Posture Sensing Smart Chair

By

Brian Hill

Geonil Kim

Steven Zhou

Final Report for ECE 445, Senior Design, Spring 2020

TA: Jonathan Hoff

08 May 2020

Project No. 69

Abstract

The posture sensing chair is a system that utilizes computer vision to analyze one's posture and audio feedback subsystem to notify the user when the posture is poor. In our society, back problems have become one of the biggest issues as more people spend their time at their desks without realizing how their postures or back problems have aggravated. The original solution to this problem was utilizing ultrasonic and pressure sensors to ensure the user is sitting upright, but our new solution uses a camera module and Raspberry Pi with audio feedback. The camera module sends images to Raspberry Pi, and the Raspberry Pi will then analyze the user's posture and communicate with the microcontroller on the chair to alert the audio subsystem when the posture goes poor. The user is also able to interact with the system in real-time through the web application that is built on Ruby on Rails. Our new solution is much more effective as it will allow for more accurate classification and specificity in the feedback to the user.

Contents

1. Second Project Motivation

1.1 Updated Problem Statement

Back problems due to slouching have become a huge cause for concern as people spend more and more time sitting in cubicles, hunched over a computer for hours upon hours every single day. In fact, according to a University of Washington study, forty-five percent of Americans between the ages of 35 and 55 suffer acute back pain each year [1]. Moreover, other studies carried out by the Social Security Administration identified back pain as the top cause of disability under the age of 45 in the United States [1]. Not only is poor posture unsightly, but it also introduces stress on the neck and spine, causing further muscular tension as the body attempts to compensate for the lack of support.

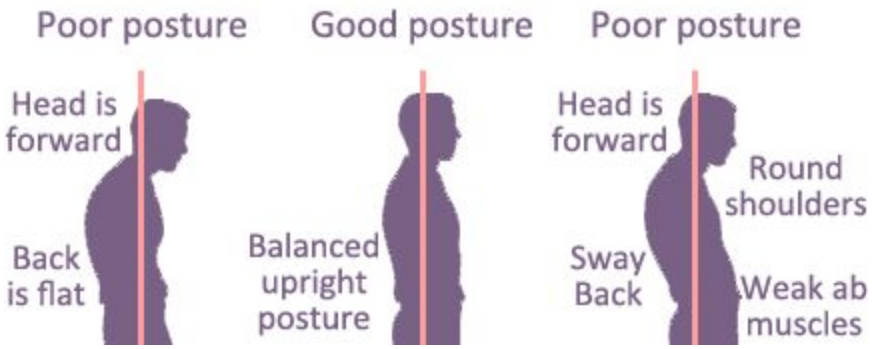


Figure 1: Good and Poor Posture

While maintaining proper posture is something that needs to be taken seriously in perpetuity, we have decided to combat this problem by focusing on time spent sitting in front of a computer, like in a traditional office environment, for example. After all, many adults spend a significant portion of their lives in this position! In response to this, we have set out to create a computer vision system that uses video input of the user's sitting position to determine whether or not they are sitting with proper posture. We will calculate the relative positions of the head, neck, shoulders, upper/lower back, and legs to determine if the user's alignment is safe and healthy. Should the user revert back to a poor sitting position, they will be notified via a combination of flashing lights and vibrations originating from within their chair.

1.2. Updated Solution

From our own experiences with posture-related back problems, we know the hardest part about changing our habits is accountability. While strengthening postural muscles is important in correcting kyphotic posture, it is useless if you are unable to *remember* to put those muscles

to use. By using our product, we hope to make people more aware of the way they are sitting and allow them to monitor their progress as they try to break the habit.

Because poor posture is such a widespread problem there currently exist many solutions on the market. A large portion of these solutions fall under the umbrella of electronic wearables [3], like the Upright GO, shown in figure 2, which attaches to the user's back between the shoulder blades. While we did initially consider the idea of designing our own wearable, we ultimately decided against it because we suspect that many people would find them uncomfortable. Other solutions involve harnesses that physically pull the shoulder blades back, shown in figure 3. The problem with these lies in the fact that they don't support the development of stronger postural muscles and therefore won't actually fix the problem. The last major class of solutions to poor posture in the workplace are ergonomic workstations and include things like standing desks and using an inflatable ball instead of a chair. While these solutions have the right idea in mind, the average person does not want to stand or roll around on a ball for eight hours a day; it's exhausting and unsustainable for many! We believe our proposed solution would have many advantages over the existing products on the market.

Figure 2: Upright Go

Figure 3: Harness

Originally, we wanted to approach this chair with a robust method of using various sensors to read a user's sitting posture, but after more research we figured this could be improved for more accuracy and reliability using computer software. Our first proposed idea had two separate subsystems to gather user information -- a back subsystem and a seating subsystem. Although this method may work, it would be more efficient and reliable to reduce the two measuring systems into one. By having just the software subsystem, we do not have to worry about faulty/inaccurate measurements from the sensors nor do we need the subsystems to be dependent on each other. Our computer vision subsystem will be a module that is placed a few

feet away from the side of the chair. We believe this is by nature the best way to measure since the camera would not be able to see all of the user's segments from other placements (front and back). In addition, computer vision allows a better visual display of a user's sitting posture. Instead of gathering raw numbers to show the user, there are now live images that detect the different segments and check its alignment. Finally, our new proposed system has the advantage over the previous proposal in that the chair will no longer be anchored by a cord as all communication to the chair will be handled using bluetooth and all power going to the chair will come from a Li-ion battery.

