Quaternion based Orientation Extraction from a Human Skeletal Pose Estimate using 2-D Cameras

Sriram Radhakrishna
Department of Computer Science and Enginering
P.E.S University, Banashankari
Bangalore, India
sriram.radhakrishna42@gmail.com

Adithya Balasubramanyam
Department of Computer Science and Engineering
P.E.S University, Banashankari
Bangalore, India
adithyab@pes.edu

***Abstract*—In this paper, we present a novel implementation of an algorithm to extract a quaternion from a two dimensional camera frame for estimating a contained human skeletal pose. The problem of pose estimation is usually tackled through the usage of stereo cameras for obtaining depth and euclidean distance for measurement of points in 3D space. However, this task comes with a high computational cost with a relatively high detection latency as well, requiring the use of expensive hardware components. By making use of MediaPipe, a framework for building perception pipelines for human pose estimation, the proposed algorithm extracts a quaternion from a 2-D frame capturing an image of a human object at a sub-fifty millisecond latency while also being capable of deployment at edges with a single camera frame and a generally low computational resource availability.**

***Index Terms*—quaternions, 2-D camera, pose estimation, MediaPipe, low-power, low-latency, embedded computer vision.**

I. Introduction

Essential scene-analysis tasks such as pedestrian detection and localization generally involve the generation of a quaternion to estimate the orientation of a target object. At times, these techniques involve the usage of stereo cameras to match feature points [1] or other such methods involving expensive hardware components with a relatively large latency and computational resource utilization.

Intuitively speaking, reducing the number of calculations and general computational work involved in the deployed algorithm would be the way forward to tackle the cost and latency issues. In order to do this, the hardware reliant inputs being accepted by existing pose estimation models must be taken note of. For this particular use case of human pose estimation, the depth of the points located within a targeted pose object and the simultaneous threading of two image streams at minimum stands out. In order to address these issues, a system containing a single 2-D camera with a human subject in frame on the hardware side to minimize costs was opted for. Additionally, a specialized deep learning based solution on the software side was implemented to account for the loss of depth perception as well as maintain latency on edge devices with a lower resource availability.

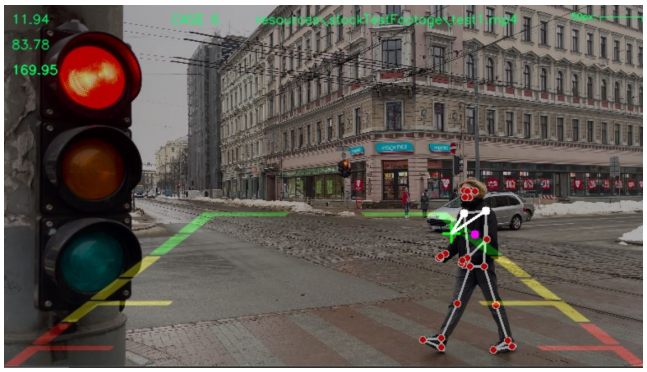

Fig. 1. A demonstration of the novel implementation for the use case of calculating the angle of orientation of a pedestrian object from a generated quaternion.

II. Related work

Pavvlo et al. in their efforts to develop a quaternion based recurrent model for human motion, made the observation that human motion is a stochastic sequential process with a high level of intrinsic uncertainty [2]. While deep learning based approaches have been very successful in predicting the pose of a human skeleton in both short term [3] [4] and long term [5] applications, they require multi-threaded computations and are generally costlier to implement on hardware. This is especially true for use cases in fields like mobile robotics and autonomous vehicle navigation where the cost factor is usually compromised on in favour of larger core counts on the main edge processing unit.

Although this trade-off has recently found more justifications with the dropping retail costs of Nvidia GPU based edge computing units [6], the gap is best bridged by adapting CPU based systems to more efficiently handle the calculations necessary. It was accurately noted by Eberly, D. that rotation matrices and quaternions take over 61 percent fewer calculations to obtain than angle-axis representations of vector rotating operations [7]. For context, table 1 illustrates the

The authors are with the Center for Internet of Things, Department of Computer Science and Engineering, PESU Ring Road Campus (Bangalore, India).
Our code repository can be found at : https://github.com/SR42-dev/human-pose-quaternion-extraction

number of operations to be carried out per vector rotation operation.

