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, SQLite**

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 the settings, 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.

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.

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 multiple workout options and have the option to save their profile and biometric information. Workouts will have feedback and statistics can be viewed at a later time.

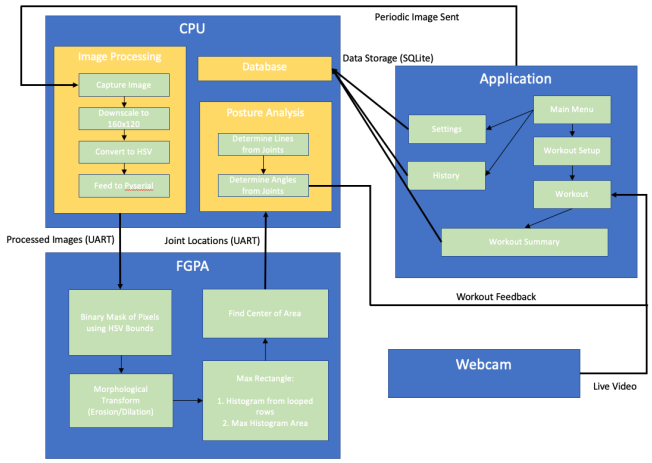


Figure 1. System picture.

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 security purposes.

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.

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

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

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

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 RAM, 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.

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. *Posture Analysis*

The output of the FPGA will feed the joint locations to the posture analysis component of the computer. 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 2, the shoulder and hip joints will form Line 1. With these lines, angles and slope can be calculated and compared to our predetermined models. For example, the second move of the pushup Figure 2, Line 1, 2, 3, and 4 will have to be parallel to each other. The angles at the hip and knee would have to be around 180 degrees while the angle at the elbow will be around 90 degrees. There will be thresholds for each check to account for the different human anatomy and angles of the webcam. We haven't determined the precise thresholds yet, but an example may be to classify an angle within 80 to 100 degrees to be perpendicular, Last but not least, if the necessary checks haven't been met, the algorithm would output feedback in the form of a list of strings, such as "Raise your legs higher".

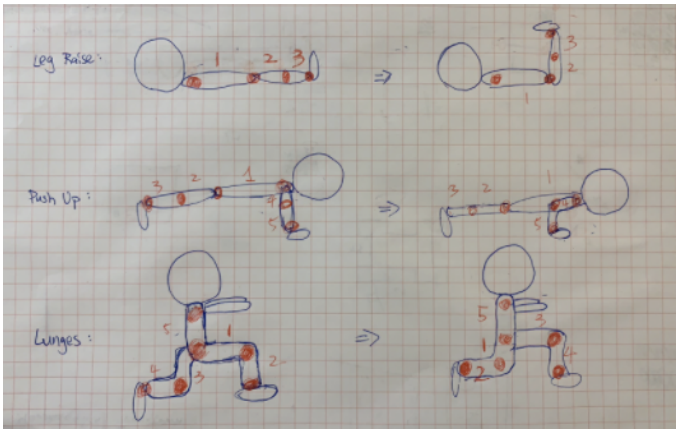


Figure 2. Hand-drawn Sketches

D. *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. 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 3). During the workout pictures will be taken for every rep and sent to the FPGA to calculate feedback.

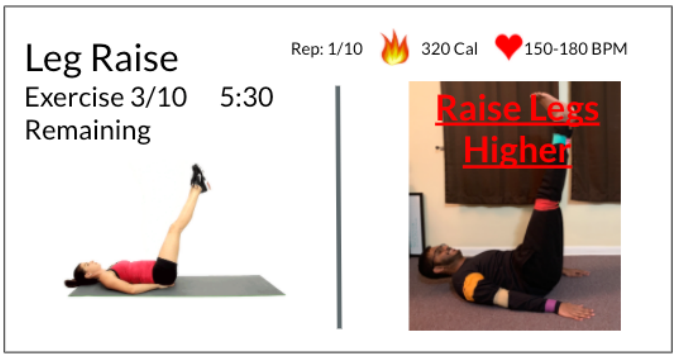


Figure 3. Mockup of the UI during a workout

VI. Project Management

A. *Schedule*

Currently, we have finished implementing the joint tracking algorithm and have roughly tuned the HSV bounds to track the color. The next steps will be to learn Vivado High Level Synthesis in order to convert the currently written Python code into RTL for the FPGA. We also have the posture analysis to be able to take in positions and output feedback; however, it has yet to be tested. On the hardware side, we have gotten the licenses and framework set up and have been able to send pixels over and configure our necessary baud rate. In terms of the User Interface, we have a basic structure that can take in video and has a model exercise. The schedule for this project is located at the end of this report.