Using computer vision to estimate a human's pose is definitely a well-researched topic with plenty of open source software and data sets that we will be able to utilize in our design. Some of the more well known data sets that are available to the public are the COCO 2018 Keypoint Detection, the MPII Human Pose Dataset, and VGG Pose Dataset[10]. Furthermore, there are even several open source models that are pre trained using these datasets which would eliminate any need for us to tackle this very complex machine learning problem ourselves[10]. The image below shows the results of one model that is trained using different datasets.

Figure 4: same model trained with different data sets

Easy implementation of models such as the one depicted above are possible using OpenCV's plethora of useful functions. Originally developed by Intel, OpenCV is an open source computer vision and machine learning software library containing interfaces for several languages --

including python, C++, Java, and MATLAB -- as well as support for Windows, Linux, Android and Mac OS users alike [11]. OpenCV can provide the means to utilize these models to effectively classify the user's posture in real-time. By finding pixel locations of several key points on the user we will be able to calculate different angles formed by these points and use these angles to classify posture. This method of calculating angles would provide a strong advantage over our previous method of using pressure and range sensors because it allows us to better *quantify* their posture and give feedback on what the user should specifically improve on. For example, consider a situation in which the user's shoulders begin to round and their posture falls out of the range of what is determined to be acceptable. While both solutions would be able to inform the user of their poor posture, the computer vision solution would ideally be able to specifically notify the user that they need to realign their shoulders.

1.3. Updated High-Level Requirements

- Using computer vision software, the system must be able to accurately classify the users posture as either good or poor with at least 70% accuracy.
- The computer must be able to collectively analyze the data provided from the computer vision subsystem and send feedback back to the chair to notify the user when their posture is poor by sounding a small speaker.
- The data analyzed must be shown visually through a graphical display -- available via web application -- showing the user's posture over time.

1.4. Updated Visual Aid

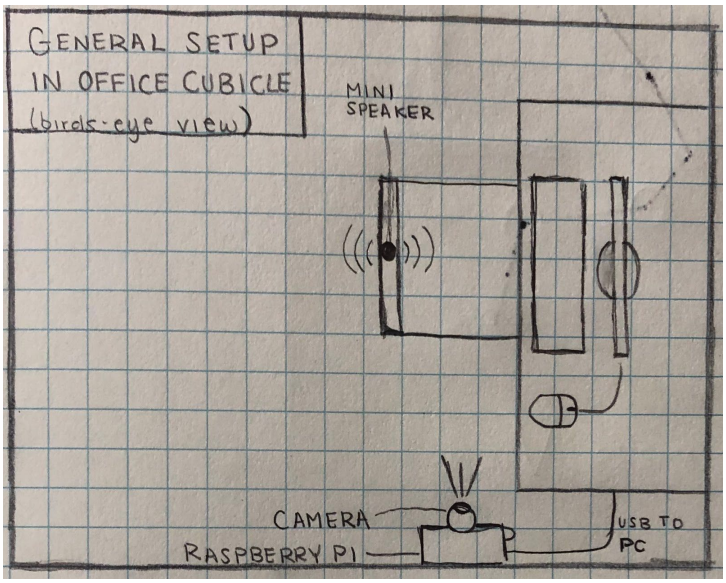


Figure 5: Visual example of how the system might be set up in an office environment

