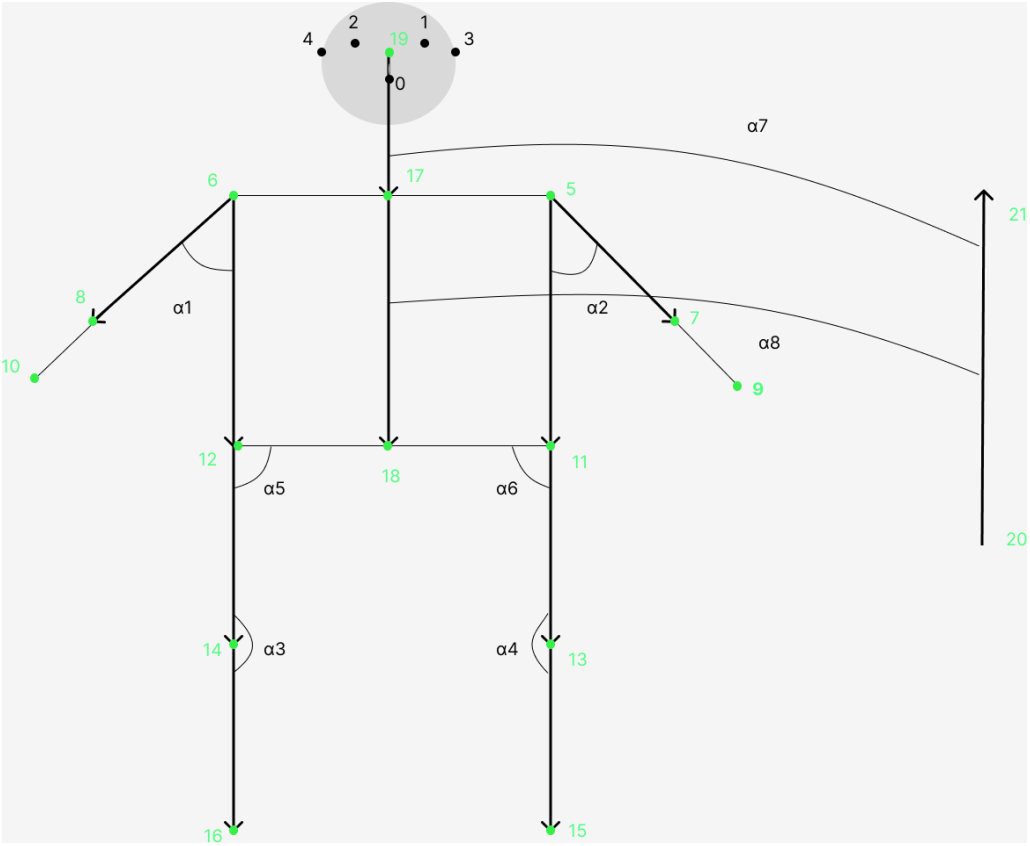
Fall Detection

Automated fall detection is for assisting elderly in daily life. It captures fall activity and notifies about that.

How it works

According to the "Privacy Preserving Automatic Fall Detection for Elderly Using RGBD Cameras" paper by Chenyang Zhang, Yingli Tian, and Elizabeth Capezuti, ‘fall’ causes much more deformation of the joint structures than other daily activities. To control the joint structures, we obtain the key points of a human skeleton. The model we used to get the key points is OpenPifPaf 0.13.4. OpenPifPaf returns seventeen key points. Having obtained the key points, we produce vectors defined in advance, which are most likely to be dramatically changed during a fall. OpenPifPaf returns negative coordinates for the key points that were not detected. We take nans instead of negative coordinates of the key points and then handle them.

In addition to the provided key points, we add extra ones that are probably important for fall detection. It includes, the midpoint of the shoulders, the midpoint of the hips, two random points with equal x coordinate that will produce a vertical vector and mean of the head points.



We calculate eight angles based on the specified pairs of the chosen vectors. Those eight angles are calculated by the norms of the chosen vectors and dot product of the vector pairs.

$$Angle = \arccos\left(\frac{\langle \vec{u}, \vec{v} \rangle}{\|\vec{u}\| * \|\vec{v}\|}\right)$$

Based on the angles several features are being calculated:

1) costMean (The mean of eight angles)

$$\frac{1}{n}\sum_{i=0}^{n} \alpha_i^{<t>}$$

2) divisonCost (Each angle is divided on by its previous frame's same angle, and then all resulted angles are summed up)

$$\sum_{i=0}^{n} \frac{\alpha_i^{<t+1>}}{\alpha_i^{<t>} + 10^{-6}}$$

3) differenceMean (The absolute value of the difference between each angle of previous and current angles are taken, then mean of eight differences are calculated and multiplied by the current fps. It represents a number close to the derivative)

$$\frac{1}{n}\sum_{i=0}^{n} |\alpha_i^{<t>} - \alpha_i^{<t+1>}|$$

4) differenceSum (The absolute value of the difference between each angle of previous and current angles are taken, then sum of eight differences are calculated)

$$\sum_{i=0}^{n} |\alpha_i^{<t>} - \alpha_i^{<t+1>}|$$

5) meanDifference (The means of eight angles of previous and current frames are calculated and absolute value of difference of those means is taken)

$$\frac{1}{n}\sum_{i=0}^{n} \mid \alpha_i^{<t>} \mid - \frac{1}{n}\sum_{i=0}^{n} \mid \alpha_i^{<t+1>} \mid$$

Where <t> superscript is the previous frame, <t+1> superscript is the current frame. Alpha is the angle calculated between two vectors. We are using 8 pairs of vectors, so there are 8 angles calculated between two frames, so n=8. In the case, when more than 6 angles are not found for some frame, the frame is skipped. Moreover, if the cost is NaN, we compute nothing and assign the current cost to the previous one. If the missing one is the first cost, we take a default number.

We use weights to give respective importance to each angle calculated, since not all angles are equally important for fall prediction. Weights can be optimized using labeled train data that will produce the weights that provide right importance to the angles.

Since falling is a sequential activity and it may last a number of frames. For that reason, we keep the costs for the last 6 frames (duration of 1 second) and multiply with different weights and calculate the mean.

$$[c_0, c_1, c_2, c_3, c_4, c_5] \rightarrow C_0$$

$$C_0 = \frac{c_0 * w_0 + c_1 * w_1 + c_2 * w_2 + c_3 * w_3 + c_4 * w_4 + c_5 * w_5}{6}$$

After getting one number from each cache as the main cost of the current frame, threshold is applied to the costs, and if they extend the threshold then a notification shows fall activity. Moreover, we remove too big and too small costs by bounding the result in [clip_from, clip_to] interval using the numpy clip function, which is the acceptable interval where the cost value can be.

Optimal threshold and bounding interval can be found using grid search on labeled training data using grid search.