AI & multi-resolution Temporal Processing for Accurate Counting of Exercises Repetitions

Youssef Fayad*
Air Defense College
Alexandria, Egypt
dr.youssef.6455.adc@alexu.edu.eg

Ahmed Khairy Mahmoud
Air Defense College
Alexandria, Egypt
Ahmedkhairy2@gmail

Shady El Gohary
Air Defense College
Alexandria, Egypt
sh_elgohary@yahoo.com

Hayman El-Sayed Hassan
Air Defense College
Alexandria, Egypt
g.d.Hayman98@gmail.com

2021 International Telecommunications Conference (ITC-Egypt) | 978-1-6654-4574-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITC-EGYPT52936.2021.9513940

Abstract—**Artificial intelligence has its importance in the field of objects' movement tracking, which is extensively applied in the fitness field. Applying machine learning technology in fitness field could introduce an auto judge by counting accurately the repetitions of any exercise. On the other hand, camera resolution depends heavily on number of frames per second (fps) which may affect the AI decision accuracy due to the possibility of losing some frames. In this paper a new algorithm relay on temporal processing for the combination of Computer Vision and deep machine learning techniques are introduced to analyze frames data accurately and report a feedback on the repetitions of exercises.**

Keywords—AI, Temporal subspace, Motion tracking, Computer Vision, deep machine learning.

I. INTRODUCTION

Great efforts have been done to improve the AI decision accuracy when applying with computer vision [1-4]. The majority of researches were aimed to improve the processing time in order to introduce a real time decision which had been at the expense of accuracy.
In [2], Khurana, and et al. had introduced a Gym Cam algorithm to detect, recognize and track exercises. In this algorithm a neural network had been trained via using a recorded video. Also, in [3] Soro, and et al. had represented a deep learning approach for exercise recognition and repetition counting depending on training a neural network with data extracted from a smart watch. But these two previous methods are not able to indicate correct/incorrect repetitions.

Chen, Steven, Yang, and Richard introduced a pose trainer algorithm [4] which detects the exercise pose and provides in details a useful feedback. But the main disadvantage of this method is that the relative position of the camera with respect to the person should be the same each time we execute the algorithm.

Talal Alatiah, Chen Chen[1] used an algorithm depends on open pose estimation technique to track body motions. Real-time network detects human poses and extracts their skeleton key points (wrists, elbows, knees, and ankles) from an input video or an external camera. This method measures, filters, and smooth the angles of the major joint for the performed exercise. Then, it counts the correct repetitions of the exercise. They stated that the algorithm achieved an error of counter within ±1 reps. But this algorithm doesn't avoid the main drawback of the net pose estimation technique that its model gives different results with different environments, and physiology specifics. For example, the model may not count some repetitions even if the exercise had performed correctly due to the difference in men's and women's body postures during physical training are big. Gunnar Farneback [5] introduced a more robust and dynamic algorithm to estimate motion of interesting features via approximating each neighborhood of both frames by quadratic polynomials using the polynomial expansion transform, and then applied a series of refinements. Gunnar stated that the problem of too large displacement can be reduced by doing the analysis at a coarser scale, but at the same time the accuracy decreases.

In order to benefit from Gunnar Farneback ability to deal with large displacement and avoid its low accuracy problem, this paper introduces a hybrid algorithm that applies the polynomial expansion transform of Gunnar Farneback method within temporal subsamples algorithm [6]. This algorithm enables three dimensions analysis including time resolution with small steps between pixels of the two consecutive frames. A video of the tracked motion is captured. Each group of the captured frames has been packed online as a subgroup. Then the algorithm represents a tempo-spatial resolution for each, whereas the algorithm picks a tempo samples for the signal enclosed by each spatial window. That enables a tempo resolution for each spatial window in order to increase motion detection resolution. The Gunnar Farneback method is applied on each tempo-spatial frame layer which also represents a large displacement as a group of small perturbations. Finally, the algorithm employs the deep learning technique to solve the Gunnar Farneback optical flow accuracy problem. The deep learning model has got its training dataset from more than 500 photos processed also with Gunnar Farneback optical flow. Then, the algorithm predicts the body movement between frames with more accurate repetitions counting.

The rest of the paper is organized as follows: section 2 presents methodology and main system design, section 3 shows the experimental results analyses, and section 4 introduces conclusion.