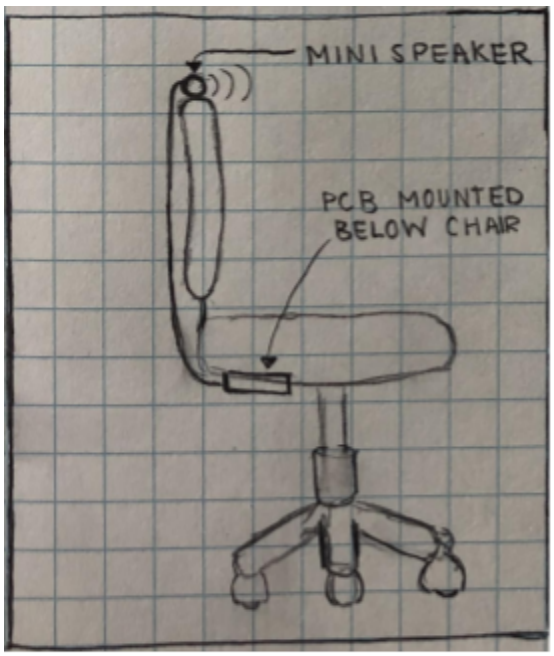


Figure 6: Placement on chair

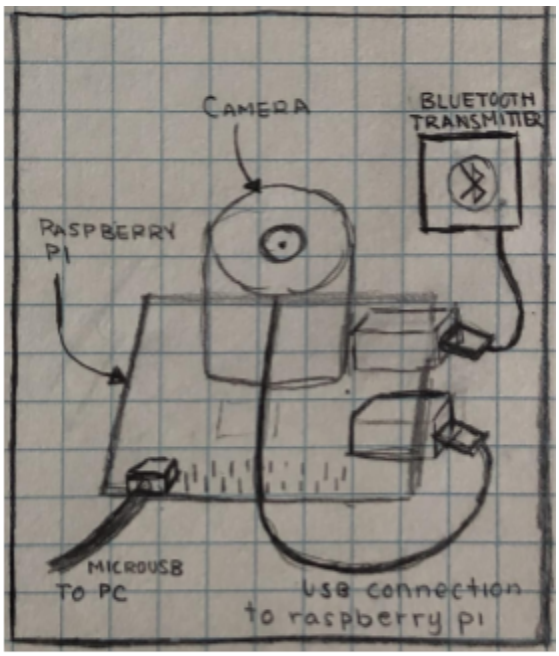


Figure 7: Computer Vision subsystem

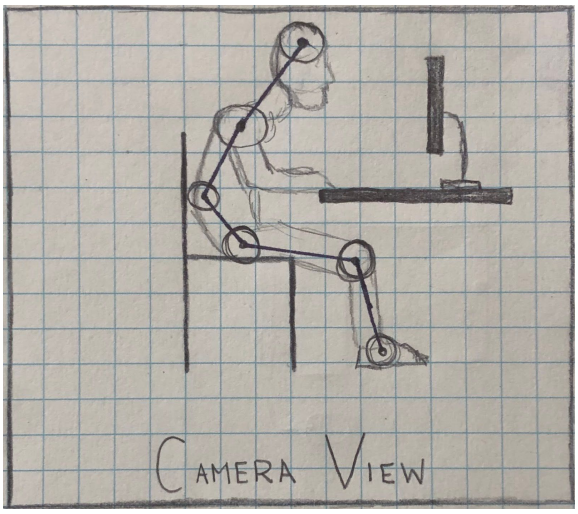
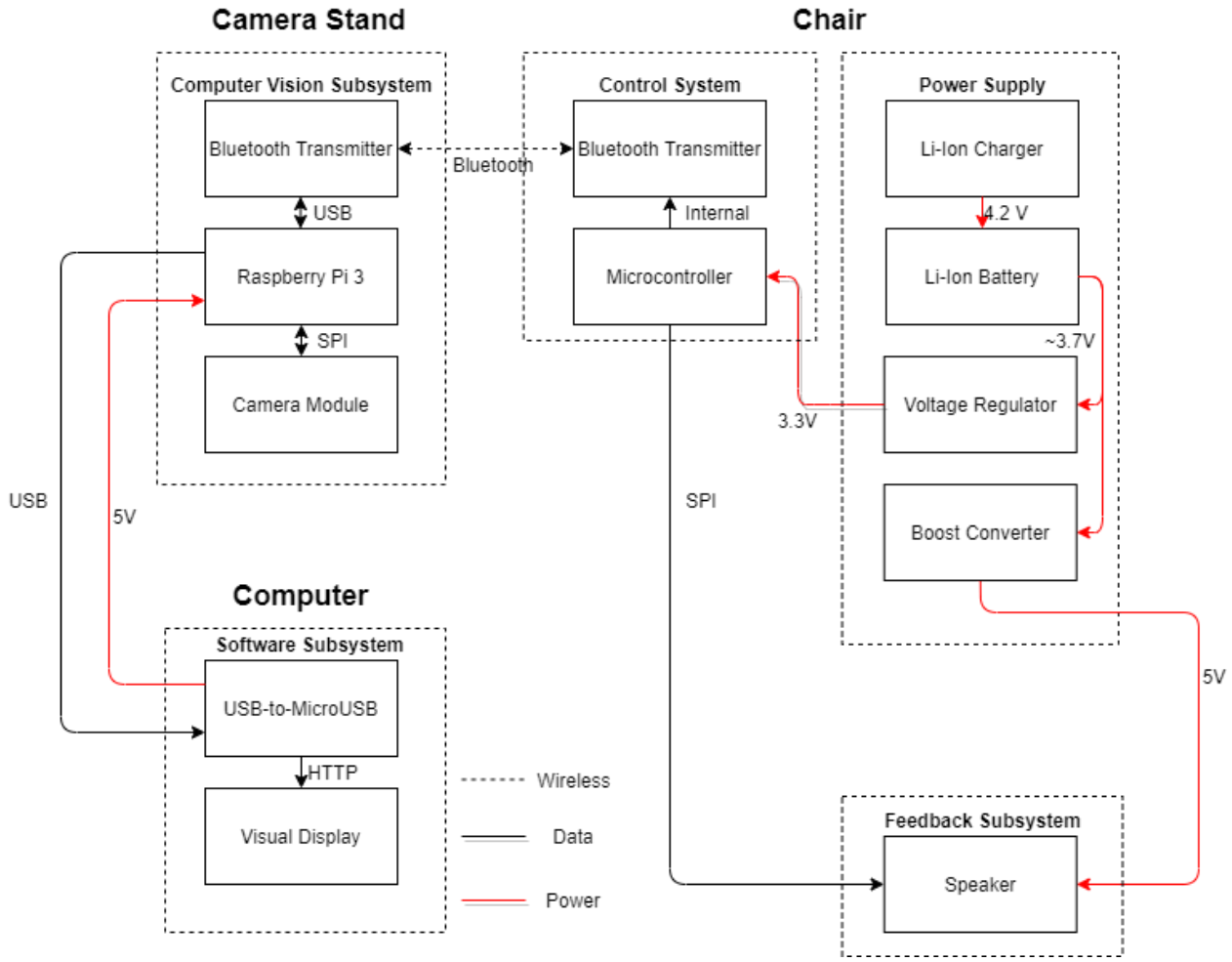


