Proceedings of The 2019 IEEE International
Conference on Real-time Computing and Robotics
August 4-9, 2019, Irkutsk, Russia

Fast and Fluid Human Pose Tracking

Lorenzo Landolfi[1], Paolo Tripicchio[1], Alessandro Filippeschi[1] and Carlo Alberto Avizzano[1]

***Abstract*—Recent advancements in human motion behaviors based on camera images made human motion tracking much more robust than in the past. However, they are still computationally expensive and they do not allow for online reconstruction of the human pose. In this context, this work presents a framework for tracking 2D human pose at high frequency while keeping the robustness of state-of-the-art methods. We achieved this result by combining, by means of Kalman filtering, the recent success of OpenPose, a robust deep learning-based 2D human pose estimation technique, with the fast Lucas-Kanade features matching method. This allows for processing images at different framerates, thus having multirate measurement updates. The frequency of the estimation is kept high by setting the Kalman filter time step. The method was tested on videos of several activities which include both fast and slow motion. The results show an improvement on the reconstruction error, an increased speed of reconstruction, better tracking during fast motion, and the capability to cover loss of tracking from the two measurements. The achieved intraframe (between two available measurements) trajectory estimation frequency was as high as 1 kHz.**

***Keywords*—Real time 2D pose tracking, multithreading, data fusion**

I. INTRODUCTION

The exploitation of the massive computational parallelism given by graphical processing units (GPU) for general purpose algorithms [12] and the use of Convolutional Neural Networks (CNN) [7] for automatic feature extraction, opened the way to recent remarkable advancements in object localization [8], [14], which in turn allowed the development of innovative techniques for quasi real time 2D human pose estimation from RGB images. [3], [4], [10]. Human pose estimation has been targeted since long. Recent advancements are based either on wearable sensors [15] or RGB image analysis, as a branch of object identification in images [11].

Human pose estimation is indeed a key component that can allow machines to get a better understanding of the behavior of people appearing in videos and images. As a confirmation to the above statement, it has been shown by Fanelli and others [1] that in general any action/activity recognition/anticipation system is greatly fostered by the use of body pose related features. Automatic activity recognition has been recognized as an important field of studies as it has been successfully applied in a vast number of fields such as assisted living systems for smart homes, healthcare monitoring, surveillance systems for indoor and outdoor activities, and tele-immersion applications [13].

Anyway, state of the art pose estimation systems are still not able to reach real-time performance, specially with machines equipped with off-the-shelf hardware. Often the user is required to arrange a trade off between desired accuracy and detection frequency. Moreover, even the best one-shot detectors do not exploit temporal information or relations between consecutive frames. Neglecting temporal information may easily lead to noisy measurements and to the confusion between body joints that are visually similar (e.g. left-right shoulder in human body). Important works have been made to lower the measurement noise with recurrent deep learning networks [6], [5]. However, these methods still require high computational burden, thus not being suitable for real-time applications.

Moving from the achievements of the deep learning-based methods, this work presents a framework for tracking 2D human pose at high frequency while keeping the robustness of state-of-the-art methods.

The proposed approach exploits a Kalman filter in order to fuse human pose estimation based on OpenPose (OP) [3], a robust deep learning-based technique, with the fast Lucas-Kanade (LK) [9] features matching method. This allows for processing images at different framerates, thus having multirate measurement updates. The frequency of the estimation is kept high by setting the Kalman filter time step. This asynchronous computation system has been realized as a collection of threads and shared queues that ensures fluidity of the pose estimation. This ensures that the tracker can be applied for both real-time and offline applications. The offline use allows faster and still accurate processing of motion data, and can be beneficial for human performance analysis. Conversely, the real-time application allows real-time feedback and live interaction with human or machine partners, for example in gaming or social robotics.