III. Concept Theory and Implementation Methodology

The following section introduces the thought processes, assumptions and applied quaternion mathematics behind the novel implementation.

A. Overview

The challenges faced in this approach lie in the implementation of algorithmic solutions to finding the inputs mentioned in the introductory paragraph. First, the problem of calculating the orientation of the human pose object came with the issue of localization of body landmarks with acceptable standards of latency. Following this, was the conception of a mathematical function that extracted the angle of orientation from the pose data of these points.

To solve the problem of estimating the orientation, we employed MediaPipe, a framework by Alphabet Inc. that provides deployed solutions from deep learning models trained on data for human pose object detection. This package is aimed at edge devices with a low resource utilization. [8]. While the framework achieves admirable detection latencies, an inherent flaw in the system when porting to use cases requiring quaternion generation like robotic navigation [9] and such due to its basis in 3-D cartesian space. An added advantage provided by the novel implementation is that portability of the model is maintained even when implemented on dynamic frames of reference, e.g.; on a mobile robot in motion.

B. Quaternion Transformations

To provide some context here, the two points chosen to evaluate the pose of our target human are the shoulder points as their relative positions are an accurate descriptor of the orientation of said target [10].

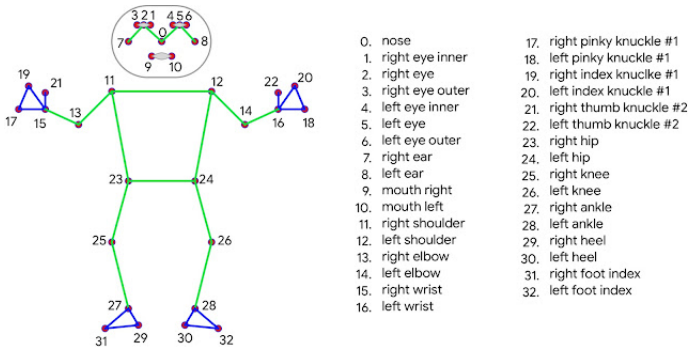

Fig. 2. BlazePose 33 keypoint topology for a human skeleton as COCO (colored with green) superset [11]

To narrow down this choice, all the provided options in the MediaPipe pose solutions documentation were evaluated [11] for appropriate pairs of points whose projections are sure to change location on a 2-D frame when the target's roll, pitch or yaw changes. Additionally, it was also arbitrarily determined that symmetry must be maintained for these points across the mid-line of the body as this seemed like the intuitive guideline to maintain given the potential use cases of our implementation.

After obtaining the coordinates for these points from the cartesian space estimated by the framework, a rotation matrix for the skeleton was generated by taking the coordinates of the two shoulder points mentioned before. Let us denote the points for the left and right shoulder points on the frame as (xl, yl, zl) and (xr, yr, zr) respectively. Therefore, the Z-axis vector of the rotation matrix can be calculated using the difference between the two points as

$$\vec{z} = \begin{bmatrix} xl & yl & zl \end{bmatrix} - \begin{bmatrix} xr & yr & zr \end{bmatrix} \tag{1}$$

$$\widehat{z} = \begin{cases} \frac{\vec{z}}{|\vec{z}|}; \, \Delta z \neq 0 \\ \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}; \, \Delta z = 0 \end{cases} \tag{2}$$

Similarly, the X and Y axis vectors are generated as such -

$$\widehat{x} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \times \widehat{z}; \, iff \, \widehat{x} \neq 0 \, else \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{3}$$

$$\widehat{y} = \widehat{z} \times \widehat{x} \tag{4}$$

... which allows us to derive our rotation matrix [12] as

$$R_{3,3} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} \widehat{x} & \widehat{y} & \widehat{z} \end{bmatrix} \tag{5}$$