Figure 8: Camera View

1.5. Updated Block Diagram



2. Second Project Implementation

2.1. Implementation Details and Analysis

2.1.1 Implementation

For the implementation portion of our project, one of the main things we focused on was human pose estimation using OpenPose. This is arguably the most important part of our project because our algorithm for classifying posture is totally dependent on our ability to measure various joint angles with respect to the horizontal access in the camera frame. Consequently, measuring joint angles first requires pixel locations of different key points on the body. In order to identify pixel locations of these aforementioned keypoints, we used an open-source implementation of Open Pose containing pretrained models using tensorflow. The model uses the COCO 2018 Keypoint Detection dataset in order to isolate the following points on the user's body: nose, neck, right shoulder, right elbow, right wrist, left shoulder, left elbow, left wrist, right hip, right knee, right ankle, left hip, left knee, left ankle, right eye, left eye, right ear, and left ear[12]. Overall, open pose yields very accurate results under most circumstances[13]. Results from human pose estimation can be seen below in figures 9 and 10.

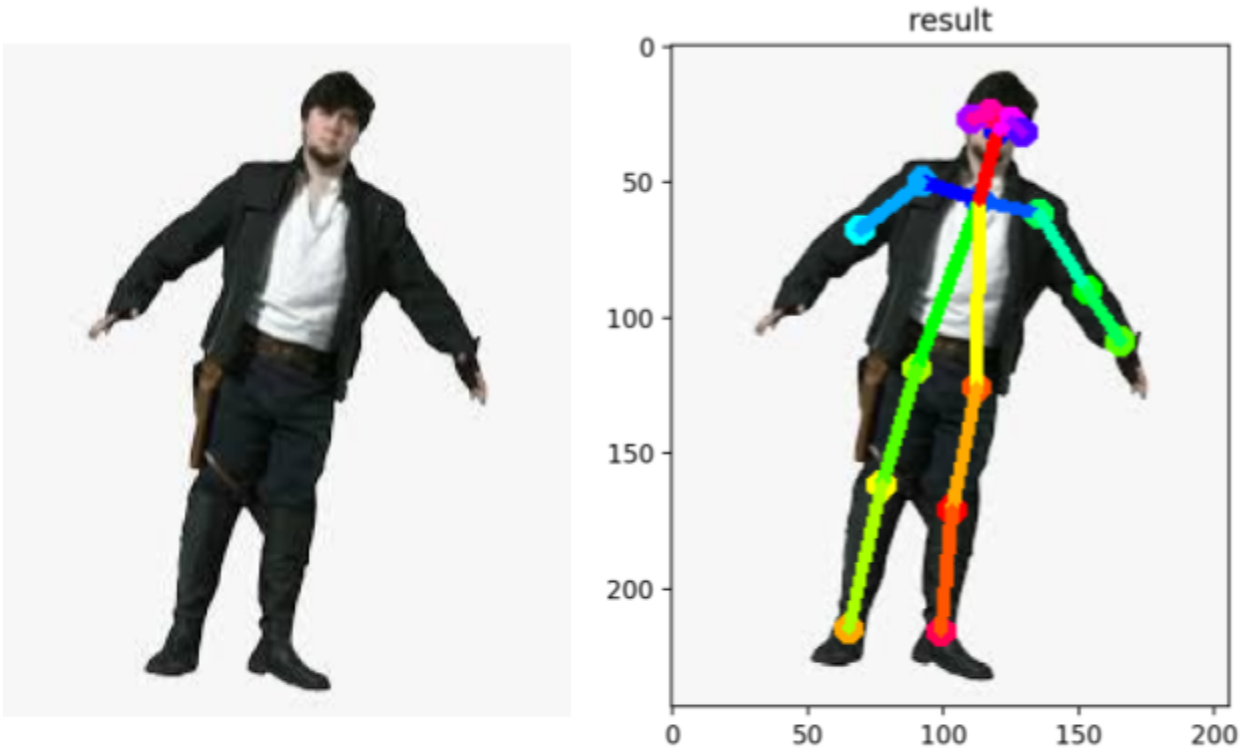


Figure 9: Results of Open Pose with key points and lines drawn (example 1)

