Optimal video handling in on-line hand gesture recognition using Deep Neural Networks

1st Dimitrios Makrygiannis
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
dimimakr@csd.auth.gr

2nd Christos Papaioannidis
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
cpapaionn@csd.auth.gr

3rd Ioannis Mademlis
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
imademlis@csd.auth.gr

4th Ioannis Pitas
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
pitas@csd.auth.gr

***Abstract*—Deep Neural Networks (DNNs) are machine learning models with a myriad of uses, such as enabling tools that support professional workers or facilitating new modes of human-computer communication. Hand gesture recognition from RGB video feed is one of the most important relevant tasks where DNN models tend to excel, with many application domains typically demanding their real-time execution (e.g., in human-robot interaction scenarios). However, several operational parameters of similar models, especially concerning the way the input video data are handled, have not yet been sufficiently investigated with a clear goal of identifying best practices. For instance, various design choices about how video frames are fed to the DNN model during its training and its evaluation (e.g., using the traditional temporal subsampling per video approach, or employing a temporal sliding window) directly affect method accuracy. This paper aims at empirically finding optimal strategies regarding such operational parameters for input video handling when training and evaluating learning models that perform real-time, on-line gesture recognition, using typical CNN-based approaches on a relevant dataset.**

***Index Terms*—real-time gesture recognition, deep learning, neural networks, Long Short-Term Memory, Multi-Layer Perceptron, sliding window**

I. Introduction

Hand gesture recognition is one of the most important tasks where Deep Neural Networks (DNNs) are typically used nowadays. It is of immense importance in modern, non-verbal human-computer/robot interaction, with most related applications requiring on-line, real-time gesture recognition, possibly on embedded hardware.

A good, representative example would be collaboration between a human worker and an autonomous camera-equipped drone/Unmanned Aerial Vehicle (UAV): such a setup is expected to become common in the near-future, since current commercial drones are rather inexpensive and almost always prepackaged with at least an RGB camera, while their cognitive autonomy functionalities are rapidly increasing [1]–[12]. Their autonomous interaction with a human worker would offer significant advantages over traditional approaches in many industries, mainly due to the drones' ability to reach places that are inaccessible to people, their easy deployment and their aerial point of-view. Human collaborators should be able to give specific instructions to such UAVs, while the drones should in turn be able to interpret them. Visual gestures are an ideal mechanism for facilitating this process in an intuitive manner, even in noisy environments, while removing the need for: a) the worker to carry a communication device, and b) for any form of mediating Graphical User Interface (GUI).

A different example comes from novel forms of human-computer interaction in an indoors setting, with commercial products such as Kinect having already proven the worth of hand gesture recognition during the past decade. In this scenario, the relevant algorithms are typically aided by the possibility of using RGB-D sensors instead of pure RGB cameras. Several application areas have already been significantly boosted by exploiting the possibilities opened-up in this manner, from smart homes [13] to elderly care [14].

Assuming the most inexpensive scenario of videos captured on-the-fly by a simple RGB camera, on-line gesture recognition can be defined as the following task: at each time instance, given the sequence of video frames up to now, predict the hand gesture currently performed by a human captured in the video feed, among a set of predefined gesture classes. This is a rather difficult problem, since at different time instances the respective person may appear in visually variable scenes and under inconstant scale, clothing and lighting conditions, while different people may perform each gesture in a slightly different manner.

DNN-based machine learning models are the most typically used algorithms for hand gesture recognition. Several architectural alternatives can be employed: for instance, combining a regular Convolutional Neural Network (CNN) which separately processes each video frame to extract a semantically meaningful representation with a Long Short-Term Memory (LSTM) network [15] predicting a gesture class label (gesture ID), a convolutional LSTM or a 3D CNN with spatiotemoral convolutional filters. A CNN-LSTM hybrid is the most typically utilized architecture for on-line gesture recognition when real-time performance is essential, especially on embedded AI hardware, since ConvLSTMs and 3D CNNs are noticeably slower, both at the training and at the inference stage. The traditional way of approaching the problem under a CNN-LSTM framework is to define a temporal sliding window of fixed length $N \in \mathbb{N}$ (in consecutive video frames) and sequentially feed the LSTM classifier with the convolutional representations of each video frame in the window. Thus, a gesture ID is predicted at each time instance, given the last N video frames. However, typically, this process is only employed during actual model deployment, while both training and evaluation on datasets are performed by simply temporally subsampling each training/test sequence to a fixed number of video frames, given that neighboring frames contain visually redundant content.