Subsequently, the quaternion $\mathbf{Q}$ was obtained in a standard form from this rotation matrix [12] as

$$\mathbf{Q} = a + bi + cj + dk \tag{6}$$

... where the coefficient values can be mapped according to equations 7 through 10.

$$a = \frac{1}{2}\sqrt{|1 + r_{00} + r_{11} + r_{22}|} \tag{7}$$

$$b = \frac{r_{21} - r_{12}}{4a} \tag{8}$$

$$c = \frac{r_{02} - r_{20}}{4a} \tag{9}$$

$$d = \frac{r_{10} - r_{01}}{4a} \tag{10}$$

In order to extract a usable real world statistic from these equations and to test the practicality of the novel implementation, the direction being faced by the human pose

TABLE I
PERFORMANCE COMPARISON OF VECTOR ROTATING OPERATIONS

Method	Multiplications	Additions/Subtractions	Trigonometric operations	Total operations
Rotation matrix	9	6	0	15
Quaternions without intermediate matrix	15	15	0	30
Quaternions with intermediate matrix	21	18	0	39
Angle/axis without intermediate matrix	18	13	2	30 + 3
Angle/axis with intermediate matrix	21	16	2	37 + 2

object in the camera frame was extracted by applying certain transformations to the obtained quaternions (hereby referred to as the angle of orientation of the human pose object). Let it be noted at this point that the quaternion values returned do in fact contain some noise due to implicit measurement uncertainities in xl, yl, zl, xr, yr and zr (refer to equation 1) as illustrated in the section IV, sub-section D 'Kalman Filter to eliminate quaternion noise'. These uncertainties were compensated for with the implementation of a 1-D Kalman filter for the angle of orientation of the human pose object as it was emperically the most accurate approach found post-testing.

This was done by extracting the angle of rotation from the angle-axis form of the quaternion, as the axis vector itself was implicitly made to be oriented upwards from the head of the human object in the frame when the rotation matrix was extracted from the model.

Assuming our quaternion to be of the form ...

$$\mathbf{Q} = cos\theta + sin\theta(xi + yj + zk) \tag{11}$$

Theta was extracted as ...

$$\theta = \arccos(\frac{r}{\sqrt{r^2 + x^2 + y^2 + z^2}}) \tag{12}$$

Theta was then transformed to obtain the angle of orientation as realAngle in our frame of reference such that the axis of reference was the horizontal across the camera feed window.

$$realAngle = \frac{\theta \cdot \frac{180^2}{\pi}}{45} - 180 \tag{13}$$

The transformation applied here takes care of the conversion from radians to degrees as well as the fact that an x degree rotation in the quaternion translates to a rotation of 2x in a real life scenario.

C. Algorithm

Prior to covering the main algorithm, do note that the *getRotationMatrix* function accepts 6 floating point values denoting xl, yl, zl, xr, yr and zr respectively (Refer to equation 1) and returns the rotation matrix formulated in equation 5 by iterating through the calculations in equations 2 through 4. The *calculateQuaternion* function accepts a rotation matrix R as given in equation 5, calculates **Q** as in equation 6 by calculating its components according to equations 7 through 10 and returns the required quaternion.

The aggregated algorithmic flow for quaternion generation from the 2-D image of the skeletal pose was framed to proceed as follows -

Input: Video feed API v
Output: **Q**
Initialisation :
v = VideoCapture Class [13]
LOOP Process :
while True **do**
  img = v.frame // assigning an image requested from the API to the variable img at the time of request
  pose = poseDetector(img) // detects the existence of a skeletal pose in the frame
  // getting the requested landmarks from the MediaPipe framework
  if pose != None **then**
    xl, yl, zl = pose.getLandmarks('Left Shoulder')
    xr, yr, zr = pose.getLandmarks('Right Shoulder')
    R = getRotationMatrix(xl, yl, zl, xr, yr, zr)
    Q = calculateQuaternion(R)
    return **Q**
  end if
end while

