for the different human anatomy and angles of the webcam. This means that if the angle is less than 165 degrees the feedback will be appended to a feedback list. There are a total of three different checks for each workout. If the angle at the hips are more than 105 degrees which is 15 degrees from being perpendicular, then the feedback "Raise your Legs Higher" would be appended to the feedback list. If the angle at the hips is less than 87 degrees which is 3 degrees from the perpendicular, then the over-extending feedback would be returned.

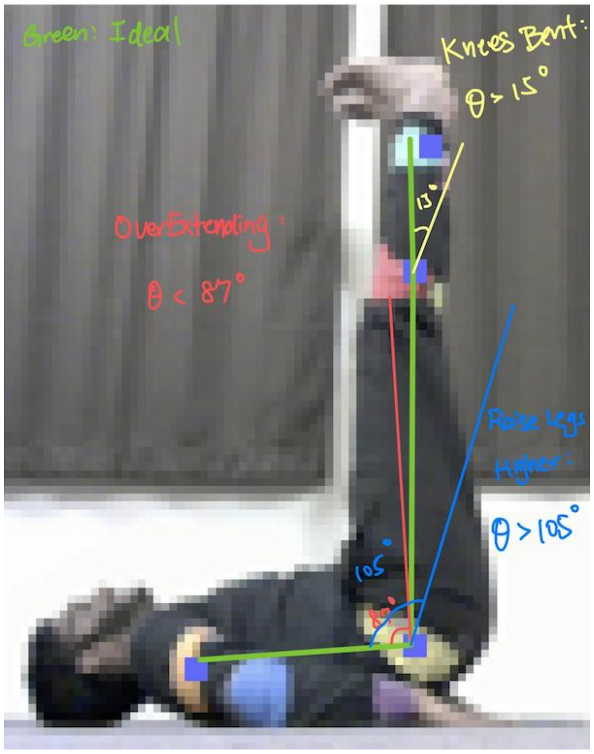


Figure 5: Leg Raise Analysis

For a lunge (Figure 6) the user will be reminded to aim for roughly a 90 degree angle on both legs. If the front knee protrudes further than the ankles more than 10 downscaled pixels, then the feedback "Front Leg too Forward" will be triggered. If the angle of the back knees is greater than 105 degrees which is 15 degrees from being perpendicular, then the users back legs are probably too far extended back. Therefore, the feedback "Back Legs too Backward" will be triggered. If the slope of the thighs in the front leg is less than -0.75, then the user doesn't have the front thighs parallel to the ground. Therefore, the feedback "Go Lower" will be triggered.

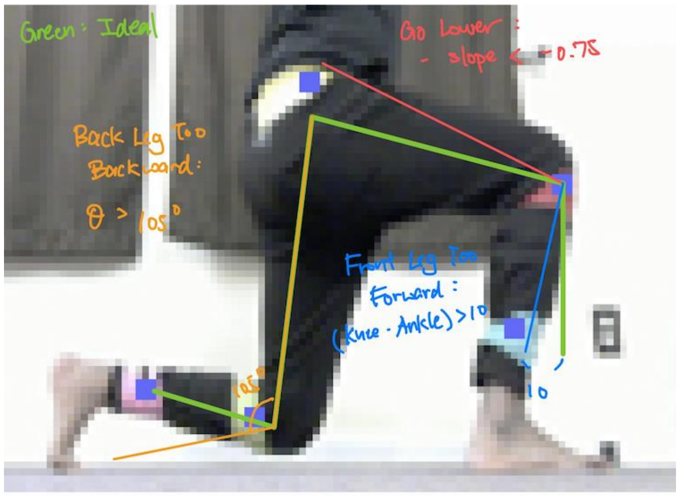

Green: Ideal
Go Lower: - slope < -0.75
Back Leg Too Backward: θ > 105°
Front Leg Too Forward: (Knee - Ankle) > 10
105°
10

Figure 6: Lunge Analysis

For a pushup (Figure 7) the user will not want to hurt their joints by positioning their hands too forward. Therefore, if the wrist joint protrudes further than elbow joint by more than 3 pixels, the feedback "Hand Too Forward" will be triggered. If the elbow joint is higher than the shoulder joint by more than 6 downscaled pixels, then the user will have to "Go Lower". Finally, if the angle at the hip is less than 170 degrees which is a 10 degree threshold from being parallel as well as if the slope of the ankle joint to the hip joint is less than -0.1, then the feedback "Butt Too High" will be triggered.

