

UNIVERSITY OF VERONA
DEPARTMENT OF COMPUTER SCIENCE

GRADUATE SCHOOL OF NATURAL SCIENCES AND ENGINEERING
DOCTORAL PROGRAM IN COMPUTER SCIENCE
CYCLE XXXVII°

GESTURAL INTERACTION IN EXTENDED REALITY

Coordinator: Prof. Ferdinando Cicalese

Tutor: Prof. Andrea Giachetti

Co-Tutor: Prof. Umberto Castellani

Doctoral Student: Marco Emporio

S.S.D. INFO-01/A

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License, Italy. To read a copy of the licence, visit the web page:
<http://creativecommons.org/licenses/by-nc-nd/3.0/>

-  **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
-  **NonCommercial** — You may not use the material for commercial purposes.
-  **NoDerivatives** — If you remix, transform, or build upon the material, you may not distribute the modified material.

Algorithms and data structures for indexing, mining, and coding of sequential data — MASSIMILIANO ROSSI
or
Combinatorics and Algorithmics of Sequential Data — Prefix normal words, colored strings, and space-aware
encodings — MASSIMILIANO ROSSI
PhD Thesis
Verona, March 10, 2025
ISBN <ISBN>

Abstract

This thesis focuses on gesture-based interaction, an important topic of Human-Computer Interaction, within the context of extended reality environments. Recent advancements in sensing technologies, computational power, and extended reality platforms have significantly increased the interest in gesture recognition research. Despite this progress, several challenges remain, including the continuous and accurate detection of gestures and the development of systems capable of real-time adaptation and robust performance in diverse contexts. To address these issues, this work provides a comprehensive review of the state-of-the-art in continuous gesture recognition detection, analyzing the methods proposed, identifying their strengths and limitations and discussing the problems related to their evaluation, providing an analysis of all the publicly available benchmarks.

The thesis also introduces novel methods for skeleton-based continuous hand gesture detection and a novel benchmark for evaluating the task, including heterogeneous gestures designed to interact with extended reality applications. The novel methods integrate multi-task learning and enhanced feature extraction. The results obtained on our benchmark show that our methods outperform state-of-the-art techniques in terms of detection accuracy and number of false positives. These contributions provide a bridge between theoretical advancements and practical implementations, highlighting the transformative potential of gesture recognition in real-world scenarios. Additionally, data augmentation techniques were introduced to enhance the robustness and adaptability of recognition systems. The gesture recognition methods developed have been demonstrated practically in industrial extended reality environments, showing how gesture-based interaction can improve user experience, operational efficiency, and workflow design. The outcomes of this work contribute to the understanding and development of gesture-based interaction, setting a foundation for future research. Key directions for further exploration include addressing scalability challenges, improving adaptability to diverse users and environments, and deepening the integration of human-centric factors to fully realize the potential of gesture recognition in extended reality and beyond.

Abstract (Italian)

Questa tesi si concentra sull'interazione basata su gesti, un tema importante nell'ambito dell'interazione uomo-computer, nel contesto degli ambienti di realtà estesa. I recenti progressi nelle tecnologie di rilevamento, nella potenza di calcolo e nelle piattaforme di realtà estesa hanno significativamente aumentato l'interesse nella ricerca sul riconoscimento dei gesti. Nonostante questi progressi, permangono diverse sfide, tra cui il rilevamento continuo e accurato dei gesti e lo sviluppo di sistemi capaci di adattarsi in tempo reale e garantire prestazioni robuste in contesti diversi. Per affrontare queste problematiche, questo lavoro offre una revisione completa dello stato dell'arte nel riconoscimento continuo dei gesti, analizzando i metodi proposti, identificandone i punti di forza e le limitazioni, e discutendo i problemi legati alla loro valutazione, fornendo un'analisi di tutti i benchmark pubblicamente disponibili.

La tesi introduce inoltre nuovi metodi per il riconoscimento continuo di gesti manuali basati su scheletri e un nuovo benchmark per valutare questo compito, includendo gesti eterogenei progettati per interagire con applicazioni di realtà estesa. I nuovi metodi integrano l'apprendimento multi-task e un'estrazione delle caratteristiche migliorata. I risultati ottenuti sul nostro benchmark dimostrano che i nostri metodi superano le tecniche all'avanguardia in termini di accuratezza del rilevamento e riduzione dei falsi positivi. Questi contributi rappresentano un ponte tra i progressi teorici e le implementazioni pratiche, evidenziando il potenziale trasformativo del riconoscimento dei gesti in scenari reali. Inoltre, sono state introdotte tecniche di data augmentation per migliorare la robustezza e l'adattabilità dei sistemi di riconoscimento. I metodi sviluppati per il riconoscimento dei gesti sono stati dimostrati praticamente in ambienti industriali di realtà estesa, mostrando come l'interazione basata su gesti possa migliorare l'esperienza dell'utente, l'efficienza operativa e la progettazione dei flussi di lavoro. I risultati di questo lavoro contribuiscono alla comprensione e allo sviluppo dell'interazione basata su gesti, ponendo le basi per ricerche future. Tra le principali direzioni di esplorazione vi sono l'affrontare le sfide di scalabilità, migliorare l'adattabilità a utenti e ambienti diversi, e approfondire l'integrazione di fattori centrati sull'uomo per realizzare pienamente il potenziale del riconoscimento dei gesti nella realtà estesa e oltre.

Acknowledgements

This study was partially carried out within the PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043).

This research is under the National Recovery and Resilience Plan (NRRP), Mission 4, Component 2 Investment 1.4, funded by the European Union – NextGeneration EU, project “Interconnected Nord-Est Innovation Ecosystem”.

Contents

List of Figures	IV
List of Tables	VI
Publications	VII
1 Introduction	1
2 State of the art in Continuous Gesture Recognition	5
2.1 Gestures Recognition surveys	5
2.2 Input modalities and data stream encoding for Gestures	7
2.2.1 Acquisition devices	7
2.2.2 Input stream encoding	8
2.3 Hand gestures' taxonomy and gesture dictionaries	9
2.4 Benchmarks Evaluation metrics	10
2.5 Existing benchmarks and results	11
2.5.1 Literature search methodology	12
2.5.2 Benchmarks	13
2.5.3 Continuous gesture recognition methods and best results on the surveyed benchmarks	22
2.6 Data Augmentation	30
2.7 Discussion	32
2.7.1 Characteristics of the benchmarks	32
2.7.2 Performance of gesture spotting methods	35
3 Benchmarking Continuous Hand Gesture Recognition	39
3.1 SHREC 2022	39
3.2 Novel dataset, task and evaluation	40
3.2.1 Task and evaluation	42
3.3 Participants and proposed methods	42
3.3.1 Group 1: Two-stage ST-GCN (2ST-GCN)	42
3.3.2 Group 2: Causal TCN	45
3.3.3 Group 3: Transformer Network + Finite State Machine (TN-FSM) ..	46
3.3.4 Our baseline: Stronger	49
3.4 Results	49
3.5 Contest's outcomes	52
3.6 Post Contest improvements	53
4 Continuous Gesture Recognition Methods	55
4.1 STRONGER: Simple TRajjectory-based ONline GESture Recognizer	55
4.2 Proposed approach	56
4.2.1 System architecture	56
4.2.2 Gesture classification network	57

4.2.3	Online recognition and the SHREC'21 dataset	59
4.2.4	Mixed reality interactive mockup	61
4.3	Results	61
4.3.1	Implementation details	61
4.3.2	Offline evaluation of the gesture classifier	61
4.3.3	Online evaluation	62
4.3.4	System prototype	64
4.4	OO-dMVM: A Deep Multi-view Multi-task Classification Framework for Real-time 3D Hand Gesture Classification and Segmentation	66
4.5	Our pipeline	68
4.5.1	The views	68
4.5.2	The tasks	68
4.5.3	Model training	69
4.5.4	Model inference	69
4.6	Experiments	70
4.6.1	SHREC'22 results	70
4.6.2	SHREC'19 results	72
4.6.3	Ablation studies	73
4.7	SHREC 2024	74
4.8	Discussion	74
5	Data Augmentation	77
5.1	Method	77
5.1.1	Sliding window detector	78
5.2	Augmentation details	78
5.2.1	Global trajectory augmentations	78
5.2.2	Hand orientation augmentation	80
5.2.3	Articulation Noise	80
5.2.4	Dataset Augmentation Strategies	81
5.3	Experiments	81
5.3.1	Establishing a Baseline	81
5.3.2	Generation of the augmented datasets, fine-tuning of the augmentation ranges and online detection results	82
5.3.3	SHREC'17 comparison	84
5.4	Discussion and Future Work	84
6	Practical Applications of Gesture Recognition in Industrial XR Environments	87
6.1	Integration of Extended Reality with a Cyber-Physical Factory Environment and its Digital Twins	88
6.2	State-of-the-Art Surveys on eXtended Reality Integration with Digital Twins and Cyber-Physical Factories	88
6.3	The ICE lab of the University of Verona: a complete Industry 4.0/5.0 technology demonstrator	89
6.4	The laboratory graphical twin	92
6.5	Virtual Reality application	94
6.5.1	Interaction Design	94
6.5.2	VR Demo: virtual visit with real-time machines' status visualization	94
6.6	Augmented Reality application	94
6.6.1	Interaction Design	95
6.6.2	Demo tasks for the AR app	96
6.7	Discussion	97
7	Conclusion	99
	References	101

A	Evaluation metrics	113
A.1	Frame-Based classification evaluation	113
A.1.1	Accuracy.	113
A.1.2	Precision and Recall.	113
A.1.3	F-Score	114
A.1.4	Area Under the curve (AUC)	114
A.1.5	Jaccard Index (JI)	115
A.1.6	Levenshtein Distance (LD)	115
A.2	Event-Based classifier evaluation	116
A.2.1	Detection rate (DR)/ True Positive Rate (TPR)/Event- based recall	116
A.2.2	False Positive Rate (FPR)	116
A.2.3	Levenshtein accuracy (LA)	117
A.2.4	ATSE, ATSR, Perf	117
A.2.5	Temporal delay	118
A.2.6	Normalized time to detection	118
A.3	Efficiency measures	118
A.3.1	Classification time.	118
A.3.2	Hardware resource demand	119
B	Classifiers	121
	Appendices	113

List of Figures

2.1	Criteria used to classify the existing benchmarks for continuous gesture spotting/recognition	12
2.2	Recording setup for the ChAirGest Benchmark. Left: image captured by the Kinect. Right: external view of the same subject. Wearable IMUs are visible on his right arm [1].	14
2.3	Example gestures from the Montalbano dataset. They are performed before a Microsoft Kinect while speaking Italian [2].	15
2.4	Left: depth maps from the ConGD dataset. Right: the corresponding RGB images. From http://www.cbsr.ia.ac.cn/users/jwan/database/congd.html	15
2.5	Top left: recording setup for the NVGesture dataset. The subject uses his left hand to drive and performs the gestures with his right hand only. NVGesture includes multi-modal acquisitions. Data are captured with a SoftKinetic DS325 recording front-view RGB-D videos (A), and with a DUO 3D stereo-IR camera mounted above the user (B). From https://www.v7labs.com/open-datasets/nvgesture	16
2.6	Example RGB (top row) and depth images (bottom row) from the EgoGesture dataset. From http://www.nlpr.ia.ac.cn/iva/yfzhang/datasets/egogesture.html	17
2.7	The dictionary of the SHREC'19 benchmark features 5 gesture types characterized by simple 2D paths (red lines) hidden in longer 3D non-gesture trajectories (blue). From [3].	17
2.8	The different data types included in the IPN Hand distribution. From left, RGB video frames, optical flow maps and segmentation masks. From https://gibranbenitez.github.io/IPN_Hand/	18
2.9	Sample RGB and thermal frames from the front sensor cluster (top row) and top sensor cluster (bottom row) included in the Mlgesture dataset (from [4]).	19
2.10	The gesture dictionary of the LD-ConGR dataset (from [5]).	19
2.11	The gesture dictionary of SHREC '22 include static (top row), dynamic coarse (middle row) and dynamic fine and periodic gestures (bottom row). From [6].	20
2.12	The different ways to perform gestures in the OHDG (and DHG 14/28) dataset: using (a) one finger or (b) the whole hand (from http://www-rech.telecom-lille.fr/DHGdataset/).	21
2.13	Examples of gestures sequences included in the ZJUGesture dataset, captured by different mobile devices (from [7]).	22
2.14	(a)	32
2.15	(b)	32
2.16	(a) Data types used as input for the recognizers in the different benchmarks. Blue bars indicate video streams, and orange bars key point data streams. Most of the benchmarks are based on videos. (b) Most of the benchmarks feature data acquired from an exocentric view.	32

2.17 (a)	34
2.18 (b)	34
2.19 (a) Most of the benchmarks feature a relatively small number of gestures classified in the dictionary (< 15), even if typically of heterogeneous types. (b) The number of subjects performing gestures is extremely variable.	34
2.20 Occurrence of the different evaluation metrics used to assess the method's performances in the 12 benchmarks reviewed. JI=Jaccard index, FPR=false positive ratio, AUC=area under receiver-operator characteristic curve, NTtd=normalized time-to-detect, LA=Levenshtein Accuracy, mAP=mean area precision. Orange bars indicate frame-based metrics, blue bars indicate event-based metrics, and green bars are efficiency-based metrics.	35
2.21 Occurrence of different classifier types in the 23 methods performed applied on the proposed benchmarks and reviewed in our work.	36
3.1 Gesture acquisition: the subject hears vocal commands suggesting pre-defined sequences of non-significant movements and gestures.	40
3.2 The 16 gestures in SHREC'22 vocabulary: static (top row), dynamic coarse (middle row), dynamic fine (bottom left) and periodic (bottom right).....	40
3.3 <i>The workflow of our two-stage hand gesture detection architecture.</i>	43
3.4 <i>Spatial-temporal graph neural networks for hand gesture recognition.</i>	43
3.5 <i>The modified architecture of ST-GCN with both regression and classification heads</i>	44
3.6 Segments joining given anatomical landmarks. Angles between all pairs as well as with the fixed global axis (arrow right) serve as input features for gesture detection.	45
3.7 Detection rate	49
3.8 Jaccard Index	49
3.9 False Positives' score	49
3.10 Performance metrics per class, averaged on all the test sequences.	49
3.11 Detection rate	50
3.12 False positives	50
3.13 Scores by gesture type	50
3.14 The Detection Rate of each method as a function of the minimum overlap ratio between the gestures' windows, for the detected gesture to be marked as correct detection.	51
3.15 The delay in frames of the last frame used by each method from the start and end marks of Ground Truth gestures, shown by gesture type. Chart bars are centered on the last frame used.....	52
4.1 Snapshots showing the online display viewed by the subject wearing a Hololens 2 running the online recognizer demo.	56
4.2 The client-server application: the Hololens2 app sends the joints' stream to the PC and receives the classification results to be displayed.	57
4.3 Our classifier adds to the DDNet architecture two branches with 1D convolutions processing palm orientation data and joint pairs' directions.	58
4.4 The novel feature are the time evolution of the palm orientation (left) and of the directions defined by selected pairs of joints (JPD, right)	59
4.5 Gesture dictionary of SHREC'21 (from [8])	59
4.6 Top: confusion matrix of DD-net on SHREC 2021 classes. Bottom: confusion matrix of the modified network with the additional features PO and JPD. ...	62
4.7 Both the features added to the original DDNet contribute to improve the SHREC '21 benchmark scores. Top: Jaccard Index per class for the original DDNet, DDNet+PO, DDNet+JPD and both PO+JPD. Middle row: Bottom: Per class false positive ratio for the original DDNet, DDNet+PO, DDNet+JPD and both PO+JPD.	65

4.8	On top, the 3D hand gesture real-time classification and segmentation scenario: in red are non-gestures, in green the segmented gesture. The task is difficult: below, we show samples from SHREC'19 (a), where it is possible to see long gestures with a static pose of the hand and the correct start of the gesture cannot be easily detected; in (b) a sample from SHREC'22, where gestures express more pose dynamics.	66
4.9	The pipeline of our framework, is composed of three steps: 1) Multi-view feature extraction, 2) OO-dMVM per-frame inference, 3) Post-processing. In red is highlighted the path used during the testing time, after the OO-dMVM model has been trained.	69
4.10	Detection rate per gesture class for SHREC'22. Points are connected with a line for clarity. On the x-axis, a colored dot indicates, for each class, the method that achieved the best detection rate. Only the best 5 methods by DR are shown for clarity.	71
4.11	Jaccard index for SHREC'22 as a function of the minimum overlap ratio. For clarity, we show only the best 5 methods.	72
4.12	Stacked False Positives for SHREC'22 per model. The gestures are divided into 4 types (colors). The FPs for each type are then summed to show the total value. Lower is better.	72
5.1	Trajectory noise.	79
5.2	Trajectory Scaling.	79
5.3	Trajectory Orientation.	79
5.4	Trajectory Resampling.	79
5.5	In this figure, the four trajectories augmentation types are visualized. They represent Figure 5.1 Position noise, Figure 5.2 Scaling, Figure 5.3 Orientation, Figure 5.4 Resampling.	79
5.6	Hand pose orientation.	80
5.7	Articulation noise.	80
5.8	Figure 5.6 Hand orientation noise is added by rotating the skeleton around the wrist node. Figure 5.7 Fingers'articulation noise is added by summing random components to the PDM coefficients.	80
5.9	The central column of this grid represents the original pose (PINCH). The rows show the variations along the first three principal components of the PDM built for the PINCH gesture, labeled as PC_1 , PC_2 , and PC_3 . The columns represent the effect of perturbations at different standard deviation offsets from the mean, specifically -2σ , -1σ , 0σ , 1σ , and 2σ	81
6.1	(a)	90
6.2	(b)	90
6.3	Two views of the ICE laboratory: (a) shows the milling machine close-by with the feeding arm in the forefront, the polyjet and stereolithographic printers in the middle, the autonomous ground vehicles and the vertical automated warehouse at the back; (b) shows the mini-pallet conveyor line on the left and the robotic arms in the middle.	90
6.4	Functional schema of the ICE lab. The AR and XR applications communicate both with the physical devices and the industrial digital twin, enabling augmented interaction, monitoring and control.	91
6.5	The graphical twin of the ICE lab includes all the laboratory assets modeled as triangle meshes at different levels of detail to support rendering on low-end devices adapting the quality to the hardware resources. The full-resolution model is currently composed of approximately 5.1M triangles. .	92
6.6	(a)	93
6.7	(b)	93
6.8	(c)	93
6.9	(d)	93

6.10	The VR application provides a realistic visualization of the simulated environment on the Head-Mounted display, with physically-based illumination effects and high-resolution details (a,b). The application allows to navigate inside the ICE lab using instant teleport by throwing a ray with a gesture and clicking when the ray intersects the floor in the desired position (c), and the interactive visualization of both offline description and real-time information on the single machines, as the active tray in the vertical warehouse (d).	93
6.11	(a)	95
6.12	(b)	95
6.13	(c)	95
6.14	(d)	95
6.15	Example headset views of the basic AR application of the ICE lab. The user can see augmentations (a) and/or semitransparent models of the selected (b) devices. In the examples in the bottom row, the operator checks the status of a segment of the conveyor (c). In case of alarm, a red arrow guides the operator to the critical point (d).	95
6.16	The 16 static and dynamic gestures modeled in [6] that are used for touchless control.	96
6.17	(a)	97
6.18	(b)	97
6.19	(c)	97
6.20	(d)	97
6.21	Demo of the gesture-based production recipes' management. The boxes in the figure show the user view. (a) The worker opens the menu of the active tasks with a menu gesture. (b) They start the transport line operation with a "knob" gesture. (c) With a specific gesture, they call pallet '1' in the local bay for repair. (d) After completing the task, they can see the updated list of available actions.	97
6.22	(a)	98
6.23	(b)	98
6.24	(c)	98
6.25	(d)	98
6.26	Remote inspection with extended reality. (a) The headset scans an image target. The pane shows the user's view. (b) Subsequent user view where the device's description is displayed together with the 3D representation of the same machine. (c) The user performs a command with a gesture (the pane shows the user's view). (d) Subsequent user view where the status of the machine, updated in real-time, is displayed.	98

List of Tables

2.1	Characteristics of the gesture dictionaries used in the reviewed benchmarks: types of gestures (S=static, DC=dynamic coarse, DF=dynamic fine, DP=dynamic periodic) and invariance (P=position, O=hand orientation) ...	13
2.2	Characteristics of the acquisition setup for the datasets included in the reviewed benchmarks.	13
2.3	Characteristics of the sets of sequences/gestures included in each benchmark. Figures in brackets specify the training/test splitting of sequences and gesture samples. Gps means Gestures per sequence (V= Variable)	14
2.4	Performances of the best method on the ChAirGest benchmark	23
2.5	Performances of the best methods on the Montalbano Track 3 benchmark ...	24
2.6	Performances of the best methods on the ConGD benchmark	24
2.7	Performances of the best methods on the NVGesture benchmark. (*) metric from [9]	25
2.8	Performances of the best methods on the EgoGesture benchmark. (*) metric from [9]	26
2.9	Performances of the best methods tested on the SHREC'19 benchmark. (*) metric from [10]. CD: percentage of gestures detected in the correct time window and correctly classified, ML: gestures detected in the correct time window and mislabeled, FP: percentage of false positives encountered before the true gesture, M: percentage of missed detections, T1 Err.: average error in estimating the initial mark T1, T2-e: average distance, in time, of the detection frame from the end mark, T2-s: average distance of the detection frame from the start mark.	27
2.10	Performances of the best methods tested on the IPN Hand benchmark.....	27
2.11	Performances of the best methods on the Mlgesture benchmark for the low-latency continuous detection task	28
2.12	Performances of the best method tested on the LD-ConGR benchmark.	29
2.13	Performances of the best methods on the ODHG benchmark	30
2.14	Performances of the best methods on the ZJUGesture benchmark	30
3.1	Minimum, maximum and average lengths of the gestures (background colors indicate gestures' types.	41
3.2	The table shows the summary of the results for each group method and all the variants proposed averaged on all the gestures. The metrics displayed are the detection rate, the false positives' scores, the Jaccard Index and the delay (in frames) between the start of a gesture marked by the method and the last frame reached at the moment of the marking.	50
3.3	Average delays in frames from ground truth start and end of annotated gestures derived by algorithm design (excluding computation time).....	51

3.4	Performances of the best methods tested on the SHREC'22 benchmark. DR: Detection Rate, FPR: False Positive's Ratio, JI: Jaccard Index, D-End/D-Start: average delays in frames from the ground truth end/start of the annotated gestures (depending on the algorithms' design).	53
4.1	Duration of the 18 gestures of the SHREC'21 dataset. There is a huge variability among the gesture classes.	60
4.2	The additional features added to DDNet strongly improve the classification accuracy on SHREC 2021 gestures.	62
4.3	Results on SHREC 2017 (Using 3D skeletons only)	63
4.4	Average scores for the SHREC'21 benchmark compared with contest participants. It should be noted that the method of Group 4 includes a gesture segmentation module to reduce false positives.	63
4.5	Ablation study showing that the novel 1D features strongly increase the detection rate while the learning of the Window Threshold Array is effective in reducing the false positives rate.	63
4.6	Per-class evaluation of the scores of the SHREC2021 contest obtained with the current STRONGER recognizer architecture.	64
4.7	Classification results of SHREC'22 benchmark. The metrics are the detection rate (DR), false positives scores (FP), the Jaccard Index (JI), the delay in frames, and per-frame processing time. In brackets are the standard deviations. Columns SV/MV refers to the single-view or multi-view learning approach, while ST/MT refers to single-task or multi-task learning approach.	71
4.8	The metrics are the detection rate , false positives scores , and inference time. Parameters with (*) comes from [3] and may not be accurate. Dash (–) values were not reported in the original benchmark. Standard deviation is reported between brackets.	73
4.9	Classification results on SHREC'22 with different Multi-task heads. Standard deviation is reported between brackets.	73
4.10	Classification results on SHREC'22 with different Regression Heads training procedures. Standard deviation is reported between brackets.	74
4.11	Comparison of total accuracy, number of parameters, and training time in seconds for a singular fold.	74
5.1	Ranges tested in the fine-tuning for each augmentation method used to synthesize gesture and non-gesture.	82
5.2	Continuous detection results (Detection Rate (DR), False Positive Score (FPS), and Jaccard Index (JI)) obtained on the SHREC'22 benchmark with the classifier presented in [10] using different training sets. Top block: scores obtained training with the examples of a single subject (baseline). Middle block: results obtained augmenting the training gestures with all proposed methods (in the optimal parameters range shown in the second column). Bottom: results obtained augmenting the non-gesture segments with all proposed methods (in the optimal parameters range shown in the second column).	83
5.3	This table reports the online evaluation scores (DR = Detection Rate, FPS = False Positive Score, and JI = Jaccard Index) of the Baseline, the union of all augmented gestures (AAD), the union of all augmented non-gestures (NGAD), and the combined union of both (AAD + NGAD).	83
5.4	This table reports the online evaluation scores (DR = Detection Rate, FPS = False Positives Score, and JI = Jaccard Index) of the Baseline of Subject 2 and Subject 3 from the training set of SHREC'22 and the augmented versions obtained using AAD + NGAD.	84
5.5	Classification accuracy of DDNet on SHREC'17 dataset using the full trainset and a single-user setup, with and without augmentation.	84

Publications

Throughout this thesis, I will draw upon and build upon the themes explored in my previous works, which serve as the fundamental basis for the analyses and methodologies presented. In particular, many of the concepts, techniques, and results discussed are inspired by and overlap with my previously published research, enriched and reworked to provide a more comprehensive and cohesive view of the challenges addressed during the PhD. Here is the list of the published papers where I was one of the authors:

- **Enhancing Safety and Privacy in Industry 4.0: The ICE Laboratory Case Study [11]**
 - **Publisher:** IEEE.
 - **Year of Publication:** 2024.
 - **Type:** Journal article.
 - **Journal:** IEEE Access.
 - **DOI:** <https://doi.org/10.1109/ACCESS.2024.3479411>
- **ICELab Demo: an industrial digital-twin and simulator in VR [12]**
 - **Publisher:** ACM (Association for Computing Machinery).
 - **Year of Publication:** 2024.
 - **Type:** Conference paper.
 - **Conference:** VRST 2024 (Virtual Reality Software and Technology), Demo section.
 - **DOI:** <https://dx.doi.org/10.1145/3641825.3689946>
- **SHREC 2024: Recognition of dynamic hand motions molding clay [13]**
 - **Publisher:** Elsevier.
 - **Year of Publication:** 2024.
 - **Type:** Journal article.
 - **Journal:** Computers & Graphics.
 - **DOI:** <https://doi.org/10.1016/j.cag.2024.104012>
- **Integration of Extended Reality with a Cyber-Physical Factory Environment and its Digital Twins [14]**
 - **Publisher:** ACM (Association for Computing Machinery).
 - **Year of Publication:** 2024.
 - **Type:** Journal article.
 - **Journal:** Proceedings of the ACM on Human-Computer Interaction.
 - **DOI:** <https://doi.org/10.1145/3660246>
- **Demo: Gesture-based interaction with the Hololens 2 [15]**
 - **Publisher:** ACM (Association for Computing Machinery).
 - **Year of Publication:** 2023.
 - **Type:** Conference paper.
 - **Conference:** CHIItaly '23: Proceedings of the 15th Biannual Conference of the Italian SIGCHI Chapter
 - **DOI:** <https://doi.org/10.1145/3605390.3610827>
- **OO-dMVMt: A Deep Multi-view Multi-task Classification Framework for Real-time 3D Hand Gesture Classification and Segmentation [10]**

- **Publisher:** ACM (Association for Computing Machinery).
- **Year of Publication:** 2023.
- **Type:** Conference paper.
- **Conference:** Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)
- **DOI:** <https://doi.org/10.48550/arXiv.2304.05956>
- **SHREC 2022 track on online detection of heterogeneous gestures [6]**
 - **Publisher:** Elsevier.
 - **Year of Publication:** 2022.
 - **Type:** Journal article.
 - **Journal:** Computers & Graphics.
 - **DOI:** <https://doi.org/10.1016/j.cag.2022.07.015>
- **STRONGER: Simple trajectory-based online gesture recognizer [16]**
 - **Publisher:** The Eurographics Association
 - **Year of Publication:** 2021.
 - **Type:** Conference paper.
 - **Conference:** Smart Tools and Apps for Graphics (STAG), Eurographics Italian Chapter.
 - **DOI:** <https://doi.org/10.2312/stag.20211481>

Finally, I want to point out that two other papers are submitted right now that must be reviewed: a gesture augmentation paper explained in [Chapter 5](#) and a survey on online gesture benchmarks [\[17\]](#) that inspired [Chapter 2](#) (see also <https://doi.org/10.2139/ssrn.4797435>)

Introduction

Human-Computer Interaction (HCI) studies how designers create interactive computer systems and how end users interact with those systems. It focuses on understanding the relationship between humans and computers, aiming to enhance how they communicate and work together. With the advancement of computing devices and related technologies, HCI has become an integral part of daily life [18], encompassing activities like work, shopping, and communication. While graphical user interfaces that rely on physical input devices (e.g., mouse and keyboard) remain the most common, new interaction methods continue to emerge. The next generation of HCI should be human-centered, deviceless, and intelligent, requiring new interface paradigms, innovative user interfaces, and advanced interaction methods.

Human interaction involves multiple aspects, including language, body movements, and emotional expression. One such aspect is hand gestures, which are often used during communication to draw attention and convey messages. People naturally use their hands to perform actions like grasping, moving, and rotating objects in everyday life. Gestures serve as a way for individuals to express their intentions. They are the most used form of non-verbal communication, so it is reasonable to rely on them to build intuitive and natural computer interfaces [19].

Given the natural use of gestures in everyday human interactions, gestural interaction is expected to become a popular option to interact in Extended Reality (XR) [20], and is already present in a variety of applications, such as Virtual Reality (VR) and Mixed Reality (MR) [21], industrial environment [22], automotive interfaces [23] and multimedia [24, 25] control, public kiosks [26], making it possible to interact with computers naturally without the need for input devices. The growing adoption of gestural interaction is made possible by advances in tracking technologies. Recent headsets now feature accurate finger-tracking capabilities [27, 28], while reliable software tools can perform real-time tracking of body limbs or fingers from RGB (D) streams [29]. Consequently, capturing a continuous stream of hand skeleton poses enables the recognition of diverse gesture types, leading to the development of skeleton-based continuous hand gesture recognition methods in recent research [30–34].

Another reason for the increasing popularity of gestural interfaces across diverse domains is driven by advancements in gesture recognition technologies. Gesture recognition is a technique that translates human gestures, such as hand or body movements, into labels that can be used as input commands. For instance, in gesture recognition systems, movements like raising a hand, swiping, or pointing can be used to scroll a webpage, navigate a menu, or control a virtual object. It relies on sensors, cameras, or other input systems to track human movements. Gesture recognition faces several challenges, particularly when it comes to capturing the full complexity and variability of human movements. One of the main challenges of 3D gesture recognition is to create approaches that are able to handle the large variety of gestures present in a heterogeneous dataset. The recognition is also challenging because movements in 3D space involve varied paths, speeds, and directions, making accurate tracking and interpretation tricky for each gesture. Furthermore, gestures can differ widely between individuals in style and physical traits [35], and even vary from one instance to another by the same person due to fatigue, time pressures, or body position.

In XR applications, gestures must be recognized in a *continuous* stream rather than as isolated instances, which requires temporal segmentation and characterization of non-gesture movements [36, 37]. This recognition can be achieved through a sliding window approach or separate segmentation modules, though this is complicated when dealing with diverse gesture types. The sliding window approach continuously classifies fixed or variable-sized gesture segments using methods trained to distinguish gesture subparts from "non-gesture" windows [38–40]. However, the literature reveals that most algorithms are not designed to detect and recognize gestures in a continuous data stream; instead, they mainly address the classification of already segmented gestures. This focus does not take into account the more complex challenge of gesture recognition in continuous time, which is essential for real-world applications where gestures must be detected and classified in real-time with minimal delays and errors. Many widely used benchmarks for evaluating gesture recognition methods offer datasets of segmented gestures for training and testing algorithms, assessing classification accuracy. Examples include VIVA [41], DHG 14-28 [42], SHREC'17 [43], and Jester [44]. Notably, the segmented classification approach remains significantly more popular even for benchmarks like EgoGesture [36] that provide both continuous and segmented gesture recordings. This fact is also surprising because when an algorithm has to recognize a gesture in a practical scenario, it should be able to distinguish a gesture performed among nonsignificant hand movements in real-time. For this reason, gesture recognition methods to solve the so-called "continuous" or "online" gesture recognition problem require event detection in the input stream, segmenting and classifying the gestures correctly, providing feedback with low latency, and minimizing recognition errors to avoid activating unwanted actions. In some applications—such as XR gaming or controlling robots in hazardous environments—recognition delay can be a significant issue. Offline benchmarks are not suited for evaluating these delays, as they do not allow comparison between the actual gesture start/end times in the ground truth and the algorithm's detected times. Evaluating continuous gesture recognition is complex, and all these aspects should be considered, not just classification accuracy. Unfortunately, as discussed in Chapter 2, most existing surveys do not focus on continuous gesture recognition. As a result, they are not helpful for interaction designers looking to develop gesture-based user interfaces. In any context, gestures have to be detected and classified within a continuous stream of hand movements that can be captured by cameras or other sensors. Affordable sensors/APIs are available to enhance different applications with this data. MR headsets like Meta Quest 3 or Hololens 2 are equipped with sensors that are able to accurately track the hands' joints, capturing finger poses at high frame rates.

While gesture recognition has been extensively studied, most existing approaches focus on classifying pre-segmented gestures rather than recognizing them in continuous streams. This limitation poses a challenge for real-world applications where gestures must be detected and classified in real-time with minimal errors and delays. Given this research gap, developing an additional benchmark that accounts for the specific challenges of continuous recognition was necessary. Furthermore, existing methods from the literature had to be adapted to achieve reliable results across key performance metrics essential for continuous gesture recognition. This thesis addresses these challenges by proposing two novel approaches designed to improve the accuracy and robustness of gesture recognition in continuous settings.

In this work, it will be presented and discussed suitable approaches to build 3D natural interfaces for XR applications that work in continuous time and use reliable recognition methods (these approaches must commit a *low amount of errors and have a real-time latency*) of mid-air heterogeneous hand gestures as a fundamental input.

To better understand these challenges and the solutions proposed, this thesis presents a literature review Chapter 2 that provides a comprehensive review of the state-of-the-art research in gesture recognition, its associated technologies, and their integration into practical applications. Following the review, the Benchmarks Chapter 3 discusses the creation and evaluation of the SHREC22 benchmark, which serves as an evaluation platform for gesture recognition algorithms. Chapter 4 introduces two approaches, STRONGER and OO-dMVM, developed to address the limitations identified in existing methods. Finally,

the Data Augmentation [Chapter 5](#) outlines various strategies to generate synthetic data, and the Demo and Applications [Chapter 6](#) showcase real-world use cases, such as a Cyber-Physical Factory, illustrating the potential of these systems in industrial settings. The paper concludes by summarizing the key findings and discussing future research directions [Chapter 7](#).

State of the art in Continuous Gesture Recognition

In recent years, significant efforts have been dedicated to the field of gesture recognition, driven by the increasing demand for natural and intuitive human-computer interaction. While substantial progress has been made, most of the existing research primarily focuses on the classification of segmented gestures, where each gesture is isolated and identified individually. However, there remains a critical need to advance recognition techniques that operate effectively on continuous gesture streams, where gestures are performed without clear separations.

This chapter is adapted from [17] and provides a comprehensive overview of the current literature on gesture recognition. The review begins with an exploration of gesture recognition surveys (Section 2.1), summarizing existing efforts to map the field and identify trends in methodology and application. In Section 2.2 then are discussed input modalities and data stream encoding for gestures, addressing the diverse ways gesture data can be captured and represented. Next, the taxonomy of hand gestures and gesture dictionaries is presented in Section 2.3. A structured classification of gestures helps standardize their definition, enabling consistent use across studies. Gesture dictionaries play a pivotal role in training and evaluating recognition systems, particularly when dealing with heterogeneous and application-specific gestures. The discussion proceeds to benchmarks and evaluation metrics (Section 2.4), focusing on the datasets and metrics used to assess gesture recognition systems. Gesture online benchmarks are then analyzed in detail in Section 2.5, emphasizing their importance for real-time and continuous gesture recognition scenarios. In addition, the role of data augmentation in gesture recognition is explored in Section 2.6. Augmentation techniques are crucial for enhancing the robustness and generalization capabilities of recognition systems, particularly when training data is limited or imbalanced. Finally, the chapter concludes with a discussion on the integration of XR technologies with cyber-physical factories and digital twins (Section 6.2), showcasing how gesture recognition is increasingly being applied in complex, real-world environments. This section highlights the synergies between XR technologies, digital twins, and gesture recognition, underscoring their transformative potential in Industry 4.0 and 5.0 contexts.

2.1 Gestures Recognition surveys

Several surveys on hand gesture recognition analyze existing literature, offering varying levels of detail and focus. These surveys address aspects such as acquisition techniques, segmentation, gesture taxonomies, and application domains, reflecting the diverse challenges in the field.

The most cited survey is by Mitra et al. [45], which covers body and face recognition methods, with a limited focus on hand gestures. While it offers a clear definition of the problem, it does not discuss gesture taxonomies or benchmarks. Furthermore, the recognition methods discussed are now largely outdated, as the paper was published in 2007.

A noteworthy yet old survey on 3D gesture recognition was conducted by LaViola [46]. This work reviewed state-of-the-art sensor technology for capturing hand and body move-

ments up to 2013 and presented classification methods and future research directions. While it is comprehensive of many research challenges, it lacks discussions on benchmarking continuous gesture recognition, and the methods reviewed are no longer used in the literature.

Rautaray et al. [47] provided a survey focused on vision-based gesture recognition, highlighting various application domains and related platforms and frameworks. They also reviewed existing commercial products for gesture tracking but did not explore validation processes or benchmarks in depth.

Anthes et al. [48] offered an overview of technologies developed for virtual reality, including precise hand joint tracking. Their work introduced the impact of input-output devices on gesture recognition and provided insights into gesture taxonomies. Although it mentioned the issue of 3D gesture datasets, it did not address continuous gesture recognition evaluation or relevant datasets, with methods now considered outdated.

Kakkoth et al. [49] focused on real-time hand gesture recognition, categorizing gestures into dynamic and static types and organizing detection methods into sensor-based, vision-based, and depth-based. Their survey includes a comparative summary of these technologies but lacks any discussion on benchmarking or continuous recognition.

Escalera et al. [50] presented an overview of challenges in multimodal gesture recognition. As organizers of several contests in the multimodal gesture and action domain, they discussed the outcomes of these events in this work, analyzing the existing datasets and evaluation metrics. Although the focus of the contests differed from continuous gesture recognition, this survey introduced a taxonomy of gesture recognition issues that has influenced subsequent research. Despite this, the methods discussed are used in contests organized from 2011 to 2015 that are now outdated, and there is limited emphasis on benchmarking.

Gu et al. [51] offered a survey focused primarily on action recognition, with an interesting emphasis on evaluating neural network models. While informative about recognition methods, the evaluation methods discussed are not suitable for continuous gesture recognition, making this survey less relevant for our purposes.

Al-Shamayleh et al. [52] conducted a systematic review of vision-based gesture recognition techniques, addressing topics such as hand gesture taxonomies, recognition algorithms, application domains, and commercial systems. However, their focus was broad and the segmentation of continuous gestures received just a little mention.

Ouadah et al. [53] reviewed research across various recognition techniques, including color segmentation, motion-based and skeleton-based recognition, and deep-learning approaches, classifying a considerable number of research papers. Yet, they did not address benchmarking challenges or continuous gesture detection.

Shi et al. [54] surveyed gesture recognition using deep learning in video-based data, categorizing methods based on neural network architectures (two-stream networks, 3D convolutional networks, and Recurrent Neural Networks). They listed several benchmarks but did not differentiate between segmented and continuous gestures, nor did they analyze the evaluation methods of these benchmarks. Sarma et al. [55] provided a detailed review of methodologies in vision-based hand gesture recognition, discussing gesture acquisition systems, preprocessing, representation, feature extraction, and recognition. However, their survey barely addressed benchmarks and evaluations.

Zholshiyeva et al. [56]’s work is focused on gesture detection and recognition methods, particularly those relevant to sign language applications. While they offered insights into segmentation techniques, feature extraction, and application areas, they did not tackle continuous recognition challenges or relevant benchmarks. Jain et al. [57] reviewed various techniques and benchmarks used for video-based gesture classification, blending continuous and non-continuous approaches without discussing the differences in evaluation strategies. Vatavu’s work [58] on gesture-based interaction provides a foundational perspective on the role of gestures as an input modality, offering insights into gesture classification, usability, and system performance evaluation. However, his study covers a broad range of gestural interactions, and it isn’t primarily focused on continuous recognition.

In practical applications, it is often necessary to design methods capable of solving the continuous gesture recognition problem, where gestures must be detected within an ongoing input stream. The continuous nature of these tasks requires a comparison of actual start

and end times, as defined by the ground truth, against those identified by the recognition method. Evaluating an algorithm for continuous gesture recognition is not straightforward; it must account for all the aforementioned aspects, not just classification accuracy.

The numerous surveys illustrate the growing interest in gesture recognition, yet up-to-date and practical information for designing interactive systems remains scarce. Many surveys are outdated, not reflecting recent algorithmic developments or data captured using modern sensors. Others adopt a broad scope, leading to superficial discussions of continuous gesture recognition and its evaluation. To date, no survey comprehensively reviews recognition methods for continuous gesture recognition systems with detailed discussions on all related aspects.

To address this literature gap, we created an updated survey [17] focused on the continuous recognition of intentional gestures for Human-Computer Interaction. Our work emphasizes communication and system command gestures, utilizing off-the-shelf technology. We conducted a systematic search to identify specific benchmarks suitable for this task, classifying them based on a taxonomy that describes their key characteristics. For each benchmark, we reviewed and analyzed the top-performing methods according to the evaluation metrics they employed, highlighting the main features of the proposed gesture detection approaches.

2.2 Input modalities and data stream encoding for Gestures

Various devices are employed to capture gestures, each utilizing distinct data structures to encode the streaming input for spotting and recognizing gestures. In this section, we will explore these devices and encoding methods in more detail.

2.2.1 Acquisition devices

Gesture-capturing devices can be broadly categorized into camera sensors and wearable sensors.

Cameras

Existing types of camera systems provide video streams that can be used either as raw data or pre-processed through computer vision techniques to create input streams for gesture detection. Below, is a list comprehensive of common camera types used for gesture capture:

- **RGB cameras.** Standard RGB cameras used in an everyday scenario, such as webcams for desktop applications or headset-mounted cameras for VR/AR applications, can be used for gesture recognition.
- **IR/Thermal Cameras.** Infrared (IR) refers to electromagnetic radiation characterized by wavelengths that are longer than those of visible light and shorter than radio waves. This type of radiation is not perceptible to the human eye, and it is emitted by bodies as a result of temperature. IR cameras capture thermal radiation, which hands emit due to their relatively higher temperature, making them effective for capturing gestures at close range. Different types of thermal sensors have been tested for gesture recognition [4]. The LeapMotion hand tracker [59], used to collect data for the SHREC'19 benchmark [3], operates with a stereo IR camera.
- **Depth Sensors.** Depth sensors, now widely accessible, facilitate gesture recognition by simplifying the spatial segmentation of arms and hands. There are multiple depth-sensing technologies available:
 - **Stereo Cameras** calculate depth through computer vision algorithms applied to two views. Devices like the UltraLeap hand tracker use stereo cameras for gesture interfaces [60].
 - **Active IR Sensors** use IR illumination patterns and single IR cameras for depth estimation, as seen in devices like the original Microsoft Kinect [61] and Intel RealSense, which was used to capture the DHG14-28 gesture dataset [42].

- **Time-of-Flight (ToF) Cameras** measure the travel time of emitted light to determine depth. The Kinect 2 camera [62] uses ToF technology, as does the HoloLens 2 headset’s finger-tracking module.

While camera-based gesture capture is simple and non-invasive, it can be affected by occlusions, potentially degrading data quality [63]. The effectiveness of image-based gesture recognition relies on advanced computer vision methods to locate and track hands. Camera viewpoint is another critical consideration, as it can influence recognition performance depending on the application. For this reason, it is possible to define two main categories for camera viewpoint:

- **Egocentric** viewpoints use one or more cameras placed on the user’s body. Head-mounted cameras, such as those in studies like [36] and [6], are useful to create Egocentric benchmarks that can evaluate performances for XR applications.
- **Exocentric** viewpoints employ external or off-body cameras, as in [64], and are well-suited for applications in Human-Robot Interaction, public kiosks, multimedia, or automotive controls.

Wearable Sensors

Human body movements and pose capture are also possible through wearable sensors, which, unlike cameras, are not susceptible to occlusions but are more intrusive. Common types of wearable sensors include:

- **Inertial Measurement Units (IMUs)**. These devices measure physical attributes of bodies such as acceleration, rotation speed, and orientation using accelerometers, gyroscopes, and sometimes magnetometers. IMUs are affordable and easy to integrate into wearables. In ChAirGest [1], for example, 4 Xsens-MTw IMUs were used to capture arm movements.
- **Gloves**. Specialized gloves equipped with sensors (e.g., IMUs, exoskeletons, or strain sensors) capture joint motion data [65]. While traditional datagloves can be cumbersome and costly, newer models use flexible strain sensors that are lighter and more affordable for gesture recognition [66]. Notably, benchmarks specifically based on glove-captured data are scarce, and no one of them is used for capturing the online gesture datasets.
- **Surface Electromyography (sEMG)**. This type of sensor is able to detect muscle activity, and sensor as the Myo armband [67] can be used for dynamic hand gesture recognition. Although public sEMG-based benchmarks for continuous gesture recognition are limited, segmented gesture examples exist [68, 69].

2.2.2 Input stream encoding

Temporal data sequences acquired from any kind of sensor are often pre-processed or encoded before use in gesture recognition modules or provided as input data in gesture recognition benchmarks. The main data stream types processed by gesture detection systems include raw video data, optical flow, arms/hands skeleton sequences, and raw feature sequences from wearable sensors.

- **Video Data**. Video streams include spatially registered image channels for each time frame, such as RGB color channels, RGB-D (color and depth), grayscale, or temperature maps for infrared/thermal videos. Image-sequence-based gesture detection requires temporal segmentation of gestures and spatial segmentation of the hands for effective recognition.
- **Image Feature Maps**. Gesture detection modules can work using original images or feature maps derived from them. One example is optical flow, which estimates pixel-wise motion across frames, projecting the scene on the image plane. The NVGesture benchmark [37] uses RGB-D streams and an IR-stereo pair to calculate optical flow and disparity maps to use as input for gesture detection.

- **Skeleton Data.** Many vision-based hand-tracking devices or gloves preprocess raw data into skeletal hand representations, providing a structured input for gesture recognition. These skeletal representations typically consist of 3D coordinates for joints and may include quaternions for segment orientations. Devices like the HoloLens 2, Meta Quest, and Ultraleap provide real-time skeleton data at high frame rates, making hand skeletons a reliable input stream for gesture recognition. If raw data are available, tools like Google MediaPipe [29] can estimate hand skeletons. Even if skeleton representations are compact and filter out irrelevant sensor data, they may suffer from hand pose estimation errors. Several gesture recognition benchmarks include skeleton data [3, 6, 43].
- **Raw feature sequences** For non-spatial acquisition techniques (such as IMUs, sEMG), gesture detection operates directly on raw data streams. In these cases, additional information, such as sensor positioning on the body, can aid in recognizing gestures [1].

2.3 Hand gestures' taxonomy and gesture dictionaries

Movements of the arms or hands that can be identified, classified, and used for communication are "hand gestures". According to the taxonomy of [70], they can be considered "semaphoric" or "language" gestures that we must distinguish from "gesticulation" or non-gesture movements. We considered several aspects to characterize the different sub-types of these gestures found in the research papers of our interest. The first one is related to their temporal features. For this aspect, we use a taxonomy similar to the one proposed in [71], distinguishing **Static** gestures and **Dynamic** gestures. Static gestures are those where the semantics are determined by the hand pose (spatial posture of the joints), without significant movements. A static gesture is performed by holding a specific hand configuration or pose for a minimum duration. Dynamic gestures are those where the movements of the hands determine their semantics and can be further divided into up to three sub-types, as done in some existing benchmarks such as in [6]:

- **Dynamic Coarse**, characterized by the hand's global trajectory (or movement pattern), for example, drawing a cross in mid-air.
- **Dynamic Fine**, where the semantics depend on a variation in the fingers' articulation over time, such as pinch, grab, or others.
- **Dynamic periodic**, where the semantics depend on the iteration of a basic motion pattern such as wave hand, wave finger, or others.