IV. RESULTS AND SUPPORTING STATISTICS

The system was tested using a standard wide-angle USB 2.0 camera (specifications elaborate on in sub-section C) and yielded a frame rate of 24 per second on average. Taking the reciprocal of the frame rate gives us the testing latency of the algorithm as a whole. This value came out to be 41.67 milliseconds. A snapshot of the execution sequence can be seen in figure 3 where **Q** was obtained as 0.63 - 0.12i + 0.31j + 0.62k, as visualized in figures 9 and 10, with figure 9 representing the 1 + 0i + 0j + 0k state. The accuracy of the same is verified in sub-section B 'Model Accuracy Verification'. All results obtained in this section were taken from the 25th test iteration of the novel implementation after they were deemed satisfactory.

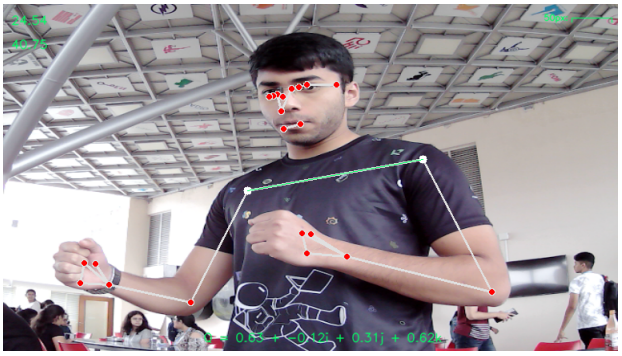

Fig. 3. A demonstration of the quaternion extraction from the pose using the novel implementation.

A. Testing Conditions

The scenes employed in the testing of the novel implementation were intentionally chosen to reflect the systems intended use case according to the authors, i.e.; estimating the angle of orientation of a pedestrian object from the point of view of an autonomous robot. As such, they were made to contain multiple human objects to demonstrate the arbitrary selection of the skeletal frame with the highest confidence value due to the singularly threaded nature off the novel implementation.

The domain taken for all angles of testing given the use case were all human pose objects with an orientation between 10 and 180 degrees with respect to the axis determined by the quaternion.

B. Model Accuracy Verification

Due to current resource limitations, the angle of orientation of the human pose object was extracted to verify the accuracy of the quaternion model by cross-checking the angle being faced by the user of the program as illustrated by equations 11 through 13.

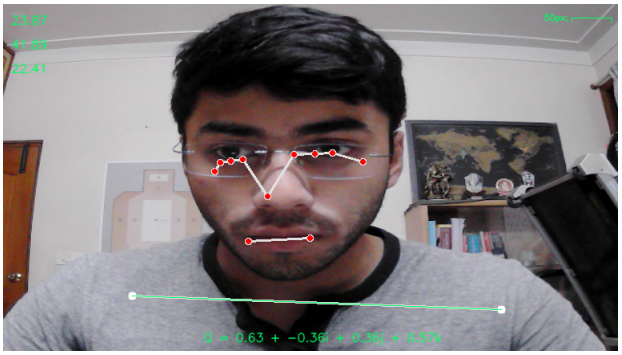

Fig. 4. A representation of a stationary human pose object taken for initial quaternion accuracy evaluation. Note that only the pose object itself and the shoulder landmarks need to be detected for all calculations following the same in the algorithm.

In a preliminary test of a stationary pose object as illustrated in figure 4, it was noted that the components of the quaternion were generated with the following variances and standard deviations over the time-spans illustrated in figures 5 through 8 -

- Real component variance and standard deviation : 3.08e-4, 1.76e-2
- i component variance and standard deviation : 9.83e-4, 3.14e-2
- j component variance and standard deviation : 3.83e-4, 1.96e-2
- k component variance and standard deviation : 2.39e-5, 4.88e-3

