Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on 3DOR2024

SHREC 2024: Recognition of dynamic hand motions molding clay

Ben Veldhuijzen [a,*], Remco C. Veltkamp [a], Omar Ikne [b], Benjamin Allaert [b], Hazem Wannous [b], Marco Emporio [c], Andrea Giachetti [c], Joseph J. LaViola Jr. [d], Ruiwen He [e], Halim Benhabiles [b], Adnane Cabani [f], Anthony Fleury [b], Karim Hammoudi [g,h], Konstantinos Gavalas [i], Christoforos Vlachos [i], Athanasios Papanikolaou [i], Ioannis Romanelis [i], Vlassis Fotis [i], Gerasimos Arvanitis [i], Konstantinos Moustakas [i], Martin Hanik [j,k,l], Esfandiar Nava-Yazdani [l], Christoph von Tycowicz [l]

[a] *Utrecht University, Department of Computing Sciences, Netherlands*
[b] *IMT Nord Europe, Institut Mines-Télécom, Univ. Lille, Centre for Digital Systems, F-59000 Lille, France*
[c] *Computer Science, University of Verona, Verona, Italy*
[d] *Computer Science, University of Central Florida, FL, USA*
[e] *Computer Science Department, École supérieure d'ingénieurs Léonard-de-Vinci (ESILV), F-75000 Paris, France*
[f] *Université Rouen Normandie, ESIGELEC, IRSEEM, 76000 Rouen, France*
[g] *IRIMAS, Université de Haute-Alsace, Mulhouse, France*
[h] *Université de Strasbourg, France*
[i] *University of Patras, Greece*
[j] *Freie Universität Berlin, Kaiserswerther Str. 16-18, 14195 Berlin, Germany*
[k] *Technische Universität Berlin, Straße des 17. Juni 135, Berlin, 10623, Germany*
[l] *Zuse Institute Berlin, Takustr. 7, Berlin, 14195, Germany*

ARTICLE INFO

Keywords:
Gesture recognition
SHREC
Hand skeleton gestures
3D shape retrieval challenge
Motion capture
Neural networks

ABSTRACT

Gesture recognition is a tool to enable novel interactions with different techniques and applications, like Mixed Reality and Virtual Reality environments. With all the recent advancements in gesture recognition from skeletal data, it is still unclear how well state-of-the-art techniques perform in a scenario using precise motions with two hands. This paper presents the results of the SHREC 2024 contest organized to evaluate methods for their recognition of highly similar hand motions using the skeletal spatial coordinate data of both hands. The task is the recognition of 7 motion classes given their spatial coordinates in a frame-by-frame motion. The skeletal data has been captured using a Vicon system and pre-processed into a coordinate system using Blender and Vicon Shogun Post. We created a small, novel dataset with a high variety of durations in frames. This paper shows the results of the contest, showing the techniques created by the 5 research groups on this challenging task and comparing them to our baseline method.

* Corresponding author.
E-mail addresses: B.veldhuijzen@students.uu.nl (B. Veldhuijzen), r.c.veltkamp@uu.nl (R.C. Veltkamp), omar.ikne@imt-nord-europe.fr (O. Ikne), benjamin.allaert@imt-nord-europe.fr (B. Allaert), hazem.wannous@imt-nord-europe.fr (H. Wannous), marco.emporio@univr.it (M. Emporio), andrea.giachetti@univr.it (A. Giachetti), jlaviola@ucf.edu (J.J. LaViola Jr.), ruiwen.he@devinci.fr (R. He), halim.benhabiles@imt-nord-europe.fr (H. Benhabiles), adnane.cabani@esigelec.fr (A. Cabani), anthony.fleury@imt-nord-europe.fr (A. Fleury), karim.hammoudi@uha.fr (K. Hammoudi), k_gavalas@ac.upatras.gr (K. Gavalas), chris.vlachos@ac.upatras.gr (C. Vlachos), up1053560@ac.upatras.gr (A. Papanikolaou), iroman@ece.upatras.gr (I. Romanelis), vfotis@ece.upatras.gr (V. Fotis), arvanitis@ece.upatras.gr (G. Arvanitis), moustakas@ece.upatras.gr (K. Moustakas), hanik@zib.de (M. Hanik), navayazdani@zib.de (E. Nava-Yazdani), vontycowicz@zib.de (C. von Tycowicz).

https://doi.org/10.1016/j.cag.2024.104012
Received 12 April 2024; Received in revised form 8 July 2024; Accepted 9 July 2024
Available online 14 July 2024

1. Introduction

The recognition of motions based on hand skeletons is becoming a more effective and intuitive tool for Human–Computer Interaction (HCI) applications. Especially in Virtual Reality (VR) and Mixed Reality (MR) devices. During the years, we have seen a higher focus on using a hand skeletal dataset for gesture recognition, however the gesture classes used in these datasets are quite simplistic and distinct [1–3]. Furthermore, most hand gesture benchmarks use only a singular hand. Hand gesture recognition has been an active research field for the past 25 years, where a range of different methods and network approaches have been proposed.