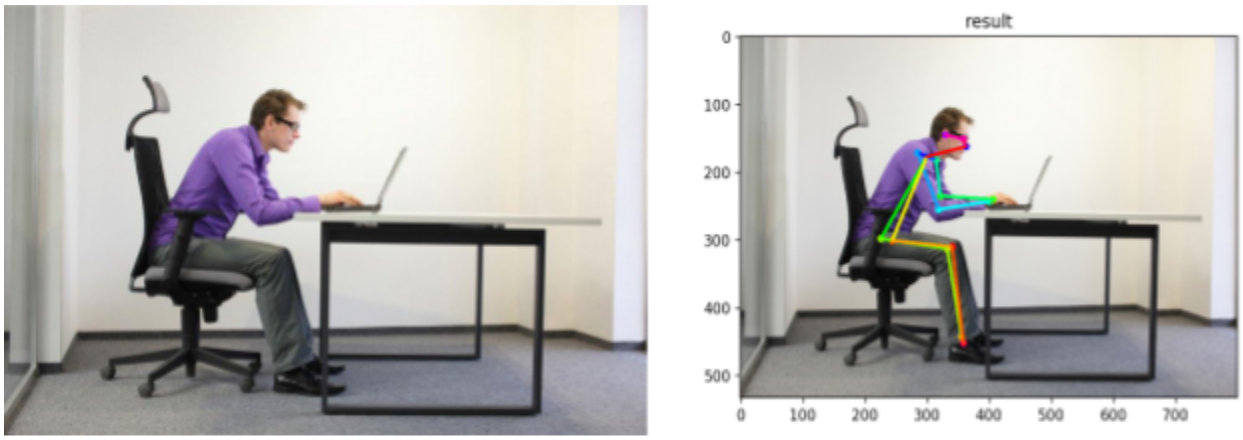


Figure 10: Results of Open Pose with key points and lines drawn (example 2). This is a good example of a typical user image would look like.

The next step in our implementation process was to find the pixel values for the key points that were found by the Open Pose algorithm. For each human found by the Open Pose algorithm, we step through each of the 18 possible key points in the Coco dataset. As an aside, our project doesn't deal with cases in which more than one human is present in the frame so this loop will only execute once. If the keypoint is not found, it is ignored and the program continues to the next key point. If the keypoint *is* present, on the other hand, the pixel value is computed by

$$(x, y) = (ceil(x_{frac} * imageWidth),\ ceil(y_{frac} * image\ Height)),$$

where x/y_{frac} = are the normalized pixel location on the image. Finally the pixel location is appended to a dictionary named keypoints with key corresponding to that body part. This is shown in the code block in figure 11.

```
keypoints = {}

for human in humans:
  for i in range(common.CocoPart.Background.value):
     if i not in human.body_parts.keys():
        continue

     body_part = human.body_parts[i]
     print(map2str(i),'pixel location (x,y):')
     center = (int(body_part.x * imageWidth + 0.5), int(body_part.y * imageHeight + 0.5))
     print(center,'\n')
     if(map2str(i) not in keypoints.keys()):
        keypoints[map2str(i)] = np.array([center[0],center[1]])
```

Figure 11: Code to find pixel locations of keypoints and store in a dictionary for further use

With the pixels stored in a dictionary, one could easily access the pixel value for a given body part by indexing the dictionary with the body part name. For example, keypoints['Nose'] would give the pixel locations of the user's nose (if it was found by the Open Pose algorithm). After finding the pixel locations of the key points the next step was finding the angle that pair of key points form with the horizontal axis of the image. This is illustrated in Figure 12 below

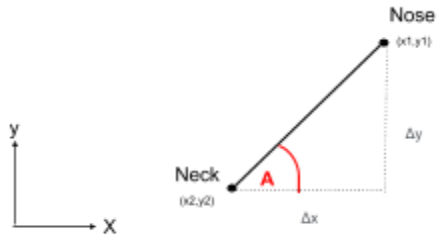


Figure 12

.

The angle of the Nose - Neck line shown above is given by:

$$A = arctan2(\Delta y, \Delta x)$$

where,

$$\Delta y = y1 - y2$$
$$\Delta x = x1 - x2$$

The angles for a selected list of pairs could then be calculated using the code block in figure 13 and stored in a dictionary, similar to the way the pixel locations were stored.

```
CustomPairs = [
  (1, 2), (1, 5), (2, 3), (3, 4), (5, 6), (6, 7), (1, 8), (8, 9), (9, 10), (1, 11),
  (11, 12), (12, 13), (1, 0), (0, 14), (14, 16), (0, 15), (15, 17), (2, 16), (5, 17)
]  #each number is mapped to a different body part as defined in the COCO dataset
CustomPairsRender = CustomPairs[:-2]

angles = {}

for pair_order, pair in enumerate(CustomPairsRender):
  if pair[0] not in human.body_parts.keys() or pair[1] not in human.body_parts.keys():
     continue

  newkey = map2str(pair[0]) + '->' + map2str(pair[1])
  pointupper = (int(human.body_parts[pair[0]].x * imageWidth + 0.5), int(human.body_parts[pair[0]].y * imageHeight + 0.5))
  pointlower = (int(human.body_parts[pair[1]].x * imageWidth + 0.5), int(human.body_parts[pair[1]].y * imageHeight + 0.5))
  if(pointupper[1]<pointlower[1]): #if point upper has a higher y value, then switch
     temp = pointupper
     pointupper = pointlower
     pointlower = temp
  angle = np.degrees(np.arctan2(pointupper[1]-pointlower[1],pointupper[0]-pointlower[0]))
  if(newkey not in angles.keys()):
     angles[newkey] = angle
```