The semantics of a gesture can be invariant with respect to specific variations in its execution. Invariance properties of the gestures are quite important as they influence the design of proper recognition tools. We can distinguish the following cases:

- **Position-invariant** gestures are those where a translation does not change the semantics. This is not true in practical recognizers where the detection is activated only in specific spatial regions so that a meaningful gesture executed in a particular location is considered non-gesture in others.
- **Orientation-invariant** gestures are those where a global hand rotation does not change the semantics. For example, a gesture dictionary that includes a thumb up or down gesture to indicate like or dislike is clearly non-invariant to orientation, as the same hand's pose changes meaning with a global rotation.
- **Direction-invariant** gestures are the dynamic ones where the correct labeling does not depend on the orientation of the global movements of the hand with respect to a fixed reference frame. The pinch gesture used in many practical VR interfaces to trigger actions has the same meaning, independent of the position and orientation of the hand.

In our classification of gesture dictionaries, we will consider Position (Orientation, Direction) invariant, a dictionary of gestures where all the items feature the same property, so that the recognizer can be trained/tested with an invariant encoding.

To allow the users to interact with gesture-based interfaces, the designers (and the creators of benchmarks) create *dictionaries* of gestures, possibly including static and dynamic ones. Dictionaries are characterized by the sets of class labels and by the origin of

their semantics. Dictionaries may assign the semantics based on simple rules such as error margins with respect to a standard pose/trajectory template. In this case, the classification errors may depend only on finger-tracking accuracy. For considered benchmarks, the semantics are data-driven and depend on a number of training templates or sequences, including examples with known temporal segmentation. This fact may introduce relevant sources of noise and biases. Different users can execute the gestures in quite different ways because of particular constraints in the individual hand articulation [72], different speeds, and different mental models of the gestures, possibly due to different cultural backgrounds. Hands morphology and size can vary, and the skin color can differ. The number and the diversity of the participants creating the template set is, therefore, an important characteristic to understand how much the results obtained on a benchmark can be generalized in a practical setting. The limited ability of the training examples to represent the target users can be a relevant issue in practical applications, requiring the collection of a huge amount of additional training data. The choice of the gesture set in the dictionary can also be a source of errors and biases. Dynamic ones can have an extremely variable time duration, which is quite challenging for online recognizers. Also, it is necessary to note that different gestures can share similar subparts.

The recognition rates for individual gestures may depend critically on the other ones included in the dictionaries, which might be optimized for the discriminability of gestures. Looking at the benchmarks included in our survey, it seems that they are not optimized in this regard. Per-class performances of the methods might give insights into optimal subsets of gestures that can be selected for practical use. Meaningful gestures of a selected dictionary cover a small part of the potential "gesture space", including all the possible hand position/orientation/pose variations. This means that in an online classifier where we need to discriminate the gestures from the "non-gesture" class, we have a large imbalance in the "gesture space" partitioning, with most of the space labeled as "non-gesture". This means that it is difficult to train classifiers with a good sampling of the non-gesture examples as the training data are necessarily limited to a few sequences (and the classifiers must deal with the class imbalance issue). This problem can be mitigated by creating synthetic examples (data augmentation) [73, 74].

The input data to train a continuous gesture should not (or not only) consist of examples of segmented gestures but long sequences provided by the input devices where the time location of the meaningful gestures is known. The characteristics of the non-gesture patterns used to interleave the gestures in these sequences should be accurately controlled to avoid unintentional executions of gestures not annotated. Ideally, the non-gesture parts should represent all the gesticulations done by all the users of the interactive system simulated, optimally sampling the "non-gesture" space. The number of sequences provided for training (and testing) is, therefore, another relevant characteristic of the benchmark, as well as the type of gesticulation done in the non-gesture parts. In the case of video data, the diversity of the visual scenarios for the data capture is also important to assess the robustness of the detector against changes in background, light conditions, and others. Therefore, we report the number of different scenes used in the datasets. In the practical design of the acquisition devices and the dictionary, many choices strongly depend on the application focus. We introduce a specific category in our taxonomy to record if a reviewed benchmark has a specific **application domain**. Specific applications for which continuous mid-air gesture recognition systems can be applied are mixed reality, automotive, public kiosks, sign language-based communication, and mobile/touchscreen interfaces.

2.4 Benchmarks Evaluation metrics

As discussed in Chapter 1, while for the segmented gesture classification task, the evaluation is simple and based only on the test gesture labeling accuracy, for the continuous classification task, the performance assessment is more complex, as it should also consider the temporal segmentation quality, the latency, and the probability of false detection.

We report in [Appendix A](#) a complete description of the metrics proposed in the reviewed benchmarks for online gesture recognition. In the appendix, the reader can find the definitions, the related use cases, advantages, and disadvantages. This section summarizes their characteristics and introduces the taxonomy used to classify them. Following Ward et al. [75], we divide the metrics in two groups:

- **Frame-Based.** For the continuous classifier labels at each time sample (frame), we can evaluate classification metrics by comparing the predicted sample/frame labels with the ground truth ones. Classic examples are the average classification scores in the evaluation of the single-frame labeling (Accuracy, Precision/Recall, F-Score, Area under curve (AUC)) or string distances (Jaccard Index, Levenshtein Distance) comparing the labeled sequences with the ground truth.
- **Event-Based.** A gesture in a continuous sequence of frames can be considered an *event* with a specific start time, end time, and an associated label. These evaluation metrics compare predicted events with ground truth ones and are less straightforward than the Frame-based ones. They should cope with the temporal location of the gestures, possibly estimating their accuracy, and must define thresholds to consider whether an event is correctly detected. Considering the input stream, the frame-based metrics compare two strings of equal length, the ground truth, and the estimated sequences of class labels of single frames. In contrast, the event-based metrics evaluate the differences between ground truth and estimated lists of events with associated attributes. These lists may have different lengths, as we can have missed events and false detection of events. For these reasons, various event-based metrics have been proposed, making it difficult to compare results obtained in different benchmarks. The Detection Rate (percentage of events correctly captured with respect to the total number in the ground truth) and the False Positives' Score (see [Appendix A](#)) are often measured, even if with variable criteria to consider a gesture detected. However, other methods can be applied, like the Levenshtein Accuracy, based on comparing strings of consecutive events. To characterize the accuracy of the temporal segmentation of the methods, some authors use the delay of the recognition feedback, while others rely on specific scores like the Accurate Temporal Segmentation Rate (ATSR), the Absolute Temporal Segmentation Error (ATSE), the Performance index ([Appendix A](#)).

In our taxonomy, we also consider the **efficiency metrics**, to evaluate how much an algorithm can be adapted to different platforms, as many benchmarks also assess the amount of resources needed by the methods in terms of memory requirements (e.g., the number of parameters of the neural models used, overall model size), computational complexity and real-time detection performances (e.g., FLOPS).

2.5 Existing benchmarks and results

According to the criteria identified in the previous sections, summarized in [Figure 2.1](#), we analyzed the benchmarks found in the literature following a specific search methodology.

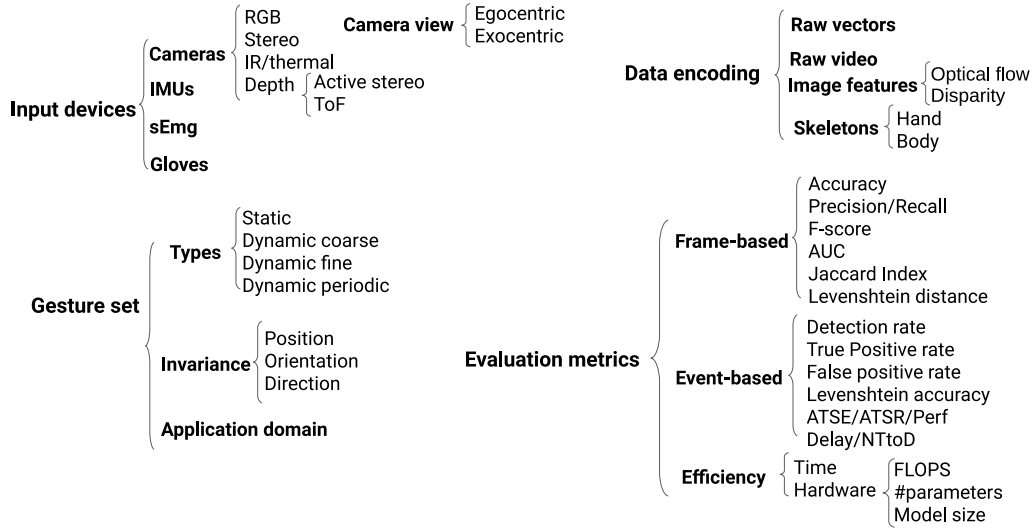


Fig. 2.1: Criteria used to classify the existing benchmarks for continuous gesture spotting/recognition

2.5.1 Literature search methodology

As explained in [17], to search for the benchmarks with the required features (with a focus on continuous recognition of mid-air hand and upper body gestures), we first used the websites that are popular for benchmarking initiatives, such as Eurographics Shape Retrieval Contests (SHREC)¹, ChaLearn², and Kaggle³, this resulted in 3 relevant benchmarks. We then conducted methodical research in generic literature archives, following the guidelines of the PRISMA 2020 methodology [76]. This identification phase aimed to form a superset of the papers that are related to "gesture recognition benchmarks and dataset" and "evaluation metrics". We defined a base *BASE* query as follows:

```

BASE= {(Hand OR (Upper and Body))
AND (Gesture OR Gestural OR Interaction OR Interactions OR Recognition)
AND (Benchmark OR Benchmarks OR Dataset OR Datasets OR (Data and Set) OR
(Data and Sets))
AND (Online OR Continuous OR Real-time OR (Real and Time))}.
    
```

A critical decision in our survey process was to use a proper search engine for the benchmarks. Google Scholar⁴ was our first option; however, using the Base query in this search engine resulted in 3,200,000 articles, most of which were not useful for our research domain. We decided to use Scopus⁵ since it is reliable, presents more relevant results, and has advanced searching capabilities that include the usage of custom queries. The query search in Scopus resulted in 2758 articles, and to find the most relevant ones, we initiated a screening phase in which we carefully read all the abstracts, keeping only the ones that matched our inclusion criteria. We define these criteria as follows:

- The paper must describe a benchmark for gesture recognition in continuous time.
- The gestures should be designed for interaction tasks. Voluntary command/communication gestures are included, action recognition and sign language are excluded.

¹ <https://www.shrec.net>

² <https://chalearnlap.cvc.uab.cat/>

³ <https://www.kaggle.com>

⁴ <https://scholar.google.com/>

⁵ <https://www.scopus.com/home.uri>

We then fully read the filtered papers to ensure that only the relevant ones are presented. Only 13 benchmark papers matched our strict inclusion criteria, and we excluded a benchmark (SHREC '21 [8]) as the benchmark's authors declared annotation issues on their website; these issues are related to what was discussed in Section 2.3.

We conducted a backtracking search of the papers that cited and used benchmarks in our list, aiming to find methods that yield the best results for each specific benchmark. Our search for these proposed methods was not limited to just one search engine (Scopus), we also used Google Scholar. These proposed methods are discussed in the subsequent subsections. This search resulted in 1645 articles overall. From this list, we selected the works that tested novel methods on the benchmarks included in the survey, and in this paper, we report a description of those that lead to the best results (up to three per benchmark).

2.5.2 Benchmarks

In this subsection, we aim to introduce the final benchmarks that are selected for our studies. We also summarize and present each benchmark's gesture dictionary characteristics (Table 2.1), acquisition setup characteristics (Table 2.2), and sequence/gesture sets characteristics (Table 2.3).

Name	Year	Type of gestures	Invariance	Application domain
ChAirGest	2013	DC, DP	P	General Interaction
Montalbano T3	2014	DF, DP	P	Verbal Communication
ConGD	2016	S, DF, DC, DP	P	Many Domains
NVGesture	2016	S, DF, DC, DP	P	Automotive
EgoGesture	2018	S, DF, DC, DP	P	Mixed Reality
SHREC'19	2019	DC	P	Mixed Reality
IPN Hand	2020	S, DF, DC	P,O	Touchless screens
MIGesture	2020	DC,DP	P,O	Automotive
LD-ConGR	2022	S, DC	P,O	Long Distance Recognition
SHREC'22	2022	S, DF, DC, DP	P	Mixed Reality
ODHG 14/28	2022	DF, DC	P,O	General Interaction
ZJUGesture	2023	DF, DC	P,O	Mobile

Table 2.1: Characteristics of the gesture dictionaries used in the reviewed benchmarks: types of gestures (S=static, DC=dynamic coarse, DF=dynamic fine, DP=dynamic periodic) and invariance (P=position, O=hand orientation)

Name	Data types	View	Input devices	FPS
ChAirGest	RGB, RGB-D, Hand pose, IMU	EGO, EXO	Kinect / Xsens MTw IMU	30/50
Montalbano T3	RGBD	EXO	Kinect	20
ConGD	RGB-D	EXO	RGB-D Camera Setup	NA
NVGesture	RGB-D, Optical flow, IR	EXO	SoftKinetic DS325 / DUO 3D	30
EgoGesture	RGB-D	EGO	Intel RealSense SR300	30
SHREC'19	Hand pose	EGO	Leap Motion	NA
IPN Hand	RGB	EXO	Webcam	30
MIGesture	RGB-D, RGB, Thermal	EXO	Industrial cameras	8/12/16
LD-ConGR	RGB-D, RGB	EXO	Kinect v4	30
SHREC'22	Hand pose	EGO	Hololens 2	20
ODHG 14/28	RGB-D, Hand pose	EXO	Intel Real-Sense	30
ZJUGesture	RGB	EXO	Mobile phone	30

Table 2.2: Characteristics of the acquisition setup for the datasets included in the reviewed benchmarks.

Name	Scenes	Subjects	Classes	Gesture Samples	Sequences	Gps
ChAirGest	1	10	10	1200 (900 - 400)	NA	V
Montalbano T3	1	10	10	13,858 (3,362 - 2,742)	563	V
ConGD	15	21	249	47,933 (30,442 - 17,491)	22535	V
NVGesture	1	20	25	1,532 (1,050 - 482)	1532	V
EgoGesture	6	50	83	24,161(19184 - 4977)	2,081	9 to 12
SHREC'19	1	13	5	195 (60 - 135)	195	1
IPN Hand	28	50	13	4,218 (148 - 52)	200	1 to 5
MIGesture	1	24	9	over 1300	over 1300	1 to 2
LD-ConGR	5	30	10	44,887 (34,315 - 10,572)	542	V
SHREC'22	1	6	16	1152 (576 - 576)	288	3 to 5
ODHG 14/28	NA	20	28	2800	NA	V
ZJUGesture	12	60	9	9892 (8290 - 1602)	NA	V

Table 2.3: Characteristics of the sets of sequences/gestures included in each benchmark. Figures in brackets specify the training/test splitting of sequences and gesture samples. Gps means Gestures per sequence (V= Variable)

ChAirGest.

The ChAirGest [1] challenge was an open research initiative aimed at motivating scholars to leverage data collected from diverse sensors to enhance and assess methods for gesture spotting and recognition. The challenge was based on a dataset created using a Kinect RGBD camera (30 fps) and four Inertial Motion Units (IMUs, 50 fps) strategically attached to the right arm and neck of the ten subjects (Figure 2.2). The dataset consists of a vocabulary of 10 one-hand dynamic gestures commonly used in close human-computer interaction (e.g., swipe, circle, push, etc.), initiated from three different resting postures and captured across two distinct lighting scenarios of the same scene. The authors collected a total of 1200 annotated gestures, organized into continuous video sequences, each containing a variable number of gestures. The evaluation is based on a hybrid approach incorporating well-established metrics (event-based precision and recall and the derived F-score) and an original time-based metric, the Accurate Temporal Segmentation Rate, described in Subsection A.2.4.

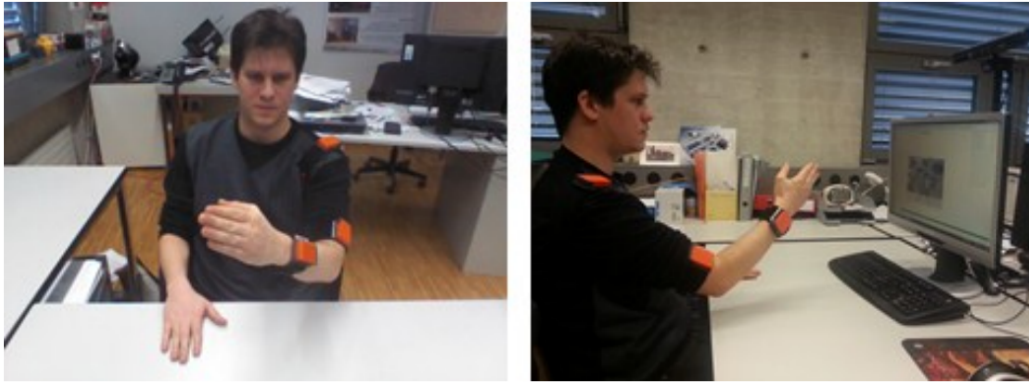


Fig. 2.2: Recording setup for the ChAirGest Benchmark. Left: image captured by the Kinect. Right: external view of the same subject. Wearable IMUs are visible on his right arm [1].

Montalbano Track 3.

The Looking at People Challenge 2014 [2] competition included three different tracks and related benchmarks. The first two tracks focus on action recognition and cannot be included in our survey; only the third complies with our inclusion criteria. In this track, the participants were tasked with recognizing 20 distinct Italian gesture categories using continuous

RGB-D video sequences captured at 20fps. The dataset comprises 13,858 gestures performed by 27 subjects in 563 sequences. From those sequences, 287 were given to the participants as training sequences and labels; the other 276 were used as test sequences. Throughout all the sequences, a single user is captured standing before a Kinect device without occlusion. The gestures in this dataset are not "commands" but are "natural" communicative gestures performed while the subjects were speaking fluent Italian (Figure 2.3). Each sequence is 1-2 minutes long, and during it, the subjects move their arms without rest, resulting in an extremely varied "non-gesture" class. This dataset exhibits a notable intra-class variability among gesture samples, meaning that gestures within the same category can vary significantly. Conversely, the inter-class variability is relatively low, indicating that some gesture categories share similarities, making the classification hard. It's worth noting that no information is available regarding the number of gestures included within each sequence. The benchmark uses the Jaccard Index to evaluate the recognition's accuracy.

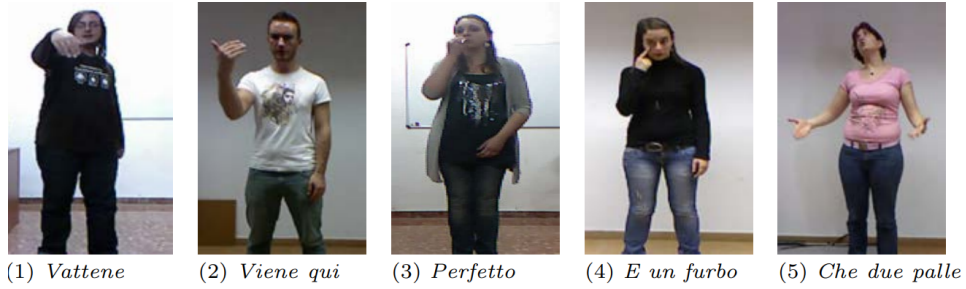


Fig. 2.3: Example gestures from the Montalbano dataset. They are performed before a Microsoft Kinect while speaking Italian [2].

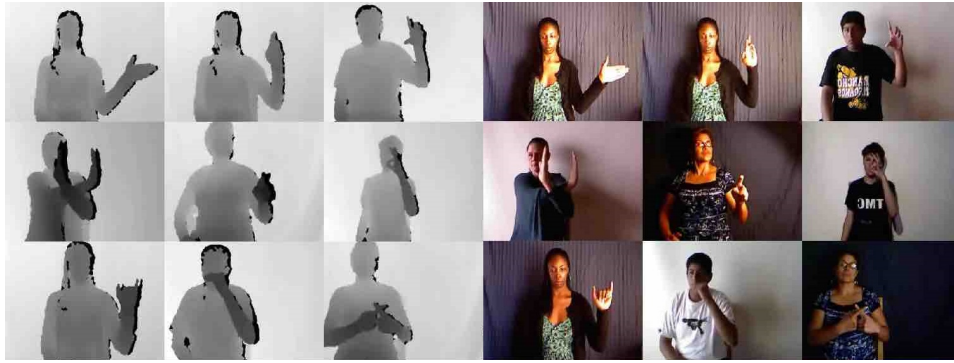


Fig. 2.4: Left: depth maps from the ConGD dataset. Right: the corresponding RGB images. From <http://www.cbsr.ia.ac.cn/users/jwan/database/congd.html>.

Continuous Gesture Recognition Dataset (ConGD)

Wan et al.(2016) [77] created two large-scale datasets by integrating all batch classes and samples from the ChaLearn gesture dataset (CGD). The CGD dataset was designed by Guyun et al. in 2013 [78] for a series of "One-shot learning" tasks, where there is only one training example for each gesture and the rest of the data is used for testing. The tasks correspond to different application domains, e.g., sign language, helicopter signals, traffic signals, and underwater sign language. For each of them, a specific batch of data with 100 example gestures from a small lexicon of 8 to 12 classes has been created. The entire dataset

consists of 289 classes from 30 lexicons and a total of 54,000 gestures recorded in 23,000 RGB-D videos. The ConGD dataset was obtained by annotating the videos with a semi-automatic procedure, combining the different batches, removing some of them, and fusing the classes with similar characteristics. The final benchmark features a dictionary with 249 gesture classes, and it is composed of 22,535 RGB video recordings and 22,535 depth videos where 21 subjects perform them for a total of 47,933 executions (Figure 2.4). The dataset is split into a training set with 30,442 gesture samples, a validation set with 8,889, and a test set with 8,602. The Jaccard Index is the only metric used to measure the continuous recognition performances on this benchmark.

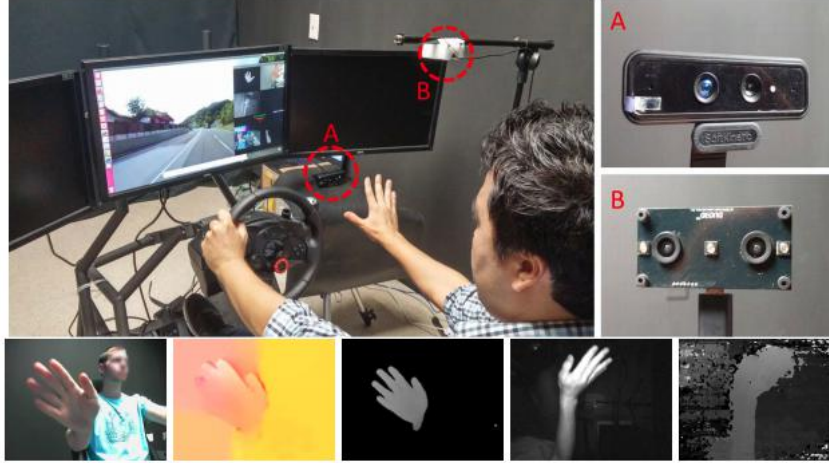


Fig. 2.5: Top left: recording setup for the NVGesture dataset. The subject uses his left hand to drive and performs the gestures with his right hand only. NVGesture includes multi-modal acquisitions. Data are captured with a SoftKinetic DS325 recording front-view RGB-D videos (A), and with a DUO 3D stereo-IR camera mounted above the user (B). From <https://www.v7labs.com/open-datasets/nvgesture>.

NVGesture.

The NVGesture dataset, released by Molchanov et al. in 2016 [37], is designed to test gesture recognition for touchless driver control. The authors recorded the gestures using multiple sensors and viewpoints; this involved 20 subjects who performed the gestures with their right hands while holding the steering wheel with their left hand. The SoftKinetic DS325 sensor was used to acquire front-view color and depth videos, and a top-mounted DUO 3D sensor was used to collect this dataset for recording a pair of stereo-IR streams (Figure 2.5). Data are captured at a frame rate of 30 fps.

This dictionary includes 25 gesture types, and the benchmark contains 1532 dynamic hand gesture samples; these samples are divided into 1050 training and 482 test samples. The metrics chosen for the evaluation are the area under the curve (AUC) of the receiver operating characteristic (ROC) curve and the normalized time to detect (NTtD) at a detection threshold ($\tau = 0.3$).

EgoGesture.

The EgoGesture dataset, released by Zhang et al. in 2018 [36], includes egocentric RGB-D videos of hand gestures and is specifically designed to evaluate methods that interact with wearable devices like VR and AR headsets. Videos in this benchmark were captured using a head-mounted Intel RealSense SR300 RGB-D camera with a 640×480 resolution and a frame rate of 30 fps (Figure 2.6). The benchmark dictionary comprises 83 static or dynamic gesture

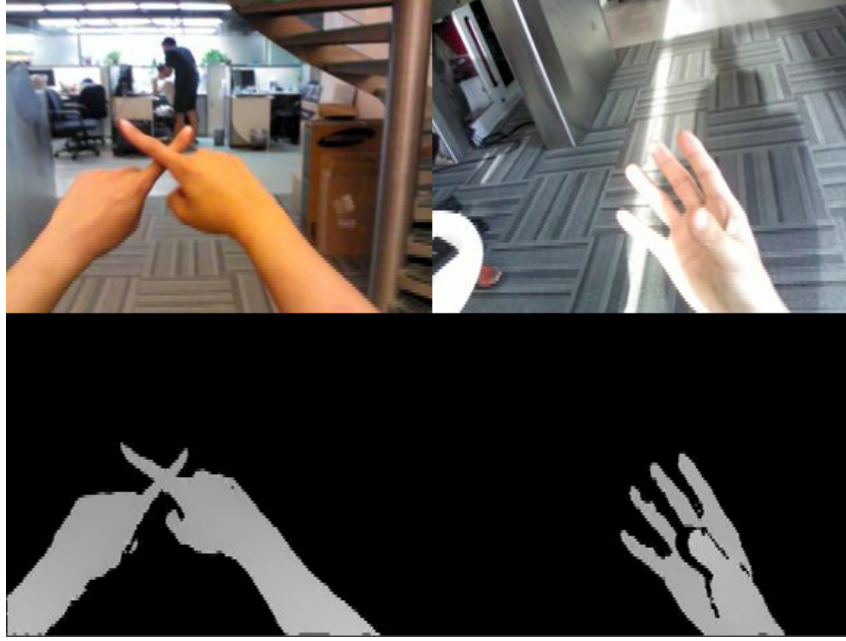


Fig. 2.6: Example RGB (top row) and depth images (bottom row) from the EgoGesture dataset. From <http://www.nlpr.ia.ac.cn/iva/yfzhang/datasets/egogesture.html>.

classes. The authors divided these classes into two types: Communicative and Manipulative. Communicative gestures represent choices or commands for interfaces (numbers, symbols). Manipulative gestures are designed to control actions on interface components, such as zoom, rotate, and open/close. The dataset includes 300 samples for each class with a large intra-class variety. Gesture samples are included in 24,161 RGB-D video sequences and are performed by 50 subjects. The video recordings have strong variations in background and illumination conditions and have been captured in indoor and outdoor environments. The start and end frame indexes of each sample are manually labeled. The minimum length of a gesture is three frames, while the maximum is 196 frames. The sequences were randomly split into a training set (1,239), a validation set (411), and a test set (431). The evaluation metrics used for this benchmark are the Jaccard Index and the execution runtime of the tested methods measured in frames per second (fps).

SHREC 2019 Track: Online Gesture Recognition (SHREC'19).

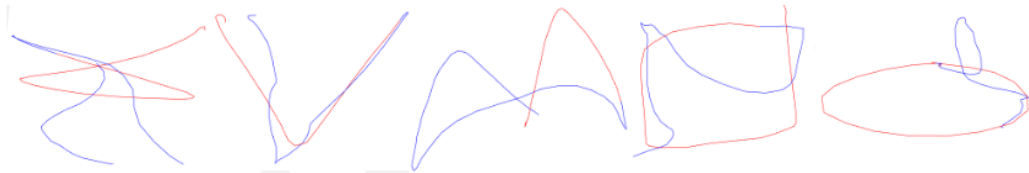


Fig. 2.7: The dictionary of the SHREC'19 benchmark features 5 gesture types characterized by simple 2D paths (red lines) hidden in longer 3D non-gesture trajectories (blue). From [3].

In 2019, Caputo et al. [3] created this benchmark by capturing the hand movements of 13 subjects while interacting with a 3D interface in an immersive Virtual Reality environment. The subjects wore a Head Mounted Display (Oculus Rift) with a Leap Motion sensor

placed on it to capture hand poses from an egocentric point of view. The dictionary is composed of 5 different coarse gestures characterized by a 2D hand trajectory (Figure 2.7). The dataset includes 195 hand pose sequences, each containing a single gesture in a different time location. The non-gesture part of the sequences is roughly determined by a set of specific actions on the virtual interfaces required by the interactive VR app. For the training set, 60 of these sequences with manually annotated start/end of the gestures are used, and the rest of the data are used as the test set. In the evaluation, a gesture is considered correctly detected and labeled if the recognition method locates it within 2.5 seconds from the actual gesture time window in the sequence. Given the detected and ground truth gesture, the following metrics are estimated to characterize the methods' performances: the percentage of correct detections, timely detections with wrong classification label, detections made before the correct time location, detections made after the correct time window or missed, the time difference between the decision time of the online algorithm and the actual start and end of the gestures. In the [10], the benchmark is used by adding the classification time and False Positive Ratio metrics.

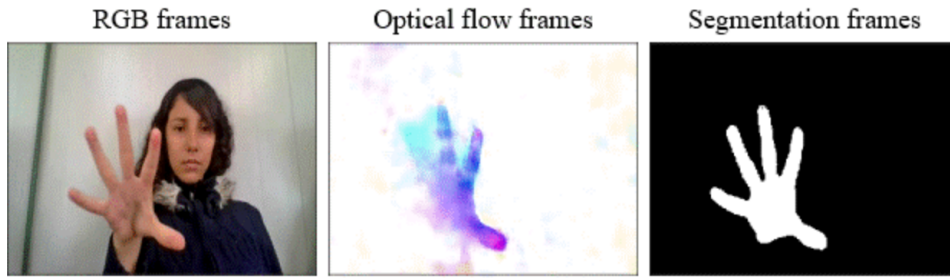


Fig. 2.8: The different data types included in the IPN Hand distribution. From left, RGB video frames, optical flow maps and segmentation masks. From https://gibranbenitez.github.io/IPN_Hand/.

IPN Hand.

This dataset, introduced by Benitez-Garcia et al. in 2020 [64], focuses on interaction with touchless screens. It contains 100 RGB video recordings of sequences, including 4,218 gesture samples, performed by 50 different individuals. The videos were captured with PC or laptop cameras, keeping a fixed resolution (640×480) and frame rate (30 fps). Authors provide not only video data and annotations but also optical flow sequences and accurate hand segmentation masks estimated as described in [64] (Figure 2.8). The sequences are captured with 28 different background environments, both static and dynamic, with strong variations in background clutter and lighting conditions (both strong and weak illumination). Each video's start and end frame index for gesture instances was manually labeled. The minimum length of a gesture is nine frames, and the maximum is 650 frames. The dataset is split into a training set with 148 videos comprising 3,117 gesture instances performed by 37 subjects and a test set with 52 videos and 1,101 gesture instances performed by 13 subjects. The 13 gesture types are designed for desktop-like interface control: two static pointing gestures (with one or two fingers) and 11 dynamic gestures (clicking with one and two fingers, throwing to four positions, double-clicking with one and two fingers, zoom-in, and zoom-out). The metrics used for the evaluation are the Levenshtein accuracy, the model size, and the inference time.

MIGesture.

Maarten et al. [4] proposed MIgesture as the first publicly available hand-gesture recognition dataset based on low-cost thermal sensors (Melexis MLX90640/MLX90641) data. The benchmark is designed to evaluate tools to control a multimedia system in a car. To investigate the impact of sensor type and viewpoint, two sensor clusters were used to capture

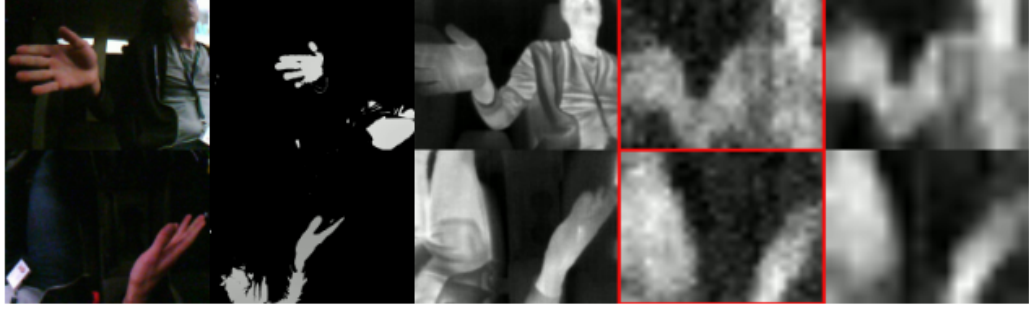


Fig. 2.9: Sample RGB and thermal frames from the front sensor cluster (top row) and top sensor cluster (bottom row) included in the Mlgesture dataset (from [4]).

data, each consisting of 5 different devices (the low-cost MLX90640 and MLX90641 thermal cameras, an OpenMV color cam, an MLX75027 time-of-flight depth sensor, a FLIR lepton thermal camera). One sensor cluster was mounted on the center of the car’s dashboard in front of the driver, while the other was mounted on the ceiling, pointing straight down (Figure 2.9). The dictionary consists of 9 different dynamic hand gestures compatible with low and ultra-low-resolution sensors. The authors acquired over 1300 RGB, RGB-D, and thermal video clips for both training and evaluation purposes, but only the thermal data are released in the public dataset. 24 different subjects performed the gestures in the videos. In addition to the gesture videos, non-gesture videos were also recorded, capturing actions like steering, gear switching, operating the radio, and using the wipers. Tests for continuous gesture detection were performed on a sequence obtained by stitching half of the test gesture and non-gesture acquisitions in a single long video and manually annotating the hand gesture nuclei in the resulting test video. In the continuous evaluation, if a correct prediction is within an annotated gesture nucleus, it is considered a True Positive, while further detections during the nucleus or detections outside the nucleus are considered False Positives. Missed gestures are considered False Negatives. In this benchmark, the metric used for the comparisons is the Mean Average Precision (plotted versus network output position graphs).

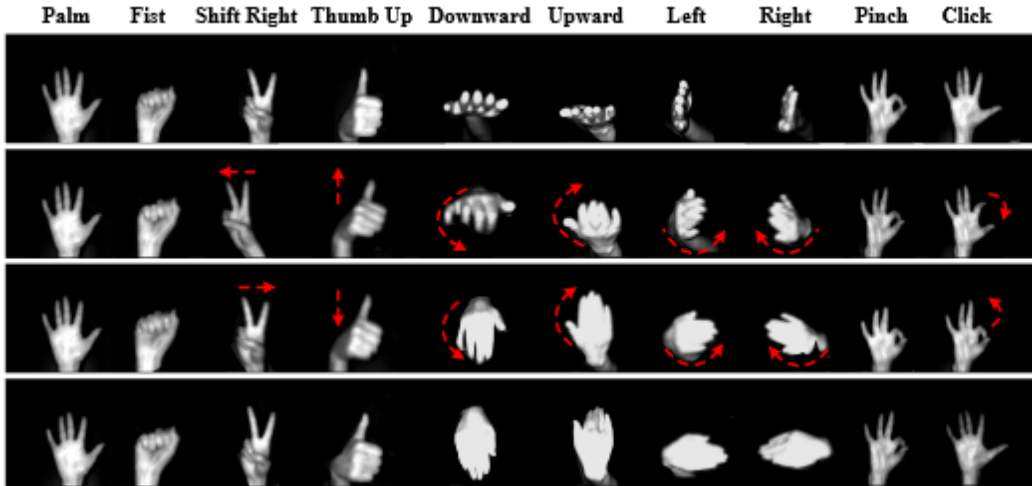


Fig. 2.10: The gesture dictionary of the LD-ConGR dataset (from [5])

A Large RGB-D Video Dataset for Long-Distance Continuous Gesture Recognition (LD-ConGR)

This benchmark, proposed by Liu et al. in 2022 [5], is characterized by the peculiar recording viewpoint where the front view of the subject is from 1 to 4 meters away. The videos are captured with a Kinect V4 device at a resolution of 1280×720 for the color streams and 640×576 for the depth streams, with a frame rate of 30 fps. The dictionary includes 10 fine-grained hand gesture classes (Figure 2.10), 3 of which are static (palm, fist, thumb up), and 7 are dynamic (shift right, downward, upward, left, right, pinch, click). The authors collected 542 videos from 30 subjects in 5 meeting rooms (with different designs and furnishing); there are 6 recording spots in every meeting room. Each video contains multiple gestures, and all the gesture instances were manually annotated with the label and the starting and ending frames. Examples of correct gesture executions were shown to the subjects before the recording started. The gestures were performed continuously, allowing short breaks between gesture instances. Given the large distance, the hands are small and difficult to recognize, and the authors added their position as an annotation for each frame. There are a total of 44,887 gesture instances in the videos. These videos are divided into training and testing sets, with 34,315 and 10,572 gestures (performed by 23 and 7 subjects), respectively. The evaluation of the continuous recognition is done with the Jaccard index.

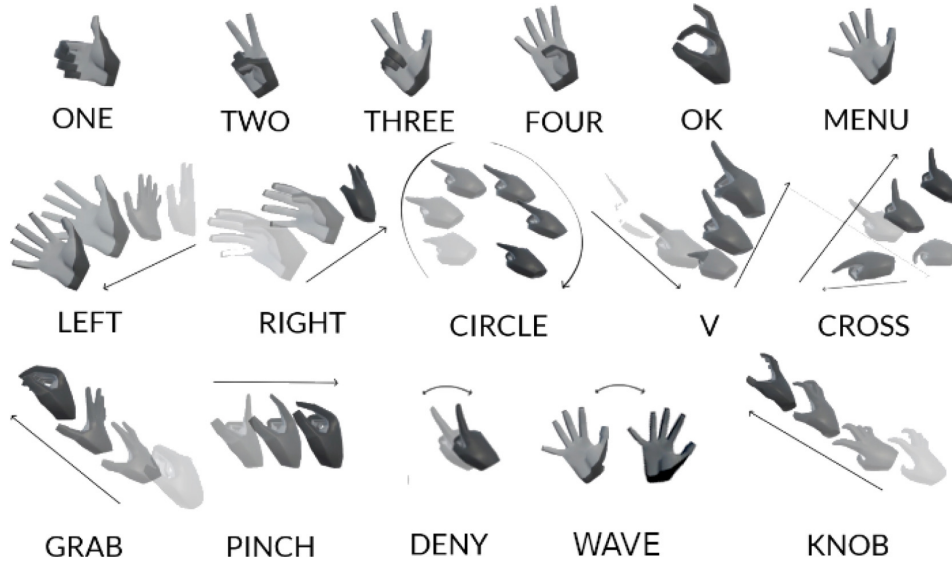


Fig. 2.11: The gesture dictionary of SHREC '22 include static (top row), dynamic coarse (middle row) and dynamic fine and periodic gestures (bottom row). From [6].

SHREC 2022 Track on Online Detection of Heterogeneous Gestures (SHREC'22).

The SHREC'22 benchmark [6] is designed to test continuous gesture recognition for a generic mixed-reality interaction. The publicly available data provided to the contest participants are hand-pose skeleton streams directly estimated by the finger-tracking system integrated into the HoloLens 2 headset. More details of this dataset will be discussed in Chapter 3

Online Dynamic Hand Gesture dataset (ODHG 14/28).

De Smedt et al. introduced the original DHG-14/28 dataset in 2016 [42]. This dataset was intended to be a segmented gesture benchmark, even though there are captured continuous sequences with both gesture and non-gesture classes. The recorded videos are RGB-D,

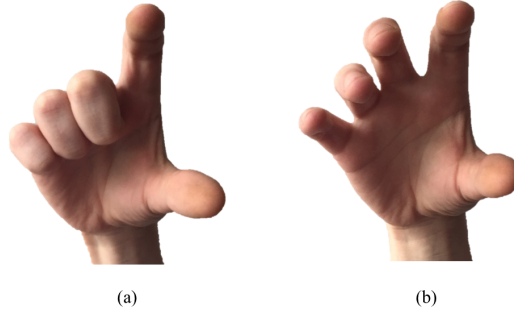


Fig. 2.12: The different ways to perform gestures in the OHDG (and DHG 14/28) dataset: using (a) one finger or (b) the whole hand (from <http://www-rech.telecom-lille.fr/DHGdataset/>).

captured with an Intel Real-Sense camera, and by using associated APIs, the corresponding hand skeleton sequences are extracted. The depth image resolution is 640×480 , and both image and hand skeletons are recorded at 30 fps. The benchmark's dictionary includes 14 gesture classes: five "dynamic fine" (Grab, Expand, Pinch, Rotation clockwise and counterclockwise), characterized by finger articulation, and nine "dynamic coarse" (Tap, Swipe Right, Swipe Left, Swipe UP, Swipe Down, Swipe X, Swipe V, Swipe +, and Shake), characterized by global hand movements. The gestures have variable execution times, ranging from 20 to 50 frames. There are two gesture execution methods: using only two fingers or using the whole hand (Figure 2.12). If the same gesture class is assigned to both methods, there would be 14 gesture labels; otherwise, there would be 28 gesture labels. The whole dataset comprises 280 sequences performed by 20 subjects and ten gestures each. Each subject executed each gesture five times. While the original benchmark only tests the classification of segmented gestures from the captured data, Wannous et al. (2022) [79] reused the sequences to create a continuous gesture detection benchmark, the Online DHG (ODHG). In this version, a class label is annotated for each frame of the sequences. The frames that occur between meaningful gestures are labeled as the "no gesture" class. During these intervals, the participants were allowed to take a resting position without any specific instruction. To evaluate the online detection and recognition, different metrics are used in [79]: accuracy, Receiver Operating Characteristic (ROC), False Positive Rate (FPR), True Positive Rate (TPR), and Normalized Time to Detect (NTtD).

ZJUGesture.

This dataset was created by Xu et al. in 2022 [7], with the primary focus on one-handed operation scenarios commonly encountered in mobile phones or tablets, and includes RGB video sequences recorded with these devices (Figure 2.13). The benchmark's dictionary includes nine categories of gestures designed to be user-friendly and easily operable. Each gesture is divided into preparation, core action, and retraction. To ensure the diversity of the captured samples, each gesture is recorded in 12 sub-scenes, which include various backgrounds and different light intensities. This approach helps to capture variations that may occur in real-world usage scenarios. All videos are labeled frame by frame with the gesture's name; each video has a resolution of 1280×720 pixels, captured at a frame rate of 30 FPS. In the ZJUGesture, 60 people performed each gesture according to their habits at three speeds: fast, normal, and slow. The Levenshtein Accuracy is used to evaluate the online task together with three efficiency measures: speed, GFLOPs, and number of parameters.

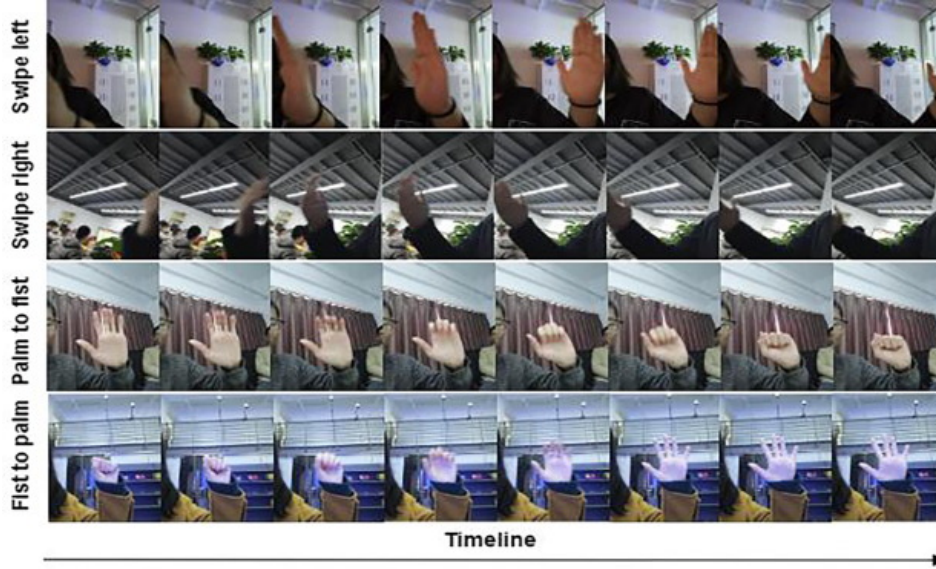


Fig. 2.13: Examples of gestures sequences included in the ZJUGesture dataset, captured by different mobile devices (from [7]).

2.5.3 Continuous gesture recognition methods and best results on the surveyed benchmarks

The methods proposed for continuous gesture recognition are more complex than those employed to recognize segmented gestures since they must also deal with segmentation in a continuous stream. If the detection and classification are performed on image data instead of hand poses or wearable sensor data streams, the methods must also detect hands with an image segmentation module before extracting gesture descriptors. To achieve this goal (independently from the hand's data encoding), a possible solution is to use **two modules**: one designed to perform the time segmentation of the gesture, and one for the classification. A typical solution for the segmentation is to estimate the boundaries based on the motion energy [80,81]. The subsequent classification can be done with classifiers trained with labeled segmented gestures (of different sizes).

Another option is to use a sliding window approach in the time domain, classifying one or more signal windows [82] ending at the current time frame with a specifically trained classifier.

When using hand pose data, the classification module can be a simple template matching approach [40], but the recent literature is dominated by methods based on neural networks. Different network architectures have been proposed, such as 1D convolutional networks [83], graph-convolutional networks [84], or transformer networks using the attention mechanism to focus on the co-occurrence of relevant features of the sequences in gesture classes [85].

2D/3D convolutional networks can be applied for RGB-D images [39]. Another possible solution for the continuous recognition is the use of a recurrent network, trained on continuous data and providing a predicted gesture label at each time step (using non-gesture as an additional class) [86–88]. In this case, the preliminary segmentation of the gestures in the test phase is unnecessary, but a further post-processing step is required to derive a reliable detection of gesture events from the frame labels.

In the following, we introduce the segmentation and classification methods that lead to the best results on the surveyed benchmarks (up to three). As these methods exploit the classifiers described above, we provide a short description of them in [Appendix B](#). Two classes of approaches can be adopted to handle the online task [50]: *direct* and *indirect* methods. Direct methods employ specialized heuristics based on speed, energy, or curvature (or trained networks) to find gesture boundaries and then send the extracted candidates

to a classification module. Although this approach is not expected to work well in the case of complex motions/gestures [89], it has been proposed in different implementations. In SHREC’19 [3](see Sec. 2.5.2), Seg.LSTM1 uses an LSTM together with a specialized segmentation network. The ST-GCN method used in [8] uses an energy-based segmentation approach, adding several ad-hoc rules. In the 2ST-GCN method of [6], an energy-based detection module is combined with a fine-grained classifier, providing gesture/non-gesture discrimination.

Indirect methods perform a continuous classification (simultaneous detection and labeling). This can be done using pre-trained classifiers working with fixed-size input subsequences and sliding windows schemes or by using recurrent networks like Long-Short Time Memory (LSTM) networks or Gated Recurrent Units (GRU). A simplified version of DeepGRU [90], combined with a smart data augmentation method and adapted for on-line detection, performed best in the SHREC’19 contest [3], but a similar solution did not perform well on complex heterogeneous gestures [8].

Best results on ChAirGest

Table 2.4 shows the metrics obtained with the best method tested on the ChAirGest benchmark. In the following, we report a description of the method.

Method	Date	ATSE	ATSR	F1 Score	Perf
CHMM [91]	2013	NA	0.923	0.907	0.912

Table 2.4: Performances of the best method on the ChAirGest benchmark

- **Concatenated Hidden Markov Models (CHMM)** by Yin et al. [91], is based on three core components: a feature extraction module, a temporal segmentation module, and a gesture spotting and recognition module. In the first step, for classification, a 12-dimensional feature vector is extracted from the Xsense data (linear acceleration, angular velocity, and Euler orientation) and Kinect data (relative position between shoulder and hand). The hand position given by the Kinect APIs was not considered to be reliable, and a custom hand segmentation method based on the RGB-D data was applied to replace it. This method exploits a salience map obtained by combining a binary skin color mask derived from the RGB images, a user mask derived from the depth image, the depth map, and its time derivative. The salience probability is higher in the skin color regions for small depths and large depth variations. The gesture spotting method exploits the training data from all the users to create Gaussian models for the ‘rest positions’ and a Gaussian model for ‘non-rest positions’ and then classify the frames accordingly. Gestures are recognized when consecutive non-rest positions are detected for over 0.25 seconds. The gesture recognition exploits multiple Hidden Markov Models (HMM) trained for the three main gesture phases: pre-stroke, nucleus, and post-stroke. The three HMMs are concatenated to form one HMM for each gesture. To detect and correctly segment the nuclei of the gestures, the Viterbi algorithm is applied for finding the most probable hidden state sequence.