Instead of training the overall CNN-LSTM for gesture recognition in an end-to-end manner, employing a CNN pretrained for 2D body pose/skeleton estimation has recently dominated the field, as a more accurate approach [16]–[21]. In this case, the LSTM is not fed latent convolutional representations of each video frame, but finalized representations of the estimated visible body joints (in 2D pixel coordinates) at each video frame. This method alleviates the negative effect of appearance variations across the data samples, especially in the typical scenario where the available domain-specific gesture recognition dataset is small, but there are large-scale, generic, publicly available benchmark datasets for 2D body pose/skeleton estimation. In such a case, end-to-end training from raw RGB video frames (without pretraining the CNN for 2D skeleton extraction) would almost certainly lead to overfitting and low generalization ability.

Despite extensive research on 2D skeleton-based CNN-LSTM algorithms, various choices about how video data are being handled and which significantly affect on-line gesture recognition performance have not been systematically investigated up to now, with only ad-hoc solutions typically applied in a haphazard manner. The most important operational parameter of this nature is how input videos are fed to the DNN model during its training and its evaluation on a dataset, where temporal subsampling to a fixed number of video frames (separately per input sequence) is the dominant approach, despite the fact that sliding windows are typically employed instead during actual model deployment.

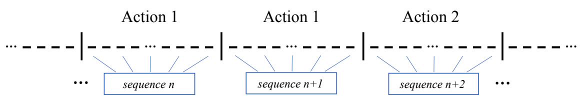
In this paper, different strategies regarding the handling of video data in the context of on-line hand gesture recognition, using fast, 2D skeleton-based CNN-LSTM neural architectures, are compared in a systematic manner, during both the training and the inference stage. This is a preliminary step towards covering the gap in relevant literature. The solutions that are being compared for feeding the input data to the DNN model are: a) traditional video subsampling, b) isolated sliding windows computed separately over each input sequence, and c) mixed sliding windows computed across a concatenation of all training/test sequences. The main difference between b) and c) is that the latter approach results in certain windows temporally extending around gesture transition/switching points, thus exposing the DNN to data resembling real-world deployment scenarios. All possible combinations of using a), b) or c) during the training and/or the evaluation stage were investigated.

Assessment of the various choices was conducted using a state-of-the-art 2D skeleton-based CNN-LSTM. All experiments were repeated with a faster, shallow Multilayer Perceptron (MLP) in lieu of the LSTM classifier neural head, since the emphasis of this paper is on gesture recognition engines with real-time performance constraints. Evaluation and identification of best choices for the investigated approaches to input video handling was performed using a recent, 6-class hand gesture RGB video dataset for human-drone interaction. The best-performing input video handling strategy is identified both for the CNN-LSTM and for the CNN-MLP scenario, while these two DNN architectures are compared in terms of accuracy in evaluation settings resembling real-world deployment.

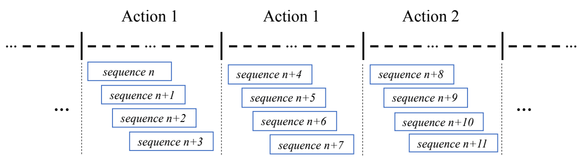
II. Related Work

In general, human activity and hand gesture recognition are very similar and well-researched problems [22], [23]. During the past years, many different video classification systems have been proposed for these tasks. For example, a Fourier Temporal Pyramid (FTP) was utilized in [24], in order to model the temporal dynamics of the extracted body joint positions. Similarly, FTP was used along with Dynamic Time Wrapping (DTW) in [25] to address specific issues, such as noise. Subsequently, using a different approach, [26] used histograms to represent 3D human skeletons, which were then given as input to a discrete Hidden Markov Model (HMM) [27] that recognized activities/gestures. HMMs were also utilized in [28] to predict sequences from high-level skeletal features.