In this track, we present a novel highly similar dynamic hand gesture dataset that provides sequences of hand skeletal data using two hands over a variable time length. We construct a novel recognition task focusing on precise hand motions using both hands and compare the results of the five groups that have been registered for this track with our baseline method.

The paper is organized as follows: Section 2 presents the related work of the previous SHREC hand gesture tracks; Section 3 presents the novel dataset; Section 4 presents the task for the participants and the evaluation method; Section 5 presents the participants and their proposed methods together with our baseline method; Section 6 presents the results; and these results will be discussed in Section 7.

2. Related work

Hand gesture recognition has been a consistent research field where several benchmarks have been created over the years. A popular benchmark is the SHREC'17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset [1], featuring dynamic gestures that will be used in interactive applications. Many methods for hand gesture classification have been evaluated on this dynamic benchmark. The benchmark proposed in the SHREC 2019 track on online gesture detection [2] was focused on gesture sequences and challenged methods to lower the number of false positives. The track SHREC'21 Skeleton-based Hand Gesture Recognition in the Wild [4] was created to test complex gestures in the form of XR interactions. Which was later improved on by the SHREC 2022 track on online detection of heterogeneous gestures [5], which removed ambiguous classes and avoided annotation issues affecting the previous SHREC 21 benchmark.

These datasets, however have weaknesses: To begin, most of these dynamic gestures are highly focused on their global motion, namely the swipes, cross and V classes. Which lowers the significance of looking at the shape of the hand. Second, these benchmarks do not contain similar or highly detailed motion classes. Lastly, all these benchmarks consist of data using a singular hand, while some gestures might require that they be performed using both hands.

3. Dataset creation

We created our novel benchmark, trying to overcome the weaknesses mentioned in the related work section. We created a novel dataset of precise, small motions that are highly similar, using both hands while keeping importance on both the global motion data as well as hand shape information.

This novel small dataset is composed of 62 motions captured using a Vicon system, consisting of 7 classes of motions divided randomly into 4 cross-validation folds. Each cross-validation fold contains 15 motions in the test set. The 7 motion classes are:

- **Pressing** the clay to make it stick to the pottery wheel.
- **Making a hole** in the clay.
- **Tightening** the cylinder of the clay.
- **Centering** the clay.
- **Raising** the base structure of the clay

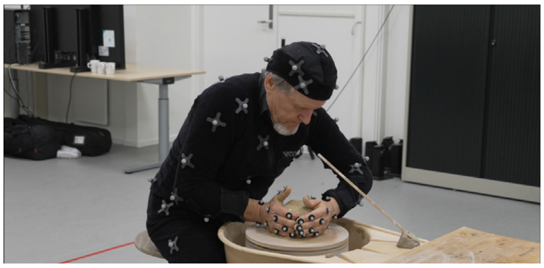
Fig. 1. Recording of hand motions using clay to create a vase by potter Kees Agterberg.

- **Smoothing** the walls
- Using the **sponge** to make the clay more moist.

We recorded the hand motions of an experienced potter who sculpted the same pot with and without clay. Our dataset is a unique set on the subject of molding clay. Molding clay is a precise and delicate act where the potter makes small and precise hand movements using both hands. Datasets of this nature are scarce in literature. This dataset has been prepared by dedicating a professional potter, technicians, and a fine acquisition platform (e.g., through a Vicon Optical Motion Capture system). The setup and acquisition protocol achievement required weeks of planning. The motions captured from the potter are perfect for our recognition benchmark since they fit all the aspects necessary to overcome the weakness mentioned earlier. Due to the potter's movements being so precise, our motions are variable in frame length and quite long. resulting in frame lengths between 29 and 3721 frames with an average of 990 frames, compared to previous benchmarks where motions had a frame length of 15 to 50 frames on average.