Best results on Montalbano Track 3

Table 2.5 shows the best results obtained on the Montalbano Track 3 benchmark. In the following, we report a description of the corresponding methods.

Method	Date	JI (%)
ConvNetsFusion [92]	2021	92.3
Temp Conv + LSTM [93]	2016	90.6
RNN + LSTM Cells [93]	2016	88.8

Table 2.5: Performances of the best methods on the Montalbano Track 3 benchmark

- The **ConvNetsFusion** method, proposed by Wang in 2021 [92], is composed of a temporal segmentation step that determines the beginning and ending frames and a gesture recognition step that assigns the gesture name. The temporal segmentation is obtained by combining two methods; the first method detects the intervals where the height of the hands, segmented with a Faster R-CNN, is above a threshold (corresponding to the values in the initial frames of the video). The second method is based on the RGB-D appearance and uses two-stream ConvNets fed with the RGB and depth channels. In the recognition step, Depth Dynamic Images (DDIs) and Depth Motion Dynamic Images (DMDIs), are generated from a depth sequence through bidirectional weighted rank pooling and are then fed into ConvNets for classification. Saliency sequences for depth and RGB sequences are also estimated using the algorithm described in [94] and are used as input on 3D ConvLSTM networks [95]. The normalized outputs of all four networks are fused with the average-score rule in an element-wise way to obtain the final gesture label.
- Both **RNN + LSTM** and **Temp Conv + LSTM Cells** have been proposed by Pigou et al. in 2016 [93]. The authors explored multiple gesture recognition techniques for video input, proposing several new neural network architectures incorporating temporal convolutions and bidirectional recurrence. In **RNN + LSTM**, they leverage the idea that a gesture becomes recognizable only after a few time steps, so they developed a method that uses a bidirectional RNN, which enables the processing of the sequences (frame by frame) in both temporal directions, with LSTM cells learning dynamic temporal dependencies. **Temp Conv + LSTM Cells** is an extension of CNN layers with temporal convolutions, capable of extracting hierarchies of motion features and capturing time-related information from the input videos. The proposed method’s 3D convolution is factorized into two-dimensional spatial convolutions and one-dimensional temporal convolutions to avoid model overfitting. Pigou et al. evaluated this method using a sliding window with single-frame steps.

Best results on ConGD

Table 2.6 shows the best results obtained on the ConGD benchmark. In the following, we report a description of the corresponding methods.

Method	Date	JI (%)
TD-Res3D + Average fusion [96]	2018	71.63
ConvNetsFusion [92]	2021	69.04
ST-GAT [97]	2022	65.78

Table 2.6: Performances of the best methods on the ConGD benchmark

- **Temporal Dilated Residual ConvNet (TD-Res3D + Average fusion)** is an approach proposed by Zhu et al. in 2019 [96]. This procedure begins with segmenting the continuous gesture sequences into isolated gesture instances with a temporal-dilated Res3D network [98]. The authors employed a balanced squared hinge loss function to handle the imbalance between boundaries and non-boundaries. The temporal dilation preserves temporal information for precise boundary detection, while a large temporal receptive field enhances the accuracy and effectiveness of the segmentation results. The

method uses this segmentation network on both RGB and depth streams. The results are then merged with the average fusion score. The classification network is a combination of a 3D Convolutional Neural Network (3DCNN), a Convolutional Long-Short-Term-Memory Network (ConvLSTM), and a 2D Convolutional Neural Network (2DCNN) for isolated gesture recognition. This architecture is adapted to learn long-term and deep spatio-temporal features, making it effective to recognize the gestures of this benchmark.

- The **ConvNetsFusion** by Wang [92], which is already described among the best-performing methods for Montalbano Track 3, also presents good results on the ConGD benchmark.
- **Spatial-Temporal Graph Attention Network (ST-GAT)** was proposed by Guo et al. in 2023 [97]. The method focuses on extracting essential features from video sequences by considering local details of the hand movements. The approach takes joint and bone information as inputs and constructs a spatial-temporal graph to capture inter-frame relevance and physical connections between nodes. It then exploits a graph-based multi-head attention mechanism with adjacent matrix calculation to effectively explore the local features. Additionally, it models the short-term motion correlation with a temporal convolutional network and uses a bidirectional long short-term memory (BLSTM) to capture long-term dependencies in the video sequences. Finally, it applies frame-by-frame the connectionist temporal classification to align the word-level sequences.

Best results on NVGesture

Table 2.7 shows the best results obtained on the NVGesture benchmark. In the following, we report a description of the corresponding methods.

Method	Date	threshold τ	AUC	TPR (%)	FPR (%)	NTtD	LA (%)*
R3DCNN [99]	2015	0.3	0.93	88	15	0.56	NA
HSTV [79]	2023	0.16	0.81	85	17	0.2158	NA
ResNet+ResNeXt [9]	2020	0.15	NA	NA	NA	NA	77

Table 2.7: Performances of the best methods on the NVGesture benchmark. (*) metric from [9]

- **Recurrent 3D convolutional neural network (R3DCNN)** was proposed by Molchanov et al. [99], in the original paper that introduced the NVGesture benchmark. The method splits the entire video into fixed-length clips, computes the class conditional probabilities set for each clip, and averages them across the modalities. The architecture includes a deep 3D-CNN for spatio-temporal feature extraction, a recurrent layer for global temporal modeling, and a softmax layer for predicting the class-conditional gesture probabilities. The authors also tested a support vector machine (SVM) instead of the softmax layer to compute the final classification score.
- **HandShapeTemporalVariation (HSTV)** is a method proposed by Wannous et al. in 2022 [79]. The authors developed a deep learning-based approach that effectively utilizes the combined description of the hand shape and its temporal variation for hand gesture recognition. To achieve this, they trained a CNN depth image dataset for hand pose estimation, the ICVL [100]. The network transforms the input data into two distinct hand representations for each time step: fine features, representing the pose, and coarse features, roughly representing the hand shape. To model the temporal aspect separately for pose and shape variations over time, they fed features into two different Recurrent Neural Networks (RNNs), RNNfine and RNNcoarse. Finally, they merged their outputs with a joint-fine-tuning method that retrain the two last softmax layers while forcing their sum to represent both networks. The online classification is handled by training the recurrent network with the gesture dictionary and a non-gesture class. The algorithm

return at each frame a class probability and a threshold for considering a gesture classification valid is obtained from the training data with an heuristic trying to maximize true positive and minimize the false positive rates.

- **ResNet+ResNeXt** is a method proposed by Köpüklü et al.(2020) [9]. It consists of a two-level hierarchical structure with a detector and a classifier. The first is a ResNet-10 trained to distinguish gestures from non-gestures. It activates the classifier model when a gesture is detected. The classifier is a ResNeXt-101 [101], chosen after testing several resource-efficient 3DCNNs [102]. The network is fed with the frames in the classifier queue sent from the detector. To prevent misclassification, the authors placed the raw softmax probability of the last detector predictions into a queue to obtain the final detector decisions with a filter applied to the values in the queue. This method is not comparable with the others because its performances are evaluated using only an event-based evaluation metric different from those used in the original paper. For this reason, we added the LA column.

Best results on EgoGesture

Table 2.8 shows the best results obtained on the EgoGesture benchmark. In the following, we report a description of the corresponding methods.

Method	Date	JI (%)	Runtime (FPS)	LA (%)*
TMMF [38]	2021	80.3	NA	NA
C3D [39]	2015	71.8	112	NA
ResNet+ResNeXt [9]	2020	NA	NA	91

Table 2.8: Performances of the best methods on the EgoGesture benchmark. (*) metric from [9]

- **Temporal Multi-Modal Fusion (TMMF)** was proposed by Gammulle et al. in 2021 [38]. It is a single-stage continuous gesture recognition framework learning the natural transitions between gestures and non-gestures without requiring a preliminary segmentation step.
Deep features extracted from windows that slide along the video. Data are processed by individual Uni-modal Feature Mapping (UFM) blocks, composed of temporal convolution layers and multiple dilated residual blocks. The output feature vectors are passed through the fusion block, which selects and extracts temporal features using the attention level parameter to create the fused feature vector. This discriminative feature vector is the input to the Multi-modal Feature Mapping (MFM) block that performs the classification. The multi-modal fusion mechanism is scalable to any number of modes.
- **3-dimensional convolutional networks (C3D)**, proposed by Tran et al. in 2015 [39] indicates a simple yet effective approach for spatio-temporal feature learning using deep 3-dimensional convolutional networks (**C3D**) trained on a large-scale supervised video dataset. C3D uses convolutional layers to extract features from a fixed-length window sliding along the video stream. This choice enables the network to capture spatial and temporal patterns in the data. In detail, the tested network is composed of eight 3D convolutional layers, one 2D pooling layer, four 3D pooling layers, three fully connected layers, and, finally, an LSTM layer. This method was applied in 2018 to the EgoGesture dataset by the benchmark’s creators and obtained the best results among the algorithms tested in the original paper [36].
- **ResNet+ResNeXt** is a method proposed by Köpüklü et al.(2020) [9]. Since this method was tested on the NVGesture dataset, we have already presented it in Section 2.5.3. This method has also been tested on the EgoGesture benchmark, and evaluated using the LA an event-based evaluation metric different from those used in the original paper. For this reason, we added the LA column. This method is not comparable with the others since metrics are different.

Best results on SHREC'19

Different methods have been applied to the SHREC'19 benchmark, and the best results are summarized in Table 2.9. In the following, we report a description of the corresponding methods.

Method	Date	CD (%)	ML (%)	FP (%)	M (%)	T1 Err. (s)	T2-e (s)	T2-s (s)	Time (ms)*	FPR(%)*
OO-dMVMVT [10]	2023	88	NA	NA	NA	NA	NA	NA	5.8	5
uDeepGRU [3]	2019	85.2	7.4	3	4.4	0.54	-1.66	0.66	3.0	10
SW-3 cent [3]	2019	75.6	16.3	2.2	5.9	0.58	-0.7	1.61	1.7	NA

Table 2.9: Performances of the best methods tested on the SHREC'19 benchmark. (*) metric from [10]. CD: percentage of gestures detected in the correct time window and correctly classified, ML: gestures detected in the correct time window and mislabeled, FP: percentage of false positives encountered before the true gesture, M: percentage of missed detections, T1 Err.: average error in estimating the initial mark T1, T2-e: average distance, in time, of the detection frame from the end mark, T2-s: average distance of the detection frame from the start mark.

- **On-Off deep Multi-View Multi-Task (OO-dMVMVT)** refers to a continuous gesture recognition framework proposed by Cunico et al. in 2023 [10], more informations about this method will be discussed on Section 4.4.
- **uDeepGRU**, by Maghoumi et al. is an extension of the Deep GRU approach proposed by the same authors in 2019 [88], used by them to participate in the SHREC'19 contest [3]. It adapts the original method to work in online application scenarios. uDeepGRU is an end-to-end deep learning-based unsegmented gesture recognizer. It features an encoder and a classification sub-network. The former uses Gated recurrent units (GRU), and the latter exploits fully connected layers followed by batch normalization and dropout. To reduce the over-fitting due to the small size of the SHREC'19 training set, the authors removed the attention sub-network used in the original DeepGRU method. To cope with the limited amount of training data, this method also puts a particular emphasis on data augmentation, using four different methods: stochastic resampling (GPSR), Fourier coefficient perturbations, time-series inversion, and rotation. This method obtained the best performances in the SHREC'19 contest.
- **SW-3cent** was the baseline method proposed in [3] and derived from a simple trajectory comparison-based technique (3-cent [40]) applied using sliding window method. The method tests sliding windows of different sizes (the average size of the gestures in the training classes), comparing them with the labeled examples. The comparison involves the resampling of the trajectories to obtain the same number of points using spline interpolation, the application of transforms to superimpose centroids, and scaling to fit a fixed-size bounding box. Finally, the trajectory distances are estimated as the sum of squared distances between pairs of points at the same index.

Best results on IPN Hand

Table 2.10 shows the best results obtained on the IPN Hand benchmark. In the following, we report a description of the corresponding methods.

Method	Date	LA (%)	Model size (MB)	Inference time (ms)
EUREKA [103]	2022	87.5	NA	NA
FASDD-Net [104]	2021	77.26	NA	28
TMMF [38]	2021	68.12	NA	NA

Table 2.10: Performances of the best methods tested on the IPN Hand benchmark.

- **EUREKA** is a method proposed by Peral et al. in 2022 [103], which consists of two main modules: the first focuses on the localization of landmarks in 2D images, and the second predicts the hand gesture class. In this architecture, the Google Mediapipe [29] detector is applied to find the hand landmarks. A scaling- and translation-invariant feature vector is then extracted from the raw landmark positions and used as the input for a densely connected convolutional neural network to classify hand gestures. To recognize dynamic gestures, encoding the spatial changes among landmarks' positions and over time, they proposed a frame selection strategy. For each gesture, they selected 15 keyframes from which they extracted hand positions and three features (Distances, DistAndTime, Dist-Time) to be used as inputs to the neural network. The keyframe selection is not trivial in continuous time, that is why a multiple-sized window approach is executed. This approach consists of taking the model's prediction result of three different-sized windows and giving priority to the window with the higher score.
- **Fast and Accurate Semantic Segmentation with Dilated Asymmetric Convolutions (FASSD-Net)**, is proposed by Benitez-Garcia et al. in 2021 [104]. This architecture employs a lightweight semantic segmentation approach to boost the accuracy of two efficient hand gesture recognition methods: the Temporal Segment Networks (TSN) and the Temporal Shift Modules (TSM). The FASSD-Net is based on the Harmonic Dense-Net architecture (HarDNet [105]) for the real-time semantic pixel segmentation (this architecture can segment pixels into the background, human shape, left and right hands). Two main modules are added to the U-shape encoder-decoder of HarDNet to increase the segmentation performance. Dilated Asymmetric Pyramidal Fusion (DAF) increases the encoder's receptive field, while Multi-resolution Dilated Asymmetric (MDA) fuses and refines multi-scale feature maps that are deeper stages of the network's outputs. To divide the sequences into clips, the authors took their inspiration from the Temporal Segment Network (TSN) [106]; this model presents a two-stream CNN combining RGB-based and Optical Flow-based networks. However, in this work, a single CNN is used (RGB input). Since the model learns frame-wise, it cannot infer any temporal relationship, so the authors added a temporal shift module (TSM) [107] to shift part of the channels along the temporal dimension, facilitating information exchanged among neighboring frames.
- **Temporal Multi-Modal Fusion (TMMF)** is introduced by Gammulle et al. in 2021 [38]. Since this method was tested on the EgoGesture dataset, we have already presented it in Section 2.5.3; this method has also been tested on the IPN-Hand benchmark, but the results are significantly worse than those obtained using the best methods.

Best results on Mlgesture

Table 2.11 shows the best results obtained on the Mlgesture benchmark. In the following, we report a description of the corresponding methods.

Method	Date	mAP (1-frame latency)
MC TCN [4]	2020	0.83
BI-LSTM [4, 108]	2020	0.74

Table 2.11: Performances of the best methods on the Mlgesture benchmark for the low-latency continuous detection task

- **Mixed-Causal Temporal Convolution Network (MC TCN)** is proposed by Maarten et al. in 2020 [4]. This method exploits a 2D CNN to generate an embedding vector for each video frame. The temporal domain is then modeled using a 1D TCN. The fundamental block of the architecture, called BB, is formed by concatenating multiple TCN basic blocks. The first layer of BB consists of a 1D dilated convolution with a kernel size of $k = 3$. The dilation factor is doubled for each subsequent BB, following a pattern

of 1, 2, 4, 8, and so on. This exponential growth in dilation allows the network’s receptive field to expand significantly. A group of consecutive BBs forms a stage. To create deeper networks, multiple stages can be stacked on top of each other, and the dilation factor is reset to 1 at the beginning of each new stage. This design enables the network to capture temporal dependencies effectively by increasing the receptive field through dilated convolutions in the TCN, facilitating better recognition and understanding of complex spatio-temporal patterns in video data. To boost the performance for predictions close to the right edge, the method uses causal convolution, which only looks at present and past activations, combined with a regular (non-causal) dilated convolution. For this reason, the configuration is called Mixed-Causal. The authors also proposed a variant of the architecture employing the SqueezeNet V1.1 [109] spatial encoder instead of the ResNet18, resulting in a large reduction in the number of parameters with limited loss in accuracy. In the proposed online recognition framework, MC TCN is used in a sliding-window based fashion over all frames in the video.

- **Bidirectional LSTM Networks (BI-LSTM)** The previous method is the top performing for the continuous task in the paper presenting the benchmark [4]. Among the other methods compared in the paper, the second best was Bidirectional LSTM Networks (BI-LSTM), proposed by Rocha et al. in 2005 [108]. In the bidirectional approach, inputs are processed in two directions: from past to future and from future to past. This bidirectional approach provides a more comprehensive understanding of the sequence and contributes to the model’s enhanced performance. The authors of [4] used BI-LSTM with the same sliding window protocol adopted for MC TCN and other classifiers for the online recognition.

Best Methods for LD-ConGR

Table 2.12 shows the best result obtained on the LD-ConGR benchmark. In the following, we report a description of the corresponding method.

Method	Date	JI (%)
ResNeXt-MMTM [103]	2022	34

Table 2.12: Performances of the best method tested on the LD-ConGR benchmark.

- **Resnet-Xt Multimodal Transfer Modules (ResNeXt-MMTM)** is the model proposed for the benchmark by Liu et al. in 2022 [5]. The authors built a baseline method on ResNeXt-101 [101] and tested the recognition performances on single (RGB or depth) input modalities. Additionally, they developed a multimodal fusion model ResNeXt-MMTM inspired by [110] fusing the features of different modalities at multiple layers through Multimodal Transfer Modules (MMTMs). MMTM learns a multimodal embedding and fine-tuning the features of each modality. In their architecture, features extracted from both the RGB and depth streams are combined by the MMTM after each ResNeXt block. After passing the fully connected layers, the feature vectors are integrated with an element-wise addition and passed through a softmax layer to have the prediction of the result. For the continuous recognition test, the classifiers are used on a 32-frame window sliding over the video sequence (with a temporal stride of 2 frames).

Best results on SHREC’22

In Section 3.6, we show the best results obtained on the SHREC’22 benchmark, and we report a description of the corresponding methods.

Best Methods for ODHG

Table 2.13 shows the best results obtained on the ODHG benchmark. In the following, we report a description of the corresponding methods.

Method	Date	AUC (%)	TPR (%)	FPR (%)	NTtD
HSTV [79]	2022	91	85	17	0.2104

Table 2.13: Performances of the best methods on the ODHG benchmark

- **HandShapeTemporalVariation (HSTV)** proposed by Wannous et al. [79] and already described in 2.5.3 was applied to this benchmark by their creators. Also in this case the authors tuned a probability detection threshold based on the ROC curve plot, obtaining the results plotted in Table 2.13. The false detection rate is clearly too high for a practical use of the system.

Best Methods for ZJUGesture

Table 2.14 shows the best results obtained on the ZJUGesture benchmark. It is necessary to note that since the ZJUGesture benchmark was introduced in 2022, the only available results are those presented in the original paper. In the following, we report a description of the corresponding methods.

Method	Date	LA (%)	Speed(seg/s)	GFLOPs	Parameters(M)
MotionNet + GestureNet [7]	2023	92.4	173	2.0	10
P3D ResNet [111]	2017	78.2	57	15.9	25
C3D [39]	2015	76.1	28	129.9	99

Table 2.14: Performances of the best methods on the ZJUGesture benchmark

- **MotionNet + GestureNet**, is an online lightweight two-stage framework for the detection and classification of dynamic gestures proposed by the creators of the benchmark [7]. The architecture is composed of a detection module and a gesture recognition module. Input videos are pre-processed by using a sliding window with length 4 and stride 2. Differential images are also estimated and concatenated to the RGB images. The detection module, MotionNet [112], aims to determine if there is a gesture execution in the raw video stream and then sends the isolated gesture to the recognition module. Their gesture classification module, GestureNet [113], is a temporal relation reasoning network that aims to process the cleaned gesture sequences and identify them. Furthermore, they have added a state machine to process the results of the gesture recognition network so that each classified gesture responds to a single-time prediction.
- **Pseudo-3D Residual Net (P3D ResNet)** proposed by Qiu et al. in 2017 [111], is a network architecture that leverages all the variations of blocks found in ResNet [114] but arranges them in different positions within the architecture to enhance the structural diversity of the network. Pseudo 3D CNN reduces the model size of a ResNet and enables the pre-training of 2D CNN from image data, increasing the leveraging power of the knowledge of scenes and objects learned from images. The original method only supported the trimmed video clips containing a single gesture, so the ZJUGesture’s authors used their detection module to cut the video into small sequences and pass them through the P3D ResNet.
- **3-dimensional convolutional networks (C3D)** which is already described in Section 2.5.3, is also tested in [7] on the ZJUGesture benchmark. The Levenshtein accuracy is lower than those obtained with the previous techniques, but the method is significantly faster than the others.

2.6 Data Augmentation

3D gesture recognition is challenging due to the complexity of movements in 3D space that involve varying trajectories, speeds, and orientations, making accurate capture and

interpretation difficult. Gestures can vary significantly between individuals in terms of style and physical characteristics [35]; they can also differ from one execution to another, even when the same user performs them, due to tiredness, time constraints, and body pose.

Gesture recognition is typically performed by using classifiers trained on a limited number of examples executed by a few subjects. A possible preprocessing tool to solve this problem is *data augmentation*, which can enrich the training data with diverse variations of labeled gestures, significantly enhancing the results. Additionally, data augmentation can mitigate overfitting by exposing the model to a broader range of inputs, allowing it to learn more effectively from the available data.

Generating synthetic examples of gestures with perturbations of the existing training data is easy. We can change the speed, position, orientation, size, or appearance of the hand, and the additional training data obtained can improve the classification results, simulating the gesture executions of different subjects. The training of recognizers with augmented sets has been proposed in video-based gesture recognition works [115–118] to improve the robustness with respect to color, illumination, and background variations, on 2D gesture recognition based on finger trajectories [119] and some approaches have been recently proposed for non-continuous skeleton-based recognition [120].

In the continuous domain, however, the real problem is different, and it is how to fill the non-gesture space with synthetic data. The characterization of sequences labeled as non-gesture and the generation of synthetic data of this kind of data is complex, as sampling the non-gesture space in the few sequences typically provided in the benchmarks is poor. The use of limited examples of non-gesture in the training phase is one reason for the non-negligible amount of false positives obtained by continuous recognizers. To the best of our knowledge, the effect of augmentation on continuous gesture recognition has not been studied in the literature. A key characteristic of the task is that gestures should be distinguished from non-gesture movements and then classified in continuous recognition. This can be obtained by adopting a sliding window approach or separate segmentation modules, which is not easy in the case of heterogeneous gesture types. The sliding window approach continuously classifies fixed or variable size gesture segments using methods that are trained to distinguish gesture subparts from "non-gestures" windows [38–40].

Gesture data streams described by keypoint trajectories (2D fingertips or 3D skeleton points) can be augmented with two approaches: classical and generative. Classical methods perturb the points in the sequence, while generative models learn the underlying distribution of the motions to produce new gestures. Generative approaches include Generative Adversarial Networks (GANs) [121–123] and non-adversarial networks [124]. Both networks require training a large model before creating new samples, meaning that a large initial gesture dataset is required. Thus, they are not applicable in customization scenarios with a small initial dataset, in which classical augmentation methods are still preferred.

The well-known classical augmentation methods are focused on trajectory perturbation. These augmentations include adding noise, interpolation, scale, rotation, translation, and other transformations commonly defined by matrix multiplications [125, 126]. Adding noise is a data augmentation method applicable to skeletal gestures [124, 127], and it was shown to increase performance over a baseline with no augmentation. We consider the combination of spatial and temporal resampling into an interpolation category [120, 128–132], because varying the number of frames in a continuous dataset while maintaining a constant FPS ratio alters the perceived speed, potentially resulting in a bad sample. Scaling augmentation can help models adapt to varying sizes of executed gestures, improving the rate of recognition [120, 128]. Methods that rotate trajectory points are commonly used in the literature [133, 134] since a gesture can be executed from different angles. However, hand gestures can be recognized based on the hand trajectory combined with hand pose, so besides the trajectory augmentation method, it is possible to create synthesized samples that augment the pose of the hand. In this work, we applied the global hand pose rotation creating a new type of augmented data. Finally, in Shen et al. [120] to change the hand’s joint pose, a random number of finger joints are selected, and the Gaussian shift is applied as noise. In our work to keep the correlation between linked finger joints, we decided to add the same random shift to every finger.

A study from 2021 tested skeletal augmentation with a custom LSTM-based recognizer [133] and found that augmenting data by reversing the trajectory and applying a shearing operation increased accuracy for the NTU RGB+D dataset. Other work applied chains of different augmentation techniques to single-2D joint time-series data [119] and found a significant performance increase. As in these studies, our intent is not to generate realistic samples, but to increase the system’s final accuracy through unsupervised augmentation methods. Another recent study of various skeletal data augmentation techniques included spatial and temporal augmentation [135]. They found that some of these techniques improve recognition when using CNNs, LTSMs, and CNN-LTSMs. However, they only tested the different techniques separately and not the union of synthetic gestures generated by all methods as is done in this paper. Notably, in their experiments, the performance of CNNs decreased below the baseline when using cropping, translation, scaling, and projection. In the paper by Rao et al. [133], rotation and Gaussian noise are used similarly to our approach. However, their work utilizes a Kinect dataset from a virtual reality environment, whereas we use the SHREC’22 dataset, which is recorded in a mixed-reality environment. In Chapter 5, we investigate the effectiveness of augmentation techniques for the online/continuous detection of heterogeneous gestures (e.g., including static and dynamic gestures characterized by changes in hand position and articulation) that can be used to interact in an XR setting. We leverage a recent benchmark designed for this task, SHREC’22 [6], testing how augmentation methods can improve the online detection performances through the specific criteria introduced by this benchmark.

2.7 Discussion

The outcomes of [17] are significant and interesting and show that while gesture recognition is a hot topic in the literature and promising results are being obtained in the field, there is a need for new efforts aimed at creating benchmarks and evaluation methods to test recognition systems designed for interactive applications in realistic settings. In this section, we discuss the characteristics of the surveyed benchmarks in Chapter 3 and Chapter 2, and the evaluation results, pointing out the main issues found in our analysis and trying to propose new research directions for future work that take inspiration from them.

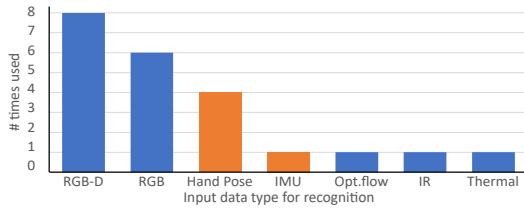


Fig. 2.14: (a)

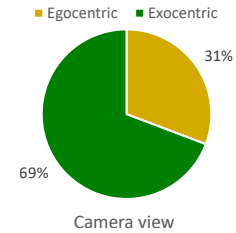


Fig. 2.15: (b)

Fig. 2.16: (a) Data types used as input for the recognizers in the different benchmarks. Blue bars indicate video streams, and orange bars key point data streams. Most of the benchmarks are based on videos. (b) Most of the benchmarks feature data acquired from an exocentric view.

2.7.1 Characteristics of the benchmarks

Few benchmarks are available

Despite our extensive search, we did not find many benchmarks, and it is clear that more effort is needed to allow the scientific community to compare different approaches for continuous gesture recognition.

Looking at [Figure 2.14](#), it is evident that the data mostly comprises of RGB-D images, and there are no public benchmarks for continuous recognition based on gloves or sEMG data. There is a reasonable number of benchmarks with hand skeleton data. This feature is helpful, as XR headsets and other devices directly provide the skeleton data, resulting in a drastically simplified input for gesture recognition. However, the related datasets feature a limited number of subjects, which may be a relevant issue for generalizing their results. A further interesting aspect of the skeleton-based benchmarks, not covered in the literature, is the investigation of how much the recognition results are affected by the inaccuracy of the hand pose estimation. Skeletons are mostly obtained with computer vision algorithms applied to RGB, IR, or RGB-D images. While different finger trackers have been evaluated and compared for generic accuracy performance [28], it is not apparent that by how much gesture recognition is robust against pose estimation errors. The hand tracking performances may heavily depend on the position/orientation of the hand to the sensor, occlusions, and illumination; this may limit the accuracy of the recognition in practical settings, even if the systems work well when the pose estimation is correct. By looking at [Figure 2.15](#) it is apparent that most of the benchmarks are recorded in Exocentric view, and all of them use cameras for recording (In [1] IMUs are used in addition to the cameras). A more significant number of egocentric datasets with different dictionaries would be helpful to assess occlusion issues for gestures captured from this viewpoint.

A possible explanation for the limited number of benchmarks and their weaknesses is the fragmented research community working on this topic. Most of the benchmarks (8) have been proposed in conferences and journals of Computer Vision and Pattern Recognition (where algorithms' benchmarking is quite popular), others (2) have been proposed within the Computer Graphics community, and only one has been introduced in an interaction focused journal.

One possible reason for the HCI community's limited effort is that using fixed dictionaries is a strong constraint for interaction designers, and a viable alternative would be to create a system enabling them to interactively generate and update sets of gestures to be recognized at runtime. Some recent research papers describe frameworks designed for this task [136–138]. This approach follows ideas that were successfully applied to 2D gesture-based interface design. However, testing proposed methods using realistic scenarios and standardized measurements for tasks that involve recognizing complex gestures is important. This approach helps in selecting the most effective and optimal gesture recognition methods.

Number of subjects, subjects' diversity

As shown in [Figure 2.17](#), only three surveyed datasets present a large number of subjects that executed the gestures (> 50). A limited number of subjects and the lack of precise characterization of the participants, such as age and cognitive and spatial ability, limit the generalization of the gesture recognition results to any target users.

Benchmarks' data should be captured on a large set of subjects representative of a generic population and collected with associated metadata, allowing an analysis of recognizers' performance for different subsets of target users. Gestures can be executed in different ways and with different speeds, and even the same person may change the execution due to varied physical or mental states. It is challenging to capture datasets with many subjects of different conditions, collecting all the related metadata. However, the availability of benchmarks with this variety would enormously help the development of new methods with good usability in the wild. None of the considered benchmarks evaluate the per-subject performances of the methods. This feature is essential to assess the potential gesture-based interface's accessibility regarding the recognizer. While considering bio-mechanical hand functions ergonomics [72], evaluating gesture dictionaries is important, especially when subjects must perform many hand poses/movements. In this situation, it is possible that the subjects cannot easily execute some of the gestures due to individual limitations. The collection of data from a large variety of subjects could also be exploited to reason about the accessibility of gestural interfaces for people with different types of abilities.

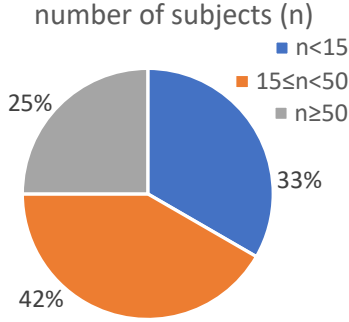


Fig. 2.17: (a)

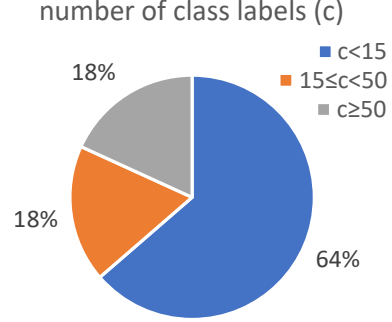


Fig. 2.18: (b)

Fig. 2.19: (a) Most of the benchmarks feature a relatively small number of gestures classed in the dictionary (< 15), even if typically of heterogeneous types. (b) The number of subjects performing gestures is extremely variable.

Limited or missing class-related analysis and evaluation of dictionaries

Most of the surveyed papers, including those that describe contests or introduce a benchmark, present a limited analysis of the results. In particular, a limited number of conducted research presents a per-class results analysis focusing on the effectiveness of the proposed methods for the different gesture labels. Some of the benchmarks feature a relatively high number of gesture classes (see Figure 2.18), and evaluating how the methods perform for individual or clustered labels could provide quite interesting results.

The performance analysis on different gesture categories (static, dynamic coarse/fine) have been presented in [137], and the obtained results are of interest since they show that different methods can perform better for different categories. Researchers can exploit these outcomes to design improved techniques by combining multiple recognizers. In [77], the dictionary consists of related classes subsets to different application domains that enable the assessment of the performances for a particular use.

However, from the interaction design point of view, an interesting research direction could be related to selecting optimal gesture subsets that are less prone to errors and more suitable for reliable user interface design. This important aspect should be considered to improve the results obtained on the benchmarks, which are far from optimal, as discussed in Subsection 2.7.2.

Evaluation metrics are not standardized

As discussed in Section 2.4, the continuous task can be evaluated with many different metrics. The surveyed benchmarks use metrics that are all different from each other. Looking at Table 2.4, Table 2.12 and Figure 2.20, we see that many benchmarks evaluate continuous gesture detection using frame-based metrics only; also, the Jaccard index, which is a frame-based metric is the most used one. We consider this to be a non-optimal choice, as the frame-based metrics do not test missed or repeated detection and present relevant biases in evaluating gestures of different time lengths.

We emphasize that evaluating false positives is as important as the classification accuracy for verifying the methods' suitability for practical use. Therefore, event-based metrics are also necessary since they are useful to estimate the feedback delay. However, since their values depend on thresholds (related to the detection criteria), a standardized way to define the "correct gesture detection" would be necessary. In practical use, continuous recognition algorithms run on machines with different hardware resources, and evaluating the computational requirements of the methods is also essential.

A reasonable design guideline for creating a continuous gesture recognition benchmark is to provide multiple evaluation metrics for correct event detection, false positives, detec-

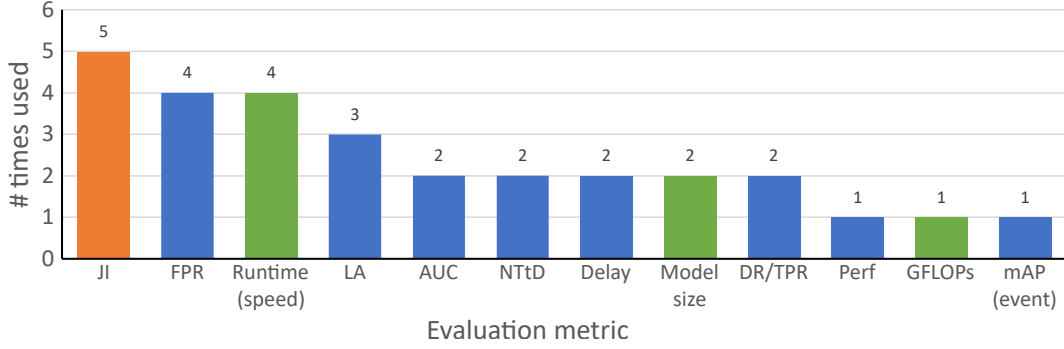


Fig. 2.20: Occurrence of the different evaluation metrics used to assess the method’s performances in the 12 benchmarks reviewed. JI=Jaccard index, FPR=false positive ratio, AUC=area under receiver-operator characteristic curve, NTtd=normalized time-to-detect, LA=Levenshtein Accuracy, mAP=mean area precision. Orange bars indicate frame-based metrics, blue bars indicate event-based metrics, and green bars are efficiency-based metrics.

tion delay, and hardware requirements. Standardizing the metrics and the gesture detection criteria would also be helpful.

2.7.2 Performance of gesture spotting methods

Results are not really good

The best results reported in [Subsection 2.5.3](#) indicate that the number of missed gestures and false positives is significant for all the benchmarks.

In many potential applications of gesture recognition, we cannot accept a detection rate of 90% or a false positive rate of 10% because, in an extended interaction where many gestures need to be recognized, a huge amount of actions would not be activated or would be wrongly activated. Note that this issue is only captured by performing a proper event-based evaluation of the methods. In some cases, these bad results depend on the fact that benchmarks are old and not many recent techniques have been tested on them. Another reason is that few authors work on continuous tasks, whereas many papers only focus on segmented tasks. A further explanation for the sub-optimal results may be due to the challenging dictionaries, often including heterogeneous gestures with different durations and semantic characterization (static, dynamic, coarse, and fine). Specific strategies could be developed to cope with this issue, for example, using different modules to detect different gesture types. Papers often report the results only as averages across all the classes in the dataset. For specific subsets of gesture labels, the accuracy in detection and the false positive ratio can be much better, meaning that it is possible to exploit these subsets to develop more reliable and usable interfaces. We did not find this kind of analysis in the reviewed works.

Comparison of different approaches

The analysis of the best-performing methods in the surveyed benchmark suggests that the two most popular approaches to address online detection, the first based on two modules (temporal detection and classification), and the second avoiding a preliminary segmentation and labeling frames or intervals with a sliding window approach, are used essentially with the same frequency. Looking at the results, there is no firm evidence of the advantages of one of the approaches, and this aspect needs further investigation. While two-module methods perform best on most of the benchmarks, in some, e.g., SHREC’22, the best method uses a single classifier model with a sliding window approach. Another relevant fact is that, in the latter work, the sliding window has been trained and is tested with fixed-length windows, meaning that the frame labeling relies on recognizing sub-parts of the gestures. This approach was successful and allows early detection with a bounded delay but could

have problems in the case of dictionaries with long gestures containing similar sub-parts. In other methods, such as EUREKA [103] or SW-3cent [3], that use both sliding windows or segmentation modules, the classifier is instead trained on complete gesture templates and tested on variable-size windows. This approach was the most successful on IPN Hand [103].

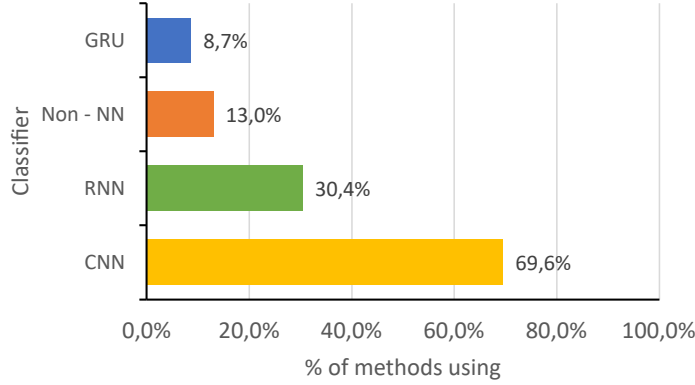


Fig. 2.21: Occurrence of different classifier types in the 23 methods performed applied on the proposed benchmarks and reviewed in our work.

CNN is the most popular choice (Figure 2.21) concerning the used classifier type for the detection/classification modules. One important reason for CNN models' popularity is that they are faster than other methods, decreasing delay time. Using recurrent networks, transformers, or graph networks to handle skeleton data does not provide relevant advantages and may increase the complexity of the architecture. This outcome may be due to networks' training difficulty with limited data. It is also worth noting that, even if most techniques are based on neural networks, the models' input does not always consist of the raw data. In many cases, especially for skeleton data, the input of the networks is not the original data sequence but a stream of handcrafted features derived from them (such as distances of finger key points or speeds).

Continuous vs. segmented benchmarks

As seen in Section 2.1, most of the methods proposed in the gesture recognition literature only focus on segmented gesture classification and are tested on related benchmarks like SHREC'17 [43], VIVA [41], LAP RGB-D Isolated Gesture Dataset (IsoGD), Jester [44]. We did not consider them in our survey as we focus on continuous evaluation, but we must recognize this large amount of literature. It is evident that a large number of Computer Vision researchers prefer to develop and test methods for this simplified task, and this is also evident considering that the most cited benchmark papers analyzed here are those including both a continuous and a segmented task evaluation (NVGesture, Montalbano, Egogesture). Most of the works citing those papers only worked on the segmented task. The reason for this is probably because it is easier to train classifiers for the segmented tasks and evaluate the results, comparing them with state-of-the-art, even if we need to cope with the continuous recognition in many practical applications. The outcome of the works dedicated to the segmented task is interesting for us, as the "continuous" and "segmented" tasks are obviously linked. We can take the best methods for segmented gesture classification and use them continuously by adding a sliding windows approach [82] or a gesture segmentation module. This procedure, however, is not straightforward. The training approach must be specifically designed for the continuous task, independently of the testing approach, as the non-gesture sequences interleave the gestures' data with non-meaningful hand movements that may be highly variable and present significant differences in training and test data. The

specific training procedures must cope with a considerable class imbalance (non-gesture is more represented). The false positives issue needs to be addressed with care. It is impossible to understand the performances of the classifiers in a realistic continuous setting based on the classification results using the segmented benchmarks.

Adapting classifiers that are used for segmented tasks to a continuous setting has been proposed in [10]. Using a sliding windows approach, the authors adapted the best methods on the DHG14-28 and SHREC'17 benchmark for working online on the SHREC'22 task. They had to retrain the networks on fixed-length windows, including a non-gesture class. Other efforts of this kind could be beneficial to develop better recognizers.

Benchmarking Continuous Hand Gesture Recognition

The **Shape Retrieval Contest (SHREC)** is an annual competition that focuses on evaluating and comparing methods for shape analysis and retrieval in various domains, including computer vision, computer graphics, and related fields. Since its origin in 2006, SHREC has expanded and has welcomed related evaluative works on classification, correspondence, and segmentation.

During my PhD research, I contributed to the SHREC'22 online gesture recognition benchmark [6], where I was involved in its design and organization, **this chapter is adapted from that benchmark [6]**, in which I detail the design, contributions, and outcomes of SHREC'22, highlighting its impact on advancing gesture recognition research. The SHREC'22 benchmark focuses on single-hand gestures and introduces a diverse gesture dictionary evaluated in mixed-reality environments using the HoloLens 2 hand tracking system. It emphasizes key metrics like false positive rates and recognition latency, which are often overlooked in previous benchmarks.

3.1 SHREC 2022

To build Gestural interfaces, it is necessary to implement effective recognizers that are able to cope with a sufficiently large dictionary of gestures of different types (hand motions, hand articulations, static poses). To evaluate the potential effectiveness of these algorithms, it is not only necessary to test the ability of the methods to classify segmented gestures but also to test how well they avoid false detections and which is their detection latency.

As we already discussed in [Chapter 2](#), not many of the existing benchmarks are optimally suited for this, so we created a novel one and organized a related contest. The main contributions of the dataset, task, and evaluation method described here are:

- The benchmark is specifically designed for generic mixed reality interaction development and directly captured with the hand tracking system of the HoloLens 2 headset, therefore featuring the same viewpoint, field-of-view, and temporal resolution. The methods tested on this benchmark could directly be applied in an interactive application developed on the same platform, locally, or via a client-server architecture, depending on the computational complexity.
- The benchmark features heterogeneous gestures, including *static* ones (fixed hand pose for at least 0.5s), *dynamic coarse* (characterized by whole hand trajectory), *dynamic fine* (where the semantic also depends on finger articulations), and *periodic* (with repeated movements) gestures. With respect to the only previous benchmark for online gesture recognition (SHREC 2021 [8]), the dictionary has been changed, removing ambiguous classes, avoiding annotation issues affecting that dataset and introducing the class of periodic gestures.
- The evaluation measures online performances, not only considering non-gestures and false positives but also evaluating the recognition latency, which is an important factor in assessing their usability in an interactive application.



Fig. 3.1: Gesture acquisition: the subject hears vocal commands suggesting pre-defined sequences of non-significant movements and gestures.

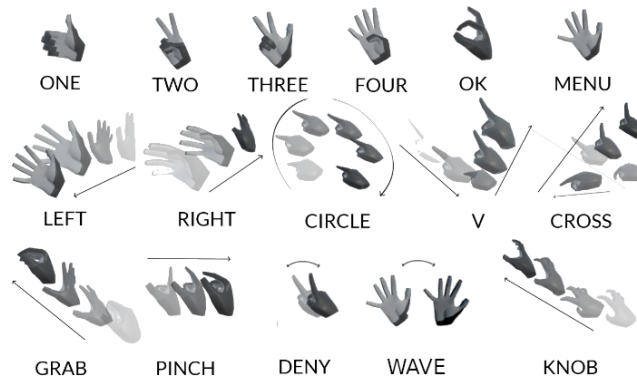


Fig. 3.2: The 16 gestures in SHREC'22 vocabulary: static (top row), dynamic coarse (middle row), dynamic fine (bottom left) and periodic (bottom right).

The task proposed in the SHREC 2019 track on online gesture detection [3]) is to find gestures in hand skeleton sequences, thus addressing the problem of avoiding false positives, but the dictionary was limited to simple dynamic gestures characterized by the mere hand trajectory. The task proposed in the SHREC 2021 track on Skeleton-based Hand Gesture Recognition in the Wild was better suited to test the potential use for XR interaction of complex gestures, as it featured static, dynamic-coarse, and dynamic-fine gestures and used an evaluation method based on detection rate, false-positive score. However, the dataset has some weaknesses: first, some gestures are ambiguous, as we found that, at the end of the execution of some dynamic gestures, there are static poses of the hand similar to the "pointing" gesture included in the dictionary but not annotated. Furthermore, the annotation of many dynamic gestures was limited to a very short segment of the whole hand movement. Finally, the evaluation did not consider the recognition latency.

3.2 Novel dataset, task and evaluation

For the reasons we mentioned, we created another dataset to try to overcome the issues of the previous ones and use the data captured by the head-mounted display directly.

The dataset is composed of 288 sequences, including a variable number of gestures (i.e., 3 to 5), divided into two subsets of the same size: a training set, with annotations provided to participants (start and end frame of the gestures with related label), and a test set where the gestures have to be found according to the task requirements.

Fingers data were captured with a Hololens 2 device simulating mixed reality interactions. The gestures in each sequence are interleaved with other hand movements. In the acquisition session, the Hololens 2 app was programmed to ask the subjects (with a vocal

	ONE	TWO	THREE	FOUR	OK	MENU	LEFT	RIGHT	CIRCLE	V	CROSS	GRAB	PINCH	DENY	WAVE	KNOB
Min length	21	24	15	23	22	20	11	11	25	17	24	19	22	31	27	37
Max length	72	59	71	69	52	59	37	32	64	37	50	70	61	81	73	79
Avg. length	34.1	37.4	38.6	37.3	34.0	33.0	19.6	18.6	45.4	27.2	37.5	39.9	36.2	47.8	45.4	54.7

Table 3.1: Minimum, maximum and average lengths of the gestures (background colors indicate gestures’ types).

command) to start specific gestures at randomized time frames (Figure 3.2). These time frames were also recorded, and, for the training set, they were given as additional data to the participants.

During the remaining time, the subjects were instructed to keep their hands in the field of view of the tracking system and to freely move the hands avoiding movements in conflict with the actual gestures from the dictionary.

The sequences have been designed so that both the training and the test set include the same number of instances of each gesture class (36).

Time sequences were saved as text files where each row represents the data of a specific time frame with the coordinates of 26 joints. Each joint is, therefore, characterized by 3 floats (x,y,z position). The frame rate of the acquisition is relatively low and not perfectly stable (approximately 20 fps), making the recognition scenario even more realistic, considering an unstable frame rate is a common condition for real applications running on stand-alone devices with limited performances. Frame data include, however, also the time stamp of the recordings, making it possible to resample the joint positions at a constant rate.

The gesture dictionary used in this work is similar to the one proposed in SHREC 2021, but with a few important modifications. Specifically, some gesture classes, including POINTING, TAP, and EXPAND, have been removed. This decision was driven by an in-depth analysis of the dataset and annotation process.

One of the key observations was that several dynamic coarse gestures, characterized by distinctive hand trajectories, often concluded with a static POINTING pose that persisted for several frames. This overlap made it difficult to reliably distinguish between gestures based solely on final hand positioning. Ideally, these poses could have been separately labeled as POINTING gestures, but doing so would introduce a high level of ambiguity, particularly in real-time recognition scenarios. Similarly, TAP and EXPAND gestures exhibited characteristics that closely resembled short segments within other gesture classes. Their brief duration and limited movement complexity increased the likelihood of misclassification, as gesture recognition models could mistakenly identify parts of other gestures as TAP or EXPAND. While it is theoretically possible for gesture recognition models to disambiguate such cases using contextual information, we opted to focus this benchmark on the recognition of more distinct, less ambiguous gesture patterns. This ensures a more reliable evaluation framework while leaving the challenge of context-based disambiguation for future research. We also found that the annotation of the dynamic-fine gestures (PINCH, GRAB) in SHREC 2021 did not include the initial global hand movements. In the novel dataset, we annotated those frames as belonging to the gesture as we consider them as characteristic of the gesture classes.

