Sitting Posture Detection and Correction Using the Microsoft Kinect

Michael Battaglia, Maria Janczak, and Brandon Slaght

Department of Computer Science
University of Rochester
Rochester, NY 14627, USA

April 20, 2015

Abstract

In this work, we are determining if a Kinect, a Microsoft motion sensing device, could be used as an effective tool for improving posture of users while sitting at a workstation or any kind of desk, table, or chair. Since the Kinect has the capability of identifying humans in its field of view and using its depth sensing capabilities to accurately track their joints, we believe that it can effectively differentiate between users sitting upright and slouching. We also discuss our implementation of a program that utilizes the Kinect to alert a user when their posture is bad.

1 Motivation

Poor posture is an often overlooked health issue which is independently linked to upper body muscle and bone disorders in heavy computer users. As computer use continues to grow, it is important to address the issue of poor sitting posture, which, when sustained, negatively impacts the physical health as well as the social an emotional health of individuals.

There are several devices and software in existence that try to improve user posture, but none seem practical. We admit that our posture is poor, and many of our colleagues will admit the same, so there is a need for a practical solution. The Kinect is an amazing device that gives a lot of power for little cost. Moreover, if an individual would like to use a Kinect to improve posture, and that individual already owns one (since Kinect's are bought and sold for various applications), there is no additional cost for that individual.

2 Prior Research and Solutions

There has been several attempts to solve this problem before. Some are with an independent device, and others are with software that is meant to use a laptop webcam.

2.1 Lumo Back - An Independent Device

This is the most popular of all posture correcting devices. This device, as of the writing this paper, sells for $79.99. As described on their website:

Lumo Back is worn on a belt around your lower back and focuses on improving lower back posture to prevent and reduce back pain. It vibrates when you slouch to remind you to sit straight and stand tall.

We find that this device can be annoying since it requires users to strap it onto themselves. Additionally, as of the writing of this paper, a used Kinect (with a return policy) can bought from GameStop for $25 dollars.

2.2 Nekoze - A Webcam Application

Nekoze is an application for OSX on the Apple App Store. It uses a device's webcam to identify bad posture and annoy the user until they correct it. This seems perfect; a drop dead simple application that operates on no additional hardware and fixes slouching. However, it functions a bit more loosely than one would first expect. It uses Apple's camera API to detect faces in the scene. Based on the app's description, it can't detect slouching, per say. It only detects when the user's face is too far to the left or the right of the webcam's field of view, or when the face gets too close to the webcam. This measurement can be thrown off by the position of the laptop relative to the user, and at best can be reliably used only when the laptop is positioned properly in front of the user. This system is not as reliable or robust as one we could create using depth information about actual joints. Also, the reviews of the app tell how poorly this software performs

2.3 Real-time Sitting Posture Tracking System, Slivovsky

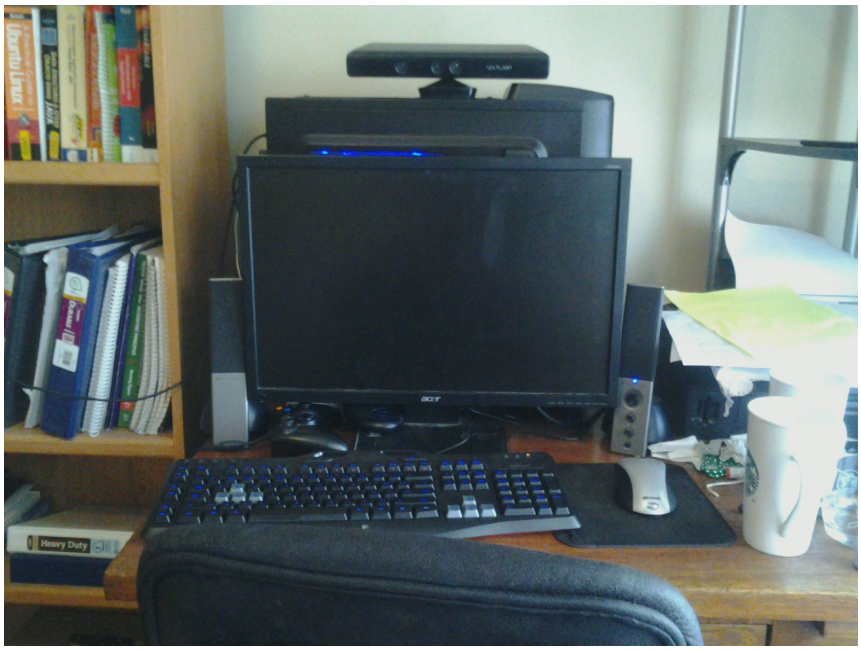
In this thesis paper, Slivovsky discusses the construction of a chair with commercially available pressure sensors to monitor a subject's posture. They generated a filtered pressure map and were able to classify which posture the user was sitting in with 90-99% accuracy depending on the posture using K-means analysis against a constructed dataset. The set of postures they used included sitting normally, leaning in any direction, and slouching, along with variations of those with the right or left leg crossed. Unfortunately there was no hard criteria for defining the posture used in the data of the training set, as the subjects were simply instructed to strike one of the postures in the list, specifics of which were left to the user's interpretation. However, this paper is encouraging because it confirms accurate machine categorization of posture is an attainable goal.