The paper is structured as follows. Section II discusses the implemented method with a focus on the development of the Kalman filter (Section II-C), the implementation of both the LK method (Section II-B) and the OP method (Section II-A), and the software architecture II-E. The experimental setup is reported in Section III, whereas the results are reported and discussed in Section IV.

II. METHOD

The method that we devised for the purpose of human motion tracking exploits three main components: detection, feature tracking, and robust estimation.

Detection is achieved with a state of the art algorithm: OpenPose. Feature tracking is achieved with the Lucas-Kanade method implemented with pyramids [18]. Finally measurement fusion for robust estimation is obtained by means of a linear Kalman Filter.

[1]PERCRO laboratory of perceptual robotics, Scuola Superiore Sant' Anna, Pisa, Italy.

978-1-7281-3726-1/19/$31.00 © 2018 IEEE

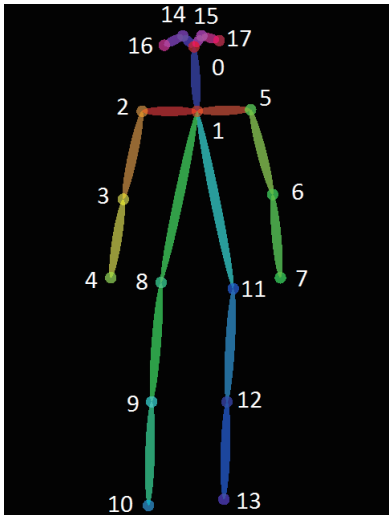

Fig. 1: OpenPose output in COCO format

A. Initial body pose Detection

Body pose key points are initially detected with OP, which outputs eighteen 2D body coordinates disposed as in Figure 1. Given an input image, OP estimates the location of the keypoints of all the bodies found in the scene and simultaneously estimates the likelihood of each key point to be connected to the other ones. As OP is a *detector*, it does not rely on any information that could be possibly given by the previous frames or by the motion of the points. It simply analyze the input RGB image with a deep convolutional neural network to locate the body landmarks and then it connects them through an heuristic algorithm. OP also emits a measure of confidence (ranging from 0 to 1) for each detected keypoint and conventionally sets to (0,0,0) any undetected point of a body.

B. Feature tracking

Once the keypoints are detected by OP, it is possible to exploit the LK tracker to find them in the subsequent frames. The LK algorithm estimates the displacement of a pixel in two consecutive frames by analyzing the variation of pixel intensity in a neighborhood. Calling u the movement w.r.t. x and v the movement w.r.t. y and $I(x, y)$ the pixel intensity at pixel (x, y) LK imposes that

$$\frac{dI}{dx}(x, y, t) \cdot u + \frac{dI}{dy}(x, y, t) \cdot v = -\frac{dI}{dt}(x, y, t)$$

As with a simple pixel we have two unknowns (u, v) and one equation, a neighborhood of pixels is needed to obtain more equations. In this way the system becomes over determined and LK proposes the least square (LSQ) solution that averages the optical flow guesses over the neighborhood. This algorithm is much faster than the detection one because its complexity is only dependent on the number of tracked points and not on the number of pixels in the image. The main issue with this tracker, as with all the trackers based on visual features, is that it cannot handle occlusions. The combination of the key points detector with the LK tracker allows to boost the speed of the pose estimator, but it does not guarantee a fluid reconstruction of human motion. The Kalman filter described in the following section and the software infrastructure described in Section II-E have been designed precisely for enhancing and overcoming the aforementioned limitations. On the one hand, they allow for obtaining a real time pose estimation without "hiccup", on the other, they allow us to get an estimate of the position of 2D body key points even during occlusions.