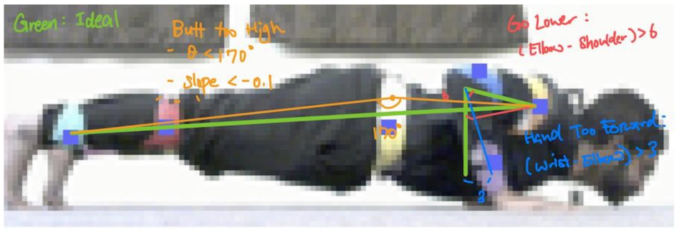

Green: Ideal
Butt too High - θ < 170° - slope < -0.1
Go Lower: (Elbow - Shoulder) > 6
Hand Too Forward: (Wrist - Elbow) > 3
170°
3

Figure 7: Pushup Analysis

Once all the checks are done, the feedback list will be forwarded to the application to output the feedback to the user.

E. User Interface

The user interface will have a few but important distinct components/pages. They will first be presented with a main menu in which there will be three widgets: start workout, view history and settings. In the settings menu the user can adjust their biometrics and change who the logged in profile is. These biometrics will be used to calculate an approximate calorie and heart rate range. The second page will be the history page in which a user can see their associated workouts they had in the past. This will include information about when the workout took place, how long and estimated calories burned. Calories will be calculated by combining the MET value for each workout with a user's weight in kg. MET values are already measured for each workout and easily accessible online. Heart rate will be calculated similarly by taking a look at the user's age, finding their Heart Rate Reserve and then combining that with the MET Reserve percentage.

In the start workout menu, a user can choose which body area they want to focus on and how long they want to work out. Once the workout starts, they will see a trainer show how the workout is performed as well as feedback as they complete reps (See Figure 8). During the workout pictures will be taken for every rep and sent to the FPGA to calculate feedback.

The feedback is returned to the user through red written words on top of the live feed as well as a computer voice. The computer voice is generated with Joanne from Amazon by using Kukarella, a text to audio converter website. Audio feedback will improve the user experience as the user may not be looking at the screen when doing certain workouts, for example a leg raises. This allows the user to still change his or her form.

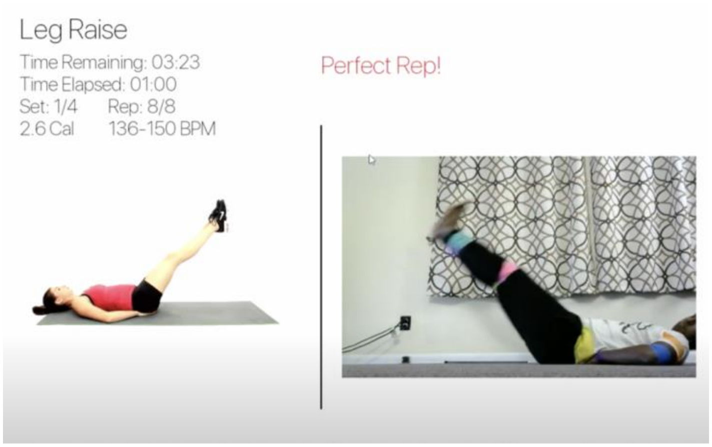

Leg Raise
Time Remaining: 03:23
Time Elapsed: 01:00
Set: 1/4 Rep: 8/8
2.6 Cal 136-150 BPM
Perfect Rep!

Figure 8: Workout UI

VI. Project Management

A. Schedule

The changes made to the schedule from the previous version in the design document to the current version located at the end of this document (Figure 10) were primarily to add more functionality to our final implementation. Vishal worked with Albert to add audio feedback while also working with Venkata to allow users to connect their Spotify account. Venkata was able to finish the FPGA portion and worked with Vishal on portions of the User Interface, namely the history pages. Vishal was able to use this time to fully integrate the database with the interface, which took a little longer than anticipated, and able to make it robust with added functionality such as being able to switch profiles.

B. Team Member Responsibilities

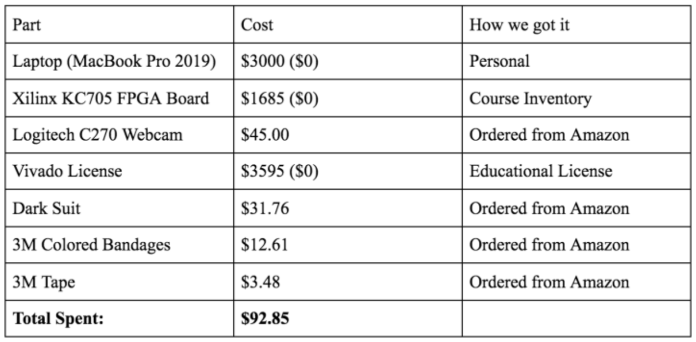
Albert - Implement Posture Analysis to provide feedback, preprocessed image, finished and fine-tuned Color Tracking algorithm to pinpoint the joints, and assist Venkata in converting Python code to C then debug on hardware. Added audio for live feedback.