3 Implementation

3.1 Component Set Up

The Kinect, even when facing the user head on, has a limited field of view. According to Microsoft documentation, the Kinect, in its seated, near-range mode, can track joint data of users at between .4 and 3 meters away. In order to get the user in this range, the Kinect needs to be positioned farther back. We found that placing the Kinect from a slightly higher position than the user's head and about a half meter away gave us good tracking. With the Kinect connected to the workstation computer and proper drivers installed, opening the application will immediately begin the calibration process.

Figure 1: An effective placement of the Kinect



3.2 Calibration

We need to calibrate our program to each user for several reasons:

1. Part of our slouching classification uses the height of the head and sternum joints to check that the user is not slouching in their seat or hunching over too much. We need to take an initial measurement of these coordinates so that we can compare these to live measurements, because not all users are going to be positioned at the same height relative to the Kinect.

2. Users can vary greatly in size, including torso broadness. Our program tracks the distance between the shoulder joints because when a user hunches forward, their shoulders tend to come closer together. Using a universal value to test this condition is problematic because of how much the shoulder distance varies between a 6.5 foot football player and a scrawny 5 foot user. Taking a measurement of this means accuracy across different body types.

3. Lateral slouches are slouches too! Again, we need a reference to compare to when checking this condition. We cannot determine this based on the user's lateral position relative to the Kinect because it is difficult to horizontally align the Kinect to center on a user.

Once the Kinect sees a user and initializes skeleton tracking, the program notifies the user to sit up straight and remain still for a few seconds, and records the position of four joints: left shoulder, right shoulder, "center" shoulder (physically located in the middle of the body at the base of the neck) and head. The program will capture three frames of these measurements, each two seconds apart, and then takes the average of all these measurements to determine the size of the user. If at any time during the process the joint data is significantly different than the previous frame, or if the Kinect loses track of the skeleton, the calibration process starts over again. This is to account for any jitteriness of the Kinect's measurements or the user shifting too much, both of which result in joint data which do not mach the user's resting position.

Saving the Y-coordinates of joints during our calibration is important for the program, so the program cannot compensate for the user adjusting the height of their chair (and with it, their body). The user must have their chair set at the height they will use it during their work for the calibration process in order for the program to work properly. If the user changes the height of their chair after calibration, it is best for the user to close and reopen the program to go through the calibration process again. This isn't a huge flaw because the entire process can be done in less than 8 seconds. (Our program was intended to have a GUI, but could not be developed in time because of our time constraint, in which it would have had a button to "re-calibrate".)

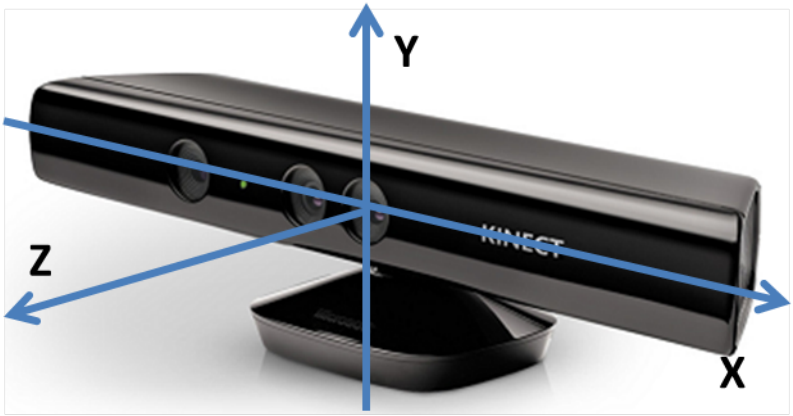
3.3 Normalization of Point Data

The goal of the normalization of the joint data is to eliminate two restrictions. The user does not need to remain in one spot, and the user does not need to face the Kinect head-on. This means the user can move their chair to a different spot in the room, and can also rotate their chair and still have the program effectively determine the user's posture. Eliminating these restrictions is important since when most people work, they will have several items along their desk and will need to shift their chair to change focus on the items along their desk. This also compensates for a desk that wraps around a corner.

