11/30/2021

Squat Analysis Program

RJ Justesen, BMEN 3325

Contents

Introduction

The purpose of this program is to use a laptop built in webcam or a webcam attached to a computer to take two snapshots at the user's discretion. It is recommended that one snapshot be of a person in a standing position and the second be of them in a deep squat. The program uses pose detection machine learning to create key points on the camera window of a person. As the person moves around the camera it will take away or add the key points depending on if they are in frame or not. As the snapshots are taken, the positions of the key points relative to the knees is used to calculate the angle of the knee in the standing position and the squatted position. If a person has muscular imbalances in their leg the angle of their knee in a squat will appear to be greater than ninety degrees. This knee angle can help physical therapists and personal trainers more easily diagnose muscular imbalances and create strength training programs to address those imbalances and prevent injury. Since this is a computer program it also opens the opportunity for long distance treatment for clients who would like to strengthen their bodies and prevent injury.

Section 1: Methods

Section 1.1: Accessing Machine Learning Database

When using machine learning there is typically a database that can be accessed for the tasks one would want to accomplish. For this particular program tensor flow has a pose detection database called PoseNet. This is a database of thousands of pictures of different people in many different poses and it detects these poses by specifying key points. These points are the nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle and right ankle. MATLAB has a toolbox called Human Pose Estimation with Deep Learning that was downloaded and used for this program. This is what made the pose detection possible.

Section 1.2: Initializing Webcam to Show Key Points.

To set up the webcam to display the key point from PoseNet a few things were initialized. PoseNet can only be used with 8-bit integer types and must have an object declared as the player. The player is initialized as the deployable video player, and it is what causes a window to pop up showing what the webcam is showing. Next an image array was declared to fit the size necessary for PoseNet. This image array is what will be used set the size of snapshot from the camera. The image array was set to receive input from the webcam. This image input was then cropped and resized so that the key points could be visualized. Next, using PoseNet the heatmaps of the image in the webcam were predicted using the machine learning databasse from PoseNet, the heatmap was then used to show the key points in the webcam. When the program runs you can see 17 red dots on each of the key points aforementioned each connected with different colored lines between each one. The key points were stored in a 17x3 array where the first two columns were the X and Y coordinates in the image and the third column being a Boolean flag where if the person in frame was only showing part of their body the key points in frame would have a 1 in the third column and the key points out of frame would have a 0 in the third column.

Section 1.3: Calculating the Angle of the Knees

Once the key points were visualized and stored in an array finding the angles was quite simple. Since the 12th to the 17th key points are the points of interest, vector calculus was used to find the angle at the knee. First the vector between the knee and hip and the knee and ankle are calculated. Then the inverse cosine of the dot products of the two vectors over the sum of the magnitudes was the equation used to find the angle. These angles were stored in an array as the angle of the knee standing was calculated along with the angles of the knees when squatting. After the angle of each knee was calculated there was a conditional statement implemented where if the angle of the right knee was greater than 90 degrees a warning statement popped up that stated that there may be muscular imbalances, the same thing was done for the left knee.

Section 2: Results

The results of this program were reached successfully. The program took two screenshots when the camera window was exited. Two sample screen shots are shown below along with the calculated angle of both knees.

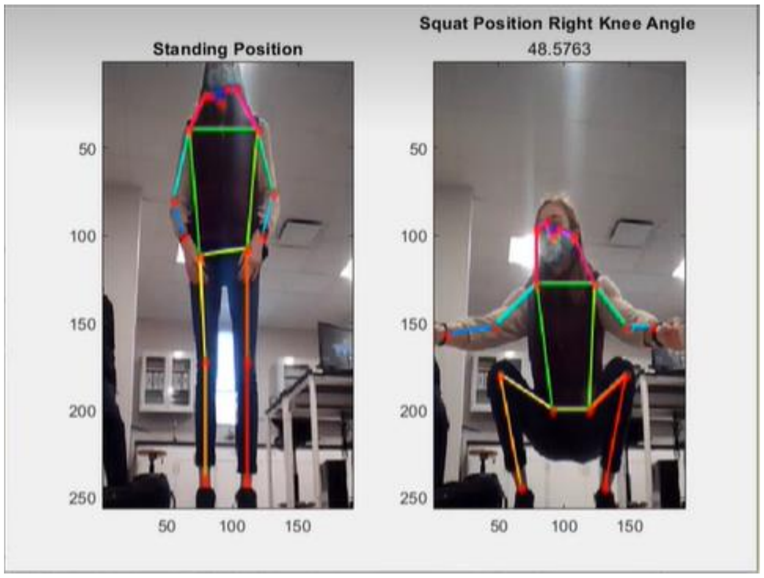
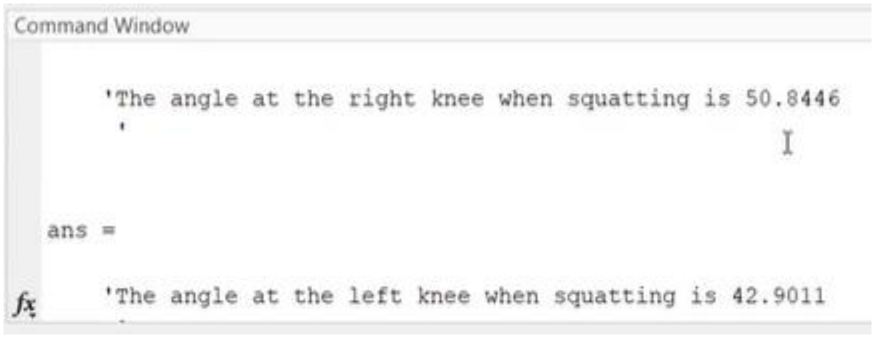