978-1-6654-4574-0/21/$31.00 ©2021 IEEE

II. Proposed Algorithm

A. The Gunner Franeback optical flow

In 1940s James J. Gibson introduced the concept of optical flow in order to study the action within the environment. Optical flow is the pattern of apparent motion of objects in a visual scene due to the relative motion between the observer and the scene. It also illustrates the apparent velocities as the brightness pattern of the image (frame).

Gunner Franeback optical flow calculates the motion between two image frames with time difference Δt depending on Taylor series which uses partial derivatives w.r.t spatial and temporal coordinates.

For 2D+t dimensional case (*x, y, t*) we can distribute each frame for voxels, each with intensity *I(x, y, t)* and moves with (*Δx, Δy,* and *Δt*) between two consecutive frames.

So, we will have *I(x+ Δx, y+ Δy, t+ Δt).*
Assuming the movement is small. So from the Taylor series we can get [5],

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \quad (1)$$

Divided on *Δt* we have,

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0 \quad (2)$$

Which results in,

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \quad (3)$$

Where V_x, V_y are the x, and y velocity components (optical flow) of *I(x, y, t)*, and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ are the derivatives of the image (frame) voxel in the corresponding directions.
Thus,

$$I_x V_x + I_y V_y = -I_t \quad (4)$$

This is an estimation equation with two unknowns.
In order to solve the optical flow estimation, Lucas-Kanade method [7-9] is used as differential method which assumes that the flow is essentially constant in a local neighborhood of the voxel under consideration.
For each point (*p*) under consideration, all the frame contents are assumed to have a small displacement (approximately constant) w.r.t two nearby frames. So, the optical flow equation (equation 4) can be assumed to hold for all pixels within a window centered at *p*.
So, the velocity vector will realize;

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &\vdots \\ &\vdots \\ &\vdots \\ I_x(q_m)V_x + I_y(q_m)V_y &= -I_t(q_m) \end{aligned} \quad (5)$$

Where;
$q_1, q_2, \ldots\ldots, q_m$ are the pixels inside the window, and $I_x(q_j), I_y(q_j), I_t(q_j)$ are the partial derivatives of the frame at point (q_j) w.r.t *x, y, t* at the current time.
In another form,

$$\begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_m) & I_y(q_m) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_m) \end{bmatrix} \quad (6)$$

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_m) & I_y(q_m) \end{bmatrix}, \; v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \; b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_m) \end{bmatrix}$$

B. Window temporal resolution

In order to increase data resolution, the (*n*) data enclosed by each window is picked up with small time steps (τ), so for each window we will have;

$$b(\tau) = \left[\begin{array}{c:c:c:c} -I_\tau(q_1) & -I_{\tau 2\tau}(q_1) & \cdots & -I_{\tau(n-1)\tau}(q_1) \\ -I_\tau(q_2) & -I_{2\tau}(q_2) & \cdots & -I_{\tau(n-1)\tau}(q_2) \\ \vdots & \vdots & \vdots & \vdots \\ -I_\tau(q_m) & -I_{2\tau}(q_m) & \cdots & -I_{\tau(n-1)\tau}(q_m) \end{array}\right]$$

By using the least square principle for each tempo-spatial layer,

$$A^T A v = A^T b \quad (7)$$

Or;

$$V = (A^T A)^{-1} A^T b \quad (8)$$

Where A^T is the transpose of A, $A^T A$ is called the structure tensor (second-moment matrix) of the image at the point *p* which summarizes the predominant directions of the gradient in a specified neighborhood of the point and the degree to which those directions are coherent.

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_j I_x(q_j)^2 & \sum_j I_x(q_j) I_y(q_j) \\ \sum_j I_y(q_j) I_x(q_j) & \sum_j I_y(q_j)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_j I_x(q_j) I_{i\tau}(q_j) \\ -\sum_j I_y(q_j) I_{i\tau}(q_j) \end{bmatrix} \quad (9)$$

j = 1, 2 , ……. , m, and i = 1, 2, ……, n.
but the least square solution stated in equation (9) gives the same importance to all pixels in the window. In order to precise the least square results it is preferred to give more weight to the pixels that are closer to the central pixel *p* symbolized as *W* which is an $m \times m$ diagonal matrix and is usually set to a Gaussian function.
Substitute in (7);

$$A^T W A v = A^T W b \quad (10)$$

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_j w_j I_x(q_j)^2 & \sum_j w_j I_x(q_j) I_y(q_j) \\ \sum_j w_j I_y(q_j) I_x(q_j) & \sum_j w_j I_y(q_j)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_j w_j I_x(q_j) I_{i\tau}(q_j) \\ -\sum_j w_j I_y(q_j) I_{i\tau}(q_j) \end{bmatrix} \quad (11)$$