The only restriction for the user is both shoulders must be within view of the Kinect, so turning to face 90 degrees away from the Kinect will not work. If the user has a corner desk, where the user will rotate 90 degrees to utilize their entire desk, we recommend placing the Kinect on the corner of their desk, so that way the user is facing the Kinect at a 45 degree angle during most of their work.

The way normalization is implemented is by changing to a coordinate system based on the user's joint positions relative to each other instead of the xyz position relative to the Kinect placement.

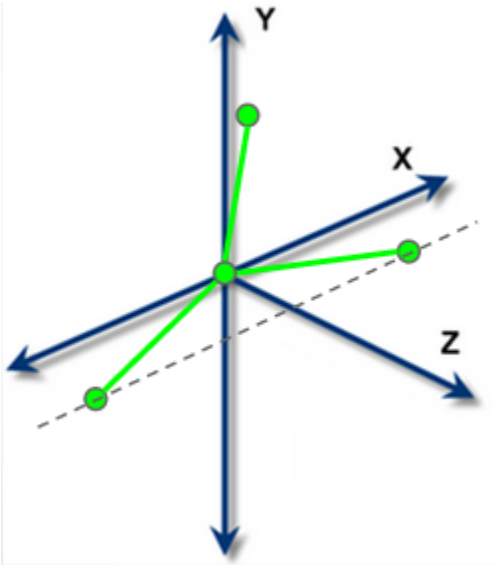
Figure 2: Coordinate System in Microsoft Kinect



Initially, the joint coordinates are simply xyz distances away from the Kinect, where x is in the left and right directions, y is in the up and down directions, and z is in the forward and back directions relative to the Kinect's camera. The goal after normalization is to have the coordinates so that the neck joint is on the y-axis (x and z coordinates are 0) with its position relative to the origin proportional to the difference in height between the current position and a position where the user sits up straight. Furthermore, in the normalized system the coordinates should be such that the line passing through the left and right shoulders is parallel to the xy plane. In this way,

the normalized version places the user always facing in the z direction. Lastly, in the normalized version, the whole coordinate system is scaled so that the length between shoulder and neck is equal to 1.0. After the normalization the coordinates should look like the figure below.

Figure 3: Normalized Skeletal Point Structure



The normalization process happens in 3 steps.

1. The first step is to shift the coordinates so that the x and z coordinates of the neck joint are 0, and the y coordinate is proportional to the change from the upright height. The last part is necessary to help detect the form of slouching where a person slowly begins to slide lower and lower in their chair. The shift is accomplished by subtracting the x and z coordinates of the neck point from the x and z coordinates of every joint point and the y coordinate of the upright calibration point from the y coordinate of every joint point. Such that
   $\Delta x = -Current.Neck.X$
   $\Delta y = -Calibrate.Neck.Y$
   $\Delta z = -Current.Neck.Z$

   and for each joint
   $Joint.X = Joint.X + \Delta x, Joint.Y = Joint.Y + \Delta y$, and $Joint.Z = Joint.Z + \Delta z$

2. The next, and most complicated step is to rotate the coordinates around the y-axis so that the line passing through the shoulders is parallel to the xy-plane. To rotate a point around the y-axis an angle θ, simply follow the transformation
   $x_{new} = z_{old} \sin\theta + x_{old} \cos\theta$
   $y_{new} = y_{old}$
   $z_{new} = z_{old} \cos\theta - x_{old} \sin\theta$

   However, to do this transformation it is necessary to find θ. The goal is to have the line passing through the shoulders be parallel to the xy-plane, and a property that a line parallel to a plane has, is that the dot product of the vector in the direction of the line and the normal vector to the plane is 0. (Since the normal vector to the plane and the vector in the direction of the line would be perpendicular.) The vector in the direction of the line between

the shoulders is $\langle x_{right} - x_{left}, y_{right} - y_{left}, z_{right} - z_{left} \rangle$. The normal vector to the xy-plane is $\langle 0, 0, 1 \rangle$. Thus their dot-product is
$\langle x_{right} - x_{left}, y_{right} - y_{left}, z_{right} - z_{left} \rangle \cdot \langle 0, 0, 1 \rangle = z_{right} - z_{left}$.