C. Measurement fusion for robust tracking

To fuse the estimates from OP and LK, as well as to decide an estimation frequency which is independent of the video framerate a linear Kalman filter is used, as described by the following equations:

$$x_k = F_k x_{k-1} + \epsilon_k \tag{1}$$

$$z_k = H_k x_k + \nu_k \tag{2}$$

where $x_k = [s_1(k), s_2(k), \dots s_n(k)]$ and $s_i = [p_i^u, p_i^v, \dot{p}_i^u, \dot{p}_i^v]$. $[p^u, p^v]$ is the position of a body key point in pixel whilst $[\dot{p}^u, \dot{p}^v]$ is the estimated velocity in the u and v components. The state-transition matrix F_k is a block-diagonal matrix, whose blocks, $F_i(k)$ for each body key point i are

$$F_i(k) = \begin{bmatrix} & \Delta t & 0 \\ I_{2\times 2} & 0 & \Delta t \\ 0_{2\times 2} & & I_{2\times 2} \end{bmatrix}$$

Being $I_{2\times 2}$ and $O_{2\times 2}$ the identity and zero matrices respectively. ϵ_k and ν_k are zero mean Gaussian random variables representing respectively the process noise and the measurement noise, whose covariances are Q_k, R_k respectively. Finally, z represents the measurements and H_k is the identity matrix.

Given the limited bandwidth of human motion dynamics, and the fact that LK measurements can be obtained faster than typical video framerates, we assume that the state estimation error between two measurements updates is sufficiently bounded using a simple kinematic model, which does not include body points acceleration.

1) Measurements: The implemented Kalman filter exploits two measurements: those received from OP, which are reliable but acquired at slow frequencies, and those received from LK: less reliable, acquired at a much higher frequency, and dependant on the initial OP estimations. In order to account for the difference in their reliability, the measurement noise factor ν_k of equation 2 has been set to ν_k^{op} for OP measurements and ν_k^{LK} for the LK measurements; being $\nu_k^{op} < \nu_k^{LK}$.

D. Fusion of multi-rate measures

The flexibility of the presented method allows for applications both *offline* and *online*. In the *offline* applications, it is possible to decide the frequency of the detector. We have identified three possible situations.

1) We process each video frame with OP to obtain z_k, and, when it is not available, we update it with LK applied to the last OP measurement. In this case the detection of landmarks is improved only in the frames in which OP

fails to reconstruct the body, there are not predict steps alone, but there are no improvements of the processing speed.

2) We process each video frame with OP and OP measures are fused with LK measures to obtain z_k. LK measures are not based on the last OP measure. In this case it is possible to obtain some improvements w.r.t. OP measures only.
3) We do not process every frame with OP, but we discard as many frame as we want in order to arrange a trade-off between having the OP accuracy and processing a video in a reasonable amount of time. In this case we have the same advantages of point 1, but with more bandwidth.

In the *real time* application of the presented tracker, we identified two possible situations.

1) We do not let the OP thread discard any frame (the size of frame queue is set to ∞). In this case the difference between the frame currently visualized and the processed one grows proportionally to the difference between computing time and input frame rate. This is useless for typical video framerate .
2) Body pose reconstruction is bound to a specific latency (implemented case, the size of the frame queue is 1), LK is applied to every frame and it is updated as soon as an OP measure is available.

Point 1 actually refers to a non real time case and it has been ignored for implementation. A possible customization on point 2 would be the possibility of bounding the queues to a specific value grater than one. In this case we would have more reliable measurements and the latency between the rendered frames and OP processed ones would not be constant but it will have a lower and upper bound depending on the video input frequency.

In the following we refer to the case reported in point 2. As it is shown in Figure 2, every component of the system works at different frequency. The OP measurements are presented at a much slower pace than the LK ones so that the Kalman filter receives OP measures relative to frame $t-k$ after it has processed the LK measures relative to frame t. In this case, velocity of the body points is set to 0 to avoid estimation drift. The advantage of using LK measures is exemplified in Figure 3, where it is possible to see that, because of the simple kinematic model, the tracked landmarks tend to drift as much as the body motion speed tend to increase.