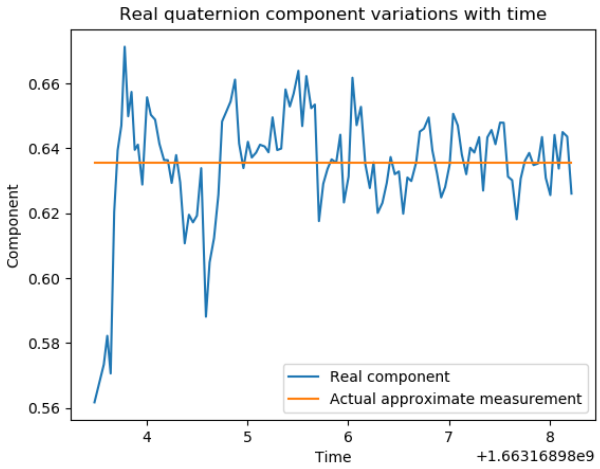

Fig. 5. A line graph representing the variation of the real component of the quaternion with respect to time.

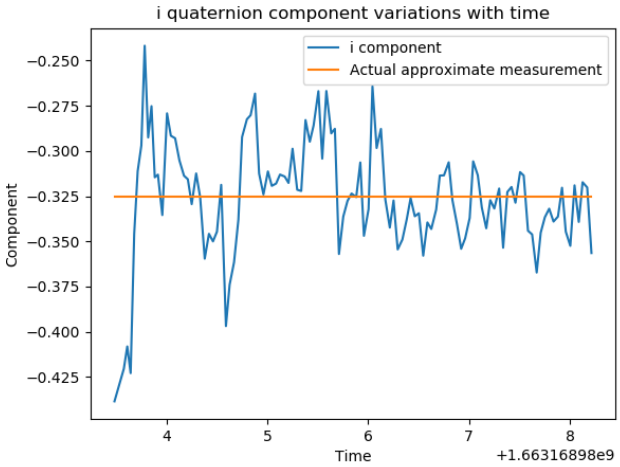

Fig. 6. A line graph representing the variation of the i component of the quaternion with respect to time.

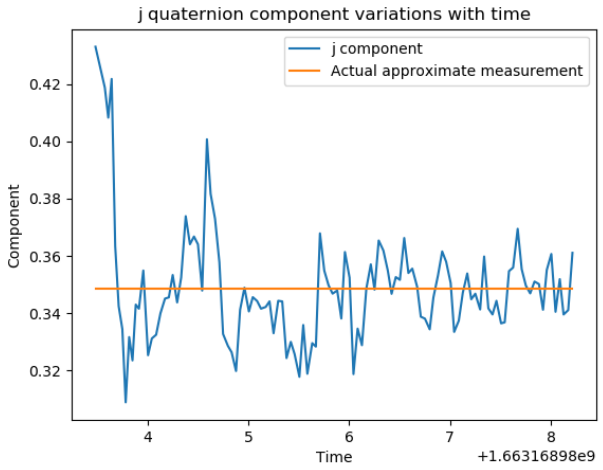

Fig. 7. A line graph representing the variation of the j component of the quaternion with respect to time.

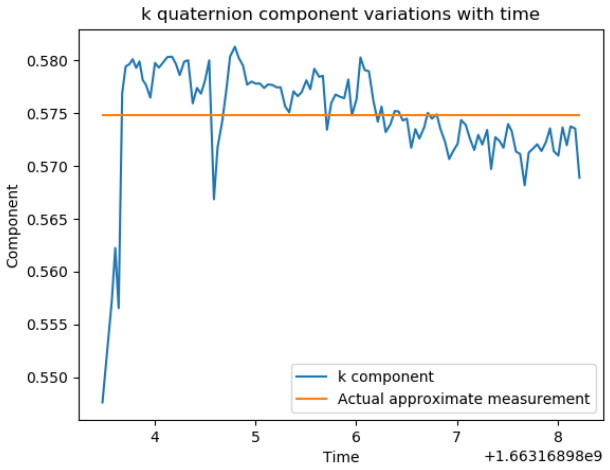

Fig. 8. A line graph representing the variation of the k component of the quaternion with respect to time.