Setting the dot-product to 0 produces
$z_{right} - z_{left} = 0$

which is equivalent to
$z_{right} = z_{left}$.

This result should not be surprising, since all it says is that both shoulders will be the same distance from xy-plane if the line passing through them will be parallel to the xy-plane. Substituting into this equation yields
$z_{right} \cos\theta - x_{right} \sin\theta = z_{left} \cos\theta - x_{left} \sin\theta$. Which can be solved for θ.
$z_{right} \cos\theta - z_{left} \cos\theta = x_{right} \sin\theta - x_{left} \sin\theta$
$(z_{right} - z_{left}) \cos\theta = (x_{right} - x_{left}) \sin\theta$
$\frac{\sin\theta}{\cos\theta} = \frac{(z_{right} - z_{left})}{(x_{right} - x_{left})}$
$\tan\theta = \frac{(z_{right} - z_{left})}{(x_{right} - x_{left})}$
$\theta = \tan^{-1} \frac{(z_{right} - z_{left})}{(x_{right} - x_{left})}$

θ is thus calculated and then plugged into the original transformation equations (at the beginning of the section). The transformation is then performed on each joint.

3. The last thing to do is to scale the data so that in the machine learning approach described below, the same training data can be used with people of various sizes. The way this is done is by scaling the data so that the distance between the shoulder and neck is 1.0. Since the distances between shoulder and neck can vary on either side, the average of the two is used. The inverse of this is taken to be the scaling factor, by which all points are multiplied.
$ScalingFactor = \frac{2}{\sqrt{(x_{right} - x_{center})^2 + (y_{right} - y_{center})^2 + (z_{right} - z_{center})^2} + \sqrt{(x_{left} - x_{center})^2 + (y_{left} - y_{center})^2 + (z_{left} - z_{center})^2}}$
Each joint point is multiplied by the scaling factor above.

3.4 Monitoring Process - Straightforward Approach

After calibration, the monitoring process is quite simple. As the new skeleton frames come in from the Kinect, we normalize them and compare them to the normalized calibrated skeleton. This approach is straightforward and works by simply finding the Euclidean distances between the corresponding joints in the calibrated and the new skeletons. If any of these distances are past a certain threshold, the posture is said to be slouching, otherwise the posture is said to be upright.

During monitoring, we look at a skeleton frame every second (as opposed to every two seconds during calibration). This is to give more immediate feedback to the user. If a new skeleton frame comes in and is significantly different from the calibrated skeleton, we flag it and wait for the next frame. If the next frame is just as different as the previous, or worse, we notify the user is slouching. We check for slouching for two consistent frames because we want to make sure the user is actually slouching and not that the Kinect read a frame poorly.

We determine if a skeleton frame is slouching by measuring the distance between each joint from their matching joint in the calibrated skeleton. If the distance is greater than a threshold, we flag it as having poor posture. If next incoming frame is significantly higher that the calibrated

frame, we assume the user is standing. The frame is then ignored. This way the user will not get false notifications if they adjust their chair, or get up to leave for a break. We think it is safe to assume this, since the user cannot slouch and end up in a higher position that their proper sitting position.

After the user has been notified of slouching, the user will be notified of sitting correctly again after the next incoming frame matches the calibrated frame closely.

3.5 Application of Machine Learning

The straightforward approach described above only takes points and compares them to the calibration. It thus solely relies on the user knowing what the correct position is. Furthermore, were there to be more than one non-slouching posture, the algorithm would not be able to handle this. The idea behind using machine learning is that by using machine learning it is not necessary to calculate the angles directly, and multiple good and bad positions can easily be accounted for. Since the problem required each position to be classified as either slouching or not slouching, a classification algorithm seemed like a good way to go, and kNN was chosen. The kNN algorithm works by taking in a training data set which is a set of points, each of which is labeled as either slouching or not slouching. The algorithm also takes in a point to classify. The distance between the point to be classified and each of the training points is measured. For our approach we used the sum of the Euclidean distances between the normalized joint locations of the training points and the the normalized joint locations of the posture whose classification is to be determined. The classification of the k training points nearest the point whose classification is to be determined, is taken into account, and if most of the k points are classified as slouching the point to be classified is also classified as slouching, and if instead most of the k points are classified as upright, the point to be classified is also classified as upright. The k we found which seemed to work well is 3. A higher k would probably work well if we had a larger training set.