As shown in Figure 3.1, we included 16 gestures divided into 4 categories: *static* characterized by a pose kept fixed (for at least 0.5 sec), *dynamic coarse*, characterized by a single trajectory of the hand, *dynamic fine*, characterized by fingers’ articulation, *periodic*, where the same fingers’ motion pattern is repeated more times. For each of these categories, the features actually determining the related semantics are different, and, in principle, it would be possible to develop specialized approaches for their classification. None of the participants, however, considered this option.

Table 3.1 shows the minimum, maximum, and average length of each gesture in the training set. It is possible to see that the static gestures are kept for variable times. Their semantics, however, do not depend on the full length. We expect, therefore, that they can be recognized just after the start frames. The algorithms should, however, avoid multiple detections within the annotated interval.

Dynamic gestures have a variable time length, and their semantics are, in principle, dependent on the full duration, with the exception of periodic ones that may be fully characterized by the first occurrence of the repeated sequence.

3.2.1 Task and evaluation

Participants were asked to create an online classification method based on training sequences and to process the test sequences, not only annotating the predicted start and end of gestures but also indicating the last frame of the sequence used to perform the prediction, giving the indication of the minimal classification delay (not including the algorithm’s execution time). The evaluation was automatically performed with a script counting the number of correct gestures detected, the false positive score, and the minimal detection delay. A detected gesture is considered correctly detected if it has the same label as the ground truth annotation and its time window has an intersection with the annotated one larger than half the ground truth length. We call *detection rate* the ratio between the number of correctly detected gestures of the class and the corresponding ground truth number. The *false positives’ score* is the ratio between the number of gesture predictions not corresponding to real ones (e.g., with no intersection with annotated ground truth gestures) and the total number of gestures of the same class included in the test set. The script also evaluates the Jaccard Index (JI), e.g., the average relative overlap between the ground truth and the predicted annotations of frames belonging to each gesture class in the sequences. This metric had been employed in other online action and gesture recognition contests [139, 140].

An important novelty of our benchmark is the evaluation of the detection delay, estimated as the difference between the actual gesture start and the reported timestamp of the last frame used for the prediction and is related to the algorithmic detection strategy. Given this estimate, it is possible to evaluate the average delay of the gestures’ detection with respect to the actual (ground truth) starts as well as the time differences with respect to the gestures’ ends. The analysis of these values gives interesting hints about the usability of the proposed methods for practical applications. To complete the temporal data analysis, we also collected the computational time for a single classification step from the participants (i.e., how much time each method takes to elaborate data for a single classification attempt).

3.3 Participants and proposed methods

Three groups were registered for the contest and submitted up to three results files obtained with different classification strategies as well as the required additional information on the algorithms’ runs. We compared their results using the previously described methods against a baseline method, a modified version of STRONGER [16], an online recognizer based on 1D convolutional neural networks.

3.3.1 Group 1: Two-stage ST-GCN (2ST-GCN)

Method Description To address the task proposed for this task effectively, Group 1 constructed their method by adapting a two-stage object detection model: the R-CNN family. Figure 3.3 illustrates the workflow of the two-stage hand gesture detection architecture.

First, the method leverages the idea of a sliding window with a small window size striding along the sequences to propose gesture candidates. Those candidates serve as inputs to a tiny classification model which results in categories of each short sequence. Next, it extends these sequences to a larger size and feeds them to a large model with both classification and localization branches. Finally, fine-tuning and matching of the outputs of tiny and large models are performed to return the final results.

Gesture Proposal Module Group 1 proposes the Gesture Proposal Module, which comprises the energy-based sliding window and a tiny classification. They borrow the idea of the energy-based function from [8] to leverage the shift of every joint of human hand

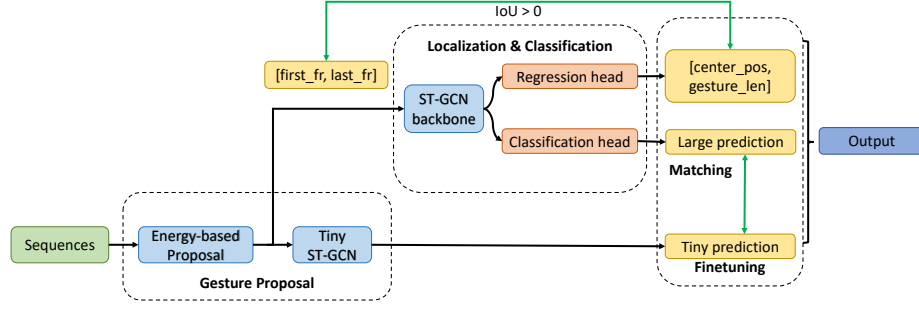


Fig. 3.3: The workflow of our two-stage hand gesture detection architecture.

over sequences of consecutive frames and calculate the amount of energy accumulated in a window with a size of l

$$E(w) = \sum_{j=1}^N \sum_{t=1}^l \frac{\|w_{j,t} - w_{j,t-1}\|}{\|w_{j,t-1}\|}$$

$$w_{j,t} = [x_{j,t}; y_{j,t}; z_{j,t}]^T$$

$$\|w_{j,t}\| = \sqrt{x_{j,t}^2 + y_{j,t}^2 + z_{j,t}^2}$$

where $w_{j,t}$ is 3D coordinates of finger-joint j at time step t and N is the number of joints in human-hand.

Next, they utilize a sliding window with a small size $l = 10$ to stride over the entire sequences

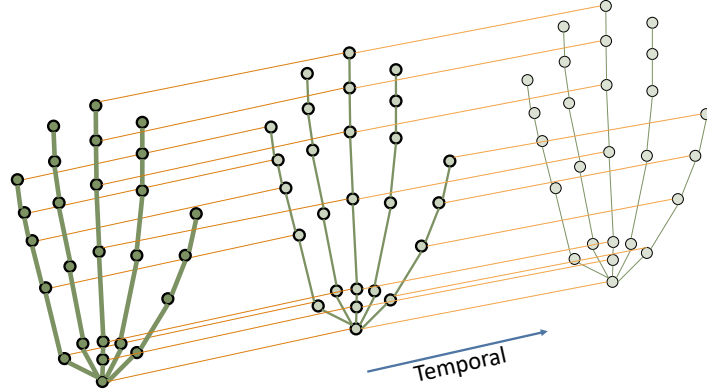


Fig. 3.4: Spatial-temporal graph neural networks for hand gesture recognition.

with $\text{step_size} = 1$ and calculate movement energy at each step. A potential segment w_i is chosen if its energy is higher than average energy from w_0 to w_{i-1} .

In the following step, those 10-frame long sequences are fed into a tiny classification model for early prediction of the gestures category. Based on the structure of the dataset, which includes 3D trajectories of finger-joints, the authors consider it reasonable to follow the natural structure of human hands and represent them as graphs containing both spatial and temporal information. Therefore, Group 1 applies Spatial-Temporal Graph Convolutional Networks (ST-GCN) [141] with the purpose of learning patterns embedded in the spatial configuration by exploring the locality of graph convolution as well as temporal dynamics. Spatio-temporal links are represented in Figure 3.4. The ST-GCN module outputs categories

along with confidence scores of each short segment. The training of this network module is based on a 10-frame sequence randomly extracted in the range $[\text{first_frame}-5, \text{last_frame}+5]$ around the labeled gestures annotated in the training set and following the protocol described in [141]. A non-gesture class is also added to the training procedure to make the predictions more accurate.

Localization and Classification Module At runtime, when the tiny ST-GCN in the proposal module classifies a small window as a specific gesture, another ST-GCN module is activated, processing a window of larger size ($L = 80$), placed in the interval $[\text{last_frame} - 80, \text{last_frame}]$. The window is defined as: $[\text{first_frame}, \text{last_frame}]$ ($\text{last_frame} - \text{first_frame} = 10$)

In this part, ST-GCN is used as a backbone feature extractor to exploit the spatial relationships between finger-joints as well as temporal information. Furthermore, the regression head for localization and the classification head with a softmax layer is appended to the graph convolution backbone. The regression head will output two values including $[\text{center_pos}, \text{gesture_len}]$ which denote the center position and the number of frames of each gesture, respectively (Figure 3.5).

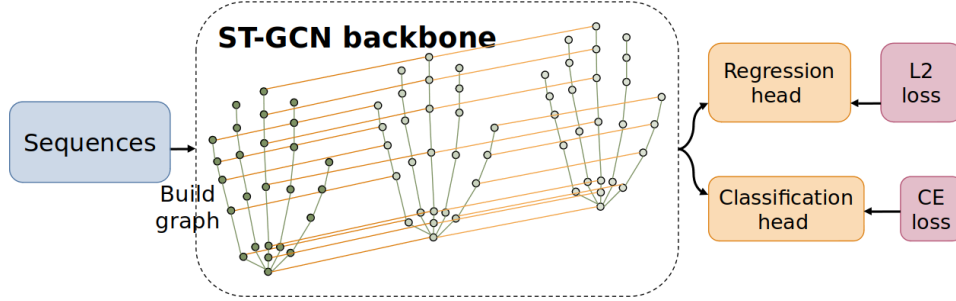


Fig. 3.5: The modified architecture of ST-GCN with both regression and classification heads

The module is trained as follows: From each annotated gesture in the training sequences, including the values of $[\text{CLASS}, \text{begin_frame}, \text{end_frame}]$, training examples of length $L = 80$ between $[\text{end_frame} - L, \text{start_frame} + L]$ are extracted. Each sample is associated with the corresponding ground-truth of center positions and length of gestures for localization branch training purposes:

$$\begin{aligned} \text{mid_frame} &= \frac{\text{start_frame} + \text{end_frame}}{2} \\ \text{GT}_{\text{center_pos}} &= \frac{\text{mid_frame} - \text{selected_start_frame}}{L} \\ \text{GT}_{\text{gesture_len}} &= \frac{\text{end_frame} - \text{start_frame}}{L} \end{aligned}$$

As in the training of the tiny module, examples of 80-frames non-gestures sequences labeled as a further class are added into the pipeline to improve the model's discriminating ability. L2 loss and Cross-Entropy loss are used for training regression head and classification head respectively.

Moreover, due to the limitation on training data, a stratified 5-fold based on class distribution is also applied to avoid under and overfitting.

Fine-tuning and Matching Module The output of tiny models from the proposal module and the output of large models from the localization and classification module are fine-tuned and go through matching condition checking to decide the last results. More concretely, the final outcomes must satisfy two matching conditions: the predicted category from the tiny model is the same as that of the large model, and the overlap ratio of short segments from the proposal module and predicted segments from the regression head is larger than zero.

$\text{cls_pred_tiny} = \text{cls_pred_large}$

$\text{IoU}([\text{first_frame}, \text{last_frame}], [\text{pred_start}, \text{pred_end}]) > 0$

The final outputs consist of 4 components:

$[\text{cls_pred_tiny}, \text{pred_start}, \text{pred_end}, \text{last_frame}]$

System Configuration Group 1 submitted the results of two experiments, one based on a single model (RUN1) and the other on the stratified 5-fold model (RUN2). The average time for the classification step was 2.1 ms in both cases, running on a single GPU NVIDIA QUADRO RTX 5000. The group actually planned to perform a further run by adding an additional feature, the Orientation Histogram [142] to discriminate gestures characterized by 2D trajectories, but the test was finally left for future work.

3.3.2 Group 2: Causal TCN

Method Description The proposed method by Group 2 is based on a temporal convolutional network (TCN) architecture that employs causal filters preventing any 'leakage' of future information to the past. In recent years, TCNs have been found to convincingly outperform canonical recurrent neural architectures across a broad range of sequence modeling tasks [143] by demonstrating longer effective memory and improved stability (mitigating the vanishing/exploding gradient problem). While there are various TCN-based action recognition approaches in the literature (see e.g. [83] and the references therein), Group 2 proposes a notably lightweight network structure that features a very low model size (only 125k parameters) and, hence, lends itself for efficient inference even on devices with limited computing power.

Feature Description Since the given per-frame landmarks are strongly coupled, this inherent structure should be exploited when designing expressive features. This coupling is furthermore naturally invariant under the Euclidean motion (e.g., due to a change of camera position and orientation), and hence, they decided to utilize angles between all anatomically definable segments as features. This representation for a skeleton configuration is not only location-viewpoint invariant but is also agnostic to its laterality. However, some of the gesture classes can only be distinguished reliably if their global context is additionally taken into account (e.g. MENU and WAVE). Therefore, another artificial segment along a fixed axis in space was added. As the 26 anatomical landmarks form 25 segments (see Figure 3.6), there are a total of 26 different joints/vectors yielding 351 angles for each input frame.

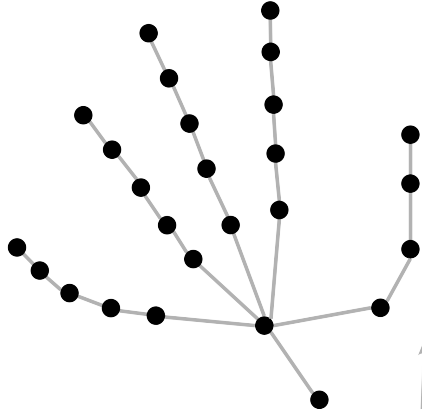


Fig. 3.6: Segments joining given anatomical landmarks. Angles between all pairs as well as with the fixed global axis (arrow right) serve as input features for gesture detection.

Network Architecture As main building blocks in the proposed model, Group 2 employs causal or unidirectional convolutional filters. Commonly dilated convolutions are used to build networks with very long effective memory. However, as the contest aims at randomized temporal sequences of gestures each of which being a short-timed activity, they opted for non-dilated convolutions. To reduce the complexity of the learning task they follow a windowing approach, i.e. conditioning the model to detect gestures for short-time windows containing n consecutive frames. In particular, each window is fed through two convolutional layers (feature dimensions 64 and 32), each with a kernel size of 5 and followed by a Leaky ReLU activation. Subsequently, a fully connected, linear layer maps the output of all frames within the window onto a 17-dim output representing a one-hot encoding of the probabilities of the 16 gesture classes and a background one. The common cross entropy, together with a L_2 regularization term, is employed as a loss function.

Training As within the training data, every gesture appears 36 times, Group 2 decided to perform a stratified six-fold approach on the occurrences to split into training and validation sets. They trained six different models, in order to take all available information from the six trained models into the prediction phase. They sample the input windows of length $n = 20$ for every gesture s.t. window and gesture are at least featuring 50% overlap in order to not provide the network with practically meaningless gesture chunks. Additionally, they sample background windows following the same rule. However, since the amount of background windows outnumbers the amount of gesture windows by an order of magnitude, only 10% of background ones are considered for training, drawn randomly. Training is performed with a batch size of 15 for 100 epochs. In every step exponential moving average is applied to the model parameters to track averaged parameters for prediction. For every 1000 batches, these average parameters are employed to determine the accuracy of the validation. Finally, the parameter set featuring the highest validation accuracy over all evaluations is chosen as the final parameter.

Online Detection For every frame, voting contributions from every window it belongs to are collected, i.e. a frame gets up to n single votes for a label within its temporal context. This indicates that there is a constant delay of $n - 1$ regarding the gesture start prediction. Each window label prediction is ensembled from the output logits across k nets. To this end, logit vectors are normalized (L_2), summed, and finally evaluated with *argmax* to assign a class label.

In order to deal with fuzzy gesture endings, Group 2 applies a simple post-processing strategy. If two consecutive frames f_0, f_1 belong to different continuous (non-background) label-chunks and (a) the chunk of f_0 is larger than the one of f_1 and the length of the f_0 -chunk is less than n then the f_1 -chunk is set to background or (b) the length of the f_0 -chunk is less than n , then the f_0 -chunk is set to background. Finally, continuous chunks of length less than 9 that are surrounded by a background are also merged into the background. These strategies technically use more than $n - 1$ future frames. However, the gesture detections that have been discarded are not captured by the evaluation protocol.

The average per-frame detection takes $\approx 2.8 \times 10^{-2}s$, mainly consumed by the temporal voting and the k-ensembling.

System Configuration Training and detection were carried out on a (Debian 11) workstation featuring an Intel(R) Core(TM) i9-10920X CPU @ 3.50GHz processor, 128GB RAM, and an NVidia GeForce RTX 3090 (24GB). The implementation was realized with Python utilizing jax/jaxlib (0.3.1), dm-haiku (0.0.6), and optax (0.1.1).

3.3.3 Group 3: Transformer Network + Finite State Machine (TN-FSM)

Method Description The proposed approach is mainly divided into three main logical blocks: the first one enriches the features coming from the input data represented by 3D hand joint positions grouped in consecutive frames. Specifically, each frame contains a set of hand joints acquired at the same time. The second block classifies the computed features, i.e. outputs a gesture or a non-gesture label for each input frame. Finally, the third block receives as input a set of consecutive frames, here referred to as *window*, in which each frame is coupled with its gesture label. As output, it provides the boundaries of each gesture, i.e.

the beginning and the end of each gesture (and therefore implicitly the presence of non-gestures).

With the Group 3 approach, the final performance of the proposed method relies on the performance of every single module, in particular, on the ability to accurately classify frames as gestures (even if with a wrong label for some frames) or non-gestures.

Feature Description As mentioned above, input data consists of the 3D positions of the hand joints grouped in frames. Formally, at a given time t , it is available a sequence of joints $J_t = \{j_i^t \mid j_i^t = (x_i^t, y_i^t, z_i^t), 1 \leq i \leq N\}$, where N is the total number of the hand joints (in this case $N = 26$ different joints acquired through a *Hololens 2* device simulating a mixed-reality interaction). This block enriches this input vector with other 4 types of features computed starting from 3D joint positions. The first two are represented by the speed (s) and the acceleration (a) of each hand joint, computed with the following equation:

$$\begin{aligned} \mathbf{s}_i^t &= [x_i^t - x_i^{(t-1)}, y_i^t - y_i^{(t-1)}, z_i^t - z_i^{(t-1)}] \\ \mathbf{a}_i^t &= [x_i^t - 2x_i^{(t-1)} + x_i^{(t-2)}, y_i^t - 2y_i^{(t-1)} + y_i^{(t-2)}, \\ &\quad z_i^t - 2z_i^{(t-1)} + z_i^{(t-2)}] \end{aligned}$$

Then, joint-to-joint distance (JD) features are added, expressed as a matrix $D = 3 \times N \times N$, containing information about the 3D distances. Each element $\mathbf{d} \in D$ is a set of three coordinates and is obtained following this equation:

$$\mathbf{d}_{i,k} = \sqrt{(\mathbf{j}_i^t - \mathbf{j}_k^t)^2}, \quad k, j \in N$$

Finally, the input feature vector is further enriched by computing the spherical coordinates (r, θ, φ) of the joints' positions starting from the 3D Cartesian coordinates (x, y, z) :

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \arctan \frac{\sqrt{x^2 + y^2}}{z} \\ \varphi &= \arctan \frac{x}{y} \end{aligned}$$

It can be observed that the value of the inverse tangent in φ can have different values depending on the correct quadrant of (x, y) . To summarize, the final input vector is a concatenation of 4 different feature vectors that represent different aspects of gestures: hand position, joint movements (in terms of speed and acceleration), and hand shape (distances can represent the state of the hand opening or closing). Therefore, the feature vector length with all types exploited is $26 \times (3 + 3 + 3 + 26 + 3) = 988$. Each joint position is then normalized to obtain a zero mean and unit variance for each 3D axis.

Frame classification In this block, each frame is classified using the features computed as detailed in the previous step. Group 3 adopts a classifier architecture divided into two main parts: the first one is a transformer-based model [144], while the second one is a fully connected layer used to output the frame classification with the correct shape. In this way, each frame is classified as non-gesture or with a gesture label.

From a formal point of view, the model is defined as:

$$Y(\mathbf{x}) = F(\text{Encoders}(\mathbf{x} + PE))$$

As mentioned, there are two main parts. The first is $F(\cdot)$, which corresponds to the fully connected layer with a softmax layer needed for the classification task. This block is applied on each time step $x \in \mathbf{x}$. The second part is a set of $\text{Encoders}(\cdot)$, consisting in a sequence of 6 transformer encoders E and the *Positional Encoding* (PE) [144].

An encoder is described by:

$$E(x) = \text{Norm}(x + \text{FC}(\text{mhAtt}(x)))$$

where $\text{FC}(\cdot)$ are two fully connected layers with 2048 units, followed by a ReLU activation function and $\text{Norm}(\cdot)$ is a normalization layer. The Positional Encoding appears to be essential in order to encode in the adopted model the temporal information of the sequence defining a vector that contains a probability distribution over n gesture classes for each time step included in $\mathbf{x}$. Finally, mhAtt is the multi-head attention block:

$$\text{mhAtt}(x) = (\text{Att}_1(x) \oplus \dots \oplus \text{Att}_8(x)) W^O$$

where

$$\text{Att}_i(x) = \text{softmax} \left(\frac{Q_i K_i}{\sqrt{d_k}} \right) V_i$$

Here, $K_i = xW_i^K$, $Q_i = xW_i^Q$, $V_i = xW_i^V$ are independent linear projections of x into a 64-d feature space, $d_k = 64$ is a scaling factor corresponding to the feature size of K_i , W^O is a linear projection from and to a 512-d feature space and $\oplus$ is the concatenation operator.

The training of the classifier is based only on the provided datasets. This is a non-trivial procedure since gestures tend to be short and then the majority of data are labeled as non-gestures. Therefore, Group 3 adopts the *Focal Loss* [145] in order to contrast the unbalanced training dataset, instead of the most common *Categorical Cross Entropy* loss for multi-classes classification scenarios. As optimizer, Group 3 use the the *AdamW* [146] algorithm, that in our experiments shows better performance w.r.t. *Adam*, with a initial learning rate of 10^{-4} , 0.5 of internal dropout and weight decay of 10^{-4} . The model is trained creating sequences with 10 consecutive frames. All the model parameters rely on a k -fold ($k = 9$) cross-validation.

Gesture Detection A *Finite State Machine* (FSM) is used to define the boundaries of each gesture, relying on the gesture labels provided by the classifier. The FSM is based on four different states, and the input is represented by a sliding window (i.e., a buffer) of 10 frames. The FSM runs only when the buffer is full. It is important to note that the final gesture class is predicted only once detected the beginning and the end of a gesture. Specifically, the final class corresponds to the most predicted class into the input window.

The first state of the FSM is used to detect the beginning of a gesture, and it is maintained as the current internal state until the window contains only frames classified as non-gesture by the transformer-based classifier.

When at least one frame is classified as a gesture, the internal state of the FSM is increased. In this second state, a check about the beginning of the gesture is conducted: this control is motivated by the possible presence of frames wrongly classified. Indeed, the beginning of a gesture is confirmed only if at least w_i different frames in 10 consecutive windows are classified as gestures, and then the FSM passes to the third state. In order to handle gestures with a very limited length, the FSM can pass directly in the fourth state if a non-gesture, i.e., a whole window with all frames classified as non-gesture, is found. In the third state, the end of the gesture is detected. In this state, the end of a gesture corresponds to a window that contains only non-gesture classifications, and when detected, the FSM internal state is increased. In the fourth state, a check at the end of a gesture is performed, since, as aforementioned, some frames can be wrongly classified. Only if w_e consecutive frames are classified as non-gesture, the FSM detects the end of the gesture. In case of a single frame is classified as a gesture, the FSM returns in the third state.

Run tests Group 3 tested three different versions of the proposed pipeline, focusing in particular on different feature types combinations used as input. It has been empirically observed that the classification performance of the transformer-based model achieves a good accuracy with the detailed setting, while for the Finite State Machine, best results are obtained setting $w_i = 5$ and $w_e = 10$. In the first solution (TN-FSM), it has been trained with a feature vector containing only position, speed, and acceleration. In the second case, features of the joint distances have also been added (TN-FSM+JD), while in the third one, spherical coordinates were added (TN-FSM+SC). **System Configuration** The proposed system ran on a computer equipped with an *Intel Core i7-7700K* and the dedicated GPU *NVidia GeForce GTX 1080 Ti*. The observed inference time on GPU is about 4.65 ± 0.39 ms and 4.72 ± 1.13 ms with only CPU (averaged on 100 different runs), denoting that the implementation of the transformer-based model is not optimized for high-level parallelism.

3.3.4 Our baseline: Stronger

The baseline we added in the comparison is a variation of the recognizer used in [16]. The method is based on a modified version of the DDNet architecture [83], customized by adding novel features and related branches and trained for online detection using a sliding window approach for continuous classification. More information about this methods are in Section 4.1

After the network training, we defined specific thresholds for the acceptance of the classification results as follows: for each gesture class, we estimated the class probabilities of the correctly estimated gestures (almost 100%) and the corresponding second highest probability among the other classes. The average of these probabilities has been taken as a threshold to discard gesture detection for that class with a lower probability. This trick is able to drastically reduce the amount of false positives. As the gesture duration is variable the test classification is performed over sliding windows of variable sizes (ranging from 5 frames to 60 frames). Feature vectors estimated in the various windows are resampled to 30 elements and sent to the trained classifier. If a gesture is detected in a window and the related class probability is higher than the related threshold, the corresponding frames are assigned to the class.

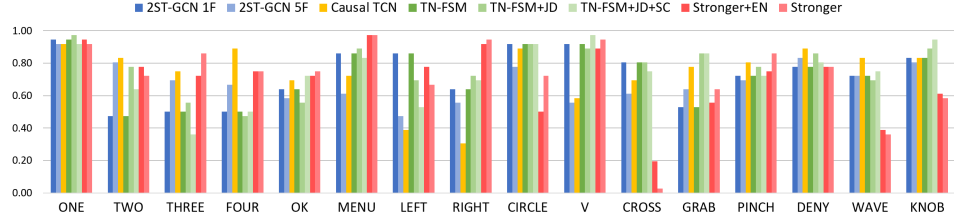


Fig. 3.7: Detection rate

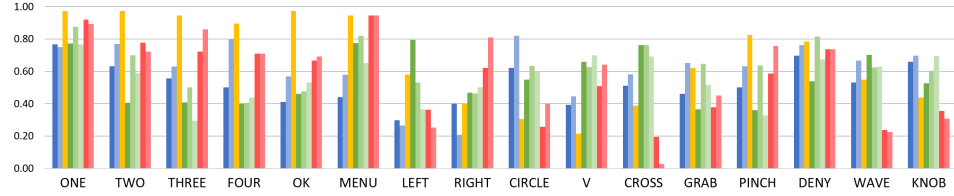


Fig. 3.8: Jaccard Index

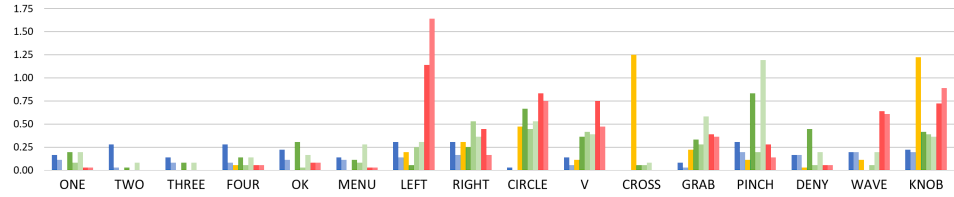


Fig. 3.9: False Positives' score

Fig. 3.10: Performance metrics per class, averaged on all the test sequences.

3.4 Results

Table 3.2 summarizes the results for the different methods and runs. The performances are rather similar on average. The table shows that, overall, Group 2 with the Casual TCN method achieved the highest detection rate (0.80). However, Group 1 and 3 managed to reach

Group	Method	DR	FP	JI	Delay(fr.)	time(ms)
B.Line	Stronger+EN	0.70	0.35	0.56	16.40	100
	Stronger	0.72	0.34	0.59	14.79	100
G1	2ST-GCN 1F	0.68	0.33	0.52	12.55	2.1
	2ST-GCN 5F	0.74	0.23	0.61	13.28	2.1
G2	Causal TCN	0.80	0.29	0.68	19.00	28
G3	TN-FSM	0.73	0.34	0.56	10.00	4.65
	TN-FSM+JD	0.77	0.23	0.63	10.00	4.65
	TN-FSM+JD+SC	0.74	0.36	0.56	10.00	4.65

Table 3.2: The table shows the summary of the results for each group method and all the variants proposed averaged on all the gestures. The metrics displayed are the detection rate, the false positives' scores, the Jaccard Index and the delay (in frames) between the start of a gesture marked by the method and the last frame reached at the moment of the marking.

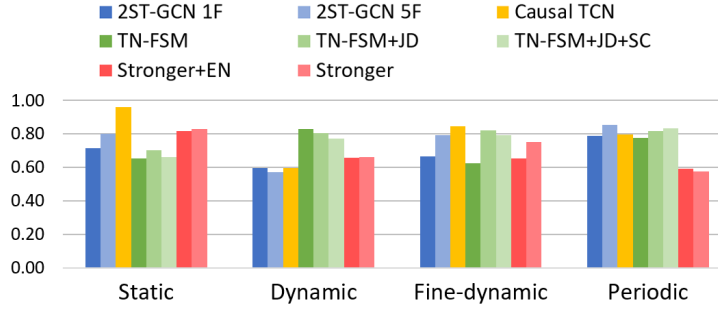


Fig. 3.11: Detection rate

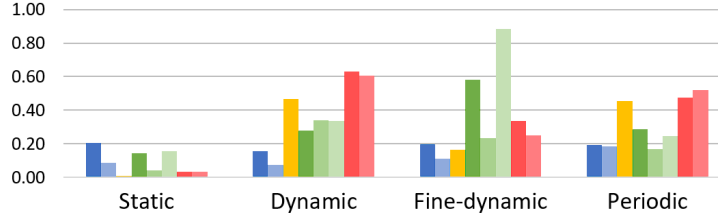


Fig. 3.12: False positives

Fig. 3.13: Scores by gesture type

the lowest amount of false positives with the 2ST-GCN 5F and TN-FSM+JD methods, with a ratio of 0.23. Looking at the Jaccard Index, Casual TCN is again the one with the best performance.

Major differences between the proposed methods appear when we look at the per-class performance metrics. In Figure 3.10 the bar charts show detection rates, Jaccard Indexes and false positives' scores for the single gesture classes. It can be noticed how no method is performing consistently across the entire gesture dictionary. The inconsistency is quite noticeable in the false positives' scores (Figure 3.9). In fact, methods like Casual TCN that have overall the highest Jaccard Index and overall FP rate in the average of the groups, when looking at the FP by-class, present near-to-zero FP for a large amount of gestures and an high FP spike for specific gestures such as CROSS and KNOB. The same behavior can be observed for the Stronger methods.

The analysis of the results grouped by gesture types can also be important to understand how the methods handle heterogeneous gestures where the semantics depends on completely different features. Figure 3.13 shows huge variability in the effectiveness of the methods for the different classes. For example Casual TCN and Stronger present good performances on static gestures, and have relevant false positive issues on coarse dynamic and

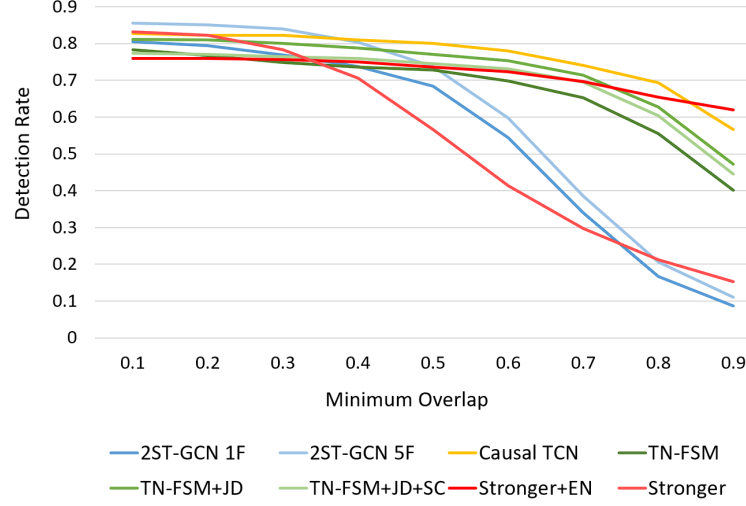


Fig. 3.14: The Detection Rate of each method as a function of the minimum overlap ratio between the gestures' windows, for the detected gesture to be marked as correct detection.

periodic gestures. TN-FSM the most consistent detection rates across the categories, being, however, the worst for static gestures and presenting relevant false positives issues for fine-dynamic gestures, with a surprising exception for the TN-FSM+JD run.

The fact that in the contest's rules the detection of gestures is considered correct if the intersection with the annotated time frame is higher than half the length of the ground truth gesture duration should be considered with care. The threshold is, in fact, arbitrary, and some of the detected gestures are discarded by not meeting its requirement and are, therefore, not classified neither as correct nor as a false positive. We analyzed the effect on the detection rate when changing this threshold. Figure 3.14 shows the detection rate of the different methods as a function of it. As expected, a lower threshold would improve the scores of most methods, however, some methods benefit from a larger improvement compared to others (up to 15% for 2ST-GCN and Stronger+EN). With small thresholds, 2ST-GCN-5F is the method providing the best results and could be the best choice for an application task that doesn't require the accurate localization of the complete gesture frame. On the other hand, Causal TCN is the method less influenced by the threshold, meaning that it provides most of the detection with a large overlap with the ground truth frame.

Group	Method	from start	from end
B.Line	Stronger+EN	10.72	-22.32
	Stronger	10.27	-22.13
G1	2ST-GCN 1F	7.85	-26.65
	2ST-GCN 5F	7.64	-22.90
G2	Causal TCN	4.36	-28.79
G3	TN-FSM	4.52	-28.63
	TN-FSM+JD	18.30	-14.73
	TN-FSM+JD+SC	10.54	-22.24

Table 3.3: Average delays in frames from ground truth start and end of annotated gestures derived by algorithm design (excluding computation time).

The evaluation of the actual delay of the recognition with respect to the ground truth annotations of gestures' starts and ends are summarized in Table 3.3 and represented in Figure 3.15 as a function of gesture types. Here the bars represent on the right (positive values) the average recognition delay with respect to the ground truth gesture start, measured in frames. In the negative part (left), they represent the advance with which they are recognized, with respect to the actual gesture end. Here, we see consistent and very low delays

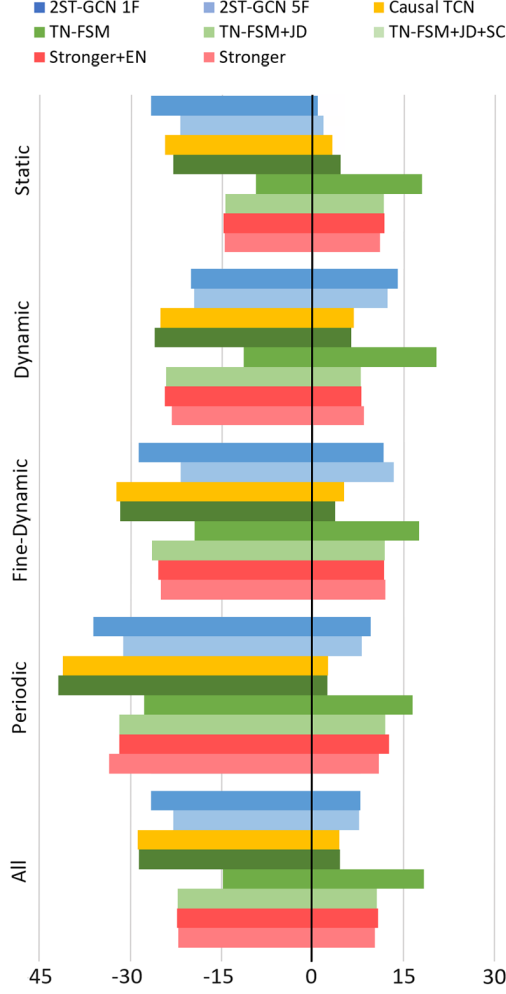


Fig. 3.15: The delay in frames of the last frame used by each method from the start and end marks of Ground Truth gestures, shown by gesture type. Chart bars are centered on the last frame used.

for the Causal TCN and the first TN-FSM run, while 2ST-GCN detects static gestures incredibly fast, but not in the other categories. We will discuss these results in detail in the next section.

3.5 Contest's outcomes

The outcomes of the SHREC '22 evaluation provide interesting insights on the feasibility of effective online recognition of heterogeneous gestures based on Hololens 2 streams or similar hand pose trackers data.

The proposed techniques employ some of the most popular network architectures: transformer networks, temporal convolutional networks (TCN), and graph convolutional networks (GCN). The methods are efficient, but the classification performances are still far from those required by usable, practical applications.

Detection rates do not exceed 80%, meaning that a relevant percentage of the gestures is missed. This happens for all the methods and all the gesture types.

The number of false positives is definitely too high for all the proposed methods to support reliable interfaces. The accuracy in the detection is also not optimal. As the false positives score is the number of false positives divided by the number of real gestures, even the lowest value obtained, 0.1 is rather high. In our dataset, we have sequences of about 30

seconds with 4 gestures on average, which means that the best method proposed would detect 4 non-existing gestures in 5 minutes of continuous hand movements. The others methods 8 or more. These figures should be reduced.

It seems that the network architecture has little impact on the classification performances, and differences seem to be mostly caused by the training and online testing procedures. Methods based on 1D convolutions (Causal TCN and STRONGER) perform best on static gestures Causal TCN performs well also on Dynamic fine and periodic gestures and has only lower scores on dynamic ones. This fact, however, does not seem to be related to the network architecture, but rather to the fact that the network is trained to recognize sequences of 20 frames, while dynamic gestures are longer. The use of more complex networks does not seem to help, as transformers and graph networks are not providing much better results. However, the method based on GCN seems to be the only one that uses raw data (coordinates as input), while the other networks process handcrafted features. Some groups tested the same methods with different feature sets, but additional features do not always help, as in the case of energy for Stronger and the spherical coordinates for TN-FSM.

Gesture prediction times are, instead, negligible and definitely not an issue. The classification times of the methods proposed by Group 1 and Group 3 are almost negligible (2.1 *ms* and 4.65 *ms*), and the only method with a non-negligible computation time, albeit compatible with an online application is Stronger (100 *ms*). These values are small with respect to the intrinsic time delay depending on the algorithms' design and shown in Figure 3.15.

The analysis of these delays is indeed quite interesting. Some predictions are surprisingly fast and are provided after very few frames from the annotated starts. In some cases, gesture prediction anticipates the actual start of the gesture itself. Causal TCN and the first run of TN-FSM provide results with delays of less than 5 frames (250ms) consistently for all the gesture types. 2ST-GCN is the fastest method on static gestures even if not so efficient for the other classes. Dynamic coarse and fine gestures are all recognized far before their completion, even if the semantics of the gesture depend on the whole palm and finger trajectories. This shows that the methods learn to predict the complete gestures from the first frames only. This early detection is allowed by the fact that classifiers like 2ST-GCN or Casual TCN are trained to recognize not only entire gestures but also smaller time frames, including the first part of the gestures. While this provides amazing response times, it could be, on the other hand, one of the reasons for the high number of false positives. For practical applications, more complex recognition strategies increasing the time delay but increasing detection accuracy and reducing false positives would be more effective.

3.6 Post Contest improvements

Table 3.4 shows the best results obtained on the SHREC'22 benchmark. In the following, we report a description of the corresponding methods.

Method	Date	DR (%)	FPR (%)	JI (%)	Delay(fr.)	Time(ms)	D-Start (s)	D-End (s)
MVMSFN [147]	2024	93	6	88	8	6	NA	NA
TY-Net [148]	2023	89	4	86	NA	NA	NA	NA
OO-dMVMt [10]	2023	92	9	85	8	4	NA	NA

Table 3.4: Performances of the best methods tested on the SHREC'22 benchmark. DR: Detection Rate, FPR: False Positive's Ratio, JI: Jaccard Index, D-End/D-Start: average delays in frames from the ground truth end/start of the annotated gestures (depending on the algorithms' design).

- **A Multi-View and Multi-Scale Fusion Network for Online Detection of Heterogeneous Gestures (MVMSFN)**) proposed by Long et al. in 2024 [147] is a sliding-window Multi-View and Multi-Task (MVMt) model designed to address the

challenge of inaccurate gesture segmentation from background data, using MVMT [Section 4.4](#) as its backbone.

The authors introduced a Multi-View Fusion Model (MVFM) to facilitate adaptive feature fusion by incorporating a spatial attention weight matrix, enhancing information exchange across multi-view branches. They also developed a Multi-Scale Fusion Model (MSFM) to capture correlations among features at different scales and layers, addressing issues caused by inconsistent gesture lengths. To further improve performance, they designed a Decoupled Gesture Progress Module (DGPM), which employs auxiliary tasks to separately predict the execution progression of static and dynamic gestures. This mitigates performance degradation caused by uniform gesture modeling in environments where static and dynamic gestures have significantly different representations. The network consists of several non-shared parameter feature extracting branches, multi-task branches, and feature fusion modules, which are implemented with a series of 1D Conv layers, except for the fully-connected layers at the last position in the task headers, which is for result output

Results on the SHREC 22 demonstrate that MVMSFN achieves state-of-the-art performance, offering higher accuracy in gesture segmentation, a lower false trigger rate, and relatively low computational cost, making it well-suited for real-time applications.

- **Transforming YOLO for Hand Gesture Recognition (TY-Net)**, Doždor et al. [148] leveraged a recognition method that directly predicts both gesture classes and gesture boundaries from a sliding window input using a transformer-based architecture. The model is trained end-to-end with a custom loss function designed to prioritize samples containing gesture centers. This enables the model to determine whether an input contains the center of a gesture and, if so, to predict its boundaries and class. The input features are extracted from overlapping windows and include pointwise velocities of each skeleton keypoint and their pairwise Manhattan distances. Velocity captures variations in gesture speed, while Manhattan distances provide spatial information. These features are embedded via a fully connected layer and combined with temporal embeddings, which transform discrete time steps into vector representations learned during training. The Transformer encoder comprises multiple layers with two main components: a multi-head self-attention mechanism and a position-wise feed-forward network. The output from the final time step of the encoder is processed by a fully connected classifier to predict confidence, gesture center, offset, and class probabilities. In an online setting, TY-Net uses a detection strategy that accumulates predictions with confidence scores exceeding 0.5 over time. When the same class is predicted for T consecutive timesteps, the gesture is activated, and subsequent predictions for the same gesture are ignored, ensuring efficient and accurate recognition in real-time scenarios.
- **On-Off deep Multi-view Multi-Task (OO-dMVMT)**, described in [Section 4.4](#) is particularly effective for the SHREC'22 benchmark.

Continuous Gesture Recognition Methods

During my PhD, I developed two main recognition methods, both based on the DDNet model [149]. The first published in 2021 is STRONGER, a client-server solution for online gesture recognition from captured hands' joint sequences. **The Section 4.1 is adapted from [16].** The system leverages a CNN-based recognizer, improving current state-of-the-art solutions for segmented gesture classification. It has been trained and tested for the online gesture recognition task on a recent benchmark, including heterogeneous gestures, specifically on the SHREC 2019 and SHREC 2021 datasets. STRONGER demonstrated high classification accuracy and a low number of false positives across most gesture classes. Additionally, it has been implemented in a demo application for a Mixed Reality scenario using a HoloLens 2 optical see-through Head-Mounted Display with hand-tracking capabilities.

The second method, MVMT, was published in 2023 and achieved state-of-the-art results on both SHREC'22 and SHREC'19 benchmarks. **Detailed information about MVMT is provided in Section 4.4 that builds upon the content of [10].** Continuous mid-air hand gesture recognition based on captured hand pose streams is fundamental for human-computer interaction, particularly in AR/VR environments.

As discussed in Chapter 2, many existing online gesture recognition methods focus primarily on classification tasks, without considering the execution time. This is largely due to the fact that original benchmarks do not incorporate execution time as an evaluation metric. Prior to SHREC'22, only a few datasets explicitly considered runtime performance, such as EgoGesture [36], where C3D [39] achieved a high processing speed of 117 FPS but a relatively low Jaccard Index (JI) of 72%. Similarly, IPN Hand [64] evaluated execution time, with FASSD [104] achieving a fast inference time of 28 ms while maintaining a moderate JI of 77%. In SHREC'22 original contest, methods often suffer from high latency Table 3.2, or poor results.

To overcome this limitation, we propose the On-Off deep Multi-View Multi-Task (MVMT) paradigm. MVMT exploits multiple time-local views related to hand pose and movement to generate comprehensive gesture descriptions. It employs heterogeneous tasks to achieve high accuracy in both classification and segmentation. Unlike traditional paradigms where all tasks must be active simultaneously, MVMT allows specific tasks to be activated or deactivated based on the input, enhancing flexibility and efficiency. This approach not only improves classification and segmentation accuracy but also minimizes false positives and reduces decision latency, all while maintaining real-time operation. Our results demonstrate that MVMT sets a new state-of-the-art in continuous and online 3D skeleton-based gesture recognition. It excels in gesture classification accuracy, segmentation accuracy, false positive rates, and decision latency, as validated on the SHREC'19 and SHREC'22 datasets.

4.1 STRONGER: Simple TRajjectory-based ONline GESture Recognizer

As stated in Chapter 1, user interfaces based on 3D mid-air gestures are expected to become popular in the next years, as they are a viable solution for natural interaction in mixed

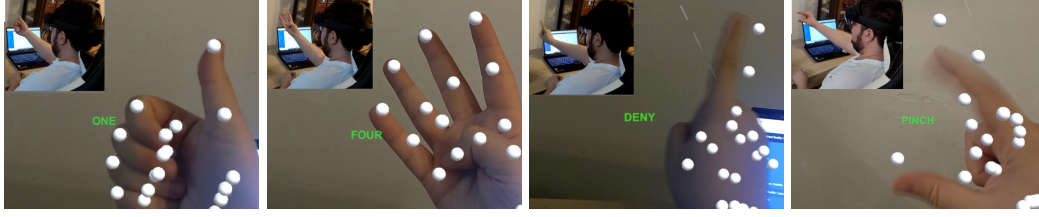


Fig. 4.1: Snapshots showing the online display viewed by the subject wearing a Hololens 2 running the online recognizer demo.

reality not requiring specific devices and being suitable for being used in crowded or noisy environments [150]. Currently, a relevant number of interactive systems in many domains already feature this kind of interface: virtual environments, smart surveillance systems, teleconferencing, home or car entertainment controls, virtual Personal Aerobics Trainers (PAT), and so on. Creating advanced touchless user interfaces is also fundamental to maintain a high level of hygiene for work environments or in public terminals, and this can solve several issues related to the pandemic emergence. These systems are promising but are typically limited to the recognition of a few simple gestures.

Skeleton-based gesture recognizers need to capture the fine differences among gestures and distinguish one gesture from another, ideally with a high degree of confidence but also being able to avoid false positives, e.g. detection of gestures corresponding to non-significant actions. This is a challenge of fundamental importance to create interfaces with a reasonable usability degree.

In 2021, we proposed STRONGER, an online skeleton-based gesture recognizer that can be used to develop mid-air device less interfaces. We test it on a recent online classification benchmark [8], and demonstrate it within a prototype mixed reality interface. The recognizer is based on a revised version of DDNet [149], modified by adding novel features to handle a larger set of gestures and to perform online classification. The prototype application is based on Hololens 2, and a client-server architecture to process hand pose streams. The results obtained in the online benchmark and our preliminary tests on the prototype show the feasibility of a gesture-based user interface for mixed reality that is able to recognize a large dictionary of heterogeneous gestures.

A relevant issue in the recognition of heterogeneous hand gestures (for example, compared with action recognition) is that the gesture classes may be differentiated by different features, which may be, in some cases, subtle. Heterogeneous gesture recognition benchmarks like Shrec'17 [43] or SFINGE3D [151] feature "coarse" gestures characterized by long whole-hand trajectories (lasting 1 second and more) and "fine" gestures characterized by fast changes of finger articulations (lasting 100 milliseconds or less).

Yang et al. [149] noted that a very simple network architecture based on 1D convolutions, fed with simple features derived from the hand joints sequence and using a motion summarization module to reduce noise from non-relevant frames, can provide state-of-the-art results with reduced computational complexity.

A benchmark specifically designed for the online classification of heterogeneous hand gestures, including coarse, fine, and also static gestures, has been proposed in [8]. This dataset includes training and test sequences, including gestures and non-gesture frames, and can be used to validate online classification approaches that can directly be used to build gestural interfaces. We exploited this dataset to train our system and used it to assess online performance.