C. System specifications

This system was tested on a platform with the following specifications -

- Intel Core i5 7200U processor
- 8GB RAM
- 512 GiB Solid State Drive
- External USB 2.0 30 FPS 2MP 'Passport' camera with a resolution of 1920x1080 and view angle of 110 degrees.

D. Kalman filter to eliminate quaternion noise

In order to obtain the angles of orientation quoted in sub-section B, a Kalman filter in one dimension was applied to the realAngle readings in order to get a stable feed [15].

To re-iterate, the angle measurements were transformed as given in equation 13 to account for the fact that all values

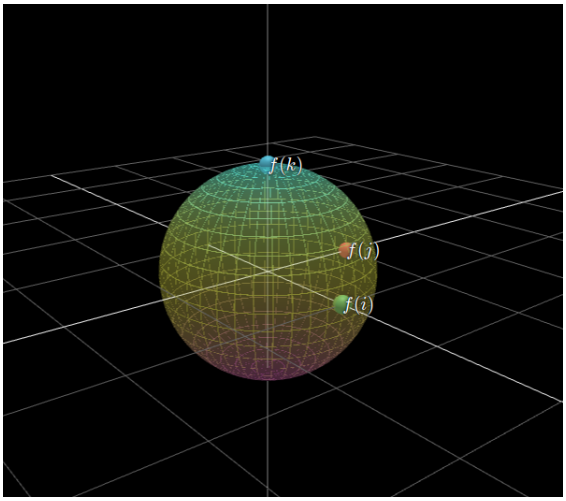

Fig. 9. A reference figure for what the quaternion representation looks like initially on the visualization platform at the **Q** = 1 state [14].

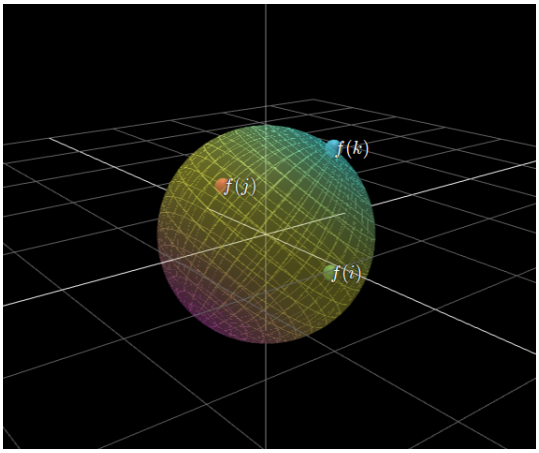

Fig. 10. A representation of the quaternion generated from the pose estimate in figure 3 [14].

for realAngle were taken with respect to an axis horizontally bisecting the frame, which was effectively achieved by taking test cases of human pose objects with erect postures. Refer to figure 11 for a visualization of the same.

The filter designed was made to be dynamic for a system assumed to be perfect, i.e.; the predicted covariance equation assumed the model uncertainty to be zero. This was because of the random nature of the system and no viable physical equations present to accurately define the motion of the pose object. Hence, the filter was made to accept the average of the last 10 readings in the window as the prediction when the real measurement was deemed to be not viable, as demonstrated in figure 11. For scale, 24 measurements were calculated from the generated quaternion every second.

Equations 14 through 16 cover the updation of the Kalman gain, state and covariance respectively, where k is the Kalman gain, p is the covariance, r was the angle measurement

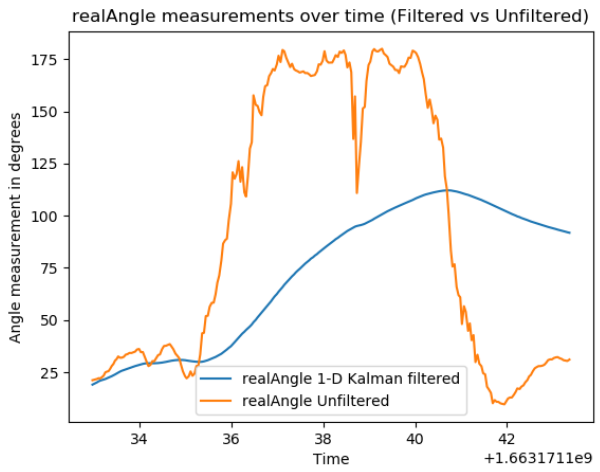

Fig. 11. A line graph representation of the filtered values from the realAngle measurement.