Figure 13

2.1.2 Results

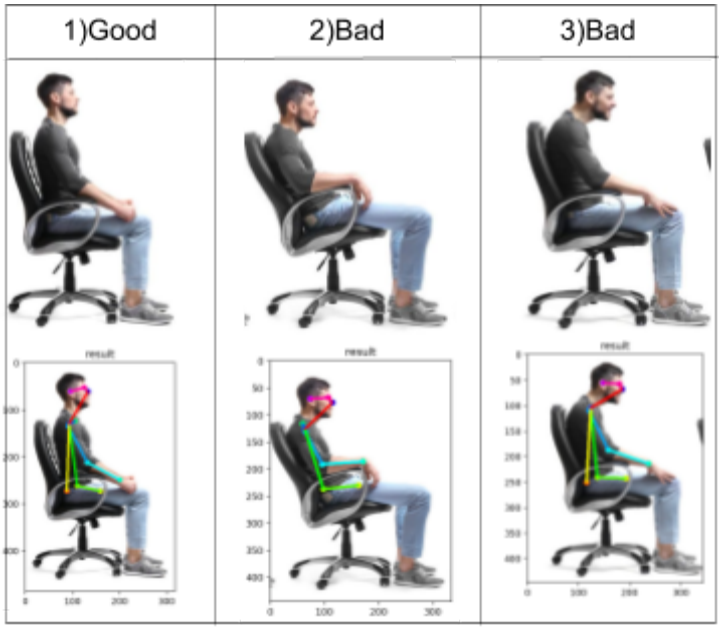


Figure 14

	1)Good	2)Bad	(1) - (2)	3)Bad	(1)-(3)
Nose->Neck	120.101	136.146	-16.045	149.172	-29.071
Nose->RShoulder	119.578	132.797	-13.219	147.131	-27.553
Nose->LShoulder	113.962	144.660	-30.698	NOT FOUND	N/A
Neck->RShoulder	111.801	78.690	33.111	120.964	-9.163
Neck->LShoulder	153.435	68.198	85.237	NOT FOUND	N/A
Neck->LHip	92.003	NOT FOUND	N/A	93.136	-1.133
Neck->RHip	82.661	72.969	9.692	84.540	-1.879
Neck->REar	92.490	100.124	-7.634	116.095	-23.605
RShoulder->RElbow	62.549	62.676	-0.127	64.335	-1.786
RShoulder->RHip	81.491	72.707	8.784	83.037	-1.546
RShoulder->REar	93.865	98.392	-4.527	116.565	-22.7
RElbow->RWrist	28.201	176.186	-147.985	18.868	9.333
LShoulder->LHip	96.508	NOT FOUND	N/A	NOT FOUND	N/A
RHip->RKnee	12.652	173.853	-161.201	2.045	10.607
RHip->REar	86.016	81.822	4.194	93.403	-7.387

Figure 15: Joint angles corresponding to images in figure 14

*More Sample Results are available in the appendix

2.1.3 Significance of Implementation

Although it is difficult to classify posture as good or poor just from the joint angles, our implementation would be extended (given more time) to include a calibration module in order to get a better estimate of what the joint angles correspond to good posture for any given individual. The calibration module would work as follows:

1. Collect several minutes of video data of the user demonstrating good posture for *the entire time* (assume **n** frames)
2. For each frame in the video calculate all joint angles (assume **i** different angle types)
    a. At this point should have $\{A_1,A_2,A_3,...A_i\}$ where A_x is the set of all joint angles of a given type over n frames ($|A_x| = n$)

3. Calculate average and standard deviation of each joint angle type over all frames
    a. This results in $\{\mu_1, \mu_2, \mu_3, \ldots, \mu_i\}$ and $\{\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_i\}$

After Collecting this calibration data, we can use this to classify subsequent images of the user's posture by doing the following:

1. Calculate joint angles from the input image (denoted as $\{B_1, B_2, B_3, \ldots B_i\}$)
2. For x in range(0,i):

    if ($\mu_x - \mathbf{k_x}\sigma_x <= B_x < \mu_x + \mathbf{k_x}\sigma_x$):

        continue

    else:

        posture is poor
3. If all joint angles are in the given range, then posture is good

The $\mathbf{k_x}$ value above is something that we would have to determine experimentally but the idea behind it is that certain joint angles can vary more without poor posture. For example, the joint angle from the elbow to wrist can vary quite a bit throughout the day whether good posture is maintained or not. Therefore, if we want to weight a certain joint angle more heavily, we could decrease k so that the resulting range is more scrutinous.

In summary, implementing the human pose estimation and joint angle calculation is vital to carrying out our proposed algorithm for classifying posture and is therefore a necessary component in reaching our high level requirements

3. Second Project Conclusions

3.1. Implementation Summary

In our revised version, the Posture-Sensing Chair 2.0, we have successfully implemented an improved design that covered areas of cost-reduction, product quality, analysis optimization, and future development opportunities. We considered the various flaws from our first version that were fixed in this model:

- **Cost Efficiency**
  We were able to reduce the cost of the entire product by replacing the high costs of the complex grid of pressure/range sensors with a simple computer module for computer vision.