4.2 Proposed approach

4.2.1 System architecture

In our work, we not only designed and implemented a network solution for gesture recognition, but we also developed a prototype gesture-based interface for mixed reality running on

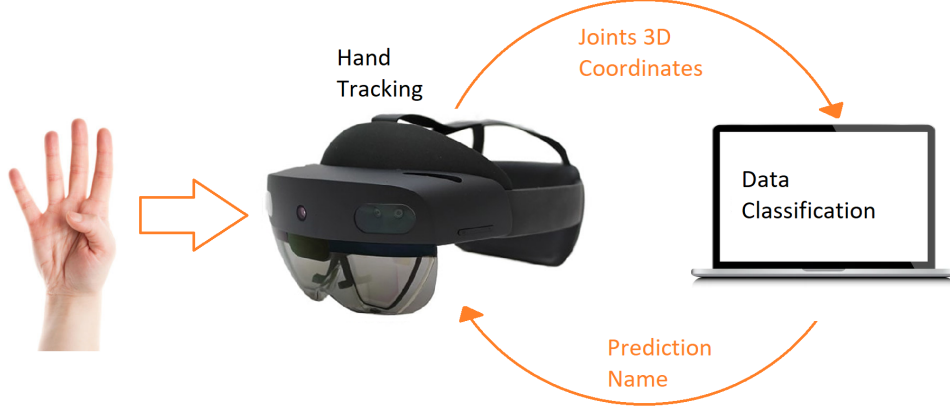


Fig. 4.2: The client-server application: the HoloLens2 app sends the joints' stream to the PC and receives the classification results to be displayed.

a HoloLens 2 head-mounted display. HoloLens 2 is provided with a hand tracking module able to capture hand skeletons (20 joints) similar to the one captured with the Leap Motion, and this allows us to use an online classifier trained on existing gesture datasets captured with the latter device for online gesture recognition on the HoloLens 2. The interactive application works with a client-server architecture, sending raw skeleton streams over TCP to a remote server where a Python application performs the recognition and sends back the results to the app running on the HoloLens. Figure 4.2 shows the operation scheme of the client-server architecture.

4.2.2 Gesture classification network

As it is based on a simpler 1D convolutional neural network and provides state-of-the-art performances on the SHREC 2017 benchmark [43], we chose DDNet [149] as the base method to build our online recognition system. However, we noted that this method does not perform well on the offline classification of the gestures of the SHREC 2021 benchmark. This is not due to limitations of the network architecture, but it depends on the data used to feed it, that it is not the complete stream, but a set of handcrafted features derived from it and does not include information that is important to disambiguate gesture classes. In detail, the two network branches of the original network process joints' velocities and between-joints distances, meaning that the evolution of the hand's orientation is lost. This makes it not possible, for example, to distinguish the classical menu gesture with the hand open with the palm in front of the user view from another static gesture keeping the hand open parallel to the floor. For this reason, we added two branches to the 1D convolutional network, one processing the sequence of palm orientations and the second processing a set of unit vectors representing the directions defined by couples of joints. The complete architecture of the proposed network is represented in Figure 4.3. Five 1D-CNN branches are used to process different 1D features derived from the input hand pose sequence, which are resampled with a fixed number of time steps. The first feature is the Joint Collection of Distances (JCD), which is the matrix of the Euclidean distances between hand joints across time flattened to become a one-dimensional vector. In particular, the matrix JCD^k (shown in Equation 4.1 as reported in [149]) is an $N-1$ -by- $N-1$ with N representing the index of the joint.

$$JCD^k = \begin{bmatrix} \left\| \overrightarrow{J_2^k J_1^k} \right\|_2 & & & \\ \vdots & \ddots & & \\ \vdots & & \ddots & \\ \left\| \overrightarrow{J_N^k J_1^k} \right\|_2 & \cdots & \cdots & \left\| \overrightarrow{J_N^k J_{N-1}^k} \right\|_2 \end{bmatrix} \quad (4.1)$$

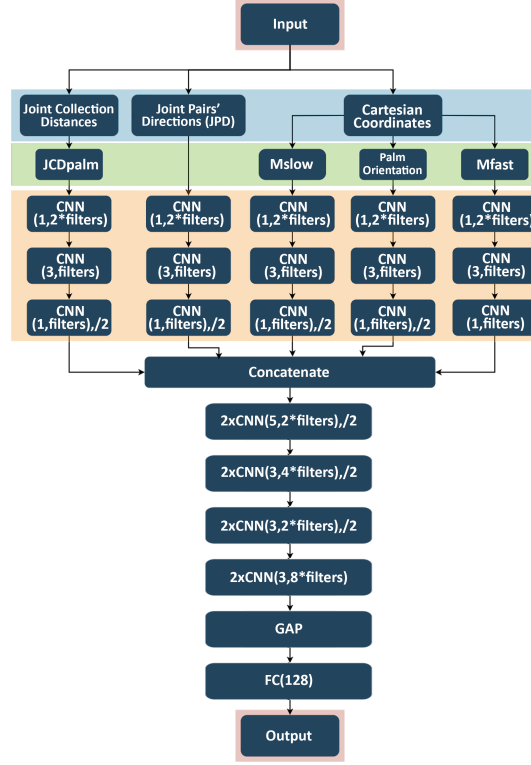


Fig. 4.3: Our classifier adds to the DDNet architecture two branches with 1D convolutions processing palm orientation data and joint pairs' directions.

That is then flattened to be a one dimensional vector to be used as network input.

$$\left\| \overrightarrow{J_i^k J_j^k} \right\|_2, \quad i \neq j \text{ and } i \in [2, N], \quad j \in [1, N-1]$$

indicates the Euclidean distance between vectors $J_i^k = (x, y, z)^k$ containing the coordinates of the joint at the frame k . The other features used also in the original DDNet are Mslow and MFast, that are joints speeds (differences between positions at consecutive times) computed at two different scales (reported again in [149]).

The two 1D inputs added are the palm orientation (PO) represented by the normal vector of the hand's palm across time ($PO^k = (x_{po}, y_{po}, z_{po})^k$), and a subset of the directions of joint pairs (Joint Pairs' Directions, JPD), obtained as the difference of selected joints' vectors

$$JPD^k = \begin{bmatrix} \overrightarrow{J_2^k J_1^k} \\ \vdots \\ \overrightarrow{J_N^k J_1^k} \dots \overrightarrow{J_N^k J_{N-1}^k} \end{bmatrix} \quad (4.2)$$

as shown in Figure 4.4. The vector components of the matrix are flattened as well to be used in the 1D convolutional network.

All the inputs are processed by similar CNN branches to derive intermediate representations that are then concatenated and passed through 3 more convolutional layers, a Global Average Pooling (GAP) and a Fully Connected layer (FC) providing the class probabilities. The training code takes hand pose time sequences corresponding to labeled gestures, re-samples the trajectories with a fixed number of elements, pre-processes them, creating the gesture descriptors feeding the network branches and training the classifier by minimizing the cross-entropy loss. To handle gesture classes together with non-gesture, we trained the network with the gesture windows corresponding to the annotations with the corresponding classes and a set of non-gesture examples cropped randomly outside the gesture time frames and assigned to a 19th class.

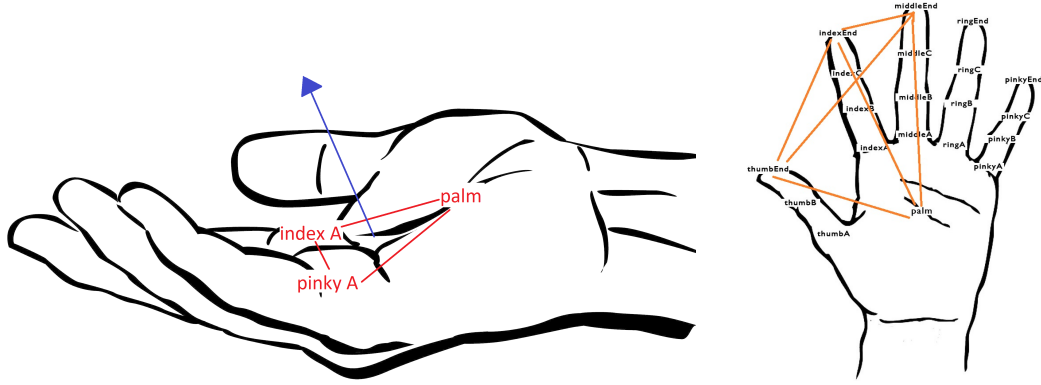


Fig. 4.4: The novel feature are the time evolution of the palm orientation (left) and of the directions defined by selected pairs of joints (JPD, right)

4.2.3 Online recognition and the SHREC'21 dataset

We decided to use the dataset of the SHREC'21 contest on Skeleton-based Hand Gesture Recognition in the Wild [8] both to train the classifier and to implement the mixed reality recognizer application, as it features a sufficiently large and heterogeneous gesture dictionary and an online evaluation protocol. This dataset features 20-node hand skeleton' sequences, including several examples of gestures interleaved with non-meaningful gesticulation, and it is divided into a training set with 108 sequences, including 24 examples for each gesture class, and a test set of 72 sequences, including 16 examples for each gesture class. Hand pose sequences in this benchmark have been acquired with a Leap Motion device mounted as in a Head-Mounted display configuration and with a frequency of 50 skeletons per second. The gesture dictionary includes 18 classes divided into 3 types:

- **Static.** 7 classes characterized by a hand pose kept fixed for at least one second (One, Two, Three, Four, OK, Menu, Pointing)
- **Dynamic coarse.** 5 classes characterized by a single global trajectory of the hand (Left, Right, Circle, V, Cross)
- **Dynamic fine.** 6 classes characterized by variations in the fingers' articulation (Grab, Pinch, Tap, Deny, Knob, Expand)

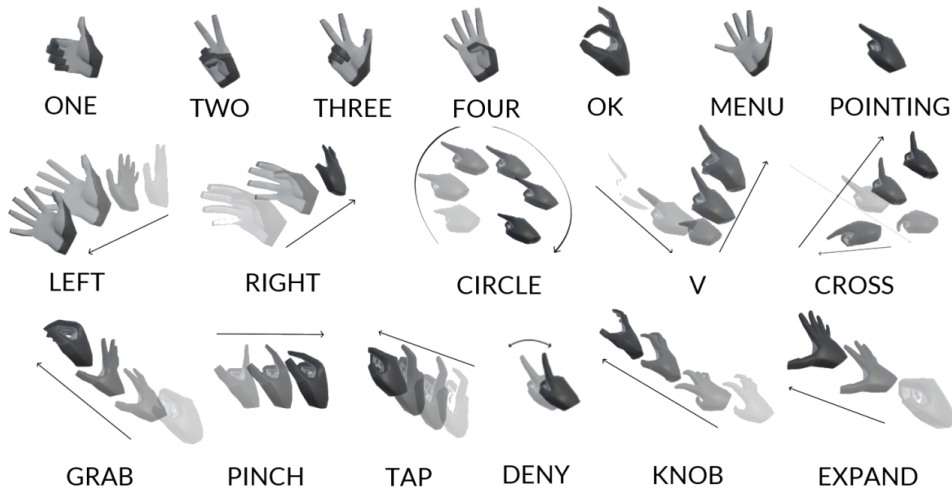


Fig. 4.5: Gesture dictionary of SHREC'21 (from [8])

These gestures, shown in [Figure 4.5](#), are interleaved in the sequences with non-significant hand motions of various types.

In our experiments on SHREC 2021, we trained the modified DDNet using cropped sequences representing both segmented gestures of the different classes defined in the dictionary and non-gesture sequences obtained by extracting sequences of random length from 0.2s. to 1.2s. The trained network can be fed with windows of the skeleton’s streams and outputs an array with the probability that it belongs to each class. For the online classification, we don’t simply assign the label corresponding to the maximal probability to the corresponding frames; we introduce a threshold for each gesture class, corresponding to the minimum probability, making the recognition acceptable.

All the gestures detected in the sliding window procedure with a probability lower than the corresponding threshold are then discarded and treated as non-gesture. This should reduce the false positive detections due to the limited representativity of the non-gesture examples. Thresholds are learned from training data as follows: during the offline training, the average probability estimated by the classifiers for the gestures belonging to that class is set as initial threshold estimates. These thresholds are then refined with a specific procedure optimizing them on the training sequences (including the labeled gestures interleaved with non-gesture movements). For each class, if the number of false positives is larger than a target value (FP-ratio > 0.5), the probability threshold is increased until the target is reached or the accuracy falls below 60%. The thresholds learned with this procedure are stored in a specific array called Probability Threshold Array (PTA) and used in online detection.

Another class-specific parameter is learned from training data and used to improve the quality of online detection. We call it the Window Thresholds Array (WTA) and define the minimum and maximum acceptable duration for each class. Each gesture to be executed needs a different number of frames, so if a short gesture is recognized in a very large window (or vice versa), and should be discarded. The time length of the gestures featured in the SHREC’21 benchmark is quite variable, ranging from 0.18s on average for gestures like EXPAND or TAP to several seconds for the static ones (see [Table 4.1](#)).

Gesture	type	avg #frames	avg. time (s)
ONE	static	149	2,98
TWO	static	169	3,38
THREE	static	166	3,32
FOUR	static	163	3,26
OK	static	148	2,96
MENU	static	151	3,02
POINTING	static	97	1,94
LEFT	dynamic coarse	23	0,46
RIGHT	dynamic coarse	17	0,34
CIRCLE	dynamic coarse	54	1,08
V	dynamic coarse	27	0,54
CROSS	dynamic coarse	43	0,86
GRAB	dynamic fine	14	0,28
EXPAND	dynamic fine	9	0,18
PINCH	dynamic fine	22	0,44
TAP	dynamic fine	12	0,24
DENY	dynamic fine	78	1,56
KNOB	dynamic fine	45	0,9

Table 4.1: Duration of the 18 gestures of the SHREC’21 dataset. There is a huge variability among the gesture classes.

We set acceptable duration limits for each class in the WTA. The minimal length in the training set diminished by 0.2 s, and the maximal length in the training set increased by 0.2 s. All the gestures recognized in the sliding window procedure are discarded if the window size of the corresponding class is outside the limits defined in the WTA.

We also set a global limit to the maximal length of the gesture tested, which is also useful to reduce the delay in the online procedure. This limit is set to 1 s. In the sliding window procedure, time samples of different sizes (from 5 frames to 50 frames with a step of 5 frames) are slid along the signal with a fixed shift of 5 frames (0.1s). If there is a gesture prediction corresponding to a window, a vote for the corresponding class is assigned to the superimposed frames, and the frame label is assigned as the one with maximal votes. A gesture is finally detected as a set of consecutive frames with the same assigned label.

4.2.4 Mixed reality interactive mockup

The client application running on the Hololens 2 device was created using the Unity game engine and the Mixed Reality Toolkit (MRTK), which was used in this project. This toolkit simplifies the development of XR headsets, providing a cross-platform input system, a basic set of components and features, and common building blocks for spatial interactions in Unity.

The application developed is a very simple mock-up superimposing to the user view 3D primitives representing the captured hand joints and a text string with the currently recognized gesture. The application is developed with Unity and features C# scripts parsing and sending to the server the joint positions stream provided by the MRTK API and getting back the current classification labels, that are displayed on the semitransparent screen (See [Figure 4.1](#)).

4.3 Results

4.3.1 Implementation details

The network code was developed using PyTorch and CUDA, and it was trained and tested on a Lenovo Legion 5 PC with a RAM of 16 GB and a Nvidia RTX 2060 (6 GB) graphics card. The network architecture features a set of parameters that have been tuned on the SHREC'21 training set, namely filters' size, number of training epochs, batch size, and, most importantly, the number of samples for input gesture resize. This value was set to 34. Different Python scripts have been developed for offline classification testing of segmented gestures, for the hand pose sequences processing following the SHREC'21 protocol using the sliding window approach, and for the online recognition receiving the Hololens 2 stream and providing it with the online classification results.

4.3.2 Offline evaluation of the gesture classifier

Before evaluating the online performances of our method, we performed a couple of tests to evaluate the offline classification performances of our implementation. [Table 4.2](#) shows that the addition of the novel features results in a 10% increase in the accuracy of the classification of segmented test set gestures of SHREC'21. The reason for the difference is the large amount of information lost in the original encoding (not including, for example, any hint on the spatial orientation of the hand).

[Figure 4.6](#), comparing the confusion matrices of the two classifiers, clearly shows that the modified version solves relevant issues of the original methods in recognizing gestures 3,5,7 (FOUR, MENU, LEFT) where the hand orientation information can disambiguate the gesture from some non-gesture movements. The classification of KNOB gestures (17) is also relevantly improved.

As the network is trained with segmented gestures, we could also evaluate the offline classification performances of our modified network on the popular SHREC '17 [\[43\]](#) benchmark. Here the improvement with respect to the original DDNet is not similarly high, but, in any case, our modified network performs slightly better than the original and better than the other method proposed in the literature (see [Table 4.3](#)).

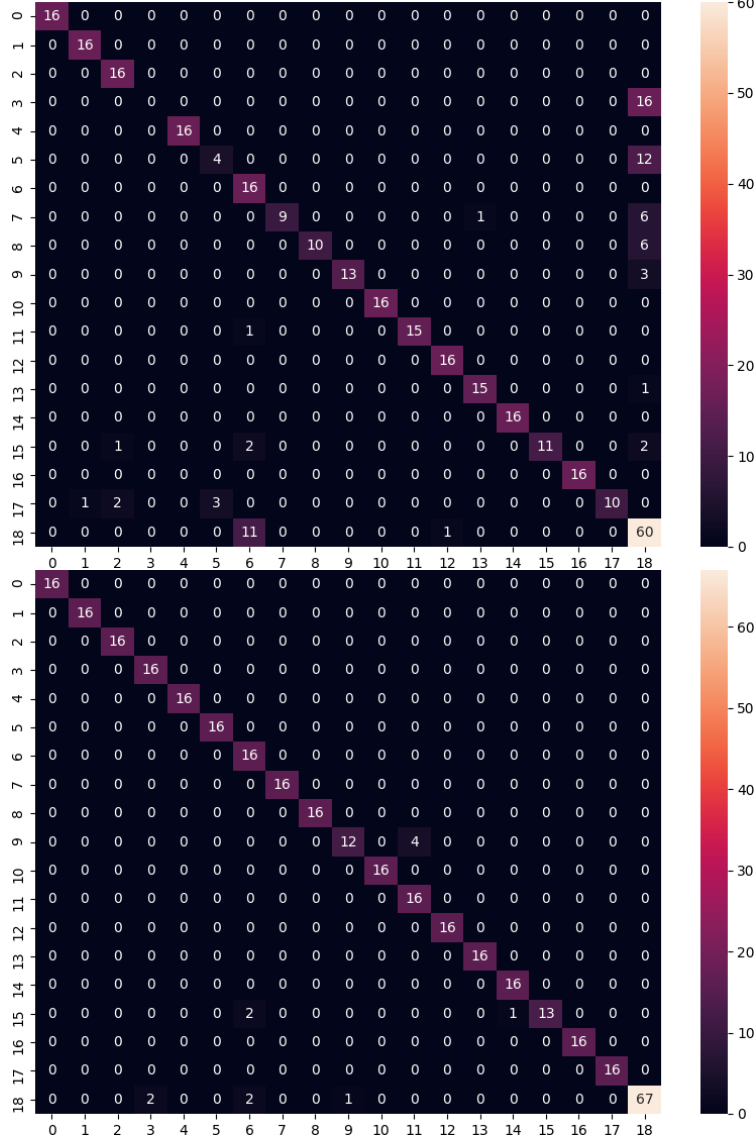


Fig. 4.6: Top: confusion matrix of DD-net on SHREC 2021 classes. Bottom: confusion matrix of the modified network with the additional features PO and JPD.

Method	Accuracy
DD-Net [149]	87.8%
Our model	97.5%

Table 4.2: The additional features added to DDNet strongly improve the classification accuracy on SHREC 2021 gestures

4.3.3 Online evaluation

For the online evaluation, we followed the protocol of SHREC '21: using the procedure described in Subsection 4.2.3, we obtained a per frame labeling of the sequences and the timestamps of gesture beginning and end. We could therefore estimate the "detection rate" in the test data, e.g., the percentage of predicted gestures (of each class) corresponding to ground truth ones correctly detected (where corresponding means at least 50% of overlap), the false-positive ratio, i.e., the ratio between the number of predictions not corresponding to ground truth ones divided by the total number of gestures of the corresponding class in the sequences, and the Jaccard Index (JI) [36, 77] measuring the average relative overlap

Methods	Accuracy
3 Cent [152]	77.9%
Key-frame CNN [43]	82.9%
Dynamic hand [153]	88.2%
CNN+LSTM [154]	89.8%
MFA-Net [155]	91.3%
Parallel CNN [156]	91.3%
STA-Res-TCN [157]	93.6%
DD-Net [149]	94.6%
Our model	95.0%

Table 4.3: Results on SHREC 2017 (Using 3D skeletons only)

Method	Det. Rate	FP Rate	JI	Time(s)
Group 1-Run 3 [8]	0.729	0.257	0.603	1.36
Group 2-Run 1 [8]	0.486	0.927	0.277	0.41
Group 3-Run 2 [8]	0.757	0.340	0.619	0.3×10^{-5}
Group 4-Run 3 [8]	0.899	0.066	0.853	0.16
Our method	0.906	0.347	0.740	0.10

Table 4.4: Average scores for the SHREC’21 benchmark compared with contest participants. It should be noted that the method of Group 4 includes a gesture segmentation module to reduce false positives.

Method	Det. Rate	FP Rate	JI
Orig.DDNet	0.858	2.052	0.353
DDNet+PO+JPD	0.944	1.896	0.431
DDNet+PO+JPD+WTA	0.906	0.347	0.740

Table 4.5: Ablation study showing that the novel 1D features strongly increase the detection rate while the learning of the Window Threshold Array is effective in reducing the false positives rate.

between the ground truth and the predicted label sequences:

$$JI_{s,i} = \frac{GT_{s,i} \cap P_{s,i}}{GT_{s,i} \cup P_{s,i}} \quad (4.3)$$

Table 4.4 shows the scores obtained by our algorithm on the SHREC’21 benchmark, compared with the best runs of the participants. Our method performs well, but the false positives are still a bit high compared to the results of the contest winners. However, it must be noted that this group used a pre-segmentation module based on gesture energy to reduce false positives, and this should be the reason for this difference. We plan to add a similar module in future versions of STRONGER. It is interesting to see how the addition of the JPD and palm orientation features, as well as the training of the Window Threshold Array, improves the results with respect to a simple application of a sliding-window DDNet. The results of an ablation study presented on Table 4.5, demonstrate the effectiveness of the added orientation-related features to improve the detection rate of the gestures and the effectiveness of the threshold training to reduce the False Positives rate. Table 4.6 shows the SHREC’21 scores obtained with STRONGER for each gesture class together with the classifier thresholds learned from the training set, i.e., the values in the WTA. It is possible to see that results are very good for many classes, while just a few gesture classes (Circle, Cross, Tap) are responsible for the decrease of the detection rate and the increase of the False Positive ratio. This means that just excluding these few classes, our system would be quite effective. Bar charts in Figure 4.7 demonstrate that the addition of the two novel features to the network input results in an increase of the detection rate and of the Jaccard Index and a reduction of false positives, with the only exception of the tap gesture, that is not well handled by our system.

Gesture	Det.rate	FP ratio	JI	Thr.
ONE	1.000	0.000	1.00	9
TWO	1.000	0.062	0.94	8
THREE	1.000	0.062	0.94	8.2
FOUR	1.000	0.312	0.76	8.7
OK	1.000	0.062	0.94	9.2
MENU	1.000	0.062	0.94	7.6
POINTING	0.937	0.062	0.88	7.5
LEFT	0.937	0.000	0.93	7
RIGHT	0.937	0.250	0.75	7.2
CIRCLE	0.750	1.500	0.30	7
V	0.937	0.187	0.78	6
CROSS	0.687	1.187	0.31	6.5
GRAB	1.000	0.375	0.72	8
PINCH	1.000	0.125	0.88	8
TAP	0.375	0.875	0.20	5
DENY	0.812	0.187	0.68	6
KNOB	0.937	0.250	0.75	8
EXPAND	1.000	0.687	0.59	7.2
Total	0.906	0.347	0.74	-

Table 4.6: Per-class evaluation of the scores of the SHREC2021 contest obtained with the current STRONGER recognizer architecture.

4.3.4 System prototype

Our recognizer can directly process gesture stream and has been used as the server-side application connected to the Hololens app described in Subsection 4.2.4. In the demo setting of the multiple windows, classification is performed every 5 frames of the continuous stream (45 fps). The prediction results are sent back to the Hololens app and are then printed on the screen on the user's display. The classification time is 0.1 seconds. An example of the online recognition system can be seen in the video at the link <https://streamable.com/5q6pl1>. Considering that no post-processing is applied the results are sufficiently good, even for some classes where the SHREC'21 testing was not optimal, such as "TAP", "PINCH", "EXPAND" and "DENY". Some gestures of the SHREC 2021 dictionary are not, however, well handled, especially dynamic coarse.

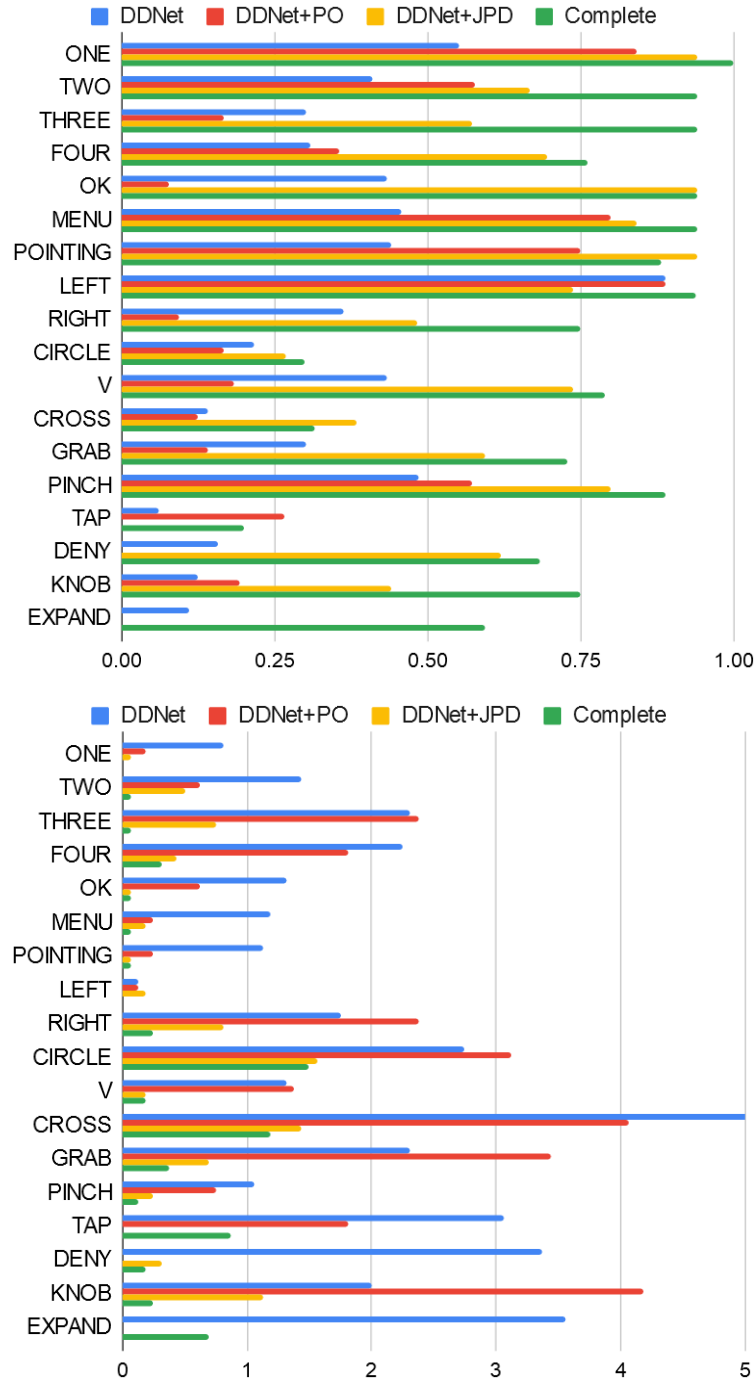


Fig. 4.7: Both the features added to the original DDNet contribute to improve the SHREC '21 benchmark scores. Top: Jaccard Index per class for the original DDNet, DDNet+PO, DDNet+JPD and both PO+JPD. Middle row: Bottom: Per class false positive ratio for the original DDNet, DDNet+PO, DDNet+JPD and both PO+JPD.

4.4 OO-dMVMT: A Deep Multi-view Multi-task Classification Framework for Real-time 3D Hand Gesture Classification and Segmentation

To recognize online heterogeneous hand gestures, in 2023, we proposed the On-Off deep Multi-View Multi-Task paradigm (OO-dMVMT). The idea is to exploit multiple time-local views related to hand pose and movement to generate rich gesture descriptions, along with using heterogeneous tasks to achieve high accuracy. OO-dMVMT extends the classical MVMT paradigm, where all of the multiple tasks have to be active at each time, by allowing specific tasks to switch on/off depending on whether they can apply to the input. We show that OO-dMVMT defines the new SotA on continuous/online 3D skeleton-based gesture recognition in terms of gesture classification accuracy, segmentation accuracy, false positives, and decision latency while maintaining real-time operation.

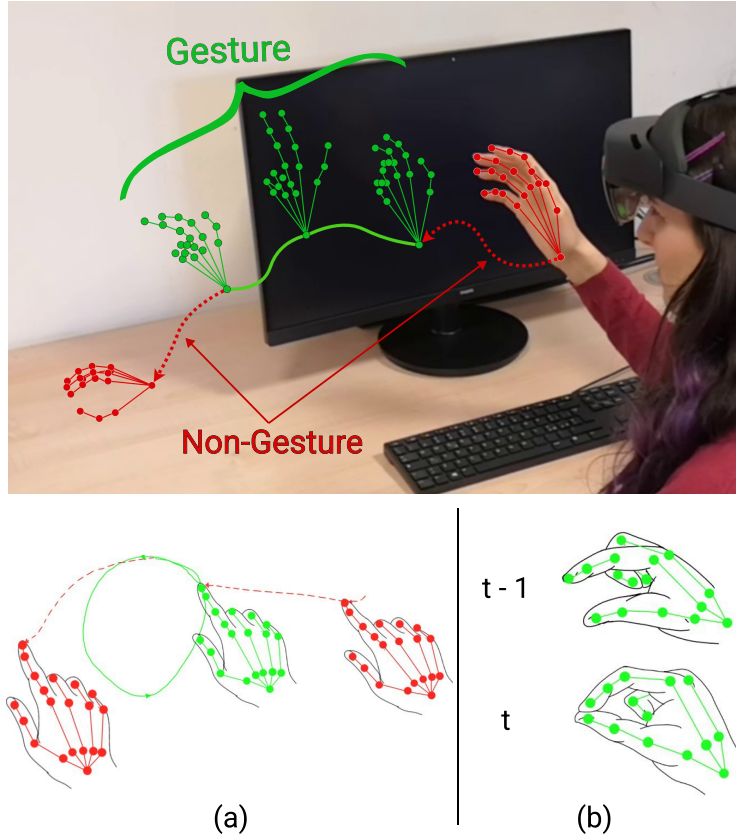


Fig. 4.8: On top, the 3D hand gesture real-time classification and segmentation scenario: in red are non-gestures, in green the segmented gesture. The task is difficult: below, we show samples from SHREC’19 (a), where it is possible to see long gestures with a static pose of the hand and the correct start of the gesture cannot be easily detected; in (b) a sample from SHREC’22, where gestures express more pose dynamics.

The current generation of Mixed Reality (MR) headsets (e.g. Meta Quest 2, Hololens 2, etc.) features accurate hand tracking capabilities, capturing finger poses at high frame rates, and cheap sensors/APIs are available to enhance different applications with this data. In addition, most of the online recognition methods (already presented in [Subsection 2.5.3](#)) use features extracted from the data, proving the importance of enriching the input rather than using all the raw data. However, none of the methods proposed in the literature has the accuracy and a false positive score good enough to guarantee a natural and reliable MR interaction. A missed or wrong gesture classification can potentially be dangerous for the

users if they have to interact with the machinery of an industry, medical operation, or any delicate situation.

In order to build an effective model, we tackled the online gesture recognition problem using a powerful yet unexplored framework called Multi-View Multi-Task (MVMT) learning paradigm [158, 159]. In MVMT, complementary views of the object of interest serve to learn a rich feature embedding that is fed into multiple tasks to borrow information across tasks that can potentially increase the predictive power of the learned models. Multi-View (MV) learning [160] is a well-established direction in machine learning, which associates multiple feature sets (called “views”) to the input data e.g. given an image, extracts spatial features, color features, and frequency features to form a single data point. Each data point is thus associated with V sets of different features where each set corresponds to a view, allowing supervised or semi-supervised learning tasks to access a more complete data descriptor. MV learning does not have to be confused with the multi-camera approach in which multiple camera views create their data point, which are then usually merged (synchronized) into a single one. Multi-Task (MT) learning aims to leverage useful information contained in multiple related tasks to help improve the generalization performance of at least one of the tasks [161].

MVMT learning is a less explored field, which extends MV learning to the MT learning setting with T tasks, where each task is a multi-view learning problem [158]. Most of the existing MVMT methods focus on proposing linear models to fit the specific application requirements, even though they are not suitable for common large-scale real-world problems in real environments. Only recently, non-linear deep models have been taken into account [162]. In our case, we take into account multiple views related to hand geometry and its global movement to create a multi-view description. Furthermore, we raise the accuracy by exploiting the relatedness of multiple tasks, such as gesture classification at different grains and regression of its start-end instants.

The issue arises when some tasks can’t be performed under certain conditions, such as the regression of the start frame of a gesture when no gestures are active. This invalidates the classical MVMT framework, which requires all the tasks to be active at each time. For this reason, we propose the On-Off deep Multi-View Multi-Task paradigm (OO-dMVMT), which extends MVMT by allowing specific tasks to switch on/off depending on whether they can process the observation sample.

OO-dMVMT deals exactly with this situation, introducing a mechanism that, at training time, switches tasks on and off depending on whether they can act on a specific data observation. Our contributions are the following:

- We individuate MVMT as the correct framework to deal with real-time hand gesture recognition.
- We extend MVMT to adapt to cases when some of the tasks cannot operate on a specific data sample, introducing OO-dMVMT
- With OO-dMVMT, we define the new state-of-the-art of hand gestures real-time classification and segmentation.

As discussed in [Subsection 2.5.3](#), many classifiers have been proposed to recognize gestures of this kind from hand poses’ sequences; most of them were not tested in a realistic scenario.

In [163], global and local motion features, along with the skeleton sequence, are fed into different branches of a Long-Short Term Memory (LSTM) network to get the predicted class of input gesture. Yang et al. [83] proposed DDNet, a simple network based on 1D convolutions, fed with multiple features (linearized joints’ distance matrix, joints speeds) with a motion summarising module to reduce noise from non-relevant frames. Trivedi et al. [164] used multiple features (joints, bones, joints’ velocity, bones velocity) to feed a Spatio-Temporal Relational Module. The use of the pre-computed features results in lighter and easier-to-train networks, and the parallel encoding of different sets of features can be interpreted as an application **multi-view** learning paradigm. Note that multi-view here does not mean that multiple features are related to different physical viewpoints (even if this idea has been proposed as well, for example, in [165]).

The benchmark we adopted to train and evaluate our system is SHREC’22 [6], where the gesture dictionary is heterogeneous, including static and dynamic gestures similar to those proposed in DHG14/28 and SHREC’17 (see Chapter 3). This benchmark improves the previous SHREC’21 [8], proposed by the same authors and with a similar structure, but solving some issues that, according to the website, made it unreliable for comparative tests. For these benchmarks, training data are long sequences of pose streams with annotated start frames, end frames, and labels of the gestures executed. The task is to detect the occurrences of gestures in test sequences using past information only. SHREC’22 data are captured with a Hololens 2 headset, *meaning that a recognizer trained for the task could be directly integrated into an interactive demo* using the same device to capture hand poses. The use of sliding windows is a simple and popular solution for the online task. Two main ideas have been proposed in this context: train the classifier on segmented gestures, handling the unknown input gesture length in the online testing [16] Section 4.1, or train it on fixed-size sub-parts (solution adopted by two methods presented in [6]).

We decided to apply a similar protocol for our online training framework, and we tested many of the models proposed for the offline setting in the continuous task, training them with fixed-size windows and adding a non-gesture class. Within this framework, we finally developed our novel On-Off deep Multi-View Multi-Task (OO-dMVMT) method, providing enhanced performances with respect to the SotA classifiers. We show the performance of our method in the realistic scenario proposed by specific benchmarks (SHREC’19 [3], SHREC’22 [6]), where gestures live within long sequences of non-significant hand movements from which they need to be distinguished.

4.5 Our pipeline

We present our proposed OO-dMVMT by first detailing the views and the tasks we have considered (Subsection 4.5.1 and Subsection 4.5.2, respectively). Then, we present how to train the model (Subsection 4.5.3) and how to infer from it for the real-time hand classification problem (Subsection 4.5.4). To fix the notation, ξ_t is the observation window that ends at frame t of length W , covering the frame interval $[t - W + 1, t]$.

4.5.1 The views

Every observation training window ξ_t is associated to $V = 3$ views: 1) *Geometric layout*. We borrow from [83] the flattened Joint Collection Distances (JCD) features, which are location-viewpoint invariant. JCD computes the Euclidean distances between a pair of collective joints, and is of size $\binom{J}{2}$ for each frame of the observation window, resulting in a tensor of $\binom{J}{2} \times W$. 2) *Short-term slow motion* M_{slow} . M_{slow} computes the 1-frame linear velocity of every single joint for all the joints, resulting in a tensor of size $J \times W - 1$. 3) *Short-term fast motion* M_{fast} . M_{fast} is similar to the short-term slow motion, but the linear velocity is computed every other frame (skipping the ones in between), with a $J \times \frac{W}{2} - 1$ resulting tensor. In practice, M_{slow} and M_{fast} model the short-term global motion of the skeleton in terms of speed. Following the multi-view learning paradigm, the three views JCD, M_{slow} and M_{fast} are each embedded into a $\mathbb{R}^{\frac{W}{2} \times 8}$ dimensional space by a set of independent encoders and concatenated in a multi-view latent pattern g_t of $\mathbb{R}^{\frac{W}{2} \times 24}$ dimensions.

4.5.2 The tasks

$T = 4$ tasks have been considered: 1) *SDN classification*. We classify gestures into three main categories: Static gestures (S), Dynamic gestures (D), and Non-Gestures (N). This macro classification is typical in heterogeneous hand gesture modeling [6, 8], and every gesture can fall in one of these superclasses. Static gestures require the user to fix the hand in a predetermined pose, keeping it still for a while. Dynamic gestures require the user to move the hand centroid, following a specific trajectory. Non-gestures are all those natural movements

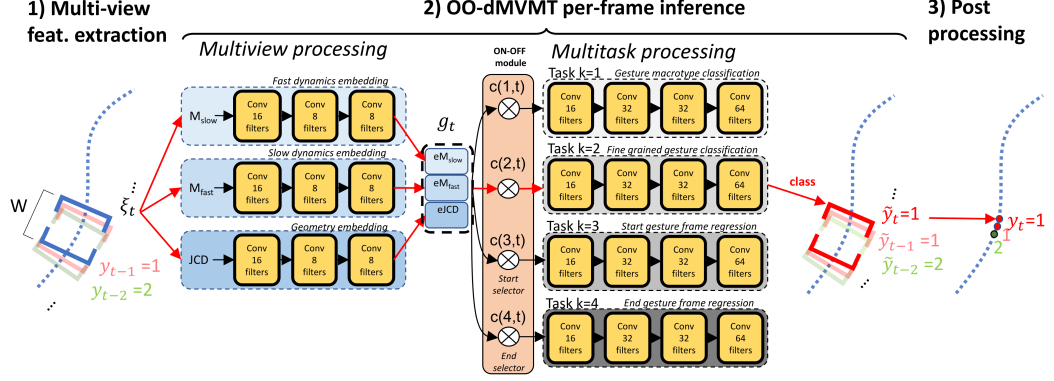


Fig. 4.9: The pipeline of our framework, is composed of three steps: 1) Multi-view feature extraction, 2) OO-dMVMt per-frame inference, 3) Post-processing. In red is highlighted the path used during the testing time, after the OO-dMVMt model has been trained.

of the hand that occur between gestures and are the most difficult class to capture. 2) *Fine-grained classification of gestures*. This task implements the fine-grained classification of the window ξ_t considering all the L classes into play (including the non-gesture class). This is the only task that is activated during testing time. 3) and 4) *Start/end gesture frame regression*. These two tasks perform the regression on the frame indices $t_{start}, t_{end} \in [0, W - 1]$ of the start and end of a gesture, respectively. The idea is to identify the patterns of the start and/or end of a gesture, thus aiding the classification tasks.

These last two regression tasks could be unable to function due to the presence of observation windows, which are completely contained in gesture (or non-gesture) streams. This configuration, dealing with multi-views data and with some tasks that cannot be associated with some (or all) of the views of the input data, is novel in the MVMt literature.

4.5.3 Model training

Let ξ_t be an input data instance sampled at time t , and k the index of the k -th task. We define the aggregated OO-dMVMt objective function to minimize as:

$$F(\xi_t) = \sum_{k=1}^T c(k, t) \cdot F^{(k)}(g_t, w_g, w_k) \quad (4.4)$$

$F^{(k)}$ is the deep objective function associated with task k controlled by $c(k, t)$, a boolean selector which is 1 if task k can work with input ξ_t , otherwise ξ_t does not contribute to the weights update for the task k . The decision of whether the task can work with the input is done at train time and depends on the semantics of ξ_t and the task itself. The term g_t represents the multi-view representation of ξ_t (see Sec. 4.5.1), w_g are the multi-view parameters which generate g_t starting from ξ_t , and w_k are the k -th task parameters that generate the output for the k -th head. The deep objective functions $F^{(k)}$ associated with our tasks are MSE for the regression tasks ($k = 3, 4$) and cross-entropy for the classification tasks ($k = 1, 2$). All the objective functions are uniformly weighted.

The idea is to treat the OO-dMVMt model as an instance of asynchronous optimization [166], where the central server begins to update the shared model w_g after it receives a gradient computation from one task node, without waiting for the other task nodes to finish their computations. In our case, the switching off of a task k corresponds to having a delay for that task, while the other active tasks contribute to the model update.

4.5.4 Model inference

The pipeline of the framework *at inference time* is summarized in Figure 4.9 and consists of three steps: in step 1) the observation window ξ_t is sampled at frame t , and the multi-view

description is extracted; in step 2) g_t flows into the fine-grained classification task, and a preliminary classification label $\tilde{y}_t$ is associated with the observation window ξ_t . The other three tasks are considered only during the training stage, as discussed in Subsection 4.5.3. 3) a post-processing step is carried out, considering the previous $W - 1$ preliminary classifications, in order to remove spurious ones. It is implemented by looking backward at W frames, assigning the class label $y_t = l$ to the observation window ξ_t if $l \in \{0, \dots, L\}$ is the most frequent label in the last W preliminary classification labels $\tilde{y}_t$. Step 3 ensures a more stable output, as the single window classifications may be noisy if the window is small and the gesture dictionary includes complex examples with similar subparts. It is worth noting that W is an important parameter of the method that needs to be tuned based on the specific characteristics of the gestures' dictionary: it needs to be long enough to capture long gestures, but as short as possible to reduce input delay.

4.6 Experiments

In Subsection 4.6.1, we report the results on the SHREC'22 benchmark; in Subsection 4.6.2, we focus on SHREC'19; finally, we present multiple ablative studies in Subsection 4.6.3. As comparative approaches, we consider several graph-based architectures like SeS-GCN [167] and MS-G3D [168], the attention-based networks DG-STA [169] and DSTA [85], a 1D convolutional network specifically designed for hand gestures, the Double-feature Double-motion Network (DDNet) [83], and a lightweight action recognition network (PSUMNET [164]). These approaches add to the competitors which can be found in the SHREC'22 challenge [6] and SHREC'19 challenge [3], for a total of 13 comparative methods.

Implementation details

The architecture of the network is shown in Figure 4.9. The view encoders are independent series of 1D Conv layers. Each task branch is a series of 1D Conv layers except for the last layers, which are fully-connected layers (with one final neuron for regression, three for SDN, and L for the fine-grained classifier). We adopted the Adafactor [170] with an initial LR of 0.004 over 100 epochs on 2 NVIDIA RTX3090.

Real-time 3D hand pose

As far as we know, the sequences in the benchmarks have been acquired with off-the-shelf devices and there are no guarantees on the accuracy of the tracking. This can possibly be one of the reasons for the non-optimal results of many methods, and OO-dMVM achieving better scores shows that it may be more robust against tracking errors. Improvements in hand tracking systems [171, 172] could further reduce the errors without changing the algorithm.

4.6.1 SHREC'22 results

In Table 4.7, we report the official gesture classification metrics as provided by the evaluation code. For each comparative model, we look for its best parameters. The W parameter should be as small as possible to lower input delay but large enough to capture significant information for longer gestures. In our case, we found $W = 16$ to be an optimal choice for SHREC'22.

OO-dMVM defines the new SOTA in all the metrics, with a fast inference time (4.1ms), and very low variances. Figure 4.10 highlights the detection rate per gesture type of the five best-performing models on SHREC'22. Notably, OO-dMVM is the most consistent top performer on this benchmark, with a detection rate that spans between 0.82 and 1, which is the thinnest min-max score range.

An important parameter of the evaluation metrics for SHREC'22 [6] is the *minimum overlap ratio* (MOR) of the detected gestures with the ground truth. When $\text{MOR} = 1$ the

Table 4.7: Classification results of SHREC’22 benchmark. The metrics are the detection rate (DR), false positives scores (FP), the Jaccard Index (JI), the delay in frames, and per-frame processing time. In brackets are the standard deviations. Columns SV/MV refers to the single-view or multi-view learning approach, while ST/MT refers to single-task or multi-task learning approach.

Method	DR $\uparrow$	FP $\downarrow$	JI $\uparrow$	Delay(fr.) $\downarrow$	time(ms) $\downarrow$	SV	MV	ST	MT
DeepGRU [90] (2018)	0.26 (.14)	0.25 (.23)	0.21 (.09)	8.00	3.1	✓		✓	
DG-STA [169] (2019)	0.51 (.10)	0.32 (.20)	0.40 (.20)	8.00	4.2	✓		✓	
SeS-GCN [167] (2022)	0.60 (.13)	0.16 (.09)	0.53 (.13)	8.00	1.8	✓		✓	
PSUMNET [164] (2022)	0.62 (.14)	0.24 (.15)	0.52 (.15)	8.00	24.4		✓	✓	
MS-G3D [168] (2020)	0.68 (.11)	0.21 (.15)	0.57 (.14)	8.00	29.3	✓		✓	
Stronger [6] (2022)	0.72 (.11)	0.34 (.26)	0.59 (.18)	14.79	100.0		✓	✓	
DSTA [85] (2020)	0.73 (.07)	0.24 (.13)	0.61 (.12)	8.00	9.2	✓		✓	
2ST-GCN 5F [6] (2022)	0.74 (.12)	0.23 (.05)	0.61 (.11)	13.28	2.1	✓			✓
TN-FSM+JD [6] (2022)	0.77 (.06)	0.23 (.12)	0.63 (.03)	10.00	4.6		✓	✓	
Causal TCN [6] (2022)	0.80 (.15)	0.29 (.22)	0.68 (.24)	19.00	28.0	✓		✓	
DDNet [83] (2019)	0.88 (.06)	0.16 (.18)	0.78 (.14)	8.00	2.2		✓	✓	
FG + SDN	0.86 (.06)	0.17 (.06)	0.75 (.11)	8.00	4.1		✓		✓
FG + SDN + GC	0.90 (.11)	0.10 (.11)	0.83 (.13)	8.00	4.0		✓		✓
OO-dMVM	0.92 (.06)	0.09 (.09)	0.85 (.11)	8.00	4.1		✓		✓

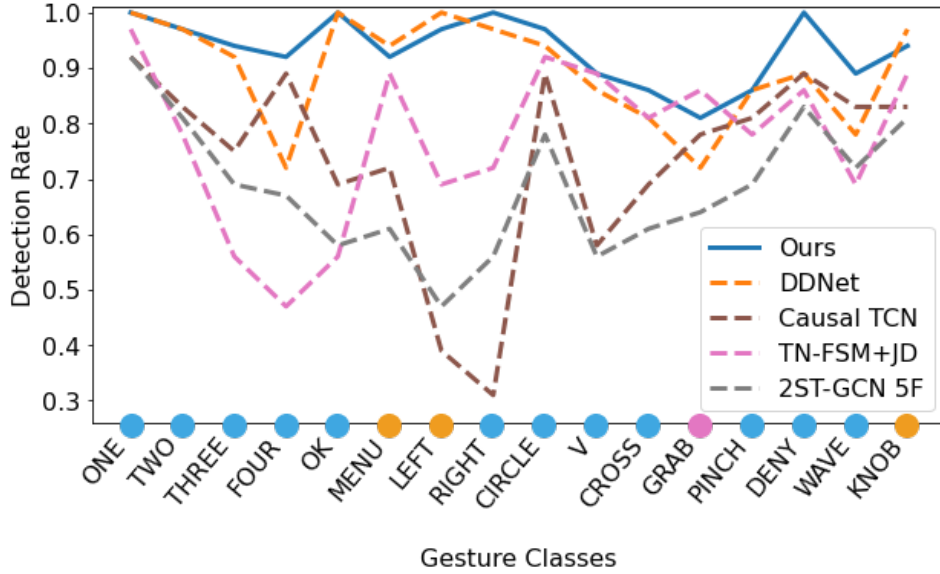


Fig. 4.10: Detection rate per gesture class for SHREC’22. Points are connected with a line for clarity. On the x-axis, a colored dot indicates, for each class, the method that achieved the best detection rate. Only the best 5 methods by DR are shown for clarity.

predicted gesture needs to completely envelop the ground truth, while a value closer to MOR = 0 relaxes this constraint. Like in the original evaluation protocol, the results are evaluated with MOR = 0.5. Figure 4.11 shows the Jaccard Index (described in Chapter 3) as a function of the Minimum Overlap Ratio. Obviously, the higher the MOR, the lower the JI of all the approaches. Until MOR = 0.7, OO-dMVM clearly dominates the competitors.