B. *Team Member Responsibilities*

Albert - Implement Posture Analysis to provide feedback, pre-process image and finish Color Tracking algorithm to pinpoint the joints and assist Venkata in converting Python code to C then optimize it into RTL.

Venkata - Learn Vivado for High Level Synthesis and the UART Protocol, send pixels to the FPGA, and work with Albert in converting image processing algorithms to RTL and optimize them.

Vishal - Design application and user interface with model exercise, live stream, and feedback, as well as create a database to store user and workout data. Integrate webcam with system.

C. Budget

The budget sheet is attached below (Figure 4).

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 4. Budget Sheet

D. Risk Management

A risk that we could potentially encounter when integrating the project is that the downscaled image would be not detailed enough for us to track the color trackers. The most reasonable approach would be to not downscale it to as low as 160 x 120 pixels. However, as mentioned earlier, the sending of pixels from the computer to the FPGA would be the biggest bottleneck. Thus, we can have a calibration system for the user to calibrate the ceiling and floor. In the pre-processing portion, we can crop out portions of the image to decrease the number of pixels sent to the FPGA.

Another potential problem that might arise is that the C code would not be able to synthesize to 300 MHz clock frequency. We would have to reduce the baud rate as a result, which would increase the overall latency. Thus, we will have to do our best to optimize and the worst case would be to modify the exercises to support a longer feedback delay, such as a plank.

The application might be at risk of being choppy because of the multiple threads that may be running at the same time. We can separate the model exercise, live stream, feedback, and workout data into different applications.

VII. Related Work

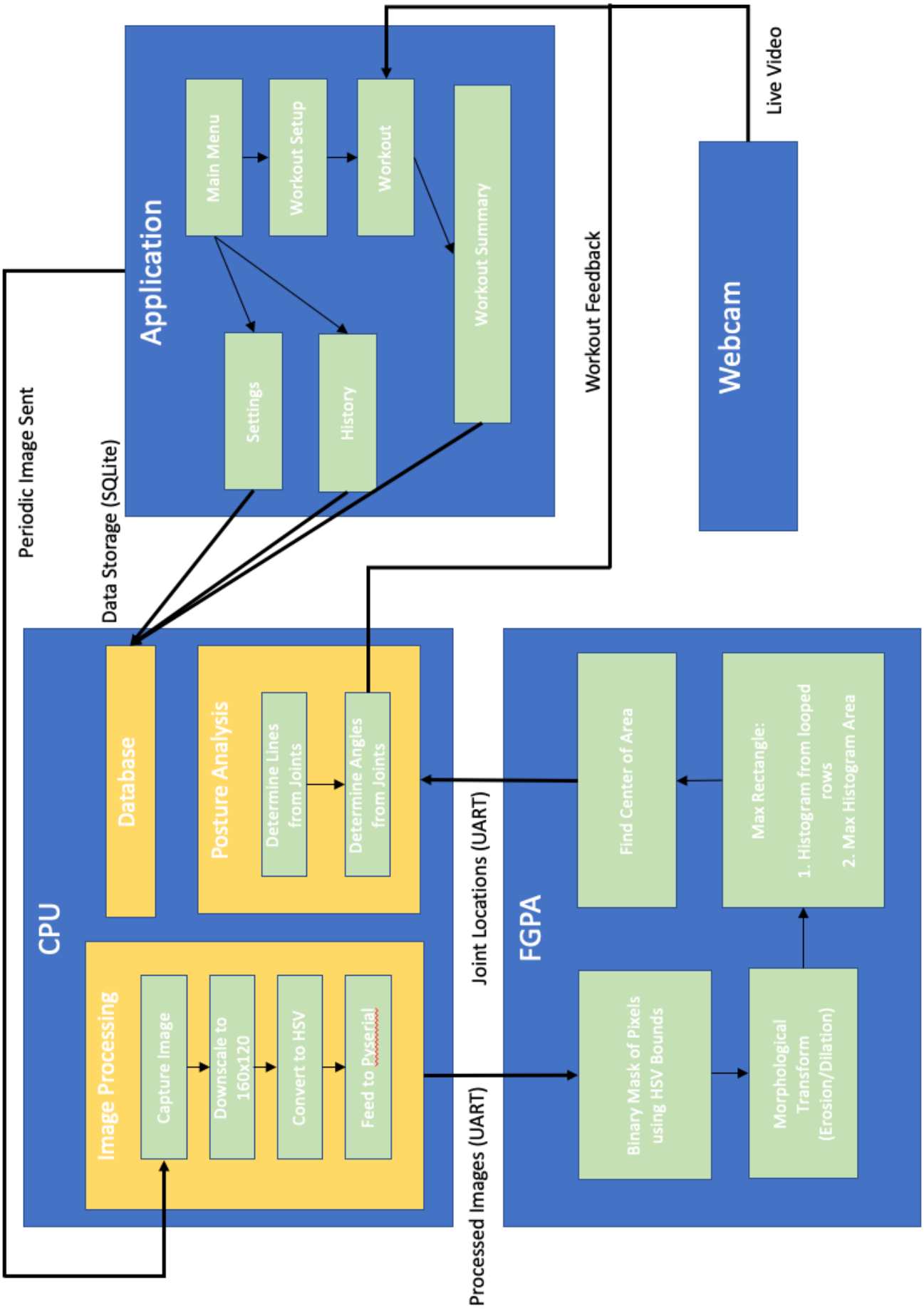
There are currently numerous home workout options available. For instance, *Mirror* is a popular home workout option that provides users with customized workouts and detailed summaries of workout sessions. It involves a large display which allows users to watch the trainers perform the exercises as they 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.