- **Product Quality**
  In our previous design we implemented the back and seating subsystems which depended on each other to determine the posture status. These subsystems were essentially connections of individual sensors that stream data to the microcontroller unit. In practice, by replacing all these various complexities with a compact camera system, the quality control and reliability exponentially increased with the benefit of lower cost.
- **Analysis Optimization**
  Using computer vision technology as an alternative to the inefficient sensor measuring system opened up many more resources and accurate methods proven by research. In our product, we decided to utilize the OpenPose software and the camera module to perform Human Pose Estimation from a side view. With its high accuracy in pinpointing the necessary joints, there are more accurate readings for posture analysis.
- **Future Opportunities**
  Implementing computer vision to this design allows us to further optimize any software or hardware utilized to help the user. Such opportunities include machine learning to further assist the learning model of creating accurate posture analysis.

With these improvements to the design, we developed a product that is more efficient and cost-friendly to assist the user with their sitting posture.

3.2. Unknowns, uncertainties, testing needed.

Due to the fact that we were unable to actually test the design physically in the lab there were many factors that were uncertain to be successful:

- Our debugging process is greatly held back without seeing how the system would work together. An example would be testing any flaws in the device and software that could affect the procedure.
- Unknown variables such as camera positioning, design, wiring flaws.

The most probable and ideal alternative to tackle these factors would be to spend extensive research on system compatibility and unit testing with parts that we could build independently. However, without the proper equipment and environment we might be hindered by an abundance of unknown variables to make sure the product works successfully.

3.3. Ethics and Safety

One of the biggest issues we need to be careful with is the IEEE Code of Ethics #9 which states to avoid injuring others by false or malicious action. Our design uses data from the Raspberry Pi and computer vision to analyze one's posture. Therefore, we need to accurately make measurements and analyze these values in order to prevent providing misinformation to the user and possibly aggravating back or neck injuries. In order to prevent these injuries, we plan to perform some tests by sitting on the chair with good and bad postures and checking if our Raspberry Pi provides the correct data.

Another important risk we need to address is the risk of electrical circuits and components that are placed on a chair where someone will sit on daily. Because we are designing this project with a camera module that is placed away from the chair instead of sensors that are directly attached, we are able to make this chair a little more safe and comfortable for a user who will be sitting on for multiple hours. However, we still need to be careful with PCBs and other small electrical components that will be placed on the chair. By ensuring that nobody will be endangered by sitting on this electrical chair with enough insulation and protection, we believe that our Posture Sensing Smart Chair is in compliance with the IEEE Code of Ethics #1 [10].

We believe that our design follows the ACM Code of Ethics 1.1 in that it contributes to society and human well-being [8]. Like it states, our design supports improving people's poor posture which is an important problem in society today. As the world is becoming more technology-based, people often tend to spend a lot of time sitting on their chair and work in their offices for several hours. Our design contributes to the society by providing feedback to their poor postures and lowering the risk of back or neck injuries.

We also intend to follow the ACM Code of Ethics 1.6: respect privacy. We will not disclose any information or data gathered through this design or share them with any other parties. The data gathered through the chairs will only be for personal use and knowledge.

3.4. Project Improvements

One improvement this project could have is the use of machine learning to analyze one's posture. Instead of our algorithm involving the lengths and angles of segments, machine learning could further improve the accuracy of determining if one's posture is good or poor.

Another improvement we could have done is a mobile application that can communicate with the chair. Being able to see the analysis of the user's posture on a phone application would be

convenient as the posture data could be accessed easily. Initially, we wanted to build a mobile application, but as none of us had enough knowledge of creating a mobile app, we decided to go with the design of web applications. If we were given more time on this project, we could have built a mobile application that provides detailed and comprehensible visual graphs of how often the user had a bad posture.

Lastly, if we had more time, we could have made a more advanced feedback system that could tell the user which part of the body needs to be corrected when the user has a poor posture. For example, when the user has a forwarded head and rounded shoulders, the web application could highlight the shoulder and neck on a body image with the color red, indicating those body parts need to be adjusted.

4. References

[1] Wang, C. (2020). *Good Posture and its Wealth of Benefits to the Workplace*. [online] Pdfs.semanticscholar.org. Available at: https://pdfs.semanticscholar.org/7755/d5c48864b44937a639fc3c72f4dd3d4d63df.pdf [Accessed 13 Feb. 2020].

[2] Acatoday.org. (2020). *Posture*. [online] Available at: https://acatoday.org/content/posture-power-how-to-correct-your-body-alignment [Accessed 13 Feb. 2020].

[3] Simpson, L., Maharaj, M.M. & Mobbs, R.J. The role of wearables in spinal posture analysis: a systematic review. *BMC Musculoskelet Disord* 20, 55 (2019). https://doi.org/10.1186/s12891-019-2430-6

[4] wikiHow. (2020). *How to Sit*. [online] Available at: https://www.wikihow.com/Sit [Accessed 28 Feb. 2020].

[5] Web.eece.maine.edu. (2020). *Product User's Manual – HCSR04 Ultrasonic Sensor*. [online] Available at: http://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf [Accessed 28 Feb. 2020].

[6] Ww1.microchip.com. (2020). *8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*. [online] Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf [Accessed 28 Feb. 2020].

[7] "IEEE Code of Ethics," *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 28-Feb-2020].

[8] "The Code affirms an obligation of computing professionals to use their skills for the benefit of society.," *Code of Ethics*. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 28-Feb-2020].

[9] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," Cornell University, November 2016. [Online]. Available: https://arxiv.org/abs/1611.08050. [Accessed Apr. 7, 2020].