Of particular importance are the FP results, as they trigger unwanted actions, making the interaction frustrating. In Figure 4.12, a stacked barplot showing the total FPs for each model is presented (lower is better). Interestingly, the macro category, which in general attracts the most false positives, is the dynamic one, probably due to the fact that a movement of the limb is necessary to start this kind of gesture. As soon as we move towards more effective approaches, it is easy to see that the number of false positive gestures considered

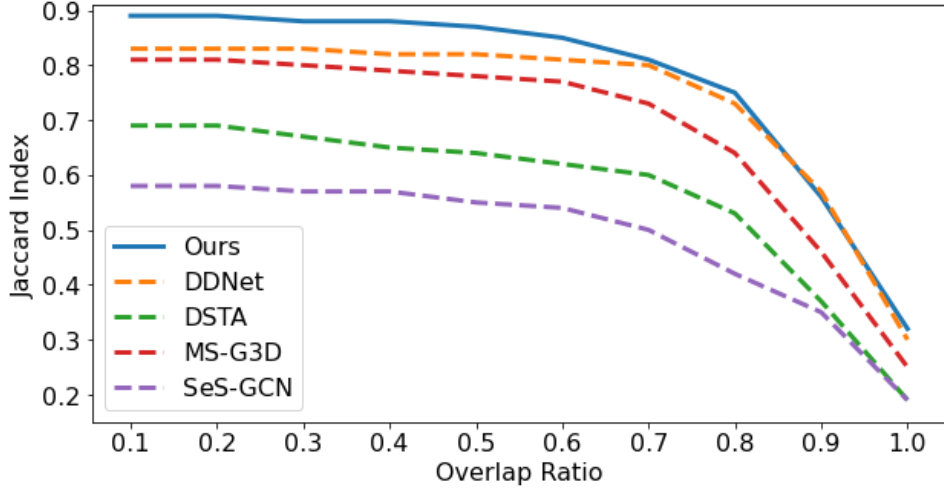


Fig. 4.11: Jaccard index for SHREC'22 as a function of the minimum overlap ratio. For clarity, we show only the best 5 methods.

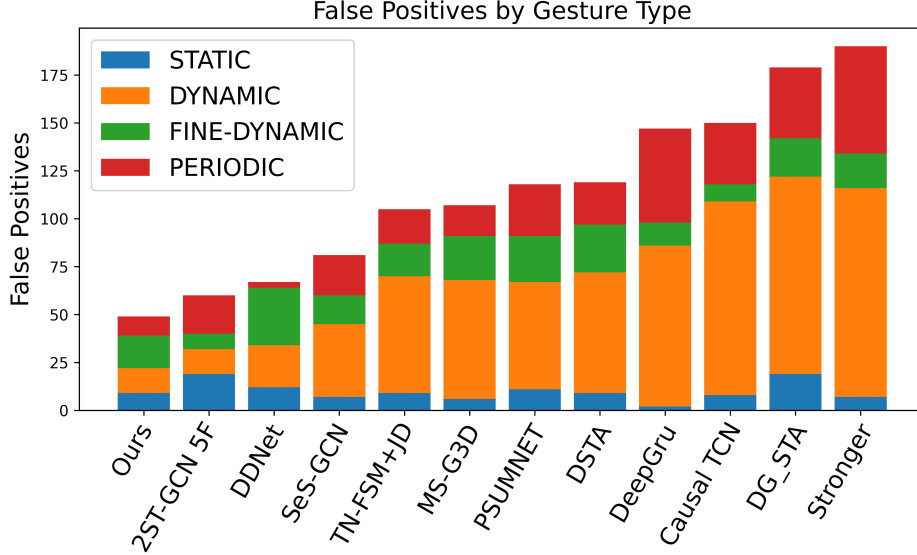


Fig. 4.12: Stacked False Positives for SHREC'22 per model. The gestures are divided into 4 types (colors). The FPs for each type are then summed to show the total value. Lower is better.

dynamic gestures diminishes drastically. Our approach has a balanced performance on all the gesture macro categories.

4.6.2 SHREC'19 results

Results of our tests on the SHREC'19 benchmark [3] are shown in Table 4.8. Some of the metrics are missing due to the unavailability of the original evaluations. The metrics differ slightly from the ones used in SHREC'22 [6] due to the differences between the SHREC'19 benchmark to its '22 counterpart, as discussed in Figure 2.5.2. For these tests, we chose a larger window size, $W = 40$. This is motivated by the longer average length of the gestures (~ 45 frames) and the fact that no short gestures are included in the dictionary. The results are consistent with the ones presented in Subsection 4.6.1 and show our model achieving state-of-the-art performances on almost all of the metrics except for the inference time.

Table 4.8: The metrics are the detection rate , false positives scores , and inference time. Parameters with (*) comes from [3] and may not be accurate. Dash (–) values were not reported in the original benchmark. Standard deviation is reported between brackets.

Method	DR $\uparrow$	FP $\downarrow$	time(ms) $\downarrow$
PSUMNET [164]	0.64 (.21)	0.22 (.20)	25.0
MS-G3D [168]	0.69 (.24)	0.25 (.22)	30.3
SeS-GCN [167]	0.75 (.20)	0.12 (.13)	2.0
SW 3-cent [173]	0.76 (–)	0.19 (–)	3.0*
DSTA [85]	0.81 (.11)	0.08 (.07)	8.8
DG-STA [169]	0.81 (.11)	0.07 (.05)	4.2
DDNet [83]	0.82 (.13)	0.10 (.09)	2.2
uDeepGRU [3]	0.85 (–)	0.10 (–)	3.0*
OO-dMVM	0.88 (.04)	0.05 (.04)	5.8

Table 4.9: Classification results on SHREC’22 with different Multi-task heads. Standard deviation is reported between brackets.

Method	DR $\uparrow$	FP $\downarrow$	JI $\uparrow$
FG	0.88 (.06)	0.16 (.18)	0.78 (.14)
FG + GS/GE	0.52 (.35)	0.48 (.31)	0.38 (.32)
FG + SDN	0.86 (.06)	0.17 (.06)	0.75 (.11)
FG + SDN + GC	0.90 (.11)	0.10 (.11)	0.83 (.13)
OO-dMVM	0.92 (.06)	0.09 (.09)	0.85 (.11)

4.6.3 Ablation studies

In this section, we show the impacts of our Multi-Task Learning On-Off paradigm, as well as discuss the impact of a correct training procedure for the regression heads in a Missing View scenario.

Task head removal

In this study, we explore the performance of OO-dMVM with some tasks removed. Results are in Table 4.9, where FG stands for fine-grained classification head only, FG + GS/GE adds the regression heads, and FG + SDN adds the coarse classification head. The FG model shows better performances than most methods of Table 4.7. In the table, we can see a pattern suggesting that the performance tends to increase by introducing the multi-view approach. The SDN module slightly decreases the performances but ameliorates the standard deviations. Surprisingly, the sole addition of the regression heads to the FG head drastically reduces the performances. This is probably due to the low task relatedness (start/end gestures are independent of the specific gesture classes), creating competing gradients during the model optimization. However, adding the SDN head allowed for reaching the optimal balance between classification and regression, with the latter improving the overall performance. It is worth noting that a grid search for the weights of the single tasks has been carried out, showing that uniform weighting was the best setup, and that each different task had a role in defining the optimal solution. Therefore, our joint OO-dMVM appears to be the right task ensemble. Additionally, we removed the regression heads in favor of a binary classifier (GC) trained to classify if a window contains the start or end of a gesture. As opposed to the always-active classification performed by GC, the On-Off regression procedure produces better performance and more stable results.

The On-Off regression head

Since the regression heads are the ones switched on and off by OO-dMVM, we further investigate them. In particular, we trained with different On-Off policies for the regression

Table 4.10: Classification results on SHREC’22 with different Regression Heads training procedures. Standard deviation is reported between brackets.

Method	DR $\uparrow$	FP $\downarrow$	JI $\uparrow$
Window Error	0.76 (.33)	0.24 (.55)	0.65 (.31)
Index Error	0.83 (.14)	0.20 (.21)	0.72 (.19)
OO-dMVM	0.92 (.06)	0.09 (.09)	0.85 (.11)

heads. The Index Error simulates inaccurate gesture segmentation by randomly changing the gesture start/end index in window ξ_t containing them. The Window Error is more dramatic since it randomly switches on/off the heads with a probability $p = 0.5$. It induces two kinds of errors since a wrongly activated window also implies a wrongly indexed gesture start. In Table 4.10, the results confirm our expectations: both errors degrade the performance, with the Window Error having a worse impact compared to the Index Error.

4.7 SHREC 2024

The SHREC 2024 [13] track focused on recognizing dynamic hand motions while molding clay, a unique task within the 3D Shape Retrieval Challenge. The competition involved classifying highly similar motions from professional potters into one of seven predefined classes. The data consisted of 3D skeletal landmarks from both hands, captured during a professional potter’s precise and synchronized movements. Challenges included the subtlety of the motions, sequence length variability, and the training dataset’s limited size, making it necessary to focus on fine motion details. In contrast to previous benchmarks like SHREC 2022, this dataset and task emphasize motions using both hands over variable time lengths.

Participants developed algorithms to classify these hand motions, submitting both their methods and results for evaluation. Accuracy was assessed using standard metrics like precision, recall, and F1 score. The competition aimed to advance motion capture and recognition technologies by exploring nuanced hand gestures in realistic, continuous interaction settings.

We adapted OO-dMVM for this specific action recognition problem. Starting from the OO-dMVM code, we adjusted it by eliminating the multi-task component, calling it: *Windowed multi view*. We got a 75% of accuracy on the SHREC24 benchmark as shown in Table 4.11.

Method	Accuracy	Params	Training time (s)
DD-Net	0.8167	27,536,263	1,210
SkelMAE	0.8667	27,767,693	3,180
Windowed Multi View	0.7167	213,308	817
DET-ACTIONS	0.9167	190,036	1,800
HMM-based approach	0.7500	2,450	804
RNN-based approach	0.8677	220,935	6,600
SE(3)-equivariant GCN	0.7500	3,003	503

Table 4.11: Comparison of total accuracy, number of parameters, and training time in seconds for a singular fold.

4.8 Discussion

STRONGER was conceptualized as a client-server architecture that employed a modified DDNet architecture with a sliding window approach. A key contribution of this system

was its ability to reduce false positives by optimizing classifier thresholds. STRONGER demonstrated the feasibility of using dictionaries of heterogeneous gestures, which are more complex than those typically employed in mixed reality applications. Despite its promising results, STRONGER faced certain limitations. In particular, it struggled with fine-dynamic gestures, where false positives were higher than desirable. Additionally, while the system’s execution time was suitable for real-time applications, it required further optimization for improved usability. Planned improvements included integrating gesture pre-segmentation inspired by the SHREC’21 contest results, capturing custom training data directly using the HoloLens 2, and exploring the possibility of training separate networks for specific gesture classes or subsets, such as static, coarse-dynamic, and fine-dynamic gestures.

Building on the foundation laid by STRONGER, MVMT (Multi-View Multi-Task) introduced significant advancements that redefined the approach to online gesture recognition. This paradigm exploited the complementary nature of geometric hand pose and motion data, processing these views concurrently across multiple tasks. This approach closely mimics human gesture interpretation, where both the start and meaning of a gesture are decoded simultaneously. MVMT also addressed scalability, enabling the system to adapt effectively to diverse and complex gesture types, and introduced a multi-task framework that handled gesture segmentation and classification concurrently, thereby reducing delays in recognition.

MVMT has consistently achieved state-of-the-art results on SHREC’22, validating its effectiveness in handling heterogeneous and complex gesture sets. Nevertheless, challenges remain. False positives for fine-dynamic gestures still require attention, and the delay in recognition is influenced by the size of the sliding window, suggesting opportunities for further refinement.

The progression from STRONGER to MVMT reflects a commitment to advancing online gesture recognition methods. However, further research is needed to address the remaining challenges and broaden the applicability of these methods. Reducing latency through gesture forecasting could eliminate delays entirely, while insights gained from detection rate analyses could inform the design of novel, more detectable gestures. In [Chapter 6](#), we discuss how we used MVMT in a real use case scenario like a Cyber-Physical Factory environment.

Data Augmentation

Data augmentation is a powerful tool in gesture recognition, especially for expanding the training dataset and improving model performance. While traditional augmentation techniques are well-explored for 2D gestures, their application to continuous mid-air 3D hand gesture recognition still needs to be explored. We analyzed the impact of various augmentation methods on a recent benchmark, demonstrating how classical augmentation techniques applied to hand trajectory, orientation, and pose can significantly enhance detection performance.

Gesture recognition is typically performed by using classifiers trained on a limited number of examples executed by a few subjects. In this context, *data augmentation* can enrich the training data with diverse variations of labeled gestures, significantly enhancing the results. Additionally, data augmentation can mitigate overfitting by exposing the model to a broader range of inputs, allowing it to learn more effectively from the available data. In our study, the training dataset is extended by applying transformations to the examples' hand trajectory and pose. Training models on the extended dataset makes them better equipped to handle different user behaviors and gesture executions, leading to higher accuracy and reliability in recognizing gestures in practical applications.

In this Chapter, we investigate the effectiveness of augmentation techniques for the online/continuous detection of heterogeneous gestures (e.g., including static and dynamic gestures characterized by changes in hand position and articulation) that can be used to interact in an XR setting. We leverage a recent benchmark designed for this task, SHREC'22 [6], testing how augmentation methods can improve online detection performances through the specific criteria introduced by this benchmark. The main contributions of our work are the following:

- a "classical" data augmentation strategy tailored for hand pose sequences and continuous recognizers decoupling global hand trajectory, local hand frame pose, and fingers' articulation
- analysis of the effect of separately augmenting gesture classes and non-gesture sequences
- evaluation of the augmentation methods on the 3D continuous gesture recognition task using its metrics.

Non-gesture examples can vary significantly and are far more represented in the training data; however, sampling only from the non-gesture space of the training data leads to poor non-gesture rejection and high false positive scores in practice. These characteristics require a careful design of augmentation techniques that preserve the distinctive features of each gesture while separating them from non-gestures.

5.1 Method

Our goal is to find a simple method for the effective augmentation of datasets used for 3D heterogeneous hand gestures recognition system training. Our aim is to provide designers of XR interactive systems with a method that enables the training of gesture recognition

methods with a limited number of examples. In this context, we focus on classical augmentation specifically designed for the task, where input data can be hand skeleton streams captured by headsets (e.g., Oculus Quest, Hololens, etc.), and the recognizer should perform continuous-time detection and classification of heterogeneous gestures. For this reason, we rely on the only existing benchmark available (SHREC'22 [6]), and, similarly to what is done in [119] for 2D finger gestures, we evaluate the effect of the proposed augmentations on a state-of-the-art online recognition method [10] trained on a very limited subset of the original training data, executed by a single user. The augmentations are designed to operate on all the aspects varied in the 3D hand gestures (hand trajectory, orientation and finger articulation), without taking into account the physiological realism but only adding perturbations as long as the new data improves the recognition model performance.

We used the well-known one-versus-all method [119, 174, 175] to obtain the baseline results of a single user and then compare them to all the singular augmentations. During the training phase, we used the gesture sequences of the first user to train and the other two users to validate. Then, we ran the trained neural network on the original SHREC'22 test set to assess the results.

5.1.1 Sliding window detector

In our test, we perform the online gesture detection/classification with a sliding window approach using the method OO-dMVM [10] described in Section 4.4, which obtained state-of-the-art performances on the SHREC'22 benchmark. This means that we trained a classifier with fixed-size windows, where a window is a fixed number of consecutive frames (recorded hand poses) sampled regularly from the training sequences. Every frame has its label (obtained from the start-end annotations, including the "non-gesture" label, assigned when most of the window is outside the annotated gestures), and the windows are labeled according to the most frequent frame label. At test time, gestures are detected using the trained classifier on the last frames of the online sequence. This creates problems and peculiarities in the classification since dynamic gestures and non-gesture classes can be longer than the window size and are recognized from their subparts. In addition, non-gesture data is much more frequent in the training data, but its variability in the training data clearly does not cover large parts of the class space (covering all the physiologically plausible movements not included in the gesture dictionary). Our augmentation procedure creates novel sequences by segmenting the gestures in the training sequences and modifying and interleaving them in non-gesture classes. Then, the additional windows extracted from the synthetic sequences are added to the set of training windows.

5.2 Augmentation details

Each sequence in the SHREC'22 dataset consists of gestures intertwined with non-gesture movements, so we divided gestures from non-gesture, creating two separate datasets to augment. In addition, when we created augmented gesture samples, we remade the 'original continuous sequence', inserting the gesture in a synthetic non-gesture background. Finally, we extract the augmented windows from these synthetic sequences to train the neural network. The sequences to be augmented are decomposed in global trajectory, local orientation, and hand pose, and we implemented specific augmentation strategies for each level of detail described in the following subsections. Each comes with a control parameter: we will explain how we selected and fine-tuned the parameters in Subsection 5.3.2.

5.2.1 Global trajectory augmentations

We created new trajectories (see Figure 5.5 for a detailed visualization) using four different augmentation methods:

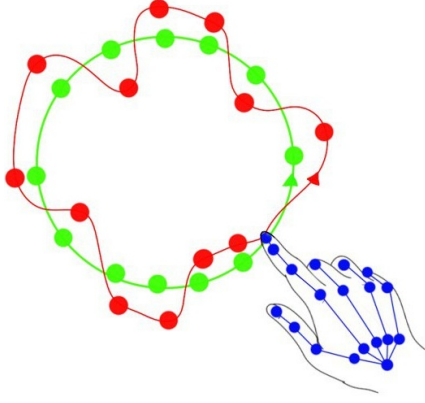


Fig. 5.1: Trajectory noise

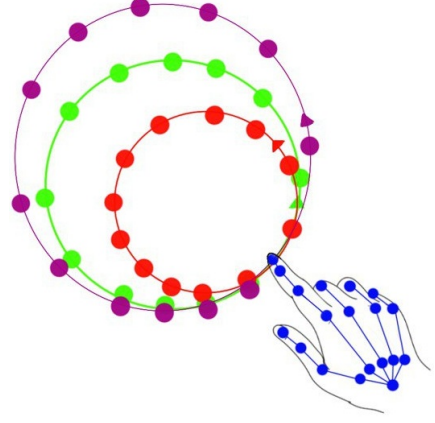


Fig. 5.2: Trajectory Scaling

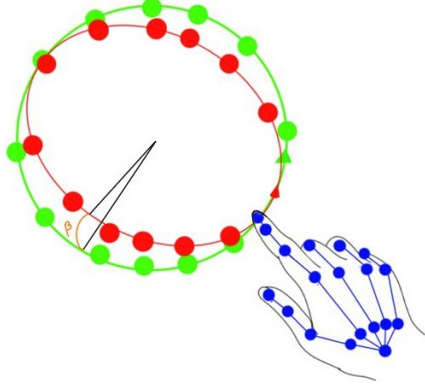


Fig. 5.3: Trajectory Orientation

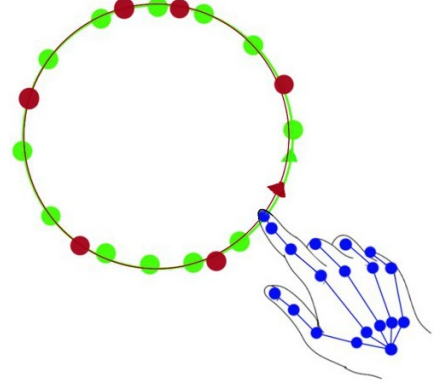


Fig. 5.4: Trajectory Resampling

Fig. 5.5: In this figure, the four trajectories augmentation types are visualized. They represent [Figure 5.1](#) Position noise, [Figure 5.2](#) Scaling, [Figure 5.3](#) Orientation, [Figure 5.4](#) Resampling.

- **Noise:** adding noise enhances the robustness of the recognition method; specifically, we introduced random 3D shifts to each time frame of the sequence, applying the same displacement to all the skeleton points ([Figure 5.1](#)). The augmentation is controlled by the parameter Δ , which defines the range of variability of the points' shift ($[-\Delta, \Delta]$, $[-\Delta, \Delta]$, $[-\Delta, \Delta]$).
- **Scaling:** augmenting the scale of trajectories enhances the recognition system's robustness to variations in hand movements across different users and environments; this variability in training improves the model's ability to generalize across a broader range of scales. It enables the system to recognize gestures consistently, whether performed with larger or smaller hand movements. We apply a common scale factor s to all the key points' coordinates or trajectories (see [Figure 5.2](#)).
- **Orientation:** users might perform gestures at different orientations due to hand positioning, user posture, or interaction context variations. This augmentation ensures accurate recognition even when gestures are performed at angles not seen during the initial training phase, effectively handling the natural variability encountered during actual use. We create augmented gestures with a perturbed orientation by applying a rotation of an angle θ around a randomly selected axis (see [Figure 5.3](#)).
- **Resampling:** Gestures can vary in speed and duration when performed by different users or in different contexts. Resampling the trajectories, assuming the same frame rate and using then fixed-size windows for the gesture classification, we can actually simulate the execution at a different speed (see [Figure 5.4](#)). To generate augmented

gestures, we applied a uniform resampling of segmented gesture/non-gesture segments, multiplying the number of sampled frames by a factor f and performing spline-based interpolation of the trajectory.

5.2.2 Hand orientation augmentation

While the semantics of some gestures change with orientation, it is clear that different subjects can perform the same gestures with slightly different hand orientations, varying with time. Looking at the executions of the gestures included in the SHREC'22 dataset, it is evident that the orientation of the hands in different executions of a CIRCLE or a WAVE gesture presents non-negligible variations.

For this reason, the addition of augmented gestures obtained by executing random pose rotations at each time frame of the sequence is expected to allow the system to identify gestures more accurately. As shown in Figure 5.6, to create the augmented gestures, we rotate the pose of the hand at each time frame of a random angle in the range $(0 \gamma]$ around a randomly oriented axis passing through the wrist joint.

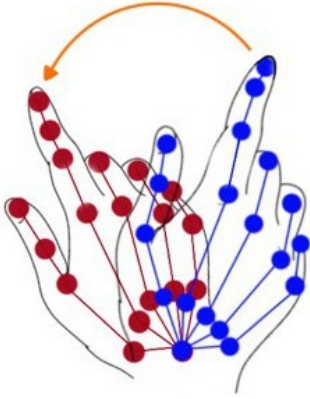


Fig. 5.6: Hand pose orientation

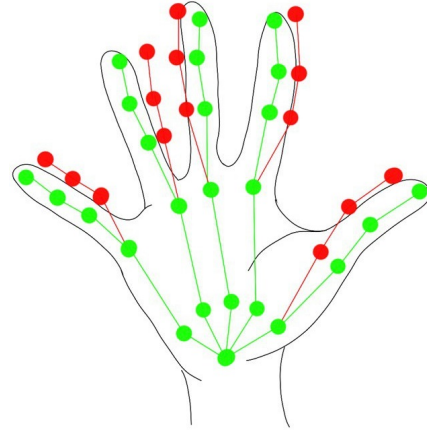


Fig. 5.7: Articulation noise

Fig. 5.8: Figure 5.6 Hand orientation noise is added by rotating the skeleton around the wrist node. Figure 5.7 Fingers' articulation noise is added by summing random components to the PDM coefficients.

5.2.3 Articulation Noise

Given that each individual is unique, there are different ways that people move and articulate their hands, which results in pose variations. In our method, we represent the class-specific variability as a Point Distribution Model (PDM) [176] obtained from the available training examples. The method projects the deviation of the joints' coordinates from the average pose on the eigenvectors of the covariance matrix, thus identifying the main modes of the allowed articulation changes. Each gesture frame can be compactly represented by the first coefficients of the PDM representation, and we alter these coefficients to create the augmentations.

We trained a PDM for each gesture class, transformed each pose into the eigenvector space, and calculated the ranges of variation for each principal component to determine the minimum and maximum values of the reasonable coefficients for the gesture class. We considered the first 10 eigenvectors, as we found that they are sufficient to explain more than 99% of the pose variability for all the classes. Augmented gestures are created by adding or subtracting to each of the 10 coefficients C_i (describing the pose at each time frame) the

value of the corresponding max-min range multiplied by a control parameter, ranging in the interval $[-\alpha, \alpha]$. Each different augmentation is generated by selecting different values for α in the range $[0, 1]$. The modified representations projected back to the original pose space result in plausible alteration of the finger articulation (as shown in Figure 5.9)

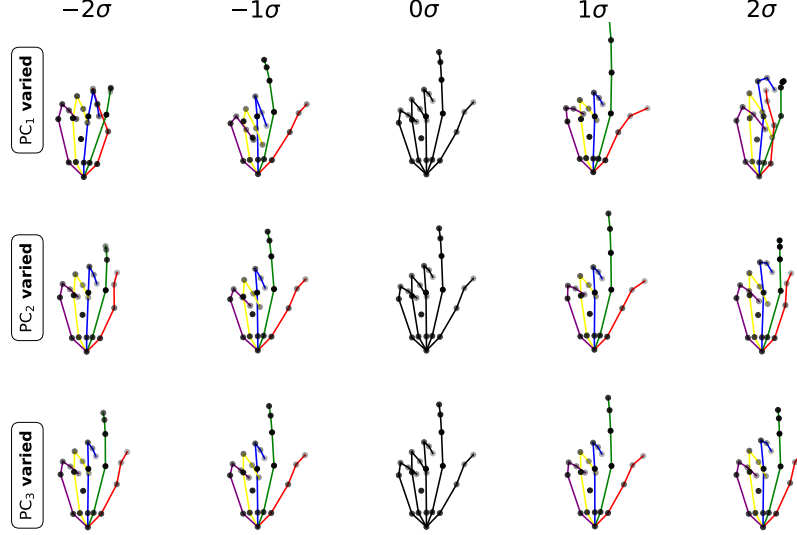


Fig. 5.9: The central column of this grid represents the original pose (PINCH). The rows show the variations along the first three principal components of the PDM built for the PINCH gesture, labeled as PC_1 , PC_2 , and PC_3 . The columns represent the effect of perturbations at different standard deviation offsets from the mean, specifically -2σ , -1σ , 0σ , 1σ , and 2σ .

5.2.4 Dataset Augmentation Strategies

In the continuous setting, we are interested in evaluating separately the effect of gesture and non-gesture augmentation. It is also important to tune separately the parameters used for the generation of the augmented dataset, as non-gesture examples should feature a larger variability. For this reason, we segmented each original training set's sequences into two distinct sets composed of gestures and non-gesture segmented sequence fragments and generated augmented samples (and finetuned parameters) separately for the two sets. From the two augmented sets, the windows used to train the sliding windows-based continuous classifier are then extracted, and the trained classifier is tested on the SHREC'22 test data. By augmenting the non-gesture data, we increased its prevalence relative to gesture data in the dataset, risking poorer accuracy in gesture class prediction due to increased class imbalance in the training set. However, the goal was to cover a wider range of non-meaningful hand poses and trajectories, thereby reducing the false positive rate. After fine-tuning, we combine all fine-tuned gesture augmentation strategies with non-gesture augmentation and report these results in Table 5.2

5.3 Experiments

5.3.1 Establishing a Baseline

We adopted a 1-user-vs-all testing approach to simulate a small amount of training data available. From the original SHREC'22 train-split, we augmented the sequences of a single subject from the training set, using the remaining two subjects as the validations set during

the offline training phase of the network. For reporting metrics, we used the same test-split and online evaluation metrics as proposed in SHREC’22. We followed this method to obtain the baseline performance: the continuous detection evaluation metrics obtained on the SHREC’22 test set using the classifier described in [10] and the single subject sequences with no augmentations applied. The baseline values appear in Table 5.1 and Table 5.2 and are $DR = 0.72$, $FPS = 0.32$, $JI = 0.58$. Improvements above this baseline can be attributed to the effectiveness of augmentation techniques.

5.3.2 Generation of the augmented datasets, fine-tuning of the augmentation ranges and online detection results

We generated the augmented training sets from the gesture/non-gestures sequence fragments of the single subject considered, adding ten augmented fragments for each of the methods described in Subsection 5.2.1 and Subsection 5.2.2, sampling the related parameters from specific ranges. To set reasonable values for these ranges, we performed a specific fine-tuning, optimizing the continuous classification performances on the validation set composed by the gestures of the remaining SHREC’22 training subjects.

For each of the values of the parameters, we started from a small $[\min \max]$ range approximately corresponding to the variability of the related property in the training set. We used two different parameter selection approaches to avoid the risk of over- or under-representing specific variations in the augmented data. For $\Delta, \theta, \gamma, \alpha$ parameters, we randomly sampled from uniform distributions in the respective ranges, deliberately excluding values close to zero. For s and f , we generated values evenly spaced within the specified range, excluding values close to one, to systematically explore the parameters in the space.

We then tested increased ranges, finally choosing the best range that minimized the Jaccard Index in evaluating the continuous classification of the validation set sequences. We fine-tuned a single augmentation method at a time, optimizing each parameter independently. Fine-tuning for the gesture and non-gesture sequence subsets was performed separately. Table 5.1 shows the ranges tested for each parameter and the one selected to generate the augmented gesture dataset.

Augmentation	Param	Ranges tested
Trajectory Noise	Δ (m)	[0, 0.001], [0, 0.002], [0, 0.003], [0, 0.01], [0, 0.1], [0, 1]
Trajectory Orientation	θ (rad)	[0, 0.01], [0, 0.02], [0, 0.04], [0, 0.08], [0, 0.16], [0, 0.32], [0, 0.64], [0, 1]
Trajectory Scaling	s	[0.5, 2], [0.6, 1.8], [0.7, 1.6], [0.8, 1.4], [0.9, 1.2]
Trajectory Resampling	f	[0.5, 2], [0.6, 1.8], [0.7, 1.6], [0.8, 1.4], [0.9, 1.2]
Hand Pose Orientation	γ (rad)	[0, 0.01], [0, 0.02], [0, 0.04], [0, 0.08], [0, 0.16], [0, 0.32], [0, 0.64], [0, 1]
Articulation Noise	α	[0, 0.1], [0, 0.2], [0, 0.5]

Table 5.1: Ranges tested in the fine-tuning for each augmentation method used to synthesize gesture and non-gesture.

It is possible to see in Table 5.2, that single augmentations provide a good increase in online classification performance. With all the augmented gestures created with the best ranges for each type, we created the *All Augmented Dataset* (AAD). While each augmentation technique separately improves the metrics over our baseline, the union of all augmentations (‘All Augmented Dataset’ *AAD*) achieves the highest detection rate. Interestingly, trajectory scaling leads to the best false positive scores, even compared to AAD. The same parameters range optimization was performed separately for the non-gesture samples. Table 5.2 shows the optimal ranges obtained and the corresponding online classification results.

Non-gesture augmentation is essential as non-gesture performed by different subjects may differ from those performed by the single training subject. Enhancing the variability of non-gesture data can improve the algorithm’s ability to differentiate between gestures and non-gesture. In theory, increasing the ratio of non-gestures to gesture data in a dataset that is already more than half composed of non-gestures could lead to worse online recognition

performance due to heavy class imbalance. Thus, our results are not obvious since we observed an improvement in both DR and FPS metrics over the baseline in each augmentation method. Also, in this case, we created a complete *Non-gestures Augmented Dataset* (NGAD) with all the augmented non-gestures sequences created with the best ranges for each type. NGAD achieves the highest DR and JI compared to applying one method at a time.

		DR↑	FPS↓	JI↑
<i>Baseline</i>		0,72	0,33	0,58
Gestures				
Augmentation	Optimal range	DR↑	FPS↓	JI↑
Trajectory Noise	[0, 0.003]	0,79	0,27	0,65
Trajectory Resampling	[0.6, 1.8]	0,77	0,31	0,62
Trajectory Scaling	[0.9, 1.2]	0,82	0,26	0,69
Trajectory Orientation	[0, 0.02]	0,79	0,29	0,64
Pose Orientation	[0, 0.32]	0,78	0,31	0,63
Fingers'articulation	[0, 0.5]	0,82	0,23	0,69
AAD (union of all)	-	0,83	0,26	0,69
Non-gestures				
Augmentation	Optimal range	DR↑	FPS↓	JI↑
Trajectory Noise	[0, 0.003]	0,81	0,22	0,67
Trajectory Resampling	[0.8, 1.4]	0,77	0,28	0,63
Trajectory Scaling	[0.6, 1.8]	0,77	0,27	0,64
Trajectory Orientation	[0, 0.16]	0,79	0,27	0,65
Pose Orientation	[0, 0.08]	0,76	0,23	0,64
Fingers'articulation	[0, 0.2]	0,79	0,30	0,64
NGAD (union of all)	-	0,84	0,25	0,70

Table 5.2: Continuous detection results (Detection Rate (DR), False Positive Score (FPS), and Jaccard Index (JI)) obtained on the SHREC'22 benchmark with the classifier presented in [10] using different training sets. Top block: scores obtained training with the examples of a single subject (baseline). Middle block: results obtained augmenting the training gestures with all proposed methods (in the optimal parameters range shown in the second column). Bottom: results obtained augmenting the non-gesture segments with all proposed methods (in the optimal parameters range shown in the second column).

Technique	DR ↑	FPS ↓	JI ↑
Baseline (no augmentation)	0,72	0,33	0,58
AAD	0,83	0,26	0,69
NGAD	0,84	0,25	0,70
AAD + NGAD	0,85	0,23	0,72

Table 5.3: This table reports the online evaluation scores (DR = Detection Rate, FPS = False Positive Score, and JI = Jaccard Index) of the Baseline, the union of all augmented gestures (AAD), the union of all augmented non-gestures (NGAD), and the combined union of both (AAD + NGAD).

Then we created an augmented dataset (*AAD + NGAD*) that is the union of the two combined gesture and non-gestures datasets (*AAD* and *NGAD*) and evaluated the performance of the online classifier trained with it. At the DR of 84%, this model outperformed all other augmentation strategies (see Table 5.3) and keeps the best FPS value that has NGAD.

The finetuned parameter ranges have also been used to augment the single-subject acquisitions of the other two subjects included in the SHREC'22 training set. Table 5.4 shows that we obtain a similar improvement in the online detection performances without subject-specific fine-tuning using the same protocol used for the first subject.

Dataset Used	DR $\uparrow$	FPS $\downarrow$	JI $\uparrow$
Subject 2 Original Data	0,73	0,39	0,56
Subject 2 AAD + NGAD	0,81	0,29	0,69
Subject 3 Original Data	0,74	0,34	0,58
Subject 3 AAD + NGAD	0,82	0,31	0,67

Table 5.4: This table reports the online evaluation scores (DR = Detection Rate, FPS = False Positives Score, and JI = Jaccard Index) of the Baseline of Subject 2 and Subject 3 from the training set of SHREC’22 and the augmented versions obtained using AAD + NGAD.

5.3.3 SHREC’17 comparison

There are not many 3D continuous gesture datasets available, so it is challenging to test our methods on a broad range of datasets. However, we have decided to also evaluate our techniques on the well-known segmented SHREC’17 [43] benchmark. The presented baseline in [120] is achieved through a CNN classifier that is not state-of-the-art (4.3% accuracy), we employed the DDNet [149] classifier, which has a 91.9% accuracy. To assess the effects of our augmentations in SHREC’17, we used the fine-tuned parameters for SHREC’22 without any modifications (reported in Table 5.2) and continued the same process by creating ten augmentations per method for each gesture. Following this procedure, we incremented the original DDNet performances, shown in Table 5.5, achieving 92.6% accuracy in only 50 training epochs. We also tested our methods using the 1-user-vs-all approach, achieving a baseline accuracy of 16.5% without augmentation and reaching 44.5% accuracy using our techniques’ union. However, since the parameters were not fine-tuned for this specific classifier and benchmark, there is a possibility that the optimal result is not achieved, and it might be possible to further improve it by fine-tuning the parameters.

Method	Full Trainset Accuracy $\uparrow$	1-User Accuracy $\uparrow$
Original Data	91,9	16,5
Augmentation	92,6	44.5

Table 5.5: Classification accuracy of DDNet on SHREC’17 dataset using the full trainset and a single-user setup, with and without augmentation.

5.4 Discussion and Future Work

Our results show that classical augmentation is a viable solution to improve online gesture recognition performance. In contrast to prior work [135] that did not find improvement while using data augmentation with a CNN recognizer, our setup improved the baseline performances on the SHREC’22 dataset despite training with only a single user’s data. The first key finding in our experiment was that a dataset that comprises all the gesture augmentations proposed guarantees better scores compared to using only the original training data. The use of an augmentation procedure that is aware of the hierarchical structure of the hand and of the constraints of the finger articulation provides better results than independent, random point shift, even if trained only on single-user data.

The pose augmentation method introduced in this work [120] applies random noise to a subset of joints by first selecting a random number of joints and then adding Gaussian noise to their positions, but we observed that it led to suboptimal results in detection performance. Specifically, the detection rate for gestures was 0.79 with a false positive rate (FPR) of 0.22, while for non-gestures, the detection rate was 0.75 with an FPR of 0.30. To improve this, we introduced a pose variation method Subsection 5.2.3 that uses a PDM, which better preserves the articulation of hand gestures while introducing realistic variations that achieve superior results.

The second key finding was the importance of non-gesture samples on the model’s performance. Looking at [Table 5.2](#), it is evident that non-gesture augmentation leads to large improvements over the baseline, and it is much more effective than gesture augmentation. Non-gestures synthesized with our method led to an increase in DR and FPS, which was not obvious because increasing the number of non-gesture samples increases the class imbalance of the training dataset. The union of all of the augmentation methods for non-gestures appears to be the most effective than every single augmentation. Instead of gestures, we got a tie on the Jaccard index of the finger articulation method that could be enough to guarantee the best result. We can assume not all augmentation types contribute equally to performance gains, it may be beneficial to apply only a subset of the proposed augmentations rather than all of them. Future work could explore model-based hand pose generation techniques that respect the physical constraints of real hands, allowing for a more controlled synthesis of novel pose streams. The augmentation techniques we studied are effective for hand trajectory and pose data collected with the HoloLens 2, but their applicability extends beyond this dataset. With the introduction of the SHREC’17 dataset and our improved data, we demonstrated their effectiveness across different sources.

Another limitation of the method is that the fine-tuning of the augmentation parameters based on optimizing the classifier performance on the validation set is time-consuming and dataset-specific. However, given the hierarchical structure and the explainability of all the parameters defining the hand pose changes, it could be possible to derive the maximum ranges of parameter variation without changing the semantics for all the gestures belonging to a specific type (static, dynamic, coarse and fine) with a specifically-designed user study. We plan to perform this study in future work.

In conclusion, our study confirms the efficacy of data augmentation in improving gesture recognition performance, even with a reduced amount of training data. Also, we have proposed an augmentation method that different skeleton model can use benchmarks to generate augmented gestures and non-gestures. This synthesized data added to the original samples notably improved the gesture recognition detection rate, false positive score, and Jaccard index. We demonstrated the utility of augmentation not only for segmented gestures, as shown in previous studies [\[119, 120\]](#) but also in the domain of continuous recognition, where reducing false triggers is paramount. This is critical in high-stakes real-time applications that leverage gestural interactions, such as operating heavy machinery in industrial environments, surgeon operations, and even video games. Our research informs future directions and applications of data augmentation in the context of online gesture recognition.

Practical Applications of Gesture Recognition in Industrial XR Environments

The research and development of gesture recognition methods, as detailed in [Chapter 4](#), focus on bridging the gap between theoretical advancements and practical applications. My work leverages robust, real-time algorithms designed for use in realistic scenarios, as showcased in two application domains: industrial environments and AR.

In [\[15\]](#), we highlight the deployment of an advanced gesture recognition system using a lightweight 1D convolutional neural network trained under the On-Off deep Multi-View Multi-Task framework [\[10\]](#) (see [Section 4.4](#)). This system supports the real-time recognition of diverse gestures, including dynamic and fine-grained hand motions, enabling effective interaction in augmented workplaces. The app was evaluated with minimal latency and demonstrated its ability to execute gestures from the SHREC'22 dictionary in industrial setups, proving its usability for controlling machines in augmented environments.

In [\[12\]](#) we presented a virtual environment that integrates real-time data from industrial machines into a detailed digital twin. Using gesture-based interactions, such as pinch gestures for navigation and data retrieval, the application facilitates efficient control and monitoring of machinery. This integration showcases the potential of gesture recognition in enhancing human-machine symbiosis in Industry 4.0 and 5.0 environments, providing a human-centric approach to industrial operations.

Within the study [\[11\]](#), I worked on integrating mixed reality technologies to improve worker safety and enhance interaction in collaborative environments. My role included incorporating HoloLens 2 into human-robot interaction tasks, creating intuitive gesture-based commands for collaborative robots, and developing safety-compliant interaction models using mixed reality. This work also addressed challenges in human-machine collaboration by ensuring the seamless interaction of physical and digital twins. These contributions have provided valuable insights into the potential of gesture-based and immersive technologies for practical, industrial use.

This chapter is adapted from [\[12\]](#), and we explore the complete integration of eXtended Reality (XR) technologies within a demonstration laboratory that simulates Industry 4.0 and 5.0 compliant machinery. Our work highlights the design and implementation of augmented and virtual reality applications that enhance real-world tasks through advanced gesture-based interaction. We've developed a sophisticated communication architecture to synchronize data between the physical factory environment, its digital twin, and the XR replicas, enabling seamless interaction.

By transforming a manufacturing plant into a Cyber-Physical Factory, workers can benefit from smart visualizations and advanced interfaces, while remote operators can monitor and control processes in real-time. Our laboratory setup at the University of Verona is a cutting-edge testbed for this technology, allowing us to study and improve XR applications in a realistic industrial setting. Though current XR technology is ready for practical use, challenges still need to be addressed, such as device comfort, battery life, and field of view limitations. Our future work will address these issues and further enhance the system's capabilities. [\[14\]](#)

6.1 Integration of Extended Reality with a Cyber-Physical Factory Environment and its Digital Twins.

Virtual Reality (VR), Augmented Reality (AR), and, more in general, eXtended Reality (XR) are fundamental enabling technologies for Industry 4.0. As pointed out in [177], "the accessibility and accelerated progress in technological devices, incorporating advanced algorithms, ergonomic features, built-in cameras, and sensors, have encouraged a massive adoption and extensive application development in a wide spectrum of industry 4.0 domains.". They are also important for the more recent Industry 5.0 paradigm, focused on sustainable, resilient and human-centric solutions, as they can be considered "Individualized human-machine interaction technologies that interconnect and combine the strengths of human and machines" [178]. Thanks to the new interaction paradigms leveraged by XR, workers can be supported by assistive tools enhancing their safety and adapting to their needs and diversity [179].

If a manufacturing plant environment is transformed into a "Cyber-Physical Factory" (CPF) [180], with all the processes monitored and synchronized between the physical factory floor and the cyber computational space, workers can be helped in their tasks with smart visualization and advanced interfaces, and remote operators can monitor in real-time the status of the production process and possibly perform actions on machines.

The concept of "Digital Twin" (DT) [181] is used to indicate the creation of a numerical representation (and simulation) of a production plant. Specific software tools are available in the industrial domain to support the organization of the work, allowing the digital control of production recipes, process monitoring and analytics, predictive maintenance, etc.

The integration of these tools with XR technologies, allowing immersive visualization and advanced interaction is, however, still limited and requires both the development of new interaction paradigms, tools and a careful user evaluation.

One of the problems limiting the research on these aspects is related to the lack of Cyber-Physical Factory demonstrators where the combination of all these technologies can be tested and evaluated.

In this chapter, we present the design of a complete framework to test applications of XR in a realistic industrial context. The framework has been created within the Industrial Computer Engineering Laboratory (ICE laboratory) of the University of Verona, an advanced demonstrator for Industry 4.0 and 5.0 technologies.

The laboratory is a functional replica of an advanced manufacturing environment with specialized hardware devices and software tools, allowing students and researchers to work with up-to-date instruments in different technological areas. In this environment, the use of XR technologies to develop and test novel Human-Centered interaction design solutions can be studied in a realistic scenario, interfaced with a complete cyber-physical factory provided with state-of-the-art industrial digital twins. We already presented a brief analysis of the literature on the integration of XR technologies with CPF/DT in [Section 6.2](#).

In the following sections of this chapter, In [Section 6.3](#), we describe the main features of our laboratory. In [Section 6.4](#), we discuss the creation of the 3D model of the plant reproducing the environment and all the machines. [Section 6.5](#) describes the VR application allowing users to navigate in the laboratory, visualize real-time information on its equipment, and possibly trigger actions on the production management. In [Section 6.6](#), we describe the AR application that allows users to walk in the physical environment, visualize information superimposed on the different machines, operate on them with simple gestures, and have feedback and alerts related to the production processes.

6.2 State-of-the-Art Surveys on eXtended Reality Integration with Digital Twins and Cyber-Physical Factories

Many research papers describe the integration of Augmented Reality and Digital Twins/Cyber Physical factories, and several literature surveys have already been published in the field.

Künz et al. [182] reviewed selected literature on AR applied to DT, finding that the main research topics considered are the visualization of information and data, the guidance of participants through given tasks, and the interaction with DT through AR. A clear weakness in the literature surveyed is that it focuses mostly on technical aspects neglecting human factors, and more than half presents only descriptions of the proposed implementation or concept as results.

Yin et al. [183] also conducted a state-of-the-art survey from the AR-assisted DT perspective across different sectors of the industrial field. They reviewed 118 papers and identified three levels of AR-assisted DT: Virtual Twin, Hybrid Twin, and Cognitive Twin, with only the latter including effective integration of machine and human intelligence and high-level support of human work. They considered the readiness level of this kind of DT as low. They also pointed out several implementation challenges, related to the limited use of human interaction with AR for DT control, the difficult adaptation of CAD-based 3D models to the AR environments, the problems in the management of real-time data communication in the integrated AR-assisted DT system.

Another literature review of papers discussing augmented reality and DT was presented by Böhm et al. [184]. Their survey focuses on how cybersecurity could benefit from the combination of the two technologies. AR can, in fact, provide situational awareness, including the actual physical situation, allowing insights derived from DT data and simulations to be visualized together with the physical assets. They also discuss on how AR and DT can be integrated with existing security mechanisms.

The use of Virtual Reality simulation integrated with smart manufacturing environments has been investigated in [185], who developed an early prototype of a VR application visualizing the data of the real-time operations of a forklift for a user equipped with a VR headset and performed preliminary evaluation of delays and acceptability.

Chandra et al. [186] surveyed papers on the usage of virtual reality (VR) in simulating a digital factory gathering recommendations for developing effective tools. They highlighted that although "VR provides the best visualization, interaction, and immersion, it still lacks in becoming a high-fidelity, industrial-grade digital factory tool".