For this project, we are using the motion capture lab at Utrecht University.[1] These recordings are done using a Vicon system.[2] This system contains 14 vantage cameras that work with the Vicon Shogun and Vicon Shogun Post software. That will track 28 reflective soft-base markers on the potter's hands and body see Fig. 1. This way, we can do full body and hand tracking in real time while the potter is at work and record high quality motion data.

We use Vicon Shogun Post to remove any stuttering found during the recording and to export the recording of the potter to a Filmbox (FBX) format. We then extract the coordinate system of the hand skeleton on a frame-by-frame basis by using a custom made small blender script. The script exports the coordinates of the markers to a text file where each row represents the data of a specific frame, followed by the 28x3 coordinate float positions (14 per hand see Fig. 2 (x;y;z)) of the markers.

The structure of the coordinate system is as follows, where L and R stand for left and right hand, respectively:

$$\begin{gathered}
Frame; LIWR(x;y;z); LOWR(x;y;z); LIHAND(x;y;z);\\
LOHAND(x;y;z); LTHM3(x;y;z); LTHM6(x;y;z);\\
LIDX3(x;y;z); LIDX6(x;y;z); LMID0(x;y;z); LMID6(x;y;z);\\
LRNG3(x;y;z); LRNG6(x;y;z); LPNK3(x;y;z); LPNK6(x;y;z);\\
RIWR(x;y;z); ROWR(x;y;z); RIHAND(x;y;z);\\
ROHAND(x;y;z); RTHM3(x;y;z); RTHM6(x;y;z);\\
RIDX3(x;y;z); RIDX6(x;y;z); RMID0(x;y;z); RMID6(x;y;z);\\
RRNG3(x;y;z); RRNG6(x;y;z); RPNK3(x;y;z); RPNK6(x;y;z);
\end{gathered}$$

For the location of the markers on the left hand, see Fig. 2. The marker locations on the right hand are in similar locations as this picture but mirrored.

[1] uu.nl/en/research/motion-capture-and-virtual-reality-lab.

[2] vicon.com/software/shogun.

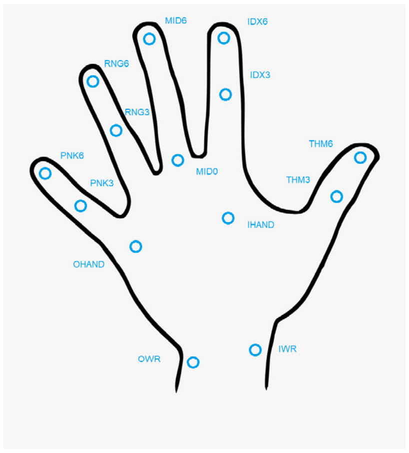

Fig. 2. Location of markers used on the left hand during recording.

4. Task and evaluation

Participants are asked to develop methods that can detect and classify hand movements based on the given skeletal coordinate system in the test set. The small size of the train set, the motion of two hands simultaneously, and the precise and highly similar motions of the potter's hands make it a novel recognition task. That requires methods to look into motion details as well as creating difficulty for training.

The result should consist of a single text file with a row representing the number of motion in the test dataset, followed by the motion class. Participants are also required to submit their algorithms and information on how to run them for verification purposes.

For the evaluation, the recognition accuracy will be computed per class as well as the total accuracy over the entire test set. We will also create a confusion matrix to extract more information from the methods. We also use the metrics precision (Eq. (1)): percentage of positive class predictions that are correct; recall (Eq. (2)): percentage of positive cases correctly predicted by the method; and F1 score (Eq. (3)): a harmonic average of the precision and recall.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{1}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2}$$

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

5. Participants and proposed methods

Five research groups were registered for the contest and sent their results. A total of six submissions have been obtained with different classification strategies or parameters. Their methods are described in the following subsections, and we compare their results against our baseline method. The five groups were composed as follows:

- *Group 1: SkelMAE: Skeleton-based MAE and STGCN*
  Omar Ikne, Benjamin Allaert and Hazem Wannous
- *Group 2: Windowed Multi View*
  Marco Emporio, Andrea Giachetti and Joseph J. LaViola Jr
- *Group 3: DET-ACTIONS: DEep-based Technique for ACTion Identification Operations from haNd-derived Skeletons*
  Ruiwen He, Halim Benhabiles, Adnane Cabani, Anthony Fleury and Karim Hammoudi
- *Group 4: HMM-based classification & RNN-based approach*
  Konstantinos Gavalas, Christoforos Vlachos, Athanasios Papanikolaou, Ioannis Romanelis, Vlassis Fotis, Gerasimos Arvanitis and Konstantinos Moustakas.