[10] V. Gupta, “Deep Learning based Human Pose Estimation using OpenCV,” *Learn OpenCV*, 29-May-2018. [Online]. Available: https://www.learnopencv.com/deep-learning-based-human-pose-estimation-using-opencv-cpp-python/. [Accessed: 18-Apr-2020].

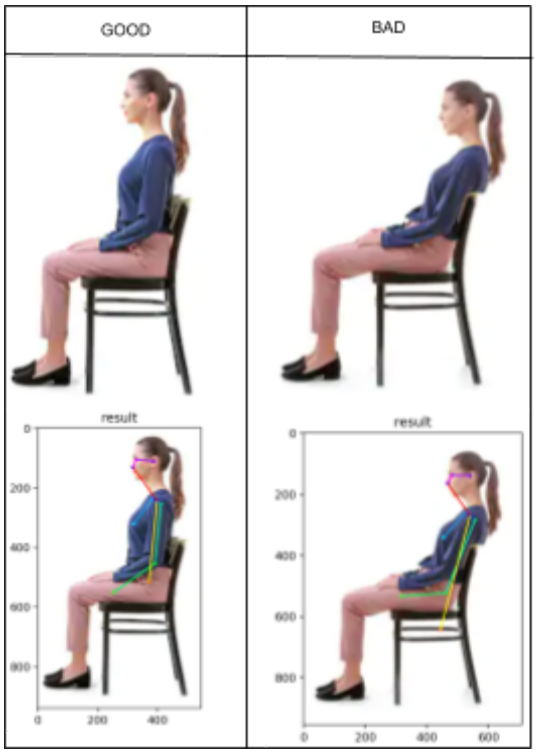
[11] “About OpenCV,” *OpenCV*. [Online]. Available: https://opencv.org/about/. [Accessed: 18-Apr-2020].

[12] D. Osokin, “Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose,” *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, 2019.

[13] L. Sigal, “Human Pose Estimation,” *Computer Vision*, pp. 362–370, 2014.

5. Appendix

5.1 More Results



JOINT ANGLES (GOOD):

{'Nose->Neck': 54.46232220802562,
'Nose->RShoulder': 57.49247557736109,
'Nose->LShoulder': 52.68159461625146,
'Neck->RShoulder': 33.69030752547975,
'Neck->LShoulder': 37.56859202867498,
'Neck->LHip': 95.68965934978917, 'Neck->LEye': 63.10376304459993, 'Neck->LEar': 84.84441568259376, 'RShoulder->RElbow': 125.90972307917767, 'LShoulder->LElbow': 95.71059313749984, 'LShoulder->LHip': 98.71316728807813, 'LShoulder->LEar': 80.08347367909274, 'LElbow->LWrist': 145.09105723888894, 'LHip->LEar': 92.12634162218517}

JOINT ANGLES (BAD):

{'Nose->Neck': 55.30489466676604,
'Nose->RShoulder': 57.60880703396235,
'Nose->LShoulder': 53.207216411918115,
'Neck->RShoulder': 40.23635830627382,
'Neck->LShoulder': 41.42386952500265,
'Neck->LHip': 104.28896423368896, 'Neck->LEye': 65.58926016050796, 'Neck->LEar': 90.0,
'RShoulder->RElbow': 130.94065360260976,
'LShoulder->LElbow': 111.8840277124547,
'LShoulder->LHip': 107.37224684609241,
'LShoulder->LEar': 83.31309097381764,
'LElbow->LWrist': 176.02750405824934, 'LHip->LEar': 100.69528868651919}

Joint Angles(Good):

{'Nose->Neck': 58.41619348116409, 'Nose->RShoulder': 63.25189583707854, 'Nose->LShoulder': 58.029194807943746, 'Neck->RShoulder': 28.56505117707799, 'Neck->LShoulder': 54.46232220802562, 'Neck->LHip': 93.70329602911423, 'Neck->LEye': 67.88083171764177, 'Neck->LEar': 85.42607874009914, 'LShoulder->LElbow': 97.06885410073368, 'LShoulder->LHip': 95.80672890553158, 'LShoulder->LEar': 81.81219799809955, 'LElbow->LWrist': 165.96375653207352, 'LHip->LKnee': 179.93514530054105, 'LHip->LEar': 91.31994566092178, 'LKnee->LAnkle': 75.31803225990636}

Joint Angles(Bad):

{'Nose->Neck': 47.10370994665107, 'Nose->RShoulder': 55.771454199319602, 'Nose->LShoulder': 41.537629943964491, 'Neck->RShoulder': 8.130102354155506, 'Neck->LShoulder': 9.162347345721171, 'Neck->LHip': 70.92845761011145, 'Neck->RHip': 85.27366570705957, 'Neck->LEye': 55.773638187776186, 'Neck->LEar': 69.75337992441179, 'RShoulder->RElbow': 91.08031189329423, 'RShoulder->RHip': 77.79653272361922, 'LShoulder->LElbow': 97.1618337274666, 'LShoulder->LHip': 76.78367560620939, 'LShoulder->LEar': 99.10263483973504, 'LElbow->LWrist': 162.37158063219337, 'LHip->LKnee': 167.8686789333675, 'LHip->LEar': 70.5999651716238, 'LKnee->LAnkle': 54.0065037420251 6}