E. Software System overview

The motion tracking system has been implemented in C++, with the open source implementation of both the pyramidal LK and Kalman filter made available by the OpenCV library [2]. The software infrastructure is a composition of threads exchanging messages through shared queues, always containing the most recent elements. Items are placed in the queues as shared pointers, hence allowing zero copy communication among software components. One thread, named *Detector*, uses OP to find the key-points from the last frame found

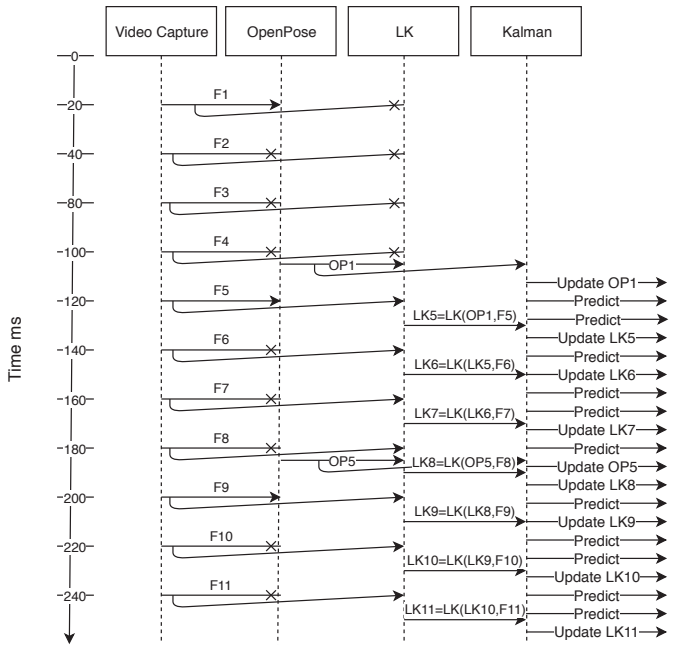

Fig. 2: UML diagram of the algorithm process. In this example the average frame inter arrival time is $20ms$ (50fps), OP service time is set to $65ms$, LK service time is $10ms$, while the KF is capable of emitting an output approximately every $2.5ms$.

in the *Frame Queue* and exposes them in the shared queue (*Detection Queue*) of detected key points. Another thread: the *Tracker*, reads the last frame present in the Frame Queue and writes the LK tracked points in the shared queue of tracked points (*Tracked Queue*). Finally, the Kalman thread performs a *predict* step at the frequency chosen by the user and an *Update* every time either the Detection Queue or the Tracked Queue is not empty. Figure 4 exemplifies the software system architecture. Notice that the modules work at different frequencies. The optical Tracker and the detector are bounded by the input frame rate frequency, whereas the Kalman output can be produced at a higher frequency.

III. EVALUATION

In order to asses the quality of our tracker, we produced ground truth landmarks for each video by processing each frame with OP, and manually labelling the undetected or mistaken points. Because of the resolution of the available videos, we neglected points 14,15,16,17 of Figure 1. Experiments were performed on a laptop equipped with an Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz and an NVIDIA 970m GPU. We point out that we had to use both the GPUs the testing machine was equipped with in order to compute and visualize landmarks on the screen. The NVIDIA GPU was used to run OP on the assigned frames, whereas the INTEL GPU was used just for frames visualization. This because both the performance of the detector and data visualization

(a) Skeleton reconstructed without using LK tracking

(b) Skeleton reconstructed with LK tracking

Fig. 3: Difference between skeleton reconstruction with and without the inclusion of LK measurements. The represented frame is frame number 966 of the dance video, processed at 50 fps. Notice in Figure 3a the drift of the right arm following the movement of the arm from down to up and from left to right.