for the additive white Gaussian noise (*N*) equation (11) will be,

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} =$$

$$\begin{bmatrix} \sum_j w_j I_x(q_j)^2 & \sum_j w_j I_x(q_j) I_y(q_j) \\ \sum_j w_j I_y(q_j) I_x(q_j) & \sum_j w_j I_y(q_j)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_j w_j I_x(q_j) I_{i\tau}(q_j) \\ -\sum_j w_j I_y(q_j) I_{i\tau}(q_j) \end{bmatrix} + \sigma^2 \Sigma_N \tag{12}$$

Note, for pixel j suitable for Lucas-Kanade method, the $A^T A$ large eigenvalues satisfy $\lambda_1 \geq \lambda_2 > 0$, which span for the pixels that are closer to the window center. In order to realize motion tracking through several frames sequence, the flow vector can be iteratively applied and recalculated till threshold approach to zero so it can be assumed that the image windows are the most similar, so for each successive tracking window the point can be tracked through several frames in a sequence until the point either obscured or goes out of frame.

C. *Deep machine learning for temporal Farneback*

Deep learning structures algorithms in layers to create an "artificial neural network” that can learn and make intelligent decisions on its own. Whereas; with machine learning systems, a human needs to identify and hand-code the applied features based on the data type. Due to this advantage, deep learning have been employed in the proposed algorithm for solving optical flow problems.

First, large amounts of Farneback optical flow training data should to be prepared in order to compute optical flow with deep neural networks. Second, each group of training data is labeled in order to address the motion of each point in the frame. Finally, a two consecutive video frames (previous (d-1) – next (d)) is taken to output the Farneback optical flow, so we will have;

$$\left(V_x, V_y\right) = f\left(I_{d-1}, I_d\right) \tag{13}$$

Where $f(I_{d-1}, I_d)$ is a neural network with two consecutive frames as its input.

We had utilized CNN as a deep learning method which also known as shift invariant or space invariant artificial neural network (SIANN). CNN enables an accurate decision with small objects, which consequently enables good discrimination between the right and wrong positions in a dynamic scene with different backgrounds and different physiology specifics.

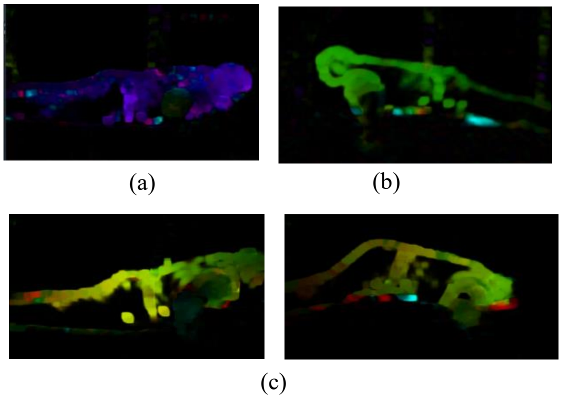
In order to have accurate results of the convolutional neural networks (CNNs), the exact motion of each voxel in the frame should be figured out by applying Gunnar Farneback with high accuracy. So, it is better to create a large training dataset with a small displacement, and trained the flow net in this data set. That had been realized by using high fps camera to capture our training data set. Lately, applying the temporal algorithm will increase the frame resolution which in turn clears out the overlapping regions of our labeled categories, and gives probability with an obvious leaning towards the correct position (label). The last step enables us to predict the state of the moving target more accurately.

III. Experimental results Analyses

The algorithm is implemented using python 3.8.5.

Three labeled data sets (for each exercise) have been created via employing HSV mask within Gunnar Farneback function in python using video captured at 30 fps. Also, 500 images are captured from the previous process for the three labeled category, then jupyter notebook is used to train the convolutional neural networks using Tensor Flow and create the model then saved as tensor flow Saved Model format to enable Keras in order to restore both built-in layers as well as custom objects. Work is validated in several steps. First, our model accuracy is compared with reference [1] model accuracy. Second, the algorithm is executed for counting pushups and pull-ups within calibrated videos from real competitions. Finally, the algorithm is executed for counting pushups and pull-ups live to the player of the Air defense college physical fitness team, and comparing results with a simultaneous manual judge.