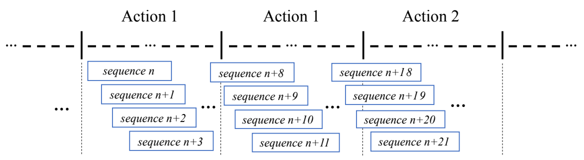
Despite the initial success of FTP, DTW and HMM in temporal dynamics modeling, many methods utilizing LSTMs emerged during the past few years, demonstrating superior performance. For example, a hierarchical LSTM network architecture was proposed in [29] to separately model the temporal dynamics of the lower body and the upper body, which were later combined together to obtain the final predictions. Having the same goal in mind, [30] proposed an end-to-end deep LSTM network. Subsequently, a two-branch stacked LSTMs network architecture for activity recognition was introduced in [31], which processed 2D human skeletons. Furthermore, [32] exploited the ability of LSTMs to use different step-sizes and model various attributes by introducing an ensemble of short-term,



(a) Temporal video sequence subsampling.



(b) Isolated temporal sliding window.



(c) Mixed temporal sliding window.

Fig. 1: Video handling strategies. Each dash represents a single video frame from the raw input videos, while each constructed sequence is a data point.

medium-term and long-term Temporal Sliding LSTMs for skeleton-based action/gesture recognition.

In [33] a model of dynamic skeletons was proposed, called Spatial-Temporal Graph Convolutional Network (ST-GCN). It automatically learns spatial and temporal patterns from given data, minimizing the computational cost and increasing the generalization capability. This method was an attempt to model the long-term correlation of features for each body part. In contrast, [21] tried to couple satisfactory gesture recognition accuracy with fast execution for real-time settings. Thus, 2D human skeleton sequence properties were analyzed by the Double-feature Double-motion Network (DD-Net) architecture. By using a lightweight model (i.e., 0.15 million parameters), DD-Net can achieve rapid execution on an ordinary GPU. In [34] it was observed that not all skeletal body joints are actually informative for gesture recognition; the irrelevant ones often just add noise to the recognition process, thus degrading accuracy. Thus, Global Context-Aware Attention LSTM (GCA-LSTM) was proposed, which is capable of selectively focusing on the useful joints in each video frame by using a global context memory cell. A recurrent attention mechanism was introduced with it, so that the model's attention focus can be enhanced progressively. In an attempt to extend traditional CNNs for human activity/gesture recognition, [35] used a two-stream CNN that extracts both spatial and temporal information.

3D CNNs have also been employed for similar tasks, although they are not the preferred solution when real-time performance is of essence. In [36], a 3D CNN combined multiple feature sources, such as dynamic image, optical flow and raw video frames, exploiting the fact that dynamic images are fast to compute and able to capture long-term temporal information of a video. In most related methods, the input is a sequence of consecutive video frames, thus the DNN model is expected to learn spatiotemporal features from the early neural layers. To facilitate this, [37] proposed a Hierarchical Max-Pooling Model (HMAX) for video classification, with predefined spatiotemporal features within the first layer. Deeper two-stream CNNs were utilized in [38], while [39] presented an end-to-end trainable 3D CNN-based gesture recognition method that uses a temporal encoding approach to model characteristics of the entire input video.

Despite the extensive literature on on-line hand gesture recognition, including several papers dealing with real-time settings using DNNs, there is a complete lack of systematic investigation regarding common operational parameters of input video handling in such methods, during both model training and evaluation, in order to identify best practices. As a result, ad-hoc solutions such as temporal input sequence subsampling to a fixed number of video frames have come to dominate the literature, despite the prevalence of sliding windows during actual model deployment. This paper is a preliminary attempt at empirically identifying optimal input video handling strategies for the popular 2D skeleton-based CNN-LSTM setting, using a state-of-the-art architecture and an extensive hand gesture dataset for human-drone interaction.

III. Video handling in on-line hand gesture recognition

The operational parameter of on-line hand gesture recognition that this paper focuses on is input video handling, during training and/or evaluation of the neural model. Three alternative strategies are compared:

- *temporal video sequence subsampling* to a fixed number of video frames N.
- *isolated temporal sliding window* of a fixed length N, separately computed across each input sequence.
- *mixed temporal sliding window* of a fixed length N, computed across a concatenation of all training/test sequences.

Subsampling is in fact the dominant approaching in model training and evaluation using LSTM-based video classifiers, with N typically equal to the length (in number of video frames) of the shortest video in the training/test dataset. However, actual deployment of such models (at inference mode only) for on-line hand gesture recognition in real-world scenarios is usually implemented using a temporal sliding window, since there is no prior knowledge about when each novel gesture begins and ends. Thus, incoming consecutive video frame representations are packaged in overlapping windows of fixed length $N \in \mathbb{N}$ which are sequentially fed to the LSTM. Then, a gesture ID is predicted at each time instance, given the last N video frames.