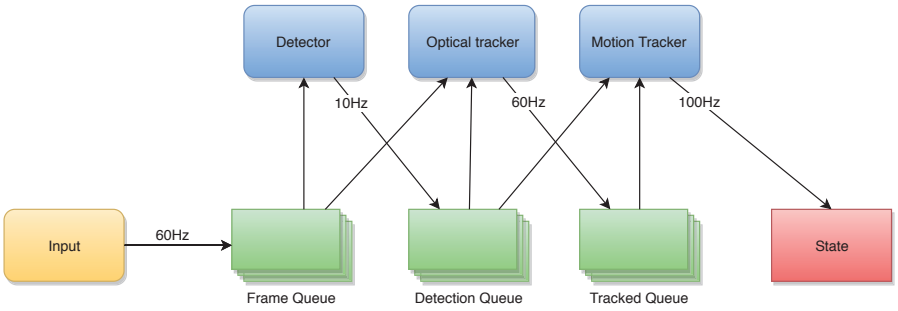

Fig. 4: Architecture of software threads with plausible working frequencies.

TABLE I: Description of videos used for test

Name	**Length** (#frames)	**Resolution**	**FPS**
Dancing	1000	490x360	30
Walking circle	500	1280x720	30
Walking frontal	600	640x480	30
Side walking	534	1280x548	24
kick	1380	720x1280	160

were severely affected by the access of the shared hardware resource.

A. Dynamic Time Warping

Dynamic time warping (DTW) [16], [17] is a technique that allows to align two signals shifted in time. Since the KF and the asynchronous measures produce a delay in the reconstruction of trajectories, we used DTW to align the ground truth trajectories with the ones produced by the tracker, then we measured the errors as the euclidean distances between those trajectories.

B. Experiments

We tested our method on the videos described in table I. Plots in Figure 5 shows the effectiveness of our tracker in reconstructing 2D body motion trajectories. Best results are achieved when there are no "holes" in the acquisition of OP measurements. Nevertheless, even when a small amount of consecutive points are missed by the detector, the Kalman based tracker is able to reconstruct plausible trajectories.

Moreover, there are reported cases in which the tracker is able to correct some wrong key point assignments made by the detector, thus making the motion tracking more robust overall. Figure 6 reports one of such cases.

Finally Figure 7 reports the error obtained by our tracker as a function of the video capture frequency. To calculate the error, we applied multi-dimensional time warping to all the tracked points simultaneously and we divided the result by the number of tracked key points. This method ensures that the warping of time is coherent across all the tracked points. It should be noted that processing the *circular walking* video the obtained error was slightly larger than with other videos given the fact that during the motion the tracked subject was exiting the camera frustum.

In Figure 7, the abscissa represent a virtual video processing speed. Said f_0 the framerate of the input video, abscissas greater than f_0 (see the square marker for each video) represent video processing at a higher speed than the online case. We highlight that the average measured frequency of output by the OP detector in the experiment was approximately 10Hz.

Finally, Figure 8 presents the errors obtained without time warping the tracked trajectories. In this case the reported error E_r is calculated as equation 3, being N the number of tracked points and T the number of frames, $p[k]_i^{gt}$ the $i-th$ ground truth point of the $k-th$ video frame and $p[k]_i^{e}$ the $i-th$