Venkata - Learn Vivado/Vitis for configuring the FPGA with the UART Protocol and the appropriate image processing defined by Albert. Work with Vishal on the user interface for features such as analyzing previous workout sessions and connecting a user's Spotify account.

Vishal - Design application and user interface with model exercise, live stream, and feedback, as well as design/implement database to store user and workout data. Implement calorie and heart rate estimation per workout, integrate Spotify and customizable profiles.

C. Budget

The budget sheet that entails our finances is attached below (Figure 9).



Part	Cost	How we got it
Laptop (MacBook Pro 2019)	$3000 ($0)	Personal
Xilinx KC705 FPGA Board	$1685 ($0)	Course Inventory
Logitech C270 Webcam	$45.00	Ordered from Amazon
Vivado License	$3595 ($0)	Educational License
Dark Suit	$31.76	Ordered from Amazon
3M Colored Bandages	$12.61	Ordered from Amazon
3M Tape	$3.48	Ordered from Amazon
Total Spent:	**$92.85**	

Figure 9: Budget Table

D. Risk Management

From the start we realized how important it was to manage risk, and we appropriately took many precautions. In terms of the overall design of our system we realized that our project involved three distinct areas: hardware for the FPGA, signals for processing the images and finally software for tying the entire project together. Since these are three distinct areas, we thought the best way to manage design risk would be for each person to take charge of an area and really master it. This would ensure we were not spread too thin and would be able to flush out our respective designs. Our schedule similarly was planned out early on with around two to three weeks of slack and we made sure to identify what MVP looked like and identified extra features to make sure we would have a presentable project by around the halfway point.

In terms of resources, we decided as a team to mostly borrow parts from the ECE department such as the Kintex-7 FPGA as it is very costly, but it allowed us to still have a large budget left in case we needed any extra items to solve future problems. In terms of time, we made sure the primary part of the project was completed before Thanksgiving as we were still in person together so that we would at least have MVP and most of the filming for the demo out of the way, with time left to put everything together.

Another useful strategy we employed was to frontload the schedule for each of us. We knew that other classes would not have picked up much yet and wanted to take advantage by getting to a basic working state for each component in the first three weeks. In terms of the signal side, we ensured that the color tracker algorithm worked in Python so it could eventually be translated to HLS. For the hardware side we ensured that the FPGA could communicate with the computer through the UART Protocol and store basic data. For software we ensured that Pygame was integrated with OpenCV to handle image captures and had basic menu flow setup. All of these strategies helped us mitigate our overall risk and create a successful project within a short time frame.

VII. Related Work

There are currently numerous home workout options available. For instance, *Mirror* is a popular home workout option that provides users with customized workouts and detailed summaries of workout sessions. It involves a large display which allows users to watch the trainers perform the exercises as they progress through the workout. The display allows the users to observe themselves as they workout and can also function as a simple mirror when not in use. Though incredibly useful, it has a high price point of $1495 and does not provide the capability to provide live feedback to users. Falcon provides the same functionality as *Mirror* by providing customized workouts but also provides live feedback to users by tracking their posture as they workout, which is an incredibly important piece of information to determine the effectiveness of a workout session.

While performing the appropriate research for our project we came across two projects that provided us with a baseline and key design decisions that helped shape our own decisions: *Identity Checker on FPGA* and *Virtual Yoga Coach*. The *Identity Checker on FPGA* is a capstone project from the Spring 2019 semester that strived to perform facial recognition by implementing the Viola-Jones algorithm on a Kintex-7 FPGA. Even though the use case of the project is fairly different from Falcon, the design choices they made (such as the use of the appropriate FPGA and communication protocol) provided insight as to how we should design our project because both projects involve the use of computer vision with an FPGA.
Falcon is also fairly similar to *Virtual Yoga Coach,* which is a project that strives to provide real-time feedback to users as they perform various yoga exercises. At a high level, both *Virtual Yoga Coach* and *Falcon: the Pro Gym Assistant* provide similar functionality in that they provide real-time feedback regarding the users' posture. However, the key difference between both projects is that the *Virtual Yoga Coach* has the image processing done on the host computer with the OpenPose library, whereas *Falcon: the Pro Gym Assistant* streams the image to an FPGA which does the image processing. Furthermore, the *Virtual Yoga Coach* provides feedback in less than 5 seconds. Since our project strives to provide feedback for exercising where each rep is significantly

less than 5 seconds, our project has a tighter window for providing feedback and cannot make the same design choices as *Virtual Yoga Coach*. Nevertheless, the project provided us with key information that shaped the information collected and the feedback provided to the users.