Given the prevalence of sliding windows during model deployment for on-line gesture recognition with LSTM-based models, it is possible that training and/or evaluating the model using a sliding window computed across the training/and test dataset as well can potentially increase accuracy, or provide a more precise assessment of model accuracy under real-world operating conditions. The main difference between isolated and mixed sliding window is that the latter approach results in certain windows temporally extending around gesture transition/switching points, thus exposing the DNN to data resembling real-world deployment scenarios.

Employing each of the three input video handling strategies in the training/evaluation stage, results in different ways to construct the training/test data points, respectively, as well as to a different number of training/test data points. However, in all cases, the corresponding datasets are constructed by a different manner of processing the same set of raw input videos, while each data point always contains a fixed number of N video frames for all three strategies. Figure 1 depicts how training/test data points are constructed from the raw input training/test videos, respectively, for each of these three strategies.

The research methodology of this paper is to assess all possible combinations of employing the three strategies during the training and/or the evaluation stage, in order to empirically identify the optimal choice.

IV. Quantitative Evaluation

Evaluation was conducted using a relevant, large-scale, RGB video dataset called *AUTH UAV Gesture Dataset* [40][1]. It is composed of 4930 videos, distributed along 6 classes: *cross arms, extend one arm to the side, namaste, thumbs up, victory, raise one arm.* The dataset is split in training/test sets according to the 80%/20% rule-of-thumb. It contains gestures performed by 8 different people and video sequences at a spatial resolution of 1920×1080 pixels, with a frame rate of 30 FPS. Recordings were conducted both indoors and outdoors, while the relative simplicity of the gestures is compensated by the fact that three out of the six classes appear to be rather visually similar to each other, making it more difficult for a gesture recognition algorithm to accurately classify such inputs.

A state-of-the-art DNN-based architecture was adopted for the evaluation process, which follows a common approach: each video frame is processed by a pretrained CNN in order to be converted into a 2D body joints list, i.e., a human skeleton, and the output is fed per-frame into a subsequent neural network that performs classification. Two different classifiers were employed: a) an LSTM network that captures temporal information recurrently, and b) an MLP network. Further details about these recognition approaches can be found in [41].

[1]For availability and distribution, please e-mail Prof. Pitas at pitas@csd.auth.gr, using "AerialCore - AUTH UAV Gesture Dataset availability" as e-mail subject.

The adopted state-of-the-art method for 2D human skeleton extraction per video frame was a CNN which outputs dense 2D body joints heatmaps for each input RGB video frame [42]. 2D pixel coordinates of each body joint can then be extracted by post-processing this heatmap. Importantly, in the gesture recognition scenario, only the 7 upper body joints estimated by [42] are of interest. This CNN was pretrained on the large Microsoft COCO 2D human pose estimation dataset [43]. In general, it is a lightweight neural architecture running very fast during inference, allowing near-real-time execution on embedded AI computational hardware suitable for autonomous UAVs[2]. Each input RGB video frame was preprocessed during training by cropping around the visible person, using automated CNN-based person detection [44]. All videos were cropped in such a way that the depicted person occupies approximately 80% of the obtained video height.

All input video frames were resized to 256×192 pixels. In the CNN-LSTM architecture, an LSTM network was attached at the end of the pretrained [42] model, receiving a vectorized list of 7 upper human body joint 2D coordinates per video frame, in a sequential manner. Thus, these lists of 2D locations were fed into an LSTM unrolling for $N = 15$ time steps, while the LSTM input dimension was 14. The network was composed of a single LSTM layer followed by a fully connected layer, each one containing 512 neurons, as well as a final fully connected, softmax classification layer. The dropout rate was set to 0.2. The batch size used for training was 256 and an SGD optimizer with momentum equal to 0.9 and weight decay equal to $1e-6$ was employed. The learning rate was 0.0001, while training was stopped after 1000 epochs.

In the alternative, faster CNN-MLP architecture, an MLP was attached as the classification head at the end of the pretrained [42] model, instead of the LSTM. It consisted of two fully connected layers, containing 512 and 64 neurons, respectively, as well as a final softmax layer, while also employing BatchNormalization [45] and Dropout [46]. In this case, all 2D upper body joints locations (in pixel coordinates) for $N = 15$ input video frames, i.e., the final output of [42], were concatenated into a single, 210-dimensional vector representing the entire input, that was fed to the classification network in one step. The dropout rate was set to 0.50 and the batch size used for training was 256, while an ADAM optimizer with initial learning rate and weight decay both equal to 0.0001 was employed. Training was stopped after 1000 epochs.