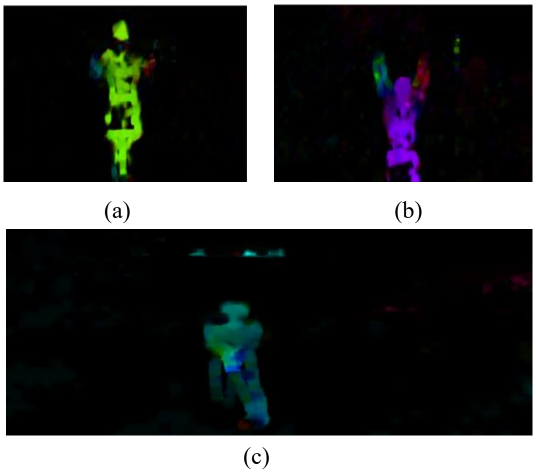
1) *Data Preprocessing*



(a) (b)

(c)

Fig. 1. Three labeled category of push-ups with temporal Gunnar Farneback (a) down (b) up (c) no move



(a) (b)

(c)

Fig. 2. Three labeled category of pull-ups with temporal Gunnar Farneback (a) up (b) down (c) no move.

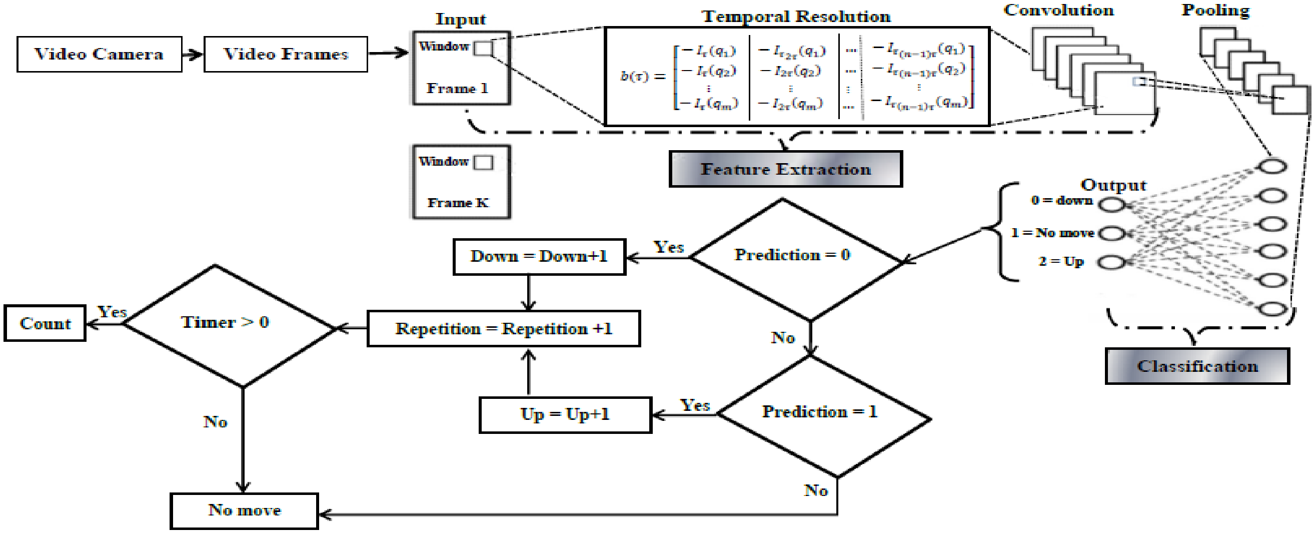
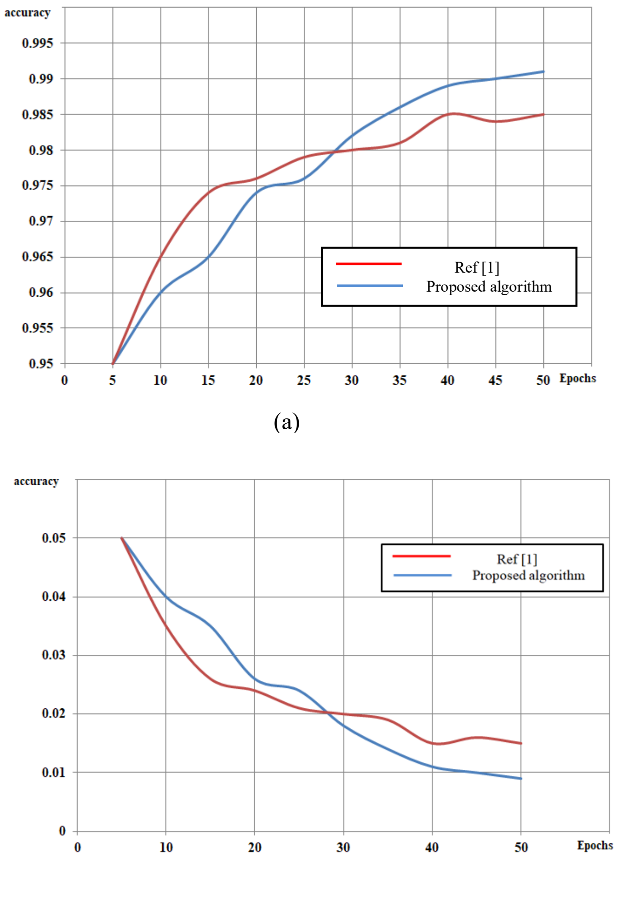