Falcon: the Gym Pro Assistant

Tasks	Week 1 8/31	Week 2 9/7	Week 3 9/14	Week 4 9/21	Week 5 9/28	Week 6 10/5	Week 7 10/12	Week 8 10/19	Week 9 10/26	Week 10 11/2	Week 11 11/9	Week 12 11/16	Week 13 11/23	Week 14 11/30	Week 15 12/7	Week 16 12/14
Milestones	**Project Planning**			**Design Implementation**						**Integration and Verification**				**Project Report and Presentation**		
Signal Processing																
Pre-processing image to downscale image																
Learn algorithm to extract joints																
Implementing algorithm to extract joints																
Implementing Posture Analysis																
Determine thresholds for posture analysis																
Hardware																
Learn Vivado/HLS																
Convert joints algorithm to FPGA code																
Optimize algorithm for better performance																
Learn communication protocol (UART)																
Implement communication protocol logic																
Software + UI																
Design the UI Model																
Setup PyGame (Basic Framework)																
Implement Camera Display + Capture Image																
Integrate Recorded Model Exercise																
Customize User Biodata and Create timed workout																
Keep Track of Data and Output Feedback																
Refining Implementation																
Extraneous Setup																
Create the Trackers																
Integration + Final Verification																
Verification of individual parts																
Integrate + Verify I/O with Image Processing																
Integrate + Verify processing output with UI																
Refining App and Integration																
Proposal/Report/Presentation																
Project Ideas																
Abstract			DUE													
Project Proposal				DUE												
Design Presentation							DUE									
Design Report								DUE								
Demo in Lab											DUE					
In Lab Demo														DUE		
Final Presentation															DUE	
Final Report																DUE

Vishal
Venkata
Albert
Everyone


Application
Main Menu
Workout Setup
Workout
Workout Summary
Settings
History
Live Video
Webcam
Workout Feedback
Periodic Image Sent
Data Storage (SQLite)
CPU
Database
Posture Analysis
Determine Lines from Joints
Determine Angles from Joints
Image Processing
Capture Image
Downscale to 160x120
Convert to HSV
Feed to Pyserial
Joint Locations (UART)
Processed Images (UART)
FGPA
Find Center of Area
Max Rectangle:
1. Histogram from looped rows
2. Max Histogram Area
Binary Mask of Pixels using HSV Bounds
Morphological Transform (Erosion/Dilation)