Given this experimental setup, all possible combinations of employing the temporal subsampling, mixed sliding window or isolated sliding window strategy for input video handling, during training and/or evaluation on the selected dataset, were evaluated separately for the CNN-LSTM and for the CNN-MLP architecture.

[2]E.g., nVidia Jetson Xavier.

TABLE I: Comparison of the three input video handling strategies, in terms of the Correct Classification Rate metric (CCR), on the two selected neural architectures (CNN-LSTM and CNN-MLP). “MSW" stands for mixed sliding window, “ISW" for isolated sliding window and “Sampling" for the traditional temporal video subsampling approach. Additionally, MSW evaluation scenarios were separately assessed with the “1-Gesture Windows" (1GW) and “Mixed-Gesture Windows" (MGS) approaches.

Model	Training Strategy	Evaluation Strategy	CCR (%)	1GW CCR (%)	MGW CCR (%)
MLP	MSW	MSW	61.25	67.4	56.7
LSTM	MSW	MSW	62.3	64.8	60.3
MLP	ISW	MSW	56.57	73.4	43.6
LSTM	ISW	MSW	57.68	71.2	47.4
MLP	Sampling	MSW	54.49	69.7	43.8
LSTM	Sampling	MSW	57.57	66.3	50.2
MLP	Sampling	Sampling	74.58	-	-
LSTM	Sampling	Sampling	71.59	-	-
MLP	Sampling	ISW	69.34	-	-
LSTM	Sampling	ISW	66.52	-	-
MLP	ISW	Sampling	74.17	-	-
LSTM	ISW	Sampling	73.76	-	-
MLP	ISW	ISW	73.28	-	-
LSTM	ISW	ISW	71.56	-	-
MLP	MSW	Sampling	70.55	-	-
LSTM	MSW	Sampling	68.80	-	-
MLP	MSW	ISW	67.13	-	-
LSTM	MSW	ISW	64.34	-	-

As it can be seen in Table I, the CNN-MLP architecture led to the overall best Correct Classification Rate (CCR) of 74.58%, when using the traditional temporal subsampling approach as an input video handling strategy during both the training and the evaluation stage. This is surprising to the extent that LSTMs are traditionally considered better at learning from sequential and timeseries data, since they store an internal state which is incrementally updated. In contrast, when using an MLP, input data from all N time instances used for predicting a class label/gesture ID have to be concatenated into a unified vector, which is fed to the classification head in a single step. Furthermore, due to the temporal subsampling video handling strategy, this unified vector represents the entire training/test video. Yet, the CNN-MLP provides slightly higher accuracy than the CNN-LSTM, with the latter architecture showing to bear no advantages. There are two complementary reasons we can identify as causes of this surprising result:

- the low-dimensionality of the input data per time instance (since only the estimated 2D upper body joint coordinates per video frame are being fed to the classification head) makes the problem easy for the MLP model to handle, despite the lack of internal state/memory and sequential processing.
- evaluating using the temporal subsampling strategy is not a setting resembling real-world deployment, since all test data inputs (composed of N time instances each) only contain a single ground-truth gesture (no transitions/switches).

Indeed, when employing a realistic mixed sliding window video handling strategy during the evaluation stage, similar to the strategy necessarily utilized during real-world deployment, the CNN-LSTM architecture surpasses its faster CNN-MLP competitor for all training-stage video handling strategy alternatives (subsampling, mixed sliding window, isolated sliding window). Thus, the inherent ability of the LSTM to model temporal dependencies via stored internal state and sequential/incremental processing comes to the forefront.