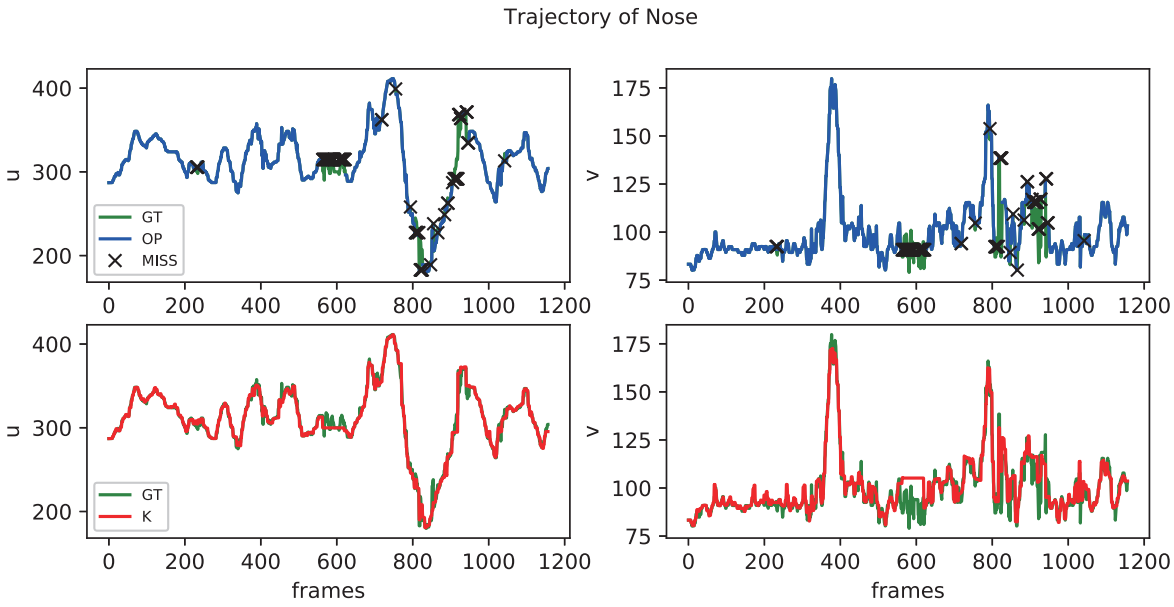

Fig. 5: Trajectory of the nose in the dance video. Comparison between the Ground truth (GT), the proposer tracker (K) run at 30fps and the OP measurements (OP). Trajectories are plotted as functions of the video frames.

Fig. 6: An example of a frame where the LK+Kalman tracker manages to fix some detector errors. The blue skeleton is the one reconstructed by OP, the green skeleton is the one reconstructed by the tracker.

estimated point at frame k. For the reconstruction framerates smaller than f_0, an average has been applied to the ground truth data, whereas for framerates bigger than f_0 we applied a linear interpolation.

$$E_r = \frac{1}{N}\sum_{i=1}^{N} e_i, e_i = \frac{1}{T}\sum_{k=1}^{T} ||p[k]_i^{gt} - p[k]_i^{e}||_2 \qquad (3)$$

In order to take into account the different aspect ratios of the videos of Table I, all the errors reported in Figures 7, 8 are normalized by the maximum possible pixel error, which is the size of the hypotenuse between the sides of the video frames.

IV. DISCUSSION

The experimental results show that the tracker can reconstruct the position of the body landmarks effectively in a reduced amount of time. The results show that the reconstruction error increases with the processing speed. The increase is almost linear with the processing speed, however, the slope is quite limited. In facts, in the worse case, it is $5\,10^{-5}[1/fps]$. We remark that the highest processing speed that we tested is about 16 times faster than OP.

The experiments show that our approach is effective in reconstructing landmarks trajectory although we did not model the kinematic model of the relationship between body joints. Indeed the choice of considering only the first derivative in the Kalman model seems to be a reasonable approach considering the time scale of body motion and the frequency of LK updates.

Given the simplicity of the model and the reduced amount of points, it is reasonable to think that the proposed tracker can reach frequencies higher than the 160 fps tested in this

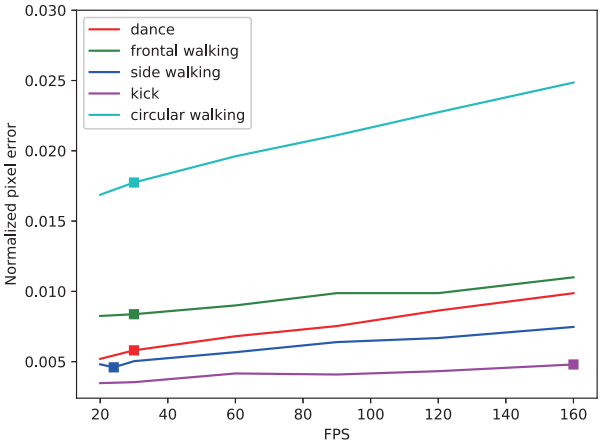

Fig. 7: Tracking error as a function of video-capture speed (frame per second) with time consistent across all the tracked body points. Square markers show the original video framerates.