Pirker et al. [187] reviewed recent papers on immersive VR applied to Digital Twins. They identified various applications fields for digital twins combined with VR that is not limited to industry, but has been extended to other domains like health, automotive, BIM, education, or even gastronomy. They found that primary use cases for VR associated to DTs are remote operation, remote collaboration or guidance, training. However, a particularly interested case where VR can play an important role is related to evaluation and to the possible use of virtual environments as safe spaces for user studies. The surveyed papers pointed out also the issues related to the usability of VR (cybersickness, user acceptance, ergonomics), to the network efficiency and security) and the specialized hardware requirements.

Many of these surveys reveal that, despite the amount of literature in the field, it seems that VR and AR technologies are still not as exploited as they could in the cyber-physical factory domain. In this context, the development of a demonstrator where different solutions are integrated with standard industrial software solutions and a rich set of machines can be a relevant contribution that can stimulate the implementation of new research for the improvement of the instruments themselves and show the potential of the technology to the industries.

6.3 The ICE lab of the University of Verona: a complete Industry 4.0/5.0 technology demonstrator

The Industrial Computer Engineering laboratory (ICE lab) is a research facility of the University of Verona, designed to serve as a demonstrator for a wide set of computational technologies applied to industrial manufacturing within the frameworks of the Fourth and Fifth Industrial Revolutions (Industry 4.0 and 5.0). Fig. 6.3 shows two views of the environment.

The main feature of the laboratory is a complete and reconfigurable production line equipped with an assortment of heterogeneous devices grouped into manufacturing cells:



Fig. 6.1: (a)

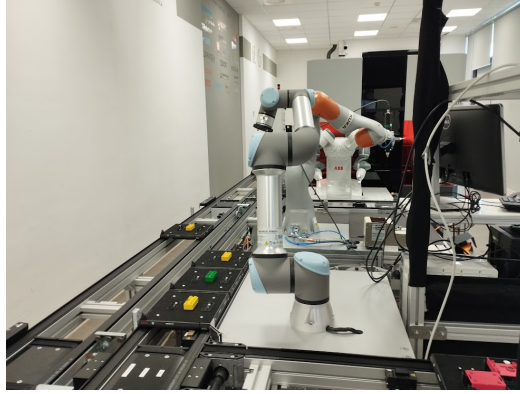


Fig. 6.2: (b)

Fig. 6.3: Two views of the ICE laboratory: (a) shows the milling machine close-by with the feeding arm in the forefront, the polyjet and stereolithographic printers in the middle, the autonomous ground vehicles and the vertical automated warehouse at the back; (b) shows the mini-pallet conveyor line on the left and the robotic arms in the middle.

autonomous storage and logistics, robotic assembly, quality control, additive manufacturing, subtractive manufacturing, and functional testing.

The autonomous storage and logistics cell supplies and circulates materials (parts and products) between the cells: a vertical automated warehouse (Ferretto VERTIMAG EF), two autonomous ground vehicles (Robotnik RB-KAIROS 5), a mini-pallet conveyor line (Bosch-Rexroth) configured as a ring. The robotic assembly cell includes two collaborative robotic manipulators (ABB YuMi IRB14000 Collaborative Robot, Kuka LBR iiwa 14 R820) with different end-effectors, such as gripper, suction and screwdriver tools. The quality control cell includes a set of cameras (optical, infrared, laser scanner) to inspect materials and products for visible defects. The additive manufacturing cell is made up of a set of 3D printing machines based on different technologies, namely fused-deposition modeling (Prusa i3 MK3S+), polyjet (Stratasys J826), and stereolithography (DWS Systems XPRO S) for building product components. The subtractive manufacturing cell has a four-axis computerized numerical control (CNC) milling machine (EMCO ConceptMill 105). The functional testing cell has a flying probe testing machine to check the quality of electronic boards (SPEA 4050 S2).

Moreover, the production line is surveyed by an array of video surveillance and RGBD cameras mounted all around the room to monitor workers and guests in the laboratory. The first array feeds a customized neural-network based tracking system that robustly follows people even if partially occluded by the equipment. The RGBD cameras instead provide skeletal-based pose estimation to analyze interactions with the machines and the environment. The laboratory is also monitored by meters in every production cell to measure energy consumption, and environmental IoT sensors within the room to record temperature, humidity, and brightness.

This set of sensors enable us not only to perform research on the efficient use of automation in industry (4.0 paradigm), but also on sustainability and inclusiveness of the factory (5.0 paradigm) that will be particularly investigated in future work.

All the equipment and sensors in the laboratory are managed by a Service-oriented Manufacturing (SoM) software architecture based on containerization and virtualization, implemented as a Kubernetes cluster with two main components: the Data Integration Hub (DIH) and the Automation Manager. The DIH monitors the equipment, stores the gathered data, and propagates to the machines the commands requested by the Automation Manager. It acts as a service platform that provides a variety of functionalities: machine monitoring and control through OPC-UA servers and clients, communication between Kubernetes nodes

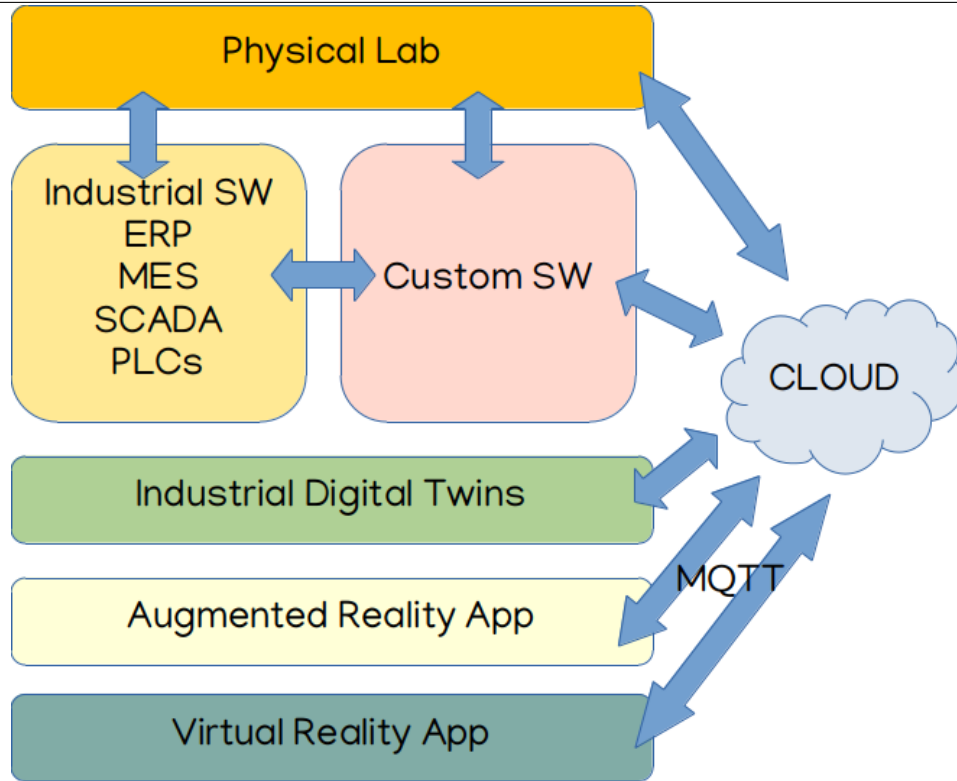


Fig. 6.4: Functional schema of the ICE lab. The AR and XR applications communicate both with the physical devices and the industrial digital twin, enabling augmented interaction, monitoring and control.

with Industrial Internet of Things (IIoT) brokers (RabbitMQ, Kafka, MQTT), storage and retrieval of time series (InfluxDB).

The Automation Manager works between the Manufacturing Execution System (MES) and the DIH to control the production plant and dynamically execute work orders, reacting to changes in the plant and in the production plan in real time. It receives from the Siemens OPCenter Execution Discrete MES the desired production data and the associated production recipes, and schedules the activities of the production cells by employing the services of the DIH.

The industrial Digital Twin for the whole production line has been created with the Siemens Tecnomatix Plant Simulation. Two DTs are actually available. The first is the Digital Shadow, communicating with the OPC-UA servers of the machines to monitor their status and visually replicate it in real-time, and allowing real-time control over the entire system. The second is the Autonomous Digital Twin, designed to simulate the plant as a whole. The information flow connecting the physical devices, proprietary and custom software layers, and the DT, is represented in Fig. 6.4. Selected data can flow from the physical factory to the digital layers and vice-versa to ensure monitoring and control.

The communication between applications and services relies on RPC and events subscriptions that generate a stream of messages managed by IIoT brokers. In particular, RabbitMQ handles the control infrastructure and Apache Kafka the data distribution. These technologies have a recognised industrial appeal for their scalability and high reliability. Alongside these systems, a lightweight messaging system based on MQTT allows communication from outside the laboratory, specifically designed to provide remote monitoring and control of the production line with mobile devices.

In the context of the ICE lab, we developed Virtual Reality and Augmented Reality applications that showcase the different opportunities that the technologies offer to train and assist workers.

These applications can be considered additional digital twins and directly communicate in both directions with the physical laboratory and the industrial digital twins.

They leverage the creation of an accurate graphical modeling of the laboratory elements, efficient network communication, and advanced interaction design solutions. The MR application is designed to help workers in the laboratory, while the VR application is designed for training but also for remote monitoring and control.

The AR and the VR applications currently run on the Meta Quest 2 and HoloLens 2 stand-alone HMDs, respectively. They can be ported easily on different platforms. The current AR application rely on the HoloLens 2 spatial mapping, but, not only similar high quality inside out tracking capabilities are available on other AR glasses (Magic Leap, Meta Quest 3, VarjoXR), but it could be also possible to use VIVE base stations mounted in the lab to provide accurate headset tracking. The interaction in AR and VR is based on gestures, and also in this case, most of the recent AR and VR headset comes with finger tracking capabilities, that can in any case added to the setup with a low cost Ultraleap device. The porting of gesture recognition algorithms based on hand skeleton data needs attention but we have successfully ported applications from Leap Motion data to the HoloLens even without the need to retrain the recognizer.

In the following sections, we describe the graphical model of the laboratory, the main characteristics of these applications, and the example demonstrations that have been performed with them and are available for the visitors.

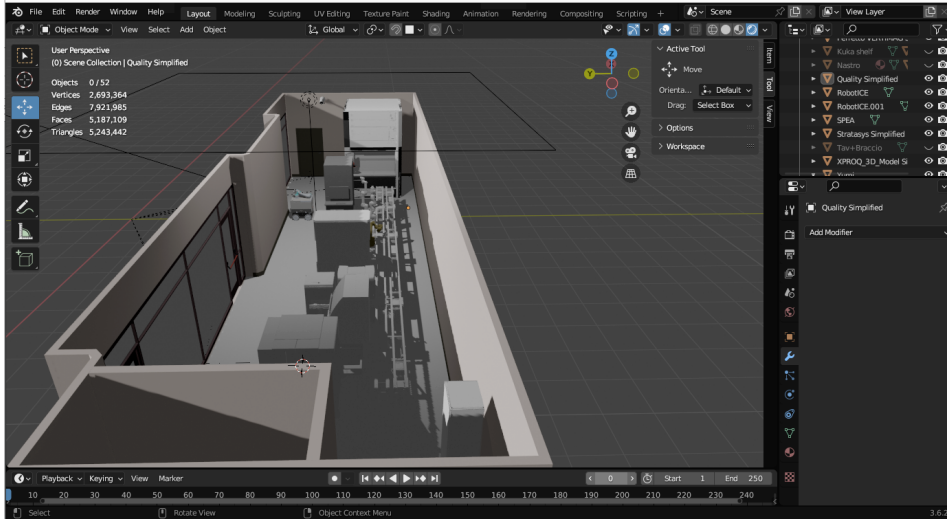


Fig. 6.5: The graphical twin of the ICE lab includes all the laboratory assets modeled as triangle meshes at different levels of detail to support rendering on low-end devices adapting the quality to the hardware resources. The full-resolution model is currently composed of approximately 5.1M triangles.

6.4 The laboratory graphical twin

A 3D digital replica of the laboratory was created to support the realization of the immersive VR tools and the creation of spatially referenced augmentations. The complete model has been designed and optimized with Blender [188], using the floor plan and on-site measurements to reproduce the room accurately (Fig. 6.5). We modeled most of the machines starting from the CAD files provided by the manufacturers, exported as surface meshes

CHAPTER 6. PRACTICAL APPLICATIONS OF GESTURE RECOGNITION IN INDUSTRIAL XR ENVIRONMENTS

using Autodesk Fusion 360 [189], and put into a hierarchical structure. The meshes had to be processed to make them manageable in interactive graphical applications running on low-end hardware, as the single parts often had several millions triangles. Due to the large diversity in models quality and complexity, the simplification had to be specialized for each model. The process was organized into two steps: i) removal of internal/non-relevant parts and possible re-meshing of the external surface; ii) simplification of the mesh with edge collapse.

With the creation of the room and the generation of the models, we obtained a basic version of the scene with all the geometric models, totaling 5.1M triangles. This amount is still very high for interactive applications on several platforms, especially for the stand-alone VR headset (Meta Quest 2) used as a target platform in our VR app (Section 6.5). For this reason, we created meshes for the different elements at four different resolutions (100%, 50%, 30% and 10% the original models' triangles) that can be used changed dynamically using the Level Of Detail (LOD) mechanism in the rendering step. The LOD mechanism adapts the resolution to the distance from the point of view. For the most complex piece, the minipallet conveyor line, which is a unique structure covering most of the room space, we had to split the model into three segments along its major length, and create a multi-resolution version of each to optimize the performances of the LOD management. To obtain a realistic rendering of the scene, we also had to assign reasonable materials to the modeled surfaces. The original CAD data mostly lacked Physically-Based materials that have been, therefore, assigned in the modeling phase and still need to be completed/optimized. We plan to develop specific image-based methods to further improve realism in future work.



Fig. 6.6: (a)

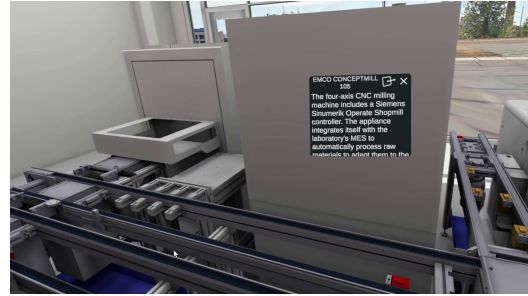


Fig. 6.7: (b)



Fig. 6.8: (c)



Fig. 6.9: (d)

Fig. 6.10: The VR application provides a realistic visualization of the simulated environment on the Head-Mounted display, with physically-based illumination effects and high-resolution details (a,b). The application allows to navigate inside the ICE lab using instant teleport by throwing a ray with a gesture and clicking when the ray intersects the floor in the desired position (c), and the interactive visualization of both offline description and real-time information on the single machines, as the active tray in the vertical warehouse (d).

6.5 Virtual Reality application

The app was developed using Unity [190], and is based on the optimized scene created with Blender. We added the ceiling of the lab with area-based lights simulating the real LED light sources. A spherical texture representing the external view of the lab has been added to provide a visually plausible rendering of the scene behind the windows and a realistic illumination. The materials of the models were fine-tuned along with a Universal Rendering Pipeline asset to find the best compromise between performance and image quality. Global illumination has been baked into light maps. The application uses the Unity LOD management system, set up with the 4 different resolutions created, and setting distance thresholds to enable lower LOD models based on a preliminary test with subjects, asking them at which distance the change in detail seemed negligible. These settings and configurations allow the app to run at 60FPS on average on the Oculus Quest 2, with the lowest performance registered at roughly 50FPS with the camera observing the scene facing the full length of the virtual laboratory. The degree of realism of the visualization is good, as shown in Fig. 6.10.

6.5.1 Interaction Design

The user can navigate the environment by walking in the real space or by changing the viewpoint with the Point and Teleport technique [191, 192]. In this case, they must perform a pinch gesture that makes the hand cast a virtual ray that can be pointed to the target destination on the floor (Fig. 6.8). The camera movement is performed upon releasing the hand pose. The users can exploit the same pinch/ray casting action to point at the different machines spread throughout the laboratory. In this case, they can activate the visualization of specific information on their status, displayed on a floating panel. We implemented two modalities for the information panels: Follow or Anchored. In the Follow mode, the panel follows the user with smooth transitions when the camera rotates or when the user moves through the environment.

6.5.2 VR Demo: virtual visit with real-time machines' status visualization

The current version of the application allows remote users to easily navigate in the laboratory, learning its layout and visualizing panels with the description of the different "virtual" machines. Furthermore, these panels can display live information retrieved from the corresponding "real" equipment. The live information update is gathered by an MQTT module that connects the application to the MQTT broker in the laboratory infrastructure that exposes data retrieval and command functions to authorized users. In Figure 6.9, the panel displays the currently active tray in the vertical automated warehouse.

6.6 Augmented Reality application

AR applications can be easily created using a see-through headset with spatial mapping capabilities like the Microsoft HoloLens 2. By using the mentioned AR headset, we built a MR system that augments the experience while being in the ICE lab, by showing real and up-to-date information about the state of the machines. Similarly to the VR app (Section 6.5), the software is developed using Unity Engine [190], importing 3D models of the machines created in Blender. Because these 3D models are superimposed over physical machines, there is no need to render them, but they can be used for visualization purposes in case of specific application requirements.

The spatial alignment of the virtual and real spaces exploits the Space Pins feature of World Locking Tools (WLT) for HoloLens 2. WLT provides a stable and reliable world-locked coordinate system. By scanning with the headset camera QR codes located in the physical world and corresponding to known virtual world coordinates, the system can recover the optimal transform matching the 3D model of the lab and the spatial map created by the headset. For high-quality alignment accuracy, it is recommended to scan two or more

CHAPTER 6. PRACTICAL APPLICATIONS OF GESTURE RECOGNITION IN INDUSTRIAL XR ENVIRONMENTS

QR codes, so we placed two QR codes in the physical lab, annotating the corresponding position in the virtual space. We estimated an alignment error of $\pm 5cm$ between the virtual model and the real machines, which we considered to be acceptable given the extent of the laboratory. The application must be initialized at startup by scanning these QR codes with the HoloLens 2 camera.

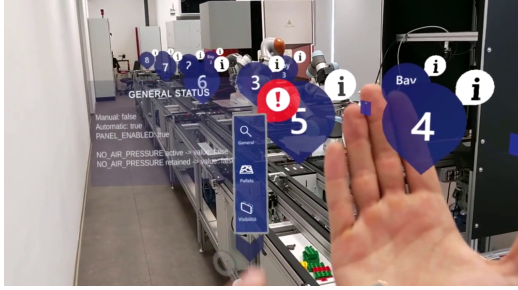


Fig. 6.11: (a)



Fig. 6.12: (b)



Fig. 6.13: (c)



Fig. 6.14: (d)

Fig. 6.15: Example headset views of the basic AR application of the ICE lab. The user can see augmentations (a) and/or semitransparent models of the selected (b) devices. In the examples in the bottom row, the operator checks the status of a segment of the conveyor (c). In case of alarm, a red arrow guides the operator to the critical point (d).

6.6.1 Interaction Design

We designed a set of augmentation options controlled by UI elements realized with the MRTK 2 toolkit. A hand menu allows the selection of the different visualization options. The user can select the augmentation panels enabled and hidden. The 3D models of the machines can be rendered with variable transparency, superimposed on the real ones, or not displayed (default option). The information panels associated with the machines are clickable with the local or remote (ray-casting) standard pinch-based interaction of MRTK to visualize related information, updated in real time.

Furthermore, our AR application features a custom gesture-based interaction system. This system detects and segments in real-time static and dynamic gestures, captured by the HoloLens 2 finger-tracking system and can map them to specific actions. The system exploits a client-server architecture and a neural network recognizer [10] to provide a low-latency detection and recognition of a dictionary of 16 heterogeneous gestures, including static (e.g., fixed hand poses) and dynamic ones, including hand articulation (pinch, grab), or whole hand movement (swipe left and right, circle). The gestures dataset used to train the network has been released for a recognition contest in 2022 [6]. The related dictionary includes gestures typically used for interaction and communication (see Fig. 6.16).

Our network could be, however, trained on different datasets/dictionaries with specific gestures and non-gestures examples, with classes optimized for the desired task.

By mapping gestures with system actions, it is possible to create easy-to-use control systems for specific tasks. This allows a touchless control (that is also optimal for hygiene) and we expect to be faster and less error-prone than clicking in mid-air. We plan to perform usability tests to verify this, comparing our gesture-based solutions to traditional mid-air clickable menus provided by the MRTK2 framework in the near future with a user study on a controlled task, measuring efficiency, errors and fatigue. The preliminary feedback of the ICE lab visitors testing the gesture-based solutions is really promising, and the fact that menu item selection with simple gestures can be faster and not less accurate than mid-air clicking has been already shown in previous work [193].



Fig. 6.16: The 16 static and dynamic gestures modeled in [6] that are used for touchless control.

6.6.2 Demo tasks for the AR app

Using the framework developed, we demonstrated different tasks in public exhibitions showing the potential impact of the use of this technology in industrial environments and showing the effectiveness of the design and implementation choices adopted.

Transport line augmentation

The first demo of the augmented reality application allows the real-time visualization of the transport line data, as shown in Fig. 6.15. The user can check the status of the pallets and of the bays. In particular, they can gather general information about the conveyor line, and inspect in more detail the position, content and destination of each pallet. When the conveyor is operational, blinking arrows on each bay signal the direction of a pallet entering or exiting. Moreover, the application warns the user about any ongoing machine alarms by placing a marker on the conveyor line and enabling the operator to inspect them in a dedicated panel. At any time, the user can enable and disable the virtual machine models, information panels, alarms and arrows.

Active monitoring and control of the production workflow with AR and gestural interface

To demonstrate the integration of the XR interfaces with the production line management and the use of natural interaction with gestures, we implemented another demo for the visitors of the laboratory where users wearing HoloLens 2 glasses can control realistic production tasks with simple gestures having visual feedback on the effects of the actions on the system.

The demo simulates the action of an operator that can select pending tasks on a list (Fig. 6.17), use a gesture (in this case "knob") to start the transport line (Fig. 6.18), select one of the available actions with a gesture (Fig. 6.19) and monitor the ongoing procedures (Fig. 6.20).

Remote monitoring/control of single machines

We implemented a specialized XR application based on our framework in occasion of a big national technology exhibition on Industry 4.0, where the ICE lab staff had the opportunity to display its solutions to the exposition visitors. For obvious reasons, the production

CHAPTER 6. PRACTICAL APPLICATIONS OF GESTURE RECOGNITION IN INDUSTRIAL XR ENVIRONMENTS

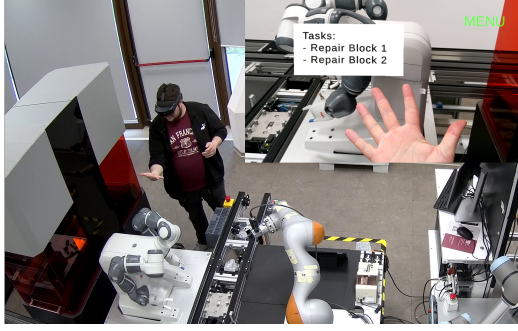


Fig. 6.17: (a)

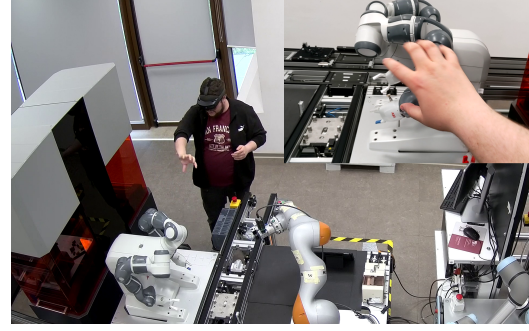


Fig. 6.18: (b)

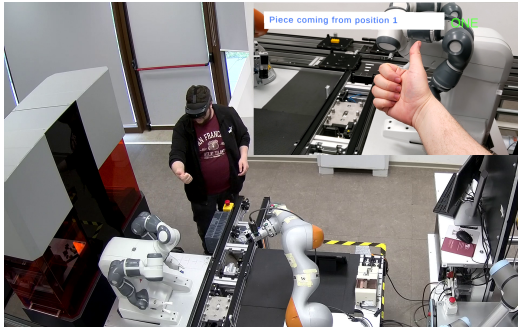


Fig. 6.19: (c)

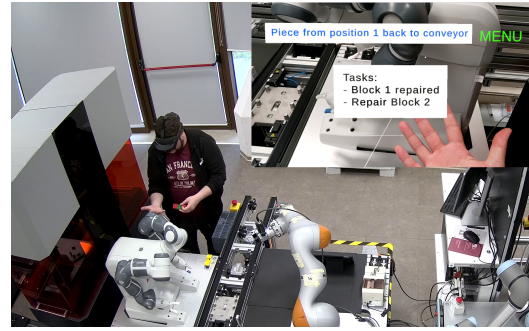


Fig. 6.20: (d)

Fig. 6.21: Demo of the gesture-based production recipes' management. The boxes in the figure show the user view. (a) The worker opens the menu of the active tasks with a menu gesture. (b) They start the transport line operation with a "knob" gesture. (c) With a specific gesture, they call pallet '1' in the local bay for repair. (d) After completing the task, they can see the updated list of available actions.

line could not be dismantled and re-assembled at the exhibition, and it was replaced with cardboard cutouts of each equipment, real-time video feeds from inside the laboratory, and remote monitoring and control of the Automation Manager described in Section 6.3. This artifice was planned in conjunction with our application to allow our guests to virtually inspect the machines and retrieve their real-time status updates during the live demonstrations.

Specifically, the cardboards were identified by unique image targets (Fig. 6.22) and recognized by the Hololens 2 app. By performing distinct gestures in this situation, the users could visualize a 3D model of the machine with associated information (Fig. 6.23) and check its real-time status (Fig. 6.25). Similarly to the VR Demo described in Subsection 6.5.2, the live connection was provided by an MQTT module that exchanged messages with the laboratory's broker to command status updates. In the future, we plan to extend these interactions in order to enable a remote operator to be fully engaged with the production activities like an in-place worker, or the same worker to enjoy an enhanced experience.

6.7 Discussion

Our results demonstrate the readiness of the current technologies for providing support for workers in industrial contexts using eXtended Reality. The integration of industrial software, digital twins, and XR applications allows the creation of specific tools to enhance productivity and safety in the factories. The use cases of our demos can be extended with many other tasks, and the controlled environment of the ICE lab would facilitate the design of specific user tests to evaluate the impact of the technologies and compare different interaction design solutions.



Fig. 6.22: (a)

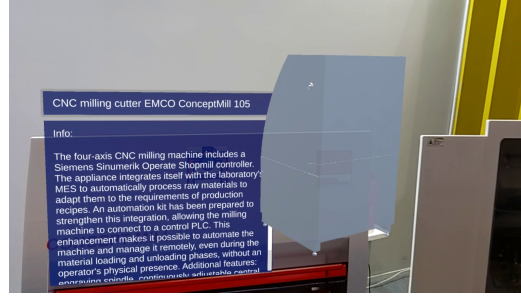


Fig. 6.23: (b)



Fig. 6.24: (c)

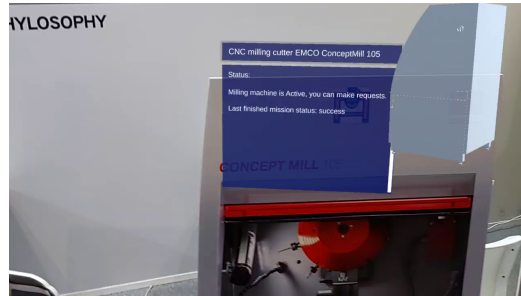


Fig. 6.25: (d)

Fig. 6.26: Remote inspection with extended reality. (a) The headset scans an image target. The pane shows the user's view. (b) Subsequent user view where the device's description is displayed together with the 3D representation of the same machine. (c) The user performs a command with a gesture (the pane shows the user's view). (d) Subsequent user view where the status of the machine, updated in real-time, is displayed.

Examples of the tasks we plan to investigate with specific experiments are the use of VR to learn the spatial layout of the environment; the effectiveness of different solutions for immersive digital twin analytics visualization; and specific solutions to synchronize collaborative work of local and remote users based on shared spatial annotations.

Other tasks to be designed will require further improvements of the system, in particular, related to the accurate simulation of the virtual robotic arms. We plan to introduce their functional models in our graphical twin soon.

Even if the current XR technology is ready for a practical application, there are still some limitations that need to be carefully considered for the interaction design. The first issue is related to potential discomfort due to the use of the headset. Its weight can lead to different issues and limit its "comfortable wear time" [194, 195]. The battery life can be another relevant issue for practical use. In the case of the HoloLens 2, it is limited to 2-3 hours. The limited field of view of the device can be a relevant problem for the effective visualization of augmentations. The next generation of see-through displays (Apple Vision Pro, Varjo XR3, Meta Quest 3) might remove this issue being based on video see-through technology, but this kind of display has the relevant drawback of excluding the user from the real environment in case of system failure.

However, while our work demonstrates the feasibility of integrating XR in industrial workflows, we acknowledge that a comprehensive evaluation of user experience with real users is still needed. This remains an essential step for assessing usability, user acceptance, and the practical impact of these technologies in real-world scenarios. Future work will focus on designing and conducting structured user studies to address this gap, leveraging the ICE lab's environment to gather empirical insights.

Conclusion

This thesis has focused on a specific area of Human-Computer Interaction (HCI), namely gesture-based interaction, addressing key challenges and exploring its potential in extended reality environments. In recent years, advancements in sensing technologies, computational power, and extended reality platforms have significantly increased research interest in gesture recognition, making it a vital component of modern interaction paradigms.

Despite this growing interest, several critical issues in gesture recognition still need to be solved. These include accurate and efficient detection of gestures in continuous streams, integration of human-centric factors, and development of robust systems capable of real-time adaptation and generalization to varied environments. These challenges underline the complexity of implementing gesture-based systems effectively in practical applications.

This work has provided a comprehensive review and evaluation of state-of-the-art approaches, identifying strengths and limitations in the existing literature. Based on these insights, novel methodologies tailored to online gesture recognition were proposed, addressing critical gaps in the field.

Significant contributions include the introduction of benchmarks like SHREC'22, which enable realistic and standardized evaluations of gesture recognition methods in continuous, interactive scenarios.

Moreover, a critical evaluation of existing methodologies was conducted, leading to the design of improved gesture recognition approaches that integrate multi-task learning and feature-enhanced classifiers to achieve better performance metrics, such as detection rate, false positive rate, and Jaccard index.

Novel data augmentation strategies were also developed to improve the robustness and training of gesture recognition systems, addressing variability in gesture execution and environmental conditions.

The potential of gesture-based interactions was demonstrated through practical implementations of our methods in industrial extended reality environments, highlighting their ability to enhance user interaction and workflow efficiency.

Through these contributions, this thesis has advanced both theoretical understanding and practical applications of gesture recognition technologies. By bridging theoretical understanding with practical applications, this thesis has advanced the field of gesture recognition, demonstrating its transformative potential in extended reality while addressing key limitations.

The experience gained in developing methods and applications and the analysis of all the available benchmarks and the performance of the different methods evaluated on them suggests some possible research directions for future work aimed at focusing on relevant problems and improving the quality of the detection results:

- Novel gesture recognition benchmarks are needed, but their creation should follow specific guidelines. Large numbers of subjects should be involved, covering different user categories, and related metadata should be collected. Multimodal capture could help to analyze how much the results depend on input data acquisition and pre-processing quality. The robustness of the methods against user diversity should be tested. Basic

research on algorithms should be brought closer to the problems of application domains by creating specific benchmarks for them.

- The evaluation should be standardized or, at least, specified for each application task in a way accepted by the scientific community. An effort to reduce the fragmentation of the community working on this topic would be desirable. We recommend the use of four primary metrics: TPR (events recognized by a method), FPR (events predicted poorly), Classification time (to understand the speed of method execution), and, finally, Temporal delay (to understand how fast an event is recognized). Once evaluation metrics are unified, one could compare the same method on multiple datasets by understanding how to improve it and the real difficulties of those benchmarks.
- Optimization of gesture dictionaries could be another goal of the evaluation. A possible idea could be to develop evaluation benchmarks/metrics for gesture dictionary usability and also for interactive gestural interface design.
- For skeleton-based classifiers, it would be important to verify the quality of the hand tracking, separating the errors due to bad tracking from the gesture classification errors.
- A careful comparison of the different continuous recognition approaches (two modules, sliding windows trained with fixed duration subparts, sliding windows trained with segmented/resampled gestures, recurrent networks) should be performed on different kinds of datasets to understand better their advantages/disadvantages
- All the methods recently proposed for the segmented task should be evaluated on properly designed continuous gesture recognition benchmarks, considering the related metrics. New research should focus more on solving the issues related to training and testing for online detection rather than on trying small changes in network architectures to solve the segmented task on old benchmarks.
- In the case of heterogeneous gestures, specialized classifiers for different gesture types included in the dataset could be considered.
- New augmentation methods should be proposed, especially for the non-gesture movements, considering the biomechanical constraints.
- Further work is required to address scalability challenges, enhancing system adaptability to diverse users and contexts.

These directions aim to ensure a broader adoption of gesture recognition technologies across various domains.

References

1. S. Ruffieux *et al.*, “Chairgest: A challenge for multimodal mid-air gesture recognition for close hci,” in *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ser. ICMI '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 483–488. [Online]. Available: <https://doi.org/10.1145/2522848.2532590>
2. S. Escalera, X. Baró, J. González, M. A. Bautista, M. Madadi, M. Reyes, V. Ponce-López, H. J. Escalante, J. Shotton, and I. Guyon, “Chalearn looking at people challenge 2014: Dataset and results,” in *Computer Vision - ECCV 2014 Workshops*, L. Agapito, M. M. Bronstein, and C. Rother, Eds. Cham: Springer International Publishing, 2015, pp. 459–473.
3. F. Caputo, S. Burato, G. Pavan, T. Voillemin, H. Wannous, J.-P. Vandeboorde, M. Maghoumi, E. Taranta, A. Razmjoo, J. LaViola Jr *et al.*, “Shrec 2019 track: online gesture recognition,” in *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2019, pp. 93–102.
4. M. Vandersteegen *et al.*, “Low-latency hand gesture recognition with a low resolution thermal imager,” *CoRR*, vol. abs/2004.11623, 2020. [Online]. Available: <https://arxiv.org/abs/2004.11623>
5. D. Liu, L. Zhang, and Y. Wu, “Ld-congr: A large rgb-d video dataset for long-distance continuous gesture recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3304–3312.
6. M. Emporio, A. Caputo, A. Giachetti, M. Cristani, G. Borghi, A. D’Eusano, M.-Q. Le, H.-D. Nguyen, M.-T. Tran, F. Ambellan, M. Hanik, E. Nava-Yazdani, and C. von Tycowicz, “Shrec 2022 track on online detection of heterogeneous gestures,” *Computers & Graphics*, vol. 107, pp. 241–251, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0097849322001388>
7. C. Xu, X. Wu, M. Wang, F. Qiu, Y. Liu, and J. Ren, “Improving dynamic gesture recognition in untrimmed videos by an online lightweight framework and a new gesture dataset zjugesture,” *Neurocomputing*, vol. 523, pp. 58–68, 2023.
8. A. Caputo, A. Giachetti, S. Soso, D. Pintani, A. D’Eusano, S. Pini, G. Borghi, A. Simoni, R. Vezzani, R. Cucchiara *et al.*, “Shrec 2021: Skeleton-based hand gesture recognition in the wild,” *Computers & Graphics*, vol. 99, pp. 201–211, 2021.
9. O. Köpüklü *et al.*, “Online dynamic hand gesture recognition including efficiency analysis,” *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 2, no. 2, pp. 85–97, April 2020.
10. F. Cunico, F. Girella, A. Avogaro, M. Emporio, A. Giachetti, and M. Cristani, “Oo-dmvm: A deep multi-view multi-task classification framework for real-time 3d hand gesture classification and segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2023, pp. 2745–2754. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2023W/CV4MR/html/Cunico_OO-dMVM_A_Deep_Multi-View_Multi-Task_Classification_Framework_for_Real-Time_3D_CVPRW_2023_paper.html
11. F. Cunico, S. Aldegheri, A. Avogaro, M. Boldo, N. Bombieri, L. Capogrosso, A. Caputo, D. Carra, S. Centomo, D. S. Cheng, E. Cinquetti, M. Cristani, M. D. Marchi, F. Demrozi, M. Emporio, F. Fummi, L. Geretti, S. Germiniani, A. Giachetti, F. Girella, E. Martini, G. Menegaz, N. Muijs, F. Paci, M. Panato, G. Pravadelli, E. Quintarelli, I. Siviero, S. F. Storti, C. Tadiello, C. Turetta, T. Villa, N. Zannone, and D. Quaglia, “Enhancing safety and privacy in industry 4.0: The ice laboratory case study,” *IEEE Access*, pp. 1–1, 2024.

12. D. Pintani, M. Emporio, A. Caputo, D. S. Cheng, L. Genghini, N. Tomasoni, and A. Giachetti, "Icelab demo: an industrial digital-twin and simulator in vr," in *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology*, ser. VRST '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: <https://doi.org/10.1145/3641825.3689946>
13. B. Veldhuijzen, R. C. Veltkamp, O. Ikne, B. Allaert, H. Wannous, M. Emporio, A. Giachetti, J. J. LaViola, R. He, H. Benhabiles, A. Cabani, A. Fleury, K. Hammoudi, K. Gavalas, C. Vlachos, A. Papanikolaou, I. Romanelis, V. Fotis, G. Arvanitis, K. Moustakas, M. Hanik, E. Nava-Yazdani, and C. von Tycowicz, "Shrec 2024: Recognition of dynamic hand motions molding clay," *Computers & Graphics*, vol. 123, p. 104012, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S009784932400147X>
14. M. Emporio, A. Caputo, D. Pintani, D. S. Cheng, T. De Marchi, G. Forte, F. Fummi, and A. Giachetti, "Integration of extended reality with a cyber-physical factory environment and its digital twins," *Proc. ACM Hum.-Comput. Interact.*, vol. 8, no. EICS, jun 2024. [Online]. Available: <https://doi.org/10.1145/3660246>
15. M. Emporio, A. Caputo, D. Pintani, F. Cunico, F. Girella, A. Avogaro, M. Cristani, and A. Giachetti, "Demo: gesture based interaction with the hololens 2," in *Proceedings of the 15th Biannual Conference of the Italian SIGCHI Chapter*, ser. CHIItaly '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: <https://doi.org/10.1145/3605390.3610827>
16. M. Emporio, A. Caputo, and A. Giachetti, "STRONGER: Simple TRajjectory-based ONLINE GEsture Recognizer," in *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*, P. Frosini, D. Giorgi, S. Melzi, and E. Rodolà, Eds. The Eurographics Association, 2021.
17. M. Emporio, A. Ghasemaghahi, J. J. LaViola, and A. Giachetti, "Continuous hand gesture recognition: Benchmarks and methods," *Computer Vision and Image Understanding*, 2024.
18. M. Pantic, A. Nijholt, A. Pentland, and T. S. Huanag, "Human-centred intelligent human? computer interaction (hci²): how far are we from attaining it?" *International Journal of Autonomous and Adaptive Communications Systems*, vol. 1, no. 2, pp. 168–187, 2008.
19. J. J. LaViola Jr, E. Kruijff, R. P. McMahan, D. Bowman, and I. P. Poupyrev, *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2017.
20. T. Wang, X. Qian, F. He, X. Hu, Y. Cao, and K. Ramani, "Gesturar: An authoring system for creating freehand interactive augmented reality applications," in *The 34th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 552–567. [Online]. Available: <https://doi.org/10.1145/3472749.3474769>
21. T. Papadopoulos *et al.*, "Interactions in augmented and mixed reality: An overview," *Applied Sciences*, vol. 11, no. 18, p. 8752, 2021.
22. J. Berg and S. Lu, "Review of interfaces for industrial human-robot interaction," *Current Robotics Reports*, vol. 1, pp. 27–34, 2020.
23. G. Prabhakar and P. Biswas, "A brief survey on interactive automotive ui," *Transportation Engineering*, vol. 6, p. 100089, 2021.
24. R.-D. Vatavu, "User-defined gestures for free-hand tv control," in *Proceedings of the 10th European conference on Interactive tv and video*, 2012, pp. 45–48.
25. G. Drossis *et al.*, "Magic: developing a multimedia gallery supporting mid-air gesture-based interaction and control," in *HCI International 2013-Posters' Extended Abstracts: International Conference, HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings, Part I 15*. Springer, 2013, pp. 303–307.
26. S. o. Huang, "To touch or not to touch? comparing touch, mid-air gesture, mid-air haptics for public display in post covid-19 society," in *SIGGRAPH Asia 2020 Posters*. SIGGRAPH Asia 2020, 2020, pp. 1–2.
27. D. Ungureanu, F. Bogo, S. Galliani, P. Sama, X. Duan, C. Meekhof, J. Stühmer, T. J. Cashman, B. Tekin, J. L. Schönberger *et al.*, "Hololens 2 research mode as a tool for computer vision research," *arXiv preprint arXiv:2008.11239*, 2020.
28. D. Schneider *et al.*, "Accuracy evaluation of touch tasks in commodity virtual and augmented reality head-mounted displays," in *Proceedings of the 2021 ACM Symposium on Spatial User Interaction*, 2021, pp. 1–11.
29. F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.
30. T. Nguyen, N.-C. Nguyen, D.-K. Ngo, V.-L. Phan, M.-H. Pham, D.-A. Nguyen, M.-H. Doan, and T.-L. Le, "A continuous real-time hand gesture recognition method based on skeleton,"

- 2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS), pp. 273–278, 2022. [Online]. Available: <https://api.semanticscholar.org/CorpusID:255266952>
31. G. Devineau, F. Moutarde, W. Xi, and J. Yang, “Deep learning for hand gesture recognition on skeletal data,” *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pp. 106–113, 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:46977161>
 32. Q. De Smedt, H. Wannous, and J.-P. Vandeboorde, “Heterogeneous hand gesture recognition using 3d dynamic skeletal data,” *Computer Vision and Image Understanding*, vol. 181, pp. 60–72, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1077314219300153>
 33. X. Chai, Z. Liu, F. Yin, Z. Liu, and X. Chen, “Two streams recurrent neural networks for large-scale continuous gesture recognition,” in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 31–36.
 34. Y. Li, G. Wei, C. Desrosiers, and Y. Zhou, “Decoupled and boosted learning for skeleton-based dynamic hand gesture recognition,” *Pattern Recognition*, vol. 153, p. 110536, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320324002875>
 35. D. Özer and T. Göksun, “Gesture use and processing: A review on individual differences in cognitive resources,” *Frontiers in Psychology*, vol. 11, p. 573555, 2020.
 36. Y. Zhang *et al.*, “Egogesture: A new dataset and benchmark for egocentric hand gesture recognition,” *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1038–1050, May 2018.
 37. P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, “Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 4207–4215.
 38. H. Gammulle *et al.*, “Tmmf: Temporal multi-modal fusion for single-stage continuous gesture recognition,” *IEEE Transactions on Image Processing*, vol. 30, pp. 7689–7701, 2021.
 39. D. Tran *et al.*, “Learning spatiotemporal features with 3d convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.
 40. F. Caputo *et al.*, “Comparing 3d trajectories for simple mid-air gesture recognition,” *Computers & Graphics*, vol. 73, pp. 17–25, 2018.
 41. E. Ohn-Bar and M. M. Trivedi, “Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2368–2377, Dec 2014.
 42. Q. De Smedt *et al.*, “Skeleton-based dynamic hand gesture recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.
 43. —, “Shrec’17 track: 3d hand gesture recognition using a depth and skeletal dataset,” in *3DOR-10th Eurographics Workshop on 3D Object Retrieval*, 2017, pp. 1–6.
 44. J. Materzynska *et al.*, “The jester dataset: A large-scale video dataset of human gestures,” in *Proceedings of the IEEE/CVF international conference on computer vision workshops*, 2019, pp. 0–0.
 45. S. Mitra and T. Acharya, “Gesture recognition: A survey,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, 2007.
 46. J. J. LaViola, “3d gestural interaction: The state of the field,” *International Scholarly Research Notices*, vol. 2013, 2013.
 47. S. S. Rautaray and A. Agrawal, “Vision based hand gesture recognition for human computer interaction: a survey,” *Artificial intelligence review*, vol. 43, pp. 1–54, 2015.
 48. C. Anthes, R. J. García-Hernández, M. Wiedemann, and D. Kranzlmüller, “State of the art of virtual reality technology,” in *2016 IEEE Aerospace Conference*, 2016, pp. 1–19.
 49. S. S. Kakkoth and S. Gharge, “Survey on real time hand gesture recognition,” in *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, Sep. 2017, pp. 948–954.
 50. S. Escalera, V. Athitsos, and I. Guyon, “Challenges in multi-modal gesture recognition,” *Gesture recognition*, pp. 1–60, 2017.
 51. F. Gu, M.-H. Chung, M. Chignell, S. Valaee, B. Zhou, and X. Liu, “A survey on deep learning for human activity recognition,” *ACM Comput. Surv.*, vol. 54, no. 8, oct 2021. [Online]. Available: <https://doi.org/10.1145/3472290>
 52. A. S. Al-Shamayleh, R. Ahmad, M. A. Abushariah, K. A. Alam, and N. Jomhari, “A systematic literature review on vision based gesture recognition techniques,” *Multimedia Tools and Applications*, vol. 77, pp. 28 121–28 184, 2018.

53. M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: A review of techniques," *Journal of Imaging*, vol. 6, no. 8, 2020. [Online]. Available: <https://www.mdpi.com/2313-433X/6/8/73>
54. Y. SHI, Y. LI, X. FU, M. Kaibin, and M. Qiguang, "Review of dynamic gesture recognition," *Virtual Reality & Intelligent Hardware*, vol. 3, no. 3, pp. 183–206, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2096579621000279>
55. D. Sarma and M. K. Bhuyan, "Methods, databases and recent advancement of vision-based hand gesture recognition for hci systems: A review," *SN Computer Science*, vol. 2, no. 6, p. 436, 2021.
56. L. Zulpukharkyzy Zholshiyeva, T. Kokenovna Zhukabayeva, S. Turaev, M. Aimambetovna Berdiyeva, and D. Tokhtasynovna Jambulova, "Hand gesture recognition methods and applications: A literature survey," in *The 7th International Conference on Engineering & MIS 2021*, ser. ICEMIS'21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: <https://doi.org/10.1145/3492547.3492578>
57. R. Jain, R. K. Karsh, and A. A. Barbhuiya, "Literature review of vision-based dynamic gesture recognition using deep learning techniques," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 22, p. e7159, 2022.
58. R.-D. Vatavu, *Gesture-Based Interaction*. Cham: Springer International Publishing, 2020, pp. 1–47. [Online]. Available: https://doi.org/10.1007/978-3-319-27648-9_20-1
59. J. Guna, G. Jakus, M. Pogačnik, S. Tomažič, and J. Sodnik, "An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking," *Sensors*, vol. 14, no. 2, pp. 3702–3720, 2014.
60. "Ultraleap web site," <https://www.ultraleap.com/>, accessed: 2023-09-10.
61. J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.
62. O. Wasenmüller and D. Stricker, "Comparison of kinect v1 and v2 depth images in terms of accuracy and precision," in *Computer Vision—ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20–24, 2016, Revised Selected Papers, Part II 13*. Springer, 2017, pp. 34–45.
63. S.-h. Lee, D.-W. Lee, K. Jun, W. Lee, and M. S. Kim, "Markerless 3d skeleton tracking algorithm by merging multiple inaccurate skeleton data from multiple rgb-d sensors," *Sensors*, vol. 22, no. 9, p. 3155, 2022.
64. G. Benitez-Garcia *et al.*, "Ipn hand: A video dataset and benchmark for real-time continuous hand gesture recognition," in *2020 25th international conference on pattern recognition (ICPR)*. IEEE, 2021, pp. 4340–4347.
65. M. Caeiro-Rodríguez, I. Otero-González, F. A. Mikic-Fonte, and M. Llamas-Nistal, "A systematic review of commercial smart gloves: Current status and applications," *Sensors*, vol. 21, no. 8, p. 2667, 2021.
66. B. Ji, X. Wang, Z. Liang, H. Zhang, Q. Xia, L. Xie, H. Yan, F. Sun, H. Feng, K. Tao *et al.*, "Flexible strain sensor-based data glove for gesture interaction in the metaverse: A review," *International Journal of Human–Computer Interaction*, pp. 1–20, 2023.
67. M. E. Benalcázar, A. G. Jaramillo, A. Zea, A. Páez, V. H. Andaluz *et al.*, "Hand gesture recognition using machine learning and the myo armband," in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 1040–1044.
68. M. Atzori, A. Gijsberts, I. Kuzborskij, S. Elsig, A.-G. M. Hager, O. Deriaz, C. Castellini, H. Müller, and B. Caputo, "Characterization of a benchmark database for myoelectric movement classification," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 23, no. 1, pp. 73–83, 2014.
69. C. Amma, T. Krings, J. Böer, and T. Schultz, "Advancing muscle-computer interfaces with high-density electromyography," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 929–938.
70. M. Karam and m. Schraefel, "A taxonomy of gestures in human computer interactions," *Electronics and Computer Science*, 01 2005.
71. Y. LI, J. HUANG, F. TIAN, H.-A. WANG, and G.-Z. DAI, "Gesture interaction in virtual reality," *Virtual Reality & Intelligent Hardware*, vol. 1, no. 1, pp. 84–112, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2096579619300075>
72. K.-S. Lee and M.-C. Jung, "Ergonomic evaluation of biomechanical hand function," *Safety and health at work*, vol. 6, no. 1, pp. 9–17, 2015.
73. M. Maghoumi, E. M. Taranta, and J. LaViola, "Deepnag: Deep non-adversarial gesture generation," in *26th International Conference on Intelligent User Interfaces*, 2021, pp. 213–223.