For completeness, all mixed sliding window evaluation scenarios were additionally and separately assessed with the “1-Gesture Windows" and “Mixed-Gesture Windows" approaches. In these cases, the test set where evaluation is conducted is a subset of the regular test set constructed by the mixed sliding window evaluation strategy. In the former/latter case (1-Gesture Windows/Mixed-Gesture Windows) only the test windows actually containing video frames from a single gesture class/temporally centered on transitions between two different gestures, respectively, are employed for evaluation. Therefore, the union of the test sets constructed by these two approaches equals the complete test set of the regular mixed sliding window evaluation strategy. Note that these two evaluation approaches (“1-Gesture Windows" and “Mixed-Gesture Windows") *are feasible only for the MSW evaluation strategy.*

As it can be seen in the two rightmost columns of Table I, training with isolated/mixed sliding windows led to the highest accuracy on the 1-Gesture/Mixed-Gesture Window test data points, respectively. This is a consistent and expected result, but the most important observation arises from contrasting it against the regular mixed window evaluation strategy: despite the fact that the 1-Gesture Window test data points are significantly more in number than their Mixed-Gesture Window counterparts, still the regular mixed sliding window evaluation (with a test set composed of the union of the 1-Gesture and the Mixed-Gesture Window test data points) indicates highest accuracy for the mixed sliding window training case. Similarly, although assessment only on the 1-Gesture Window test data points shows a slight advantage of the CNN-MLP architecture, when assessing accuracy on either the Mixed-Gesture Window test dataset or on the overall regular mixed sliding window evaluation, the CNN-LSTM seems clearly superior. This is despite the fact that the 1-Gesture Window test data points constitute the vast majority of the regular mixed sliding window evaluation test data points, since Mixed-Gesture Windows are significantly fewer in number due to their occurrence only on gesture transitions. Thus, the lead LSTMs enjoy in real-world deployment scenarios for timeseries problems, due to their inherent sequentially updated state, is validated once more.

V. Conclusions

This paper focused on identifying best practices regarding input video handling for on-line, real-time hand gesture recognition using CNN-LSTM neural architectures, by exploring and comparing alternatives to the temporal sequence subsampling to a fixed number of video frames approach, during both training and evaluation, which dominates the literature. Thus, all possible combinations of employing the temporal subsampling, mixed sliding window or isolated sliding window strategy for input video handling, during training and/or evaluation on a selected relevant dataset, were evaluated separately for a state-of-the-art CNN-LSTM architecture and for a faster CNN-MLP alternative. The underlying motivation was that the mixed sliding window strategy is more realistic, since it is in fact the strategy necessarily utilized during real-world deployment of on-line gesture recognition systems. The results indicate that the CNN-MLP architecture with a subsampling strategy during both training and evaluation provides the overall best accuracy, but the CNN-LSTM performs better when a realistic mixed sliding window strategy is applied during evaluation, no matter the input video handling scheme selected for training. From the opposite perspective, it is shown that constructing the training dataset using mixed sliding windows, instead of the typically used subsampling approach, leads to more accurate results for the most realistic evaluation scheme, both for the CNN-MLP and for the CNN-LSTM architecture. Future extensions of this research will attempt to further validate our findings, by comparing the presented CNN-LSTM/CNN-MLP architectures with similar methods, using all the examined video handling strategies.

VI. Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871479 (AERIAL-CORE). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] S. Papadopoulos, I. Mademlis, and I. Pitas, "Neural vision-based semantic 3D world modeling," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[2] S. Papadopoulos, I. Mademlis, and I. Pitas, "Semantic image segmentation guided by scene geometry," in *Proceedings of the IEEE International Conference on Autonomous Systems (ICAS)*, 2021.

[3] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous UAV cinematography: a tutorial and a formalized shot-type taxonomy," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–33, 2019.

[4] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous unmanned aerial vehicles filming in dynamic unstructured outdoor environments [applications corner]," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 147–153, 2018.

[5] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "High-level multiple-UAV cinematography tools for covering outdoor events," *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 627–635, 2019.

[6] I. Mademlis, A. Torres-González, J. Capitán, R. Cunha, B. Guerreiro, A. Messina, F. Negro, C. Le Barz, T. Gonçalves, A. Tefas, and I. Pitas, "A multiple-UAV software architecture for autonomous media production," in *Proceedings of the EURASIP European Signal Processing Conference (EUSIPCO) Satellite Workshop: Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, 2019.

[7] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "UAV cinematography constraints imposed by visual target tracking," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2018.

[8] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Shot type feasibility in autonomous UAV cinematography," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[9] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Shot type constraints in UAV cinematography for autonomous target tracking," *Information Sciences*, vol. 506, pp. 273–294, 2020.

[10] F. Patrona, I. Mademlis, A. Tefas, and I. Pitas, "Computational UAV cinematography for intelligent shooting based on semantic visual analysis," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2019.