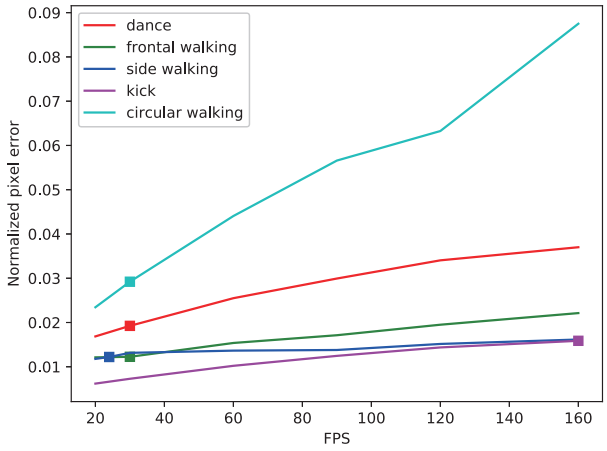

Fig. 8: Plot of real time tracking error as described by equation 3.

work. Differently from deep-learning-based skeleton detection methods, the speed may be affected by the number of tracked points.

V. CONCLUSION

In this paper we proposed a simple and interesting technique that combines robust and well established algorithms (Kalman filter, LK feature tracker) with a modern deep learning based 2D body pose landmarks detector (OpenPose). The performance achieved both in the offline experiments and online ones confirmed that our method is promising yet simple, not relying in any complex infrastructure on top of the landmark detector.

The effect of the number of landmarks has not been assessed so far and we devise to study it in a future work. Moreover, we will evaluate the effects of imposing a kinematic model to the human skeleton. Finally, we will assess the proposed method against a ground truth (e.g. optical marker-based motion tracker) to quantify the error in a metric space.

REFERENCES

[1] Gabriele Fanelli Angela Yao, Juergen Gall and Luc Van Gool. Does human action recognition benefit from pose estimation? In *Proceedings of the British Machine Vision Conference*, pages 67.1–67.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.67.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050, 2016.

[4] Yu Chen, Chunhua Shen, Xiu-Shen Wei, Lingqiao Liu, and Jian Yang. Adversarial posenet: A structure-aware convolutional network for human pose estimation. *CoRR*, abs/1705.00389, 2017.

[5] Huseyin Coskun, Felix Achilles, Robert S. DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5525–5533, 2017.

[6] Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *CoRR*, abs/1511.05121, 2015.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[9] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[10] Guanghan Ning, Zhi Zhang, and Zhihai He. Knowledge-guided deep fractal neural networks for human pose estimation. *CoRR*, abs/1705.02407, 2017.

[11] L. Peppoloni, M. Satler, E. Luchetti, C. A. Avizzano, and P. Tripicchio. Stacked generalization for scene analysis and object recognition. In *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014*, pages 215–220, July 2014.

[12] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 873–880, New York, NY, USA, 2009. ACM.

[13] Suneth Ranasinghe, Fadi Al Machot, and Heinrich C Mayr. A review on applications of activity recognition systems with regard to performance and evaluation. *International Journal of Distributed Sensor Networks*, 12(8):1550147716665520, 2016.

[14] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[15] Emanuele Ruffaldi, Lorenzo Peppoloni, Alessandro Filippeschi, and Carlo Alberto Avizzano. A novel approach to motion tracking with wearable sensors based on probabilistic graphical models. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1247–1252. IEEE, 2014.

[16] Hiroaki Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[17] Stan Salvador and Philip Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. In *KDD workshop on mining temporal and sequential data*. Citeseer, 2004.

[18] Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.