Figure 1: Sample Screenshot



```
Command Window

    'The angle at the right knee when squatting is 50.8446
     '


ans =

    'The angle at the left knee when squatting is 42.9011
```

Figure 2: Angle of both knees from figure 1.

The next screenshot shows a knee shift in a squat position along with the warning and calculated knee angle.

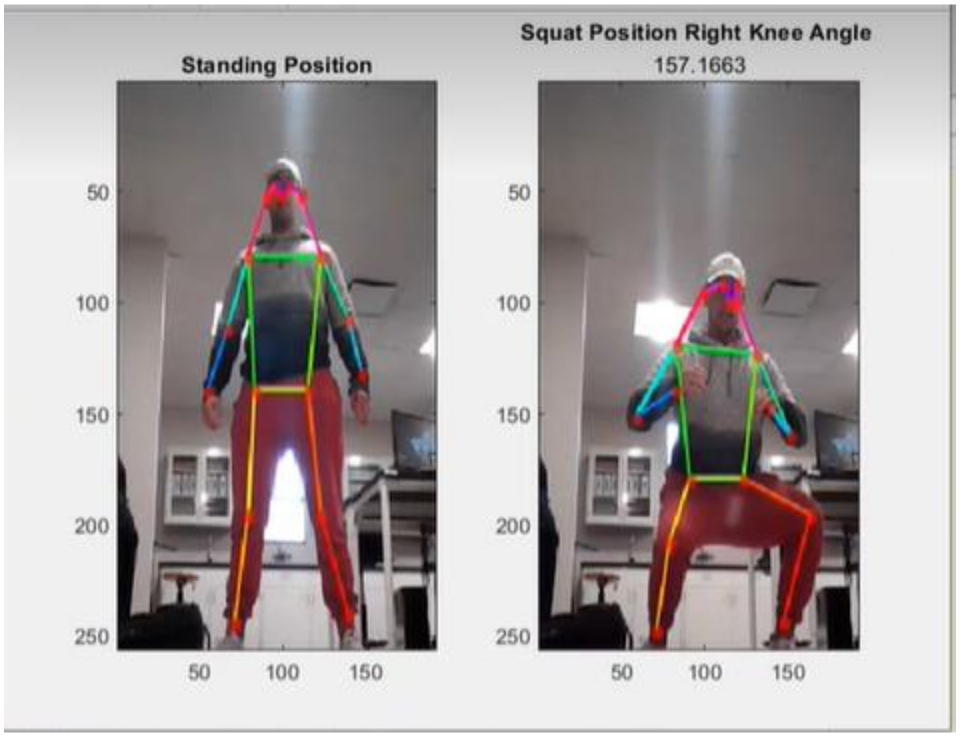
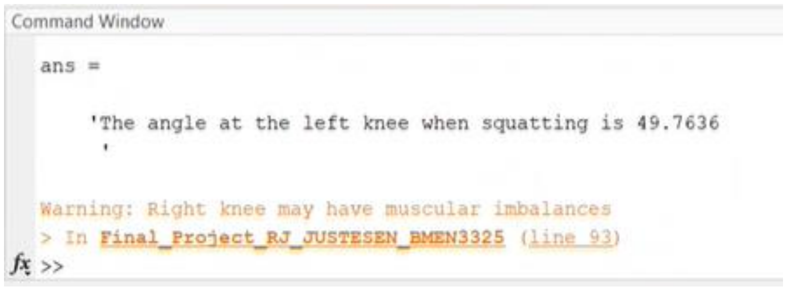


Figure 3: Screenshot with a person who has a right knee shift



```
Command Window

  ans =

      'The angle at the left knee when squatting is 49.7636
       '

  Warning: Right knee may have muscular imbalances
  > In Final_Project_RJ_JUSTESEN_BMEN3325 (line 93)
fx >>
```

Figure 4: Warning issued when knee angle is greater than 90 degrees

Overall, the results were very successful, and the program performed what it was intended to do. Future iterations would involve implementing this into a graphical user interface and having the angle of each knee display on the webcam video.

Conclusion

To bring to an end, the program was successful in its purpose. The video supplied with this paper will show the program being implemented and will show the key points being displayed, the two screenshots of the standing position and the squat position and will have an example of muscular imbalances in the knees. The purpose was fulfilled and with some polishing this program could be very helpful for personal trainers and physical therapists alike.