uncertainty (emperically determined as 0.5 after testing), x was previous measurement and z was the current measurement itself -

$$k = \frac{p}{p+r}\,;\; kalman\; gain\; calculation \tag{14}$$

$$x = x + k(z - x)\,;\; state\; updation \tag{15}$$

$$p = (1 - k)p\,;\; covariance\; updation \tag{16}$$

In order to demonstrate the translation of this system into use cases in the real world, we chose to build a preliminary prototype of a pedestrian intent classification system [16], where the co-ordinates of a pedestrian object on the frame as well as take the realAngle output value mentioned earlier to were recorded in a window of values to project the future path of motion on to the same frame. The classification was done using a fuzzy state approach taking these inputs into account. A demonstration of this is illustrated in figure 12, where one should take note of the third value in the top-right of the overlay reading 172.04. This was the angle of orientation of the pedestrian object returned by the novel implementation in an on-ground test. This can be visually verified by the fact that the referenced pedestrian object was walking almost perpendicularly across the projected path of the robot. Also note that the image here has been flipped horizontally, and hence does not seem consistent with the previously mentioned frame bisecting axis definition at first glance.

The intention classifier was able to accurately classify approximately 75 percent of the co-ordinates of the human pose object on the frame 2 seconds into the future with a radius of accuracy of 50 pixels.

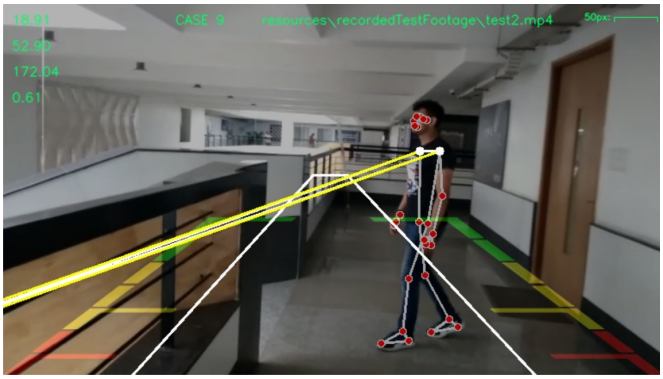

Fig. 12. A demonstration of the fuzzy pedestrian intent classification algorithm prototype implemented from a ground vehicle perspective to demonstrate the practicality of quaternion extraction from a 2-D camera frame containing a human skeletal pose object.

V. Conclusions and End Notes

Therefore, the conclusion was drawn that the novel implementation was suitable for extracting the pose of a human object using 2-D cameras and satisfied the requirements of preserving costs due to minimal hardware usage as well as those of latency and performance as the algorithm can be implemented on hardware [17] and performs within reasonable limits of accuracy. Notwithstanding the latter, its single threaded nature as well as low computational resource usage make it suitable for edge deployment applications, although it should be noted that the number of threads required increases linearly with the number of pose objects detected.

A. Optimizations and Future Scope

The novel implementation can be optimized in a variety of ways, with the primary approach being multi-threading of the video feed as well as the implementation of the algorithm for detecting multiple human objects and processing their pose estimations in parallel [18]. Expanding on the use case mentioned in section IV, sub-section A 'Testing Conditions', the model can be used to implement a more elaborate version of the pedestrian intent classifier demonstrated in sub-section D of the same section with more reliable prediction of whether or not a pedestrian object will cross the path in front of an autonomous robot such that its motion may endanger the lives of the same [19].

Additionally, optimizing the algorithm to make use of the large core count of a GPU also has promising prospects [20] [21], although a deeper literature survey into the topic would have to be conducted to say so for sure.