VIII. Summary

A. Future Work

To make our application more user friendly, an improvement to our product will be to add a skeleton on top of the live feed that shows the perfect position of a workout. Currently, the posture analysis returns feedback based on a pre-made model with a small threshold to allow for margins. There will be modifications we need to make on the posture analysis code to return the pixels as well as the feedback. The application will have to scale it back to the original image which will require some fine-tuning. The user will be moving within a workout, so the skeleton has to be synced after every rep as well as the timing when it appears to provide a user-friendly experience.

In addition to improving the user interface, we could also improve our analysis. Though we were able to satisfactorily meet our end goal with the assistance of trackers, the trackers do not make for an ideal user experience. In order to appropriately track the trackers, we did a lot of fine tuning to get rid of any background noise. Exploring a different way to track the trackers such as with the assistance of RFID would be worthwhile. Likewise, the project could also be improved by determining a better way to determine when a user is at the position that we wish to analyze. Our current implementation involves analyzing the posture during a time range when we expect the user to be at the final position. However, this could have issues in the case that users fall behind.

B. Lessons Learned

The most important lesson we learned is that time and communication are key for any project that involves large amounts of integration. Since our project had a high amount of integration with each team member working on a separate component, as explained in the Risk Management section, we often ran into issues that arose due to miscommunications and issues that we did not consider. For instance, we did not check that all of the team members were using the same version of Python and so when we started integrating, we ran into dependency issues. We then had to spend time to ensure that all of our various portions were synchronized to the same versions and still functioned appropriately. Though we placed an emphasis on proper planning at the beginning of the project, we still ran into issues and would recommend future groups to consider allocating extra time for integration to address similar issues.

Another lesson we learned was that it is important to consider future areas of growth for the project such as in the form of reach goals. We briefly discussed reach goals at the beginning of our project but did not spend a lot of time considering how they would possibly integrate into our final project. As a result, we were not sure what to do close to the end of the project. We were able to supplement our visual interface with audio through audio feedback and music but these were simple add-ons and did not significantly affect the core of our project. So, we would urge future groups to define extra features during the initial planning phase or consider working on a project that is buildable and so, adding functionality is straightforward.

IX. References

[1] B, Sam, and Andew F. *Python Pillow HSV Color Selection; Making It More Specific*. 1 May 1968, stackoverflow.com/questions/55476067/python-pillow-hsv-color-selection-making-it-more-specific.

[2] "Calories Burned for Workouts." *SparkPeople*, www.sparkpeople.com/resource/calories_burned.asp?exercise=75.

[3] "Estimate How Many Calories You Are Burning With Exercise." *Hospital for Special Surgery*, www.hss.edu/conditions_burning-calories-with-exercise-calculating-estimated-energy-expenditure.asp.

[4] *Image Module* . pillow.readthedocs.io/en/stable/reference/Image.html.

[5] Kukarella. *Audio to Text: Kukarella*. www.kukarella.com/.

[6] *Maximum Size Rectangle Binary Sub-Matrix with All 1s*. 11 June 2020, www.geeksforgeeks.org/maximum-size-rectangle-binary-sub-matrix-1s/.

[7] Mordvintsev , Alexander, and Abid K K. "OpenCV-Python Tutorials Documentation." *Build Media*, 17 Nov. 2017, buildmedia.readthedocs.org/media/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf.

[8] *Morphological Transformations*. docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html.

[9] *Multiple Color Detection in Real-Time Using Python-OpenCV*. 10 May 2020, www.geeksforgeeks.org/multiple-color-detection-in-real-time-using-python-opencv/.

[10] "Pygame Front Page." *Pygame Front Page - Pygame v2.0.1.dev1 Documentation*, www.pygame.org/docs/.

[11] Radames. "OpenCV VideoCapture Running on PyGame - Repo Ref Https://Github.com/Radames/opencv_video_to_pygame." *Gist*, gist.github.com/radames/1e7c794842755683162b.

[12] Renzym Education. AXI BRAM Controller, Custom AXI Slave - 1, Digital System Design 2018 Lec 8/30. 3 Oct. 2018, www.youtube.com/watch?v=Q-2IRM8HHtY.

[13] "Threading - Thread-Based Parallelism." *Threading - Thread-Based Parallelism - Python 3.9.1 Documentation*, docs.python.org/3/library/threading.html.

[14] Welcome to Spotipy!¶. spotipy.readthedocs.io/en/2.16.1/.