Video Camera
Video Frames
Input
Window
Frame 1
Window
Frame K
Temporal Resolution
$b(\tau) = \begin{bmatrix} -I_\tau(q_1) & -I_{2\tau}(q_1) & \cdots & -I_{(n-1)\tau}(q_1) \\ -I_\tau(q_2) & -I_{2\tau}(q_2) & \cdots & -I_{(n-1)\tau}(q_2) \\ \vdots & \vdots & \vdots & \vdots \\ -I_\tau(q_m) & -I_{2\tau}(q_m) & \cdots & -I_{(n-1)\tau}(q_m) \end{bmatrix}$
Convolution
Pooling
Feature Extraction
Output
0 = down
1 = No move
2 = Up
Classification
Prediction = 0
Yes
Down = Down+1
No
Prediction = 1
Yes
Up = Up+1
No
Repetition = Repetition +1
Timer > 0
Yes
Count
No
No move

Fig. 3. Block diagram of the proposed system.

The proposed algorithm is illustrated in Figure 3. Figure 4 plots the model accuracy & loss with different number of iterations compared with [1].



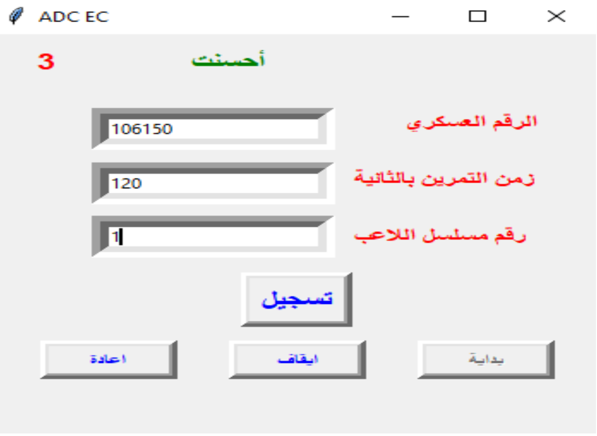
(a)

(b)

Fig. 4. (a) model accuracy on training set (b) model loss on training set.

2) Counting exercises

Mathematical function (Argmax) is the most commonly used function for finding the class with the largest predicted probability. Also, the python code is transformed and all necessary Python and Tensor Flow packages are packed to executable file in order to run it as a window program. Finally we use tkinter function to design a suitable (GU) interface that enables us to enter the required data to run the program. Figure 5 (a) displays the (GU) interface and results of program for different exercises with different backgrounds.