4 Issues

4.1 Kinect/User Positioning

As previously stated, the Kinect needs a good view of the user for accurate tracking. It can handle the user sitting at various angles thanks to the normalization process, but is unable to track users sitting perpendicular to it as it loses sight of the far shoulder. It also tends to be less accurate the larger the angle between it and the user is. This is not too big of an issue as long as the user can position the Kinect in front of them.

4.2 Bad Tracking

The Kinect often fails to properly track a joint, especially when there is not enough color contrast or depth distance between the user and the background of the scene. In these cases, it often continues tracking the joints but the coordinate data becomes very skewed and incorrect. Under these circumstances the posture classification is understandably thrown off and will often misidentify slouching. In order to combat this slightly, our program requires three frames in a row of poor posture before complaining to the user. This tends to filter out a lot of bad joint tracks because they tend to last only momentarily before snapping back to their correct coordinates. If the Kinect loses track of the entire skeleton, it is assumed that the user has moved from their seat and posture monitoring ceases until the user returns and their skeleton is properly tracked again.

We also noticed that the Kinect had difficulty tracking the neck and head joints when a user was hunched forward at an extreme angle, or were in other positions such as resting their head on their hands. This issue may be alleviated by v2 of the Kinect which includes more accurate tracking with more joints, but in our implementation and testing with Kinect v1 it is an issue, albeit not a major one, because it leaves us with an ultimatum: either ignore these bad tracks and miss classification when a user is in this hunched over position or classify bad tracks as slouching and complain even if the user is sitting straight up and it is simply the fault of the Kinect. However, as discussed later in our results section, it turns out our classification is accurate enough to properly classify forward slouching.

5 Testing

The Kinect SDK allows for the recording and playback of sensor data to the host system, which means we can record a series of movements, including sitting still for calibration, leaning forward and backward, and slouching left and right, and play it back verbatim to our application. This is a critical tool for adjusting our program's measurement constraints as it enables us to see exactly what effect a change made on posture classification.

6 Results

Our tests reported slouching in every direction when it was appropriate. It also reported corrections to posture after sitting back up. We do not have a measure for our accuracy, but in our testing, slouching is reported as soon as the user slouches slightly. It does excitingly well at recognizing poor posture. Due to our normalization, it is not set off when the user moves their chair in any direction. It does tend to have a bit bigger of a tolerance for slouching than we would have liked, but this can be changed by feeding different data to the K-NN algorithm. Also, despite the Kinect's apparent difficulty tracking a skeleton during forward slouching, our program classifies forward slouching properly thanks to the robustness of the K-NN algorithm.

7 Concluding Statements

The results of our experimentation proved that the Kinect can be a very powerful and accurate tool for monitoring posture at a desk workstation, especially when a K-NN classification algorithm is applied to a generated set of data. The biggest drawback is positioning the sensor, which is trivial in many cases. Future researchers can build on our progress and create a user friendly interface, use the Kinect v2 for more accurate tracking and tracking multiple users, and using facial recognition to track user profiles and progress over time. On the side of medical research, doctors interested in this coukd give the K-NN input a better set of points by providing data on what would be medically detrimental, as opposed to just points tht we think are slouching. Eventually, once computer vision advances to the point of doing skeleton tracking via the webcam of a laptop, we can literally apply our algorithm on the webcam data, removing the need for extra hardware and adding a new level of convenience.

References

[1] "Improve Your Posture With Nekoze". In: (). URL: http://www.makeuseof.com/tag/improve-your-posture-with-nekoze/.

[2] "Lumo Website". In: (). URL: http://www.lumobodytech.com/.

[3] "Position yourself to stay well". In: *Consumer Reports on Health* 18.2 (2006), pp. 8–9.

[4] "Skeleton Basics - WPF". In: *Kinect for Windows SDK 1.8* (2015). URL: https://msdn.microsoft.com/en-us/library/hh855381.aspx.

[5] Lynne Anne Slivovsky. "A real-time sitting posture tracking system". In: (2001), pp. 31–39, 55–57. URL: http://search.proquest.com/docview/304724369/abstract/9883CE3C2A7746F9PQ/1?accountid=13567.

[6] Karin Horgan Sullivan. "Perfect Posture". In: *Vegitarian Times* 257 (1999), p. 64.

cs