[15] XilinxInc. AXI UART Lite v2.0. 15 Apr. 2017, www.xilinx.com/support/documentation/ip_documentation/axi_uartlite/v2_0/pg142-axi-uartlite.pdf.

[16] XilinxInc. Creating a Simple MicroBlaze Design in IP Integrator. 7 Jan. 2016, www.youtube.com/watch?v=VjYdNIOyRcE.

[17] XilinxInc. "Kintex-7 FPGA KC705 Evaluation Kit." Https://Www.xilinx.com/Support/Documentation/boards_and_kits/kc705/2014_2/ug883_K7_KC705_Eval_Kit.Pdf, 22 Aug. 2014.

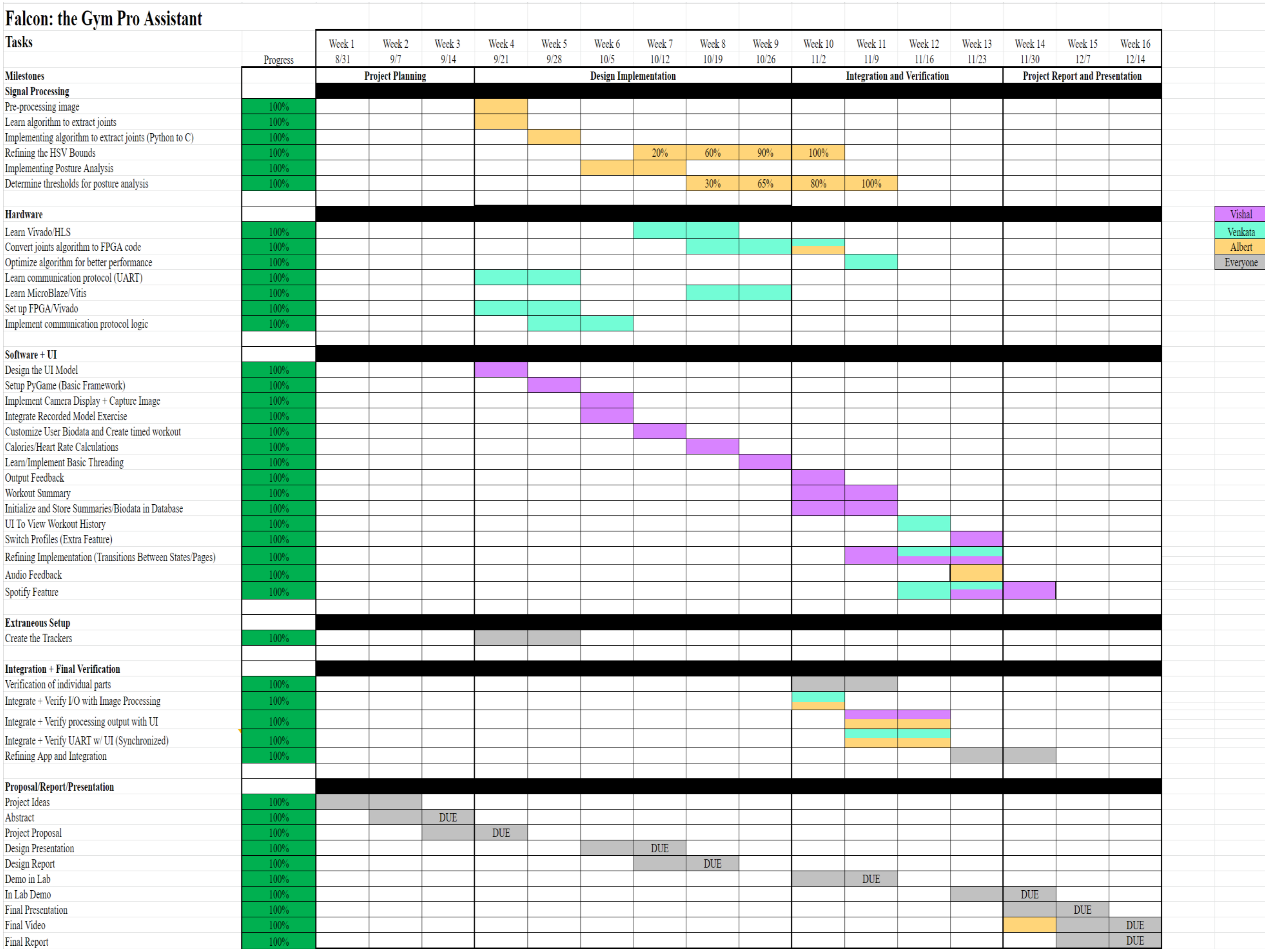


Figure 10: Gantt Chart