[11] P. Nousi, I. Mademlis, I. Karakostas, A. Tefas, and I. Pitas, "Embedded UAV real-time visual object detection and tracking," in *Proceedings of the IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2019.

[12] I. Mademlis, V. Mygdalis, N. Nikolaidis, and I. Pitas, "Challenges in autonomous UAV cinematography: An overview," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[13] S. Desai and A. Desai, "Human-Computer Interaction through hand gestures for home automation using Microsoft Kinect," in *Proceedings of International Conference on Communication and Networks*, Springer, 2017.

[14] M. Oudah, A. Al-Naji, and J. Chahl, "Elderly care based on hand gestures using Kinect sensor," *Computers*, vol. 10, no. 1, p. 5, 2021.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] G. Chéron, I. Laptev, and C. Schmid, "P-CNN: Pose-based CNN features for action recognition," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] C. Li, Q. Zhong, D. Xie, and S. Pu, "Skeleton-based action recognition with Convolutional Neural Networks," in *Proceedings of the IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 2017.

[18] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou, "Deep progressive reinforcement learning for skeleton-based action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[19] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan, "An attention enhanced graph convolutional LSTM network for skeleton-based action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] F. Yang, Y. Wu, S. Sakti, and S. Nakamura, "Make skeleton-based action recognition model smaller, faster and better," in *Proceedings of the ACM Multimedia in Asia*, 2019.

[22] I. Mademlis, A. Iosifidis, A. Tefas, N. Nikolaidis, and I. Pitas, "Stereoscopic video description for human action recognition," in *Proceedings of the IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP)*, 2014.

[23] I. Mademlis, A. Iosifidis, A. Tefas, N. Nikolaidis, and I. Pitas, "Exploiting stereoscopic disparity for augmenting human activity recognition performance," *Multimedia Tools and Applications*, vol. 75, no. 19, pp. 11641–11660, 2016.

[24] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[25] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3D skeletons as points in a lie group," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[26] L. Xia, C.-C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3D joints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012.
[27] L. Rabiner and B. Juang, "An introduction to Hidden Markov Models," *IEEE Acoustics, Speech and Signal Processing Magazine*, vol. 3, no. 1, pp. 4–16, 1986.
[28] D. Wu and L. Shao, "Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
[29] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton-based action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[30] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.
[31] D. Avola, M. Cascio, L. Cinque, G. L. Foresti, C. Massaroni, and E. Rodolà, "2D skeleton-based action recognition via two-branch stacked LSTM-RNNs," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2481–2496, 2019.
[32] I. Lee, D. Kim, S. Kang, and S. Lee, "Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
[33] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.
[34] J. Liu, G. Wang, L.-Y. Duan, K. Abdiyeva, and A. C. Kot, "Skeleton-based human action recognition with global context-aware attention LSTM networks," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1586–1599, 2017.
[35] K. Simonyan and A. Z., "Two-stream convolutional networks for action recognition in videos," *arXiv preprint arXiv:1406.2199*, 2014.
[36] L. Jing, Y. Ye, X. Yang, and Y. Tian, "3D Convolutional Neural Network with multi-model framework for action recognition," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2017.
[37] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2007.
[38] Y. Han, P. Zhang, T. Zhuo, W. Huang, and Y. Zhang, "Going deeper with two-stream ConvNets for action recognition in video surveillance," *Pattern Recognition Letters*, vol. 107, pp. 83–90, 2018.
[39] K. Liu, W. Liu, C. Gan, M. Tan, and H. Ma, "T-C3D: Temporal convolutional 3D network for real-time action recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.
[40] F. Patrona, I. Mademlis, and I. Pitas, "An overview of hand gesture languages for autonomous uav handling," in *Proceedings of the AIRPHARO Workshop on Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)*, 2021.
[41] C. Papaioannidis, D. Makrygiannis, I. Mademlis, and I. Pitas, "Learning fast and robust gesture recognition," in *Proceedings of the EURASIP European Signal Processing Conference (EUSIPCO)*, IEEE, 2021.
[42] C. Papaioannidis, I. Mademlis, and I. Pitas, "Fast single-person 2D human pose estimation using multi-task convolutional neural networks," in *(submitted)*, 2021.
[43] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2014.
[44] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single-shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2016.
[45] S. Ioffe and C. Szegedy, "Batch Normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, 2015.
[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.