74. M. E. Cabrera and J. P. Wachs, "Biomechanical-based approach to data augmentation for one-shot gesture recognition," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018, pp. 38–44.
75. J. A. Ward *et al.*, "Performance metrics for activity recognition," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 1, jan 2011. [Online]. Available: <https://doi.org/10.1145/1889681.1889687>
76. M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan *et al.*, "The prisma 2020 statement: an updated guideline for reporting systematic reviews," *International journal of surgery*, vol. 88, p. 105906, 2021.
77. J. Wan, S. Z. Li, Y. Zhao, S. Zhou, I. Guyon, and S. Escalera, "Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2016, pp. 761–769.
78. I. Guyon, V. Athitsos, P. Jangyodsuk, H. J. Escalante, and B. Hamner, "Results and analysis of the chalearn gesture challenge 2012," in *Revised Selected and Invited Papers of the International Workshop on Advances in Depth Image Analysis and Applications - Volume 7854*. Berlin, Heidelberg: Springer-Verlag, 2012, p. 186–204. [Online]. Available: https://doi.org/10.1007/978-3-642-40303-3_19
79. H. Wannous and J.-P. Vandeborre, "Continuous hand gesture recognition using deep coarse and fine hand features," in *The 33rd British Machine Vision Conference-BMVC 2022*, 2022.
80. K. Kahol, P. Tripathi, S. Panchanathan, and T. Rikakis, "Gesture segmentation in complex motion sequences," in *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, vol. 2. IEEE, 2003, pp. II–105.
81. J. Li, P. Lei, and S. Todorovic, "Weakly supervised energy-based learning for action segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6243–6251.
82. T. G. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops SSPR 2002 and SPR 2002 Windsor, Ontario, Canada, August 6–9, 2002 Proceedings*. Springer, 2002, pp. 15–30.
83. 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 asia*. ACM, 2019, pp. 1–6.
84. Y. Li, Z. He, X. Ye, Z. He, and K. Han, "Spatial temporal graph convolutional networks for skeleton-based dynamic hand gesture recognition," *EURASIP Journal on Image and Video Processing*, vol. 2019, no. 1, pp. 1–7, 2019.
85. L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Decoupled spatial-temporal attention network for skeleton-based action-gesture recognition," in *Proceedings of the Asian Conference on Computer Vision*, 2020.
86. E. Tsironi, P. Barros, C. Weber, and S. Wermter, "An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition," *Neurocomputing*, vol. 268, pp. 76–86, 2017.
87. X. Chai, Z. Liu, F. Yin, Z. Liu, and X. Chen, "Two streams recurrent neural networks for large-scale continuous gesture recognition," in *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016, pp. 31–36.
88. M. Maghoumi and J. J. LaViola, "Deepgru: Deep gesture recognition utility," in *Advances in Visual Computing: 14th International Symposium on Visual Computing, ISVC 2019, Lake Tahoe, NV, USA, October 7–9, 2019, Proceedings, Part I 14*. Springer, 2019, pp. 16–31.
89. E. M. Taranta II, C. R. Pittman, M. Maghoumi, M. Maslych, Y. M. Moolenaar, and J. J. Laviola Jr, "Machete: Easy, efficient, and precise continuous custom gesture segmentation," *ACM Trans. Comput.-Hum. Interact.*, vol. 28, no. 1, Jan. 2021. [Online]. Available: <https://doi.org/10.1145/3428068>
90. M. Maghoumi and J. J. L. Jr., "Deepgru: Deep gesture recognition utility," *CoRR*, vol. abs/1810.12514, 2018. [Online]. Available: <http://arxiv.org/abs/1810.12514>
91. Y. Yin and R. Davis, "Gesture spotting and recognition using salience detection and concatenated hidden markov models," in *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ser. ICMI '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 489–494. [Online]. Available: <https://doi.org/10.1145/2522848.2532588>
92. H. Wang, "Two stage continuous gesture recognition based on deep learning," *Electronics*, vol. 10, no. 5, 2021. [Online]. Available: <https://www.mdpi.com/2079-9292/10/5/534>
93. L. Pigou, A. van den Oord, S. Dieleman, M. V. Herreweghe, and J. Dambre, "Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video," *CoRR*, vol. abs/1506.01911, 2015. [Online]. Available: <http://arxiv.org/abs/1506.01911>

94. R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1597–1604.
95. G. Zhu, L. Zhang, P. Shen, and J. Song, "Multimodal gesture recognition using 3-d convolution and convolutional lstm," *Ieee Access*, vol. 5, pp. 4517–4524, 2017.
96. G. Zhu, L. Zhang, P. Shen, J. Song, S. A. A. Shah, and M. Bennamoun, "Continuous gesture segmentation and recognition using 3dcnn and convolutional lstm," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 1011–1021, April 2019.
97. Q. Guo, S. Zhang, and H. Li, "Continuous sign language recognition based on spatial-temporal graph attention network," *Computer Modeling in Engineering & Sciences*, vol. 134, pp. 1653–1670, 01 2023.
98. D. Tran, J. Ray, Z. Shou, S. Chang, and M. Paluri, "Convnet architecture search for spatiotemporal feature learning," *CoRR*, vol. abs/1708.05038, 2017. [Online]. Available: <http://arxiv.org/abs/1708.05038>
99. P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Multi-sensor system for driver's hand-gesture recognition," in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, 05 2015.
100. D. Tang, H. Jin Chang, A. Tejani, and T.-K. Kim, "Latent regression forest: Structured estimation of 3d articulated hand posture," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3786–3793.
101. S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5987–5995.
102. D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3D: generic features for video analysis," *CoRR*, vol. abs/1412.0767, 2014. [Online]. Available: <http://arxiv.org/abs/1412.0767>
103. M. Peral, A. Sanfeliu, and A. Garrell, "Efficient hand gesture recognition for human-robot interaction," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 272–10 279, 2022.
104. G. Benitez-Garcia, L. Prudente-Tixteco, L. C. Castro-Madrid, R. Toscano-Medina, J. Olivares-Mercado, G. Sanchez-Perez, and L. J. G. Villalba, "Improving real-time hand gesture recognition with semantic segmentation," *Sensors*, vol. 21, no. 2, p. 356, 2021.
105. P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin, "Hardnet: A low memory traffic network," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3552–3561.
106. L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*. Springer, 2016, pp. 20–36.
107. J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7083–7093.
108. G. Rocha and H. L. Cardoso, "Exploring spanish corpora for portuguese coreference resolution," *ACE*, vol. 2000, p. 22, 2004.
109. F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: <http://arxiv.org/abs/1602.07360>
110. H. R. V. Joze, A. Shaban, M. L. Iuzzolino, and K. Koishida, "MMTM: multimodal transfer module for CNN fusion," *CoRR*, vol. abs/1911.08670, 2019. [Online]. Available: <http://arxiv.org/abs/1911.08670>
111. Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5534–5542.
112. P. Wu, S. Chen, and D. N. Metaxas, "Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 385–11 395.
113. A. Chang, S. Mota, and H. Lieberman, "GestureNet: a common sense approach to physical activity similarity," *Electronic Visualisation and the Arts (EVA 2014)*, pp. 89–94, 2014.
114. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
115. P. Narayana, J. R. Beveridge, and B. A. Draper, "Continuous gesture recognition through selective temporal fusion," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

116. M. K. Bhuyan, D. Ajay Kumar, K. F. MacDorman, and Y. Iwahori, "A novel set of features for continuous hand gesture recognition," *Journal on Multimodal User Interfaces*, vol. 8, pp. 333–343, 2014.
117. B.-A. Awaluddin, C.-T. Chao, and J.-S. Chiou, "A hybrid image augmentation technique for user- and environment-independent hand gesture recognition based on deep learning," *Mathematics*, vol. 12, no. 9, 2024. [Online]. Available: <https://www.mdpi.com/2227-7390/12/9/1393>
118. B. Limonchik and G. Amdur, "3d model-based data augmentation for hand gesture recognition," located at <http://cs231n.stanford.edu/reports/2017/pdfs/218.pdf>, pp. 1–9, 2017.
119. M. Maslych, E. M. Taranta, M. Aldilati, and J. J. Laviola, "Effective 2d stroke-based gesture augmentation for rnns," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: <https://doi.org/10.1145/3544548.3581358>
120. J. Shen, J. Dudley, and P. O. Kristensson, "The imaginative generative adversarial network: Automatic data augmentation for dynamic skeleton-based hand gesture and human action recognition," in *2021 16th IEEE international conference on automatic face and gesture recognition (FG 2021)*. IEEE, 2021, pp. 1–8.
121. H. Tang, W. Wang, D. Xu, Y. Yan, and N. Sebe, "GestureGAN for Hand Gesture-to-Gesture Translation in the Wild," in *Proceedings of the 26th ACM International Conference on Multimedia*, ser. MM '18. New York, NY, USA: Association for Computing Machinery, Oct 2018, pp. 774–782.
122. A. Hoelzemann, N. Sorathiya, and K. V. Laerhoven, "Data Augmentation Strategies for Human Activity Data Using Generative Adversarial Neural Networks," *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2021.
123. J. Shen, J. Dudley, and P. O. Kristensson, "Simulating Realistic Human Motion Trajectories of Mid-Air Gesture Typing," in *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Bari, Italy: IEEE, Oct 2021, pp. 393–402.
124. M. Maghoumi, E. M. Taranta, and J. LaViola, "DeepNAG: Deep Non-Adversarial Gesture Generation," in *26th International Conference on Intelligent User Interfaces*. College Station TX USA: ACM, Apr. 2021, pp. 213–223.
125. T. Ha and H. Bunke, "Off-line, handwritten numeral recognition by perturbation method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 535–539, May 1997.
126. R. Vemulapalli, F. Arrate, and R. Chellappa, "Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, Jun. 2014, pp. 588–595. [Online]. Available: <http://ieeexplore.ieee.org/document/6909476/>
127. K. Davila, S. Ludi, and R. Zanibbi, "Using Off-Line Features and Synthetic Data for On-Line Handwritten Math Symbol Recognition," in *2014 14th International Conference on Frontiers in Handwriting Recognition*. Hersonissos, Greece: IEEE, Sep. 2014, pp. 323–328.
128. E. Ledda and L. D. Spano, "Applying Long-Short Term Memory Recurrent Neural Networks for Real-Time Stroke Recognition," in *Companion of the 2021 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '21. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 50–55.
129. J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 816–833.
130. Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1110–1118.
131. Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "Sketch-a-Net: A Deep Neural Network that Beats Humans," *International Journal of Computer Vision*, vol. 122, no. 3, pp. 411–425, May 2017. [Online]. Available: <https://doi.org/10.1007/s11263-016-0932-3>
132. S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data," *arXiv:1611.06067 [cs]*, Nov. 2016. [Online]. Available: <http://arxiv.org/abs/1611.06067>
133. H. Rao, S. Xu, X. Hu, J. Cheng, and B. Hu, "Augmented skeleton based contrastive action learning with momentum lstm for unsupervised action recognition," *Information Sciences*, vol. 569, pp. 90–109, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025521003443>

134. P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View Adaptive Recurrent Neural Networks for High Performance Human Action Recognition from Skeleton Data," in *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice: IEEE, Oct. 2017, pp. 2136–2145. [Online]. Available: <http://ieeexplore.ieee.org/document/8237495/>
135. C. Xin, S. Kim, Y. Cho, and K. S. Park, "Enhancing human action recognition with 3d skeleton data: A comprehensive study of deep learning and data augmentation," *Electronics*, vol. 13, no. 4, 2024. [Online]. Available: <https://www.mdpi.com/2079-9292/13/4/747>
136. G. B. Mo, J. J. Dudley, and P. O. Kristensson, "Gesture knitter: A hand gesture design tool for head-mounted mixed reality applications," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: <https://doi.org/10.1145/3411764.3445766>
137. J. Shen *et al.*, "Gesture spotter: A rapid prototyping tool for key gesture spotting in virtual and augmented reality applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 11, pp. 3618–3628, 2022.
138. A. Schäfer, G. Reis, and D. Stricker, "Anygesture: Arbitrary one-handed gestures for augmented, virtual, and mixed reality applications," *Applied Sciences*, vol. 12, no. 4, p. 1888, 2022.
139. J. Wan, Y. Zhao, S. Zhou, I. Guyon, S. Escalera, and S. Z. Li, "Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 56–64.
140. Y. Zhang, C. Cao, J. Cheng, and H. Lu, "Egogesture: A new dataset and benchmark for egocentric hand gesture recognition," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1038–1050, 2018.
141. S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *CoRR*, vol. abs/1801.07455, 2018. [Online]. Available: <http://arxiv.org/abs/1801.07455>
142. E. M. Thompson, S. Biasotti, A. Giachetti, C. Tortorici, N. Werghi, A. S. Obeid, S. Berretti, H.-P. Nguyen-Dinh, M.-Q. Le, H.-D. Nguyen, M.-T. Tran, L. Gigli, S. Velasco-Forero, B. Marcotegui, I. Sipiran, B. Bustos, I. Romanelis, V. Fotis, G. Arvanitis, K. Moustakas, E. Otú, R. Zwiggelaar, D. Hunter, Y. Liu, Y. Arteaga, and R. Luxman, "SHREC'20 track: Retrieval of digital surfaces with similar geometric reliefs," *Computers and Graphics*, vol. 91, pp. 199–218, Oct. 2020. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02916788>
143. S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.
144. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
145. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.
146. I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
147. H. Long, S. Wang, H. Hu, Y. Wang, H. Xu, and H. Wang, "Mvmsfn: A multi-view and multi-scale fusion network for online detection of heterogeneous gestures," in *2024 International Joint Conference on Neural Networks (IJCNN)*, 2024, pp. 1–7.
148. Z. Doždor, Z. Kalafatić, v. Ban, and T. Hrkać, "Ty-net: Transforming yolo for hand gesture recognition," *IEEE Access*, vol. 11, pp. 140 382–140 394, 2023.
149. F. Yang, S. Sakti, Y. Wu, and S. Nakamura, "Make skeleton-based action recognition model smaller, faster and better," in *ACM International Conference on Multimedia in Asia*, 2019.
150. L. Guo, Z. Lu, and L. Yao, "Human-machine interaction sensing technology based on hand gesture recognition: A review," *IEEE Transactions on Human-Machine Systems*, 2021.
151. A. Caputo, A. Giachetti, F. Giannini, K. Lupinetti, M. Monti, M. Pegoraro, and A. Ranieri, "Sfinge 3d: A novel benchmark for online detection and recognition of heterogeneous hand gestures from 3d fingers' trajectories," *Computers & Graphics*, vol. 91, pp. 232–242, 2020.
152. S. Zhang, X. Liu, and J. Xiao, "On geometric features for skeleton-based action recognition using multilayer lstm networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 03 2017.
153. C. Ma, S. Zhang, A. Wang, Y. Qi, and G. Chen, "Skeleton-based dynamic hand gesture recognition using an enhanced network with one-shot learning," *Applied Sciences*, vol. 10, no. 11, 2020. [Online]. Available: <https://www.mdpi.com/2076-3417/10/11/3680>
154. J. C. Nez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Vlez, "Convolutional neural networks and long short-term memory for skeleton-based human activity and hand

- gesture recognition,” *Pattern Recogn.*, vol. 76, no. C, p. 80–94, Apr. 2018. [Online]. Available: <https://doi.org/10.1016/j.patcog.2017.10.033>
155. C. Xinghao, G. Wang, H. Guo, C. Zhang, H. Wang, and L. Zhang, “Mfa-net: Motion feature augmented network for dynamic hand gesture recognition from skeletal data,” *Sensors*, vol. 19, p. 239, 01 2019.
 156. G. Devineau, W. Xi, F. Moutarde, and J. Yang, “Convolutional Neural Networks for Multivariate Time Series Classification using both Inter- and Intra- Channel Parallel Convolutions,” in *Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP’2018)*, Marne la Vallée, France, Jun. 2018. [Online]. Available: <https://hal-mines-paristech.archives-ouvertes.fr/hal-01888862>
 157. J. Hou, G. Wang, X. Chen, J.-H. Xue, R. Zhu, and H. Yang, “Spatial-temporal attention resnet for skeleton-based dynamic hand gesture recognition,” in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, pp. 273–286.
 158. J. He and R. Lawrence, “A graph-based framework for multi-task multi-view learning,” in *ICML*, 2011.
 159. C.-T. Lu, L. He, W. Shao, B. Cao, and P. S. Yu, “Multilinear factorization machines for multi-task multi-view learning,” in *Proceedings of the tenth ACM international conference on web search and data mining*, 2017, pp. 701–709.
 160. J. Zhao, X. Xie, X. Xu, and S. Sun, “Multi-view learning overview: Recent progress and new challenges,” *Information Fusion*, vol. 38, pp. 43–54, 2017.
 161. Y. Zhang and Q. Yang, “A survey on multi-task learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2021.
 162. Y.-F. Wu, D.-C. Zhan, and Y. Jiang, “Dmtmv: a unified learning framework for deep multi-task multi-view learning,” in *2018 IEEE international conference on big knowledge (ICBK)*. IEEE, 2018, pp. 49–56.
 163. X. Chen, G. Wang, H. Guo, C. Zhang, H. Wang, and L. Zhang, “Mfa-net: Motion feature augmented network for dynamic hand gesture recognition from skeletal data,” *Sensors*, vol. 19, no. 2, p. 239, 2019.
 164. N. Trivedi and R. K. Sarvadevabhatla, “Psumnet: Unified modality part streams are all you need for efficient pose-based action recognition,” *arXiv preprint arXiv:2208.05775*, 2022.
 165. S. Li, Z. Liu, G. Duan, and J. Tan, “Mvhanet: Multi-view hierarchical aggregation network for skeleton-based hand gesture recognition,” *Signal, Image and Video Processing*, pp. 1–9, 2023.
 166. I. M. Baytas, M. Yan, A. K. Jain, and J. Zhou, “Asynchronous multi-task learning,” in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 11–20.
 167. A. Sampieri, G. M. D. di Melendugno, A. Avogaro, F. Cunico, F. Setti, G. Skenderi, M. Cristani, and F. Galasso, “Pose forecasting in industrial human-robot collaboration,” in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 51–69.
 168. Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, “Disentangling and unifying graph convolutions for skeleton-based action recognition,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 143–152.
 169. Y. Chen, L. Zhao, X. Peng, J. Yuan, and D. N. Metaxas, “Construct dynamic graphs for hand gesture recognition via spatial-temporal attention,” *arXiv preprint arXiv:1907.08871*, 2019.
 170. N. Shazeer and M. Stern, “Adafactor: Adaptive learning rates with sublinear memory cost,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 4596–4604.
 171. S. Han, B. Liu, R. Cabezas, C. D. Twigg, P. Zhang, J. Petkau, T.-H. Yu, C.-J. Tai, M. Akbay, Z. Wang *et al.*, “Megatrack: monochrome egocentric articulated hand-tracking for virtual reality,” *ACM Transactions on Graphics (ToG)*, vol. 39, no. 4, pp. 87–1, 2020.
 172. S. Han, P.-c. Wu, Y. Zhang, B. Liu, L. Zhang, Z. Wang, W. Si, P. Zhang, Y. Cai, T. Hodan *et al.*, “Umetrack: Unified multi-view end-to-end hand tracking for vr,” in *SIGGRAPH Asia 2022 Conference Papers*, 2022, pp. 1–9.
 173. F. M. Caputo, P. Prebianca, A. Carcangiu, L. D. Spano, and A. Giachetti, “Comparing 3d trajectories for simple mid-air gesture recognition,” *Computers & Graphics*, vol. 73, pp. 17–25, 2018.
 174. R.-D. Vatavu, L. Anthony, and J. O. Wobbrock, “Gestures as point clouds: A \$P recognizer for user interface prototypes,” in *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ser. ICMI ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 273–280. [Online]. Available: <https://doi.org/10.1145/2388676.2388732>
 175. E. M. Taranta, M. Maslych, R. Ghamandi, and J. LaViola, “The voight-kampff machine for automatic custom gesture rejection threshold selection,” in *Proceedings of*

- the 2022 CHI Conference on Human Factors in Computing Systems, ser. CHI '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: <https://doi.org/10.1145/3491102.3502000>
176. T. F. Cootes and C. J. Taylor, "Active shape models—'smart snakes'," in *BMVC92: Proceedings of the British Machine Vision Conference, organised by the British Machine Vision Association 22–24 September 1992 Leeds*. Springer, 1992, pp. 266–275.
 177. L. A. Cardenas-Robledo, Ó. Hernández-Uribe, C. Reta, and J. A. Cantoral-Ceballos, "Extended reality applications in industry 4.0—a systematic literature review," *Telematics and Informatics*, p. 101863, 2022.
 178. J. Müller, "Enabling technologies for industry 5.0," *Independent Expert Report*, pp. 8–10, 2020.
 179. Y. Lu, J. S. Adrados, S. S. Chand, and L. Wang, "Humans are not machines—anthropocentric human-machine symbiosis for ultra-flexible smart manufacturing," *Engineering*, vol. 7, no. 6, pp. 734–737, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2095809921001612>
 180. J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing letters*, vol. 3, pp. 18–23, 2015.
 181. W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *Ifac-PapersOnline*, vol. 51, no. 11, pp. 1016–1022, 2018.
 182. A. Künz, S. Rosmann, E. Loria, and J. Pirker, "The potential of augmented reality for digital twins: A literature review," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2022, pp. 389–398.
 183. Y. Yin, P. Zheng, C. Li, and L. Wang, "A state-of-the-art survey on augmented reality-assisted digital twin for futuristic human-centric industry transformation," *Robotics and Computer-Integrated Manufacturing*, vol. 81, p. 102515, 2023.
 184. F. Böhm, M. Dietz, T. Preindl, and G. Pernul, "Augmented reality and the digital twin: State-of-the-art and perspectives for cybersecurity," *Journal of Cybersecurity and Privacy*, vol. 1, no. 3, pp. 519–538, 2021.
 185. E. Frontoni, J. Loncarski, R. Pierdicca, M. Bernardini, and M. Sasso, "Cyber physical systems for industry 4.0: Towards real time virtual reality in smart manufacturing," in *Augmented Reality, Virtual Reality, and Computer Graphics: 5th International Conference, AVR 2018, Otranto, Italy, June 24–27, 2018, Proceedings, Part II 5*. Springer, 2018, pp. 422–434.
 186. S. Chandra Sekaran, H. J. Yap, S. N. Musa, K. E. Liew, C. H. Tan, and A. Aman, "The implementation of virtual reality in digital factory—a comprehensive review," *The International Journal of Advanced Manufacturing Technology*, vol. 115, no. 5-6, pp. 1349–1366, 2021.
 187. J. Pirker, E. Loria, S. Safikhani, A. Künz, and S. Rosmann, "Immersive virtual reality for virtual and digital twins: A literature review to identify state of the art and perspectives," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 2022, pp. 114–115.
 188. "Blender web page," <https://www.blender.org/>, accessed: 2023-09-30.
 189. "Autodesk Fusion 360 web page," <https://www.autodesk.com/products/fusion-360/personal>, accessed: 2023-09-30.
 190. "Unity web page," <https://unity.com/>, accessed: 2023-09-30.
 191. E. Bozgeyikli, A. Raij, S. Katkooi, and R. Dubey, "Point & teleport locomotion technique for virtual reality," in *Proceedings of the 2016 annual symposium on computer-human interaction in play*, 2016, pp. 205–216.
 192. J. Frommel, S. Sonntag, and M. Weber, "Effects of controller-based locomotion on player experience in a virtual reality exploration game," in *Proceedings of the 12th international conference on the foundations of digital games*, 2017, pp. 1–6.
 193. A. Kulshreshth and J. J. LaViola Jr, "Exploring the usefulness of finger-based 3d gesture menu selection," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014, pp. 1093–1102.
 194. D. Odell and N. Dorbala, "The effects of head mounted weight on comfort for helmets and headsets, with a definition of 'comfortable wear time'," *Work*, no. Preprint, pp. 1–8, 2023.
 195. Y. Yan, K. Chen, Y. Xie, Y. Song, and Y. Liu, "The effects of weight on comfort of virtual reality devices," in *Advances in Ergonomics in Design: Proceedings of the AHFE 2018 International Conference on Ergonomics in Design, July 21–25, 2018, Loews Sapphire Falls Resort at Universal Studios, Orlando, Florida, USA 9*. Springer, 2019, pp. 239–248.
 196. M. Hossin and S. M.N, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, pp. 01–11, 03 2015.

197. Y. Sasaki, "The truth of the f-measure," *Teach Tutor Mater*, 01 2007.
198. D. M. Powers, "What the f-measure doesn't measure: Features, flaws, fallacies and fixes," *arXiv preprint arXiv:1503.06410*, 2015.
199. M. Goswami and others, "A comparative analysis of similarity measures to find coherent documents," *Appl Sci Manag*, vol. 8, no. 11, pp. 786–797, 2018.
200. L. d. F. Costa, "Further generalizations of the jaccard index," *arXiv preprint arXiv:2110.09619*, 2021.
201. V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.
202. D. Minnen *et al.*, "Performance metrics and evaluation issues for continuous activity recognition," *Performance Metrics for Intelligent Systems*, vol. 4, pp. 141–148, 2006.
203. S. R. Fanello *et al.*, "Keep it simple and sparse: Real-time action recognition." *Journal of Machine Learning Research*, vol. 14, 2013.
204. Y. Yin and R. Davis, "Real-time continuous gesture recognition for natural human-computer interaction," in *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2014, pp. 113–120.
205. M. Hoai and F. De la Torre, "Max-margin early event detectors," *International Journal of Computer Vision*, vol. 107, pp. 191–202, 2014.
206. V. Gupta *et al.*, "Progression modelling for online and early gesture detection," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 289–297.
207. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
208. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
209. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
210. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Evaluation metrics

In this appendix we present the complete definitions of the frame-based, event-based and efficiency-based metrics proposed for gesture recognition benchmarking.

A.1 Frame-Based classification evaluation

Frame-based metrics compare the strings of the per-frame predicted labels with the frame labels sequence of ground truth. These metrics are extensively used for continuous action and gesture recognition benchmarking, even if they are not optimal to tell if a method is usable in interactive systems, as events trigger interaction. The most common evaluation metrics are average accuracy, Jaccard Index, Levenshtein distance, and precision/recall.

A.1.1 Accuracy.

Accuracy is the number of correct predictions, which is the sum of correctly recognized gesture's frame (True positives **TP**) and non-gesture frame (True Negatives **TN**) divided by the total predictions which are the sum of correct predictions and wrong predictions (**WP**); wrong predictions consist of all misclassified gesture's frame (False Positives **FP**) and non-gesture frame (False Negative **FN**) [196]. It can be defined as a single class:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (A.1)$$

In a multi-class classification problem, Accuracy is calculated as the sum of TP and TN across all classes divided by the sum of TP, TN, FP, and FN across all classes.

$$Accuracy = \frac{\sum_{c \in C} (TP + TN)}{\sum_{c \in C} (TP + TN + FP + FN)}. \quad (A.2)$$

The accuracy provides an easy-to-understand measure of how well a classifier is performing. However, in imbalanced scenarios where one class is significantly more prevalent than the others, the accuracy can be misleading. In such cases, additional evaluation metrics like precision, recall, or F1 score give a more comprehensive assessment of the classifier's performance.

A.1.2 Precision and Recall.

Precision [196] is an evaluation metric that quantifies the number of correct positive predictions. Precision evaluates the fraction of correctly classified instances among the ones classified as positive as the ratio of correctly predicted positive examples divided by the total number of predicted positive examples; precision can be used for binary and multi-class classification. The formula for binary classification is:

$$Precision = \frac{TP}{(TP + FP)}, \quad (A.3)$$

where TP/FP are the true/false positives that were previously defined. The result is between 0 for no precision and 1 for full or perfect precision. In a multi-class classification problem, precision is calculated as the sum of TP across all classes divided by the sum of TP and FP across all classes,

$$Precision = \frac{\sum_{c \in C} TP}{\sum_{c \in C} (TP + FP)}. \quad (A.4)$$

Recall [196] is an evaluation metric that quantifies the number of correct positive predictions from all positive predictions that could have been made. Unlike precision, which only comments on the correct positive predictions out of all positive predictions, recall indicates missed positive predictions. In this way, recall provides some notion of the coverage of the positive class. Recall can be used for binary and multi-class classification. The formula for binary classification is

$$Recall = \frac{TP}{(TP + FN)}. \quad (A.5)$$

The result is between 0.0 for no recall and 1.0 for full or perfect recall. In a multi-class classification problem with more than two classes, recall is calculated as the sum of TP across all classes divided by the sum of TP and FN across all classes,

$$Recall = \frac{\sum_{c \in C} TP}{\sum_{c \in C} (TP + FN)}. \quad (A.6)$$

These metrics show how much the data is biased towards one class ¹, allowing us to understand whether our model performs well in an imbalanced dataset for a minority class ¹. However, if we aim to increase the recall, our precision decreases because, in addition to increasing the TP, we increase the FP, and if we aim to increase the precision, our recall decreases ².

A.1.3 F-Score

The F-score (also known as the F1 score or F-measure) [196] is a metric used to evaluate the performance of a Machine Learning model. It combines precision and recall into a single score. David D. Lewis first introduced the F-measure in 1992 [197]. It combines precision and recall into a single score and is calculated as follows:

$$F - Score = \frac{2 \cdot (Precision \cdot Recall)}{(Precision + Recall)}. \quad (A.7)$$

It is possible to adjust the F-score to give more importance to precision over recall or vice-versa by assigning different weights to precision and recall; this would lead to F2-Score, which assigns more weight to recall, or F0.5-Score that assigns more weight to precision ³. While precision and recall provide valuable insights individually, they don't always give a complete picture of a system's performance. For example, a model could achieve high precision but low recall, or vice versa. The F1 score combines these two metrics into a single value, providing a balanced measure of the model's effectiveness [198].

A.1.4 Area Under the curve (AUC)

To evaluate the temporal detection performance in their study, the MlGesture's authors [4] took a unique approach. They combined 50% of the videos from the test set into a single extended test video. This combined video was manually annotated with the locations of

¹ <https://medium.com/analytics-vidhya/performance-metrics-classification-model-69a5546b118c>

² <https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification>

³ <https://deeppai.org/machine-learning-glossary-and-terms/f-score>

hand gesture nuclei. The network was then applied in a sliding-window manner across all video frames. A single temporal output step of the network produced the prediction output. The researchers adopted a specific detection metric to ascertain the most suitable output. They calculated Precision-Recall (PR) curves for each class label (except for the non-gesture label) and then averaged them to generate a multi-class PR-curve. The mean Average Precision (mAP), defined as the Area Under the Curve (AUC), was utilized as their detection metric. Regarding the metric calculation, if the network produced a single detection during an annotated gesture nucleus, it was classified as a True Positive. However, if there was an additional detection within the nucleus or detections outside the nucleus, they were marked as False Positives. Missed gestures were categorized as False Negatives. The preference for the PR-curve over the Receiver Operating Characteristic (ROC) curve was rooted in the challenge of determining the correct number of True Negatives required for calculating the ROC curve, given the erratic nature of the Connectionist Temporal Classification (CTC) predictions. The spikiness of these predictions made accurately defining True Negatives difficult, making the PR-curve a more suitable choice for their evaluation approach.

A.1.5 Jaccard Index (JI)

Also known as the Jaccard similarity coefficient [199], it was first introduced in 1901 by Paul Jaccard [200] under the name of "coefficient de communauté". The JI measures the average relative overlap between true (ground truth) and predicted sequences of frames for a given gesture [6]. By quantifying the overlap between the recognized and ground truth gestures, it captures the degree of similarity, indicating how well the recognized gesture matches the intended gesture. Let $G_{s,i}$, and $P_{s,i}$ be binary indicator vectors for a sequence s , in which 1-value corresponds to frames labeled or predicted as the i_{th} class of gestures. Then, the JI for the i_{th} class is defined as:

$$J_{s,i} = \frac{(G_{s,i} \cap P_{s,i})}{(G_{s,i} \cup P_{s,i})}. \quad (A.8)$$

where $G_{s,i}$ is the ground truth, and $P_{s,i}$ is the prediction. Next is the **Jaccard Distance (JD)**, which is the complement of the JI and measures how dissimilar two sets are, and is defined as ⁴:

$$JD_{s,i} = 1 - J_{s,i}, \quad (A.9)$$

Higher Jaccard index values indicate a higher level of accuracy, as it signifies a greater overlap between the recognized gesture and the ground truth. In the ConGD dataset [77] they calculate the **mean Jaccard Index** $\bar{J}_S$, on all testing sequences $S = \{s_1, \dots, s_n\}$ with n samples as:

$$\bar{J}_S = \frac{1}{n} \sum_{j=1}^n J_{Sj}. \quad (A.10)$$

A.1.6 Levenshtein Distance (LD)

This evaluation metric was first introduced by the Russian scientist Vladimir Levenshtein in 1965 [201]. This method arose in speech recognition, where researchers were interested in measuring sentence-level accuracy rather than ensuring that each word or phoneme was detected at exactly the right time [202]. The LD [9] is calculated by counting the number of item-level changes, including insertion (FP), deletion (FN), or substitutions (WP), to transform one sequence into the other and to measure the distance between sequences. The LD formula is shown as [203]:

$$LD = \frac{WP + FN + FP}{M}, \quad (A.11)$$

⁴ Stephanie Glen. "Jaccard Index / Similarity Coefficient" From StatisticsHowTo.com: Elementary Statistics for the rest of us! <https://www.statisticshowto.com/jaccard-index/>

where each action is treated as a symbol in a sequence, M is the length of the ground truth sequence, and where FN/FP are the false negative/positives previously defined.

This metric was analyzed in [202], showing that it can be problematic for periodic gestures with highly variable duration. This metric is not directly used in the benchmarks surveyed, but it is the basis for computing the Levenshtein Accuracy (LA).

A.2 Event-Based classifier evaluation

The event-based metrics are estimated on the detected events instead of the label strings, characterized by an event window with start/end frames and labels. They are more directly linked to the usability of a practical gesture recognizer for interaction as they directly determine how well the system will respond to the user input. As explained in Ward's 2011 work [75], in event-based classification, there are additional types of errors beyond frame-by-frame analysis:

- **Fragmentation.** If several sub-parts recognize an event by the classifier.
- **Merge.** The recognition method can't distinguish several gesture executions and label them as one (the inverse of fragmentation).
- **Fragmented and Merged.** A recognizer can fragment and merge events at the same time.

Event-based metrics can assess if a method suffers from these errors, while this is not possible with frame-based methods. A common problem in continuous hand gesture datasets is the Timing problem of the frame-based evaluation metrics because if the recognition delay is too high or the gesture is not anticipated, there would be no recognition. However, by using the event-based evaluation metrics and also by defining a correct threshold, the gesture can still be recognized, even though there would be a minimum delay. The main issue in event-based evaluation is that there are no unique criteria to determine if an event is correctly detected and classified. The decision of whether a detected gesture event is correct or not is typically based on specific thresholds that can depend on specific design choices or the gesture dictionary characteristics. In SHREC '19 [3], for example, a gesture is considered correctly detected if the start time is closer than a threshold from the ground truth one, while in SHREC '22 [6], a "gesture is considered correctly detected if it has the same label as the ground truth annotation, and its time window has an intersection with the annotated one larger than half the ground truth length". The metrics depending on the detected gesture are strongly affected by these criteria/thresholds that need to be clearly defined and motivated. Given a rule to assess gesture detection, several metrics can be introduced for the gesture recognition benchmarks.

A.2.1 Detection rate (DR)/ True Positive Rate (TPR)/Event-based recall

Since we cannot use a one-to-one comparison in event-based metrics, we can define the detection rate as a necessary metric to assess the accuracy of a classification method. The Detection rate, used in [1, 3, 6] is defined as the number of Correctly Detected gestures (CD) for a class in the sequence, divided by the corresponding ground truth number (Occurrence Number ON),

$$DetectionRate = \frac{CD}{ON}. \quad (A.12)$$

This metric is similarly estimated in [1] where it is called "Recall" (event-based).

A.2.2 False Positive Rate (FPR)

To calculate the error rate [3, 6, 37, 79], we first need to calculate the 'false positive score' which is the ratio between the number of gesture predictions that do not correspond to real gestures (FP) and the total number of gestures of the same class present in the ON setting.

$$FalsePositiveScore = \frac{FP}{ON}. \quad (A.13)$$

It is necessary to note that event-based metrics are state-of-the-art evaluation metrics that are not used in many benchmarks, but it could be beneficial to find and remove the merge and fragmentation errors that can affect the method used.

A.2.3 Levenshtein accuracy (LA)

In IPNHand [64] and ZJUGesture [7] the **Levenshtein accuracy (LA)** was used as evaluation metrics for the continuous tasks. For continuous hand gestures, the metric measures the disparity between the sequence of predicted gestures and the ground truth gestures. This metric employs the Levenshtein Distance (LD) to quantify the distance between sequences of events by calculating the number of item-level changes (insertions, deletions, or substitutions) required to transform one sequence into another. This LD is divided by the number of all the events that are in the ground truth (ON). Formally:

$$LA = 1 - \frac{LD}{ON}. \quad (A.14)$$

The result of this formula is between 1, which means the two strings are identical, and 0, where the two strings have no common points.

A.2.4 ATSE, ATSR, Perf

In the ChAirGest benchmark [1], a novel time-based metric called the Accurate Temporal Segmentation Rate (ATSR) was introduced to assess the efficacy of algorithms in gesture spotting. This metric gauges the accuracy of temporal segmentation by evaluating how precisely the start and end events of correctly identified gesture instances are determined. The computation of ATSR involves the following steps: for each accurately detected gesture instance, the Absolute Temporal Segmentation Error (ATSE) is calculated. This error is derived by summing the absolute temporal discrepancies between the algorithm’s output and the ground truth for both the start and stop events. This sum is then divided by the total duration of the gesture occurrence, as measured from the ground truth data.

$$ATSE = \frac{\|Start_{GT} - Start_{Alg}\| + \|Stop_{GT} - Stop_{Alg}\|}{Stop_{GT} - Start_{GT}}, \quad (A.15)$$

$$ATSR = 1 - \frac{1}{n} \cdot \sum_{j=1}^n ATSE(j). \quad (A.16)$$

The ATSR provides an effective means of quantifying the performance of gesture spotting algorithms in terms of their ability to determine the temporal boundaries of correctly identified gestures precisely. In the final step, the ATSR metric is computed for a specific sequence by subtracting the average ATSE from 1. This computation is aimed at obtaining an accuracy rate. A perfectly precise segmentation would yield an ATSR of 1. To prevent minor timing discrepancies in the ground truth from resulting in undue penalties during the ATSE calculation, any temporal error for the start or end event that is less than 10% of the gesture’s duration is deemed irrelevant and is treated as 0. This mechanism helps ensure that minor deviations from ground truth timing do not disproportionately affect the accuracy of the assessment. The Performance Index (Perf) is a single metric performance that combines F-score and ATSR. The F-score formula used in the paper is the same as we present in Eq. (A.7). However, the Recall used in this case is *frame-based* and corresponds to the Detection Rate Eq. (A.12), while the precision is defined as the number of correctly detected events (CD) divided by the number of returned events by the method (RE):

$$Precision_{Event} = \frac{CD}{RE}. \quad (A.17)$$

These changes lead to a different formula for the F-score:

$$F_{1(Event)} = \frac{2 \cdot (Precision_{Event} \cdot Recall_{Event})}{(Precision_{Event} + Recall_{Event})}. \quad (A.18)$$

The new F-score is used to calculate the performance index (Perf):

$$Perf = 5 \cdot \frac{ATSR \cdot F_{1(Event)}}{4 \cdot ATSR + F_{1(Event)}}. \quad (A.19)$$

A.2.5 Temporal delay

In [6], the "delay" metric measures the average difference between gesture start and recognition feedback. In [204], the Responsiveness Score (RS) is the average difference between gesture end and recognition feedback. This crucial parameter evaluates the temporal delay when an identified gesture is compared to the actual start/end time provided in the ground truth. This depends not only on the classification time but also on the detection strategy and gesture types. Using a sliding window approach for the recognition may depend on the window size. If the method obtains a result close to zero, it means it can recognize the execution accurately. If it has a value greater than zero, it signifies that it recognizes it with a delay. Finally, if the value is less than zero, the recognizer can anticipate the gesture prediction.

A.2.6 Normalized time to detection

In [205], the authors introduced Normalized Time to Detection (NTtoD) to evaluate the timeliness of detection; this metric is defined as a fraction of an event that has occurred, and it's calculated as:

$$NTtD = \frac{t - s + 1}{e - s + 1}, \quad (A.20)$$

where t is the feedback time if the event starts at time s and ends at time e . This metric can be used in the context of gesture spotting in virtual and augmented reality applications. It is used to evaluate the early detection performance of key hand gestures, measuring the time it takes for the system to detect a gesture normalized by the duration of the gesture [137]. It can also be used as a metric to evaluate the performance of systems or algorithms in terms of early detection or prediction. It measures the ratio of the event duration that the detector observes before the event prediction [206].

A.3 Efficiency measures

Continuous gesture recognition methods' usability critically depends on real-time feedback, and it is mandatory to evaluate the factors influencing the efficiency of algorithms and implementations. We define Efficiency measures as metrics or criteria used to assess how effectively a recognizer performs its task while utilizing the least amount of resources. These measures help evaluate how efficiently resources such as time or computational power are used for the classification.

A.3.1 Classification time.

It is defined by the amount of time that each method takes to elaborate data for a single classification attempt. This metric is called differently in different benchmarks: extraction time in VIVA [41], time in Shrec 22 [6], runtime in Egogesture [36] and interface time in IPN hand [64].

A.3.2 Hardware resource demand

Each gesture detection method is also characterized by the corresponding computational complexity and by the demand of memory that may affect the potential use on different architectures. The most common metrics used to estimate them are the number of operations required, the model size and the number of parameters used in the neural network architectures.

- **Floating Point Operations (FLOPs)** [9]; this metric is used to measure the computational complexity or workload of a machine learning model or algorithm. FLOPs represent the number of floating-point arithmetic operations, such as additions and multiplications that are performed during the execution of the model.
- **Number of parameters** is a crucial metric to evaluate a machine learning model. When a model has a greater amount of parameters, it tends to be computationally heavier to run. This increased complexity can affect both training time and the computational resources required during inference. So, while having more parameters can enable a model to capture intricate patterns in data, it also comes with trade-offs in terms of computational efficiency. However, a too-large model may overfit the training data and perform poorly on unseen data.
- **Model size** refers to the amount of memory required to store all the parameters and associated data structures of the trained machine learning model. The primary reason for measuring model size in MB is to understand how much storage space is needed to save the trained model. This is important because machine learning models can have a large number of parameters, especially deep neural networks, and saving these parameters in a compact format is essential for practical deployment.

Classifiers

In this appendix we quickly introduce the most popular classification models used in the gesture recognition works reviewed.

- **Convolutional Neural Networks (CNNs):** are a type of artificial neural network specifically designed for analyzing visual data such as images or videos, and they are widely used in the gesture recognition field. They consist of multiple layers of interconnected artificial neurons arranged hierarchically. Each layer learns to extract increasingly complex features from the input data. The core component of a CNN is the convolutional layer. It applies a set of learnable filters, also called convolutional kernels, to the input. Each filter detects a specific pattern or feature. The filters are slid over the input, performing convolutions to produce a set of feature maps highlighting the presence of the learned features in different spatial locations. Typically, a CNN architecture consists of alternating convolutional layers and pooling layers. Pooling layers reduce the spatial dimensions of the feature maps, making the network more robust to translation and increasing its computational efficiency. Common pooling operations include max pooling and average pooling. After several convolutional and pooling layers, the network typically ends with one or more fully connected layers, which perform high-level reasoning and produce the final classification or regression output. The fully connected layers combine the features extracted by the preceding layers to make predictions based on the learned representations.
- **Graph-Convolutional Network (GCN):** This neural network architecture [207] is designed to learn from graph-structured data and can operate on graph data. GCNs are based on CNNs, and while CNNs operate on a grid of pixels, they operate on irregular graphs. There are two main algorithm categories of GCN, spatial and spectral graph convolutional networks; Spatial graph convolutional networks operate directly on the graph structure, while spectral graph convolutional networks use the graph's Laplacian matrix to perform convolutions.
- **Transformers:** This neural network architecture [144] is based on an encoder-decoder structure without any reliance on recurrence and convolutions for output generation. This model uses a self-attention mechanism that allows attendance to different parts of the input sequence for computing a representation of the input. In this architecture, the encoder takes an input sequence and generates a sequence of hidden representations, while the decoder takes the hidden representations and generates an output sequence. These two components are composed of multiple layers, each of which contains a self-attention mechanism and a feed-forward neural network; this self-attention mechanism allows the model to attend to different parts of the input sequence to compute a representation of the input, while the feed-forward neural network applies a non-linear transformation to the output of the self-attention mechanism.
- **Recurrent Neural Network (RNN):** Is a type of artificial neural network designed to process sequential data in various tasks, including gesture recognition. Unlike feed-forward neural networks, which process input data in a single pass, RNNs have a feedback connection that allows them to persist information across time steps. The key feature of

an RNN is its ability to maintain a hidden state that captures information from previous inputs in the sequence. At each time step, the RNN combines an input with the previous hidden state to produce an output and update the current hidden state. This process is repeated for each time step, allowing the network to capture temporal dependencies and context within the sequence. The basic building block of an RNN is the recurrent neuron, which takes an input and the previous hidden state as inputs and produces an output and the updated hidden state. The hidden state is updated using a set of learnable weights, similar to other neural network architectures.

- **Long short-term memory (LSTM):** This architecture [208] is a type of RNN, and is used to learn, process, and work with sequential data. This architecture was introduced to address the RNNs' problem for learning long-term dependencies (Vanishing gradient problem), the dependencies are learned between time steps of the data through utilizing extra gates such as memory cells, input gates, output gates, and forget gates. These gates control what information in the hidden cell is exported as output to the next hidden state.
- **Gated Recurrent Unit (GRU):** This architecture [209] is another type of RNN, and is designed as a simpler alternative to LSTM architecture with the same goal of addressing the vanishing gradient problem. GRU is easier and faster to train than LSTM because in GRU, there are only two gates available with lower parameters, and there is no output gate. The two gates are a reset gate and an update gate, where the reset gate helps the model decide how much of the past information to forget, and the update gate helps the model decide how much of the current state to keep.
- **Hidden Markov Model (HMM):** Is a statistical model that is used in machine learning and pattern recognition. There are two types of hidden variables in this model, where a probabilistic relationship between a sequence of observations and a sequence of hidden states is described. These hidden states refer to the underlying model or process that can be unknown or hidden observations and states. This model is based on two key assumptions, the first assumption is that the current state of the model depends only on the previous state, a property known as the Markov property, the second assumption is that the observations are conditionally independent given the state, meaning that the features depend only on the current state. The parameters of an HMM are of two types: transition probabilities and emission probabilities. Transition probabilities represent the likelihood of transitioning from one hidden state to another, while emission probabilities describe the probability of observing an output given a hidden state.
- **Residual Neural Network (ResNet):** Is a deep learning model that uses skip connections and residual blocks to improve training efficiency and accuracy [210]. The main idea behind ResNets is to learn residual functions with reference to the layer inputs, allowing the network to fit the residual mapping instead of the initial mapping, helping to address the vanishing gradient problem. In this model, skip connections connect the activations of a layer to further layers by skipping some layers in between. The output of the previous layer is added to the output of the layer after it in the residual block, which helps preserve the information and gradients during training. Also, a residual block in this model is a sub-network comprising one or more layers responsible for learning the residual function. The output of a residual block is the sum of its input and the output of its layers, which is then passed through an activation function.