B. Acknowledgment

This research was supported by the Center for Internet of Things, PESU-RR. We thank our colleagues from PES University who provided insights and expertise that greatly assisted the research, although they may not agree with all of the interpretations/conclusions of this paper.

REFERENCES

[1] K. Fathian, J. P. Ramirez-Paredes, E. A. Doucette, J. W. Curtis, and N. R. Gans, "Quaternion based camera pose estimation from matched feature points," *arXiv preprint arXiv:1704.02672*, 2017.

[2] D. Pavllo, D. Grangier, and M. Auli, "Quaternet: A quaternion-based recurrent model for human motion," *arXiv preprint arXiv:1805.06485*, 2018.

[3] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4346–4354.

[4] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2891–2900.

[5] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–13, 2017.

[6] D. Schneider, "Deeper and cheaper machine learning [top tech 2017]," *IEEE Spectrum*, vol. 54, no. 1, pp. 42–43, 2017.

[7] D. Eberly, "Rotation representations and performance issues," *Magic Software: Chapel Hill, NC, USA*, 2002.

[8] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee *et al.*, "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[9] S. Sarabandi and F. Thomas, "A survey on the computation of quaternions from rotation matrices," *Journal of Mechanisms and Robotics*, vol. 11, no. 2, 2019.

[10] S. Rungruangbaiyok, R. Duangsoithong, and K. Chetpattananondh, "Shoulder angle measurement (sam) system for home-based rehabilitation using computer vision with a web camera." *Songklanakarin Journal of Science & Technology*, vol. 43, no. 5, 2021.

[11] V. Bazarevsky and I. Grishchenko, "On-device, real-time body pose tracking with mediapipe blazepose," Aug 2020. [Online]. Available: https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html

[12] W. R. Hamilton, *Elements of quaternions*. London: Longmans, Green, & Company, 1866.

[13] D. L. Baggio, *Mastering OpenCV with practical computer vision projects*. Packt Publishing Ltd, 2012.

[14] B. Eater and G. Sanderson, "Visualizing quaternions, an explorable video series," Sep 2018. [Online]. Available: https://eater.net/quaternions

[15] Z. Huang, P. Du, D. Kosterev, and B. Yang, "Application of extended kalman filter techniques for dynamic model parameter calibration," in *2009 IEEE Power & Energy Society General Meeting*. IEEE, 2009, pp. 1–8.

[16] J.-Y. Kwak, B. C. Ko, and J.-Y. Nam, "Pedestrian intention prediction based on dynamic fuzzy automata for vehicle driving at nighttime," *Infrared Physics & Technology*, vol. 81, pp. 41–51, 2017.

[17] J. Hegarty, J. Brunhaver, Z. DeVito, J. Ragan-Kelley, N. Cohen, S. Bell, A. Vasilyev, M. Horowitz, and P. Hanrahan, "Darkroom: compiling high-level image processing code into hardware pipelines." *ACM Trans. Graph.*, vol. 33, no. 4, pp. 144–1, 2014.

[18] K. Roszyk, M. R. Nowicki, and P. Skrzypczyński, "Adopting the yolov4 architecture for low-latency multispectral pedestrian detection in autonomous driving," *Sensors*, vol. 22, no. 3, p. 1082, 2022.

[19] B. Völz, K. Behrendt, H. Mielenz, I. Gilitschenski, R. Siegwart, and J. Nieto, "A data-driven approach for pedestrian intention estimation," in *2016 ieee 19th international conference on intelligent transportation systems (itsc)*. IEEE, 2016, pp. 2607–2612.

[20] G. A. Laguna-Sánchez, M. Olguín-Carbajal, N. Cruz-Cortés, R. Barrón-Fernández, and J. A. Álvarez-Cedillo, "Comparative study of parallel variants for a particle swarm optimization algorithm implemented on a multithreading gpu," *Journal of applied research and technology*, vol. 7, no. 3, pp. 292–307, 2009.

[21] K. Fatahalian and M. Houston, "A closer look at gpus," *Communications of the ACM*, vol. 51, no. 10, pp. 50–57, 2008.