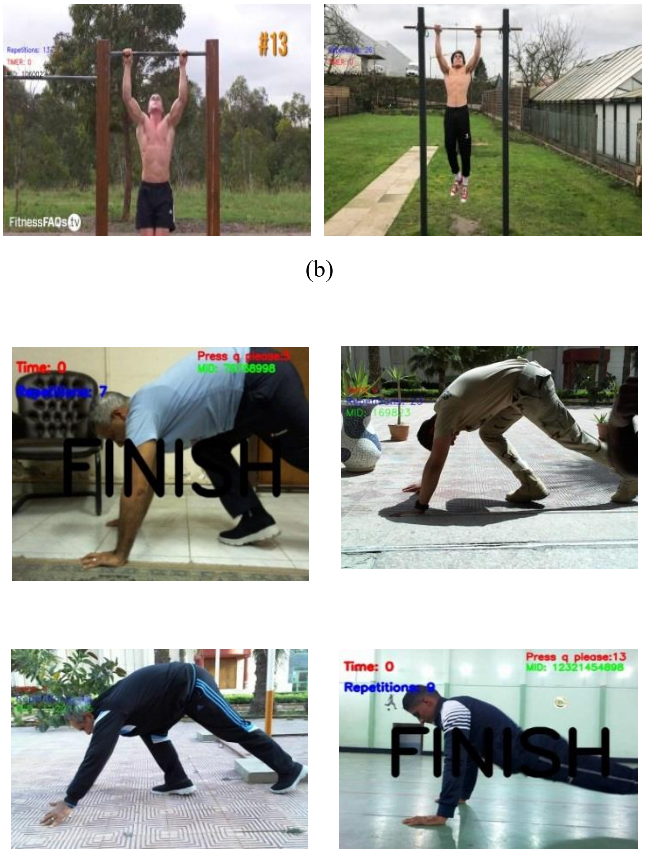
3 أحسنت

الرقم العسكري 106150

زمن التمرين بالثانية 120

رقم مسلسل اللاعب 1

(a)



(b)

(c)

Fig. 5. (a) (GU) interface (b) Snapshots of an output for a recorded video stream of pull-up exercises with two different backgrounds. (c) Snapshots of an output for a recorded video stream for push-up exercises with different backgrounds and different players have different physiology specifics.

Ref [1] algorithm depends on open pose estimation technique to track body motions. Although our proposed method has more processing time which consequently means more complexity, it is obvious from figures 4 (a, b) that the accuracy of our model is better than the accuracy of Ref [1] with 0.6%. Also, losses of our model are less than Ref [1] with about 0.55%. This improvement had been realized due to use a large amounts of accurate training data sets.

Figure 5(b) presents snapshots from a recorded video stream of pull-up exercises; figure 5(c) presents snapshots from a live video stream of push-up exercises with different backgrounds and different players have different physiology specifics. The proposed algorithm had succeeded in avoiding the main drawback of the net pose estimation technique - used in Ref [1] - that its model gives different results with different environments and physiology specifics. Also, it introduces a great performance when it is tested with both right, and wrong positions of each exercise.

IV. Conclusions

In this paper, a new method is developed based on the concept of temporal subspace. Firstly, the spatial subspaces algorithm is used to set up pixels windows. Secondly, the temporal sampling is executed to increase frame resolution. Track body motion between frames using Gunnar Farneback within the combination tempo-spatial subspaces algorithm enables better performance specially in counting only right counts and excludes the wrong count. In other meaning, our proposed algorithm presents more accurate tracking. This consequently enhances the counting accuracy. Now, our laptop GPU is PRADEON, In future work we may use NVIDIA graphics processing unit that will enable us to apply YOLO-v5 for training because it works better with NVIDIA GPU.

References

[1] Talal Alatiah, Chen Chen "Recognizing Exercises and Counting Repetitions in Real Time," arXiv: 2005.03194 [cs.CV], (2020).

[2] Khurana R., Ahuja K., Yu Z., Manko J.,Harrison C., & Goel M., "Gym-Cam: Detecting, Recognizing and Tracking Simultaneous Exercises in Unconstrained Scenes", proceeding of the ACM on interactive, mobile, wearable and ubiquitous technologies, (2018).

[3] Soro, A., Brunner, G., Tanner, S., & Wattenhofer, R. "Recognition and Repetition Counting for Complex Physical Exercises with Deep Learning", sensors 2019, vol. 19, no. 3, (2019).

[4] Chen, Steven & Yang, Richard., "Pose Trainer: Correcting Exercise Posture using Pose Estimation", arXiv: 2006.11718 [cs], (2020).

[5] Gunnar Farneback "Two-Frame Motion Estimation Based on Polynomial Expansion," Scandinavian Conference on Image Analysis (SCIA 2003), pp. 363-370, (2003).

[6] Youssef Fayad, Caiyun Wang, Qunsheng Cao Alaa El-Din Sayed Hafez "A developed ESPRIT algorithm for DOA estimation," Frequenz 2015, vol. 69, no. 3, pp. 263-269, (2015).

[7] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision". Proceedings of Imaging Understanding Workshop, pages 121-130, (1981).

[8] Bruce D. Lucas, "Generalized Image Matching by the Method of Differences", doctoral dissertation, (1984).

[9] Aurelien Plyer, Guy Le Besnerais, Frederic Champagnat, "Massively parallel Lucas Kanade optical flow for real-time video processing applications", journal of Real-Time Image Processing, vol. 11, no. 4. pp. 713-730, (2016).