Appendix I

Annotated code

```
%% Program to calculate angle at each knee in squat
position
%This program will have a person step into the webcam and
take one picture
%standing and one in a sqaut positon
%this will allow a trained professional to see the angle at
the knee and
%show let them know if the person has muscular inbalances
based on the
%angle of the knee in the squatted position. the webcam
will pop up when
%the run button is hit, by exiting out of the popup window
the first
%screenshot will be taken, then the webcam will popup again
and this
%indicates that the person should squat and by exiting the
popup the second
%screenshot will be taken.


clear all;
detector = posenet.PoseEstimator; %acesses machine learning
database from
%tensorflow trained for pose detection.

player = vision.DeployableVideoPlayer; %initialize player
I = zeros(256,192,3,'uint8'); %initialize image array as 8
bit integers,
%pose detection has size constraints that can only use 8
bit integers.
arr = {uint8(size(I)),uint8(size(I))}; %initialize cell
array for storing
%captures images.
player(I); %change resolution of webcam to fit pose
detection database
```

```
angleArr(2,2) = 0;

for i =1:2
    flag = 1; %flag used to exit while loop
    cam = webcam; %initializes variable for webcam snapshot
while flag == 1
   % for i = 1:2
    % Read an image from web camera
    I = snapshot(cam);

    % Crop the image fitting the network input size of
256x192
    Iinresize = imresize(I,[256 nan]);
    Itmp = Iinresize(:,(size(Iinresize,2)-
192)/2:(size(Iinresize,2)-192)/2+192-1,:);
    Icrop = Itmp(1:256,1:192,1:3);

    % Predict pose estimation
    heatmaps = detector.predict(Icrop);
    keypoints = detector.heatmaps2Keypoints(heatmaps);

    % Visualize key points
    Iout = detector.visualizeKeyPoints(Icrop,keypoints);
    player(Iout);
    %if keypoints(12:17,3) == 0
     %   error('Lower half of client is not in frame,
please ensure the client is in position and start again')
    %end
    %initialize x,y coordinates for the joints (lol joints
sounds like points) of interest
    rightAnkle =  keypoints(17,1:2);
    leftAnkle = keypoints(16,1:2);
    rightHip  = keypoints(13,1:2);
    leftHip = keypoints(12,1:2);
    rightKnee = keypoints(15,1:2);
    leftKnee = keypoints(14,1:2);

    %initialize distance vectors between the hips and knees
and ankles and
    %knees
    x10 = rightHip(1) - rightKnee(1);
    y10 = rightHip(2) - rightKnee(2);
    x20 = rightAnkle(1) - rightKnee(1);
    y20 = rightAnkle(2) - rightKnee(2);
    x11 = leftKnee(1) - leftHip(1);
    y11 = abs(leftKnee(2) - leftHip(2));
    x22 = leftKnee(1) - leftAnkle(1);
    y22 = abs(leftKnee(2) - leftAnkle(2));

    %calculate angle of right knee
    angle1 = atan2(abs(x10*y20-x20*y10),x10*y10+x20*y20)
*180/pi;

    %calculate angle of left knee
    angle2 = (atan2(abs(x11*y22-x22*y11),x11*y11+x22*y22)
*180/pi);

    %store angles into angle array.
    angleArr(i,1) = angle1;
    angleArr(i,2) = angle2;

    %store image into cell array.
    arr{1,i} = Iout;

    if ~isOpen(player)
       flag = 2; %flag to run through for loop a second
time
      release(player); %release player to allow webcam to
be accesed again
    end

end
%clear webcam for second iteration.
clear cam
end

sprintf('The angle at the right knee when squatting is %g
\n', angleArr(2,1))
sprintf('The angle at the left knee when squatting is %g
\n', angleArr(2,2))
if angleArr(2,1) > 90
    warning('Right knee may have muscular imbalances');
end

if angleArr(2,2) > 90
    warning('Left knee may have muscular imbalances');

end
```

```
%convert image cell array to matrix for display
A = cell2mat(arr(1,1));
B = cell2mat(arr(1,2));
str1 = sprintf('Angle of right knee: %g degrees',
angleArr(1,1));
str2 = sprintf('Angle of left knee: %g degrees',
angleArr(1,2));
%display standing image
subplot(1,2,1)
%display standing image
image(A)
title('Standing Position');
subplot(1,2,2)
%display squatting image
image(B)
title('Squat Position Right Knee Angle ',
num2str(angleArr(1,1)));
```

Appendix II: Famous Computer Scientist Alan Turing

Alan Turing is considered the father of theoretical computer science. He was crucial in World War 2 were he built an electromechanical computing machine that was able to decipher encrypted German messages. It would decipher the settings of the enigma machine that the German's were using that day so that the Allied intelligence could decipher any incoming messages. This was important in turning the tide of the war in the favor of the Allies. Turing developed the Bombe which was the machine used to decipher the German's Enigma machine that sent coded messages across public channels. The Bombe would perform a chain of logical deductions for each possible position of each rotor of the Enigma machine. This was the first of automating daily tasks as before the Bombe was made ciphers would decode messages by hand and often that meant it was too late to stop the Germans form attacking. This was the start of modern-day computing and Alan Turing's contributions paved the path for the modern computers and programming potential we have today.

Figure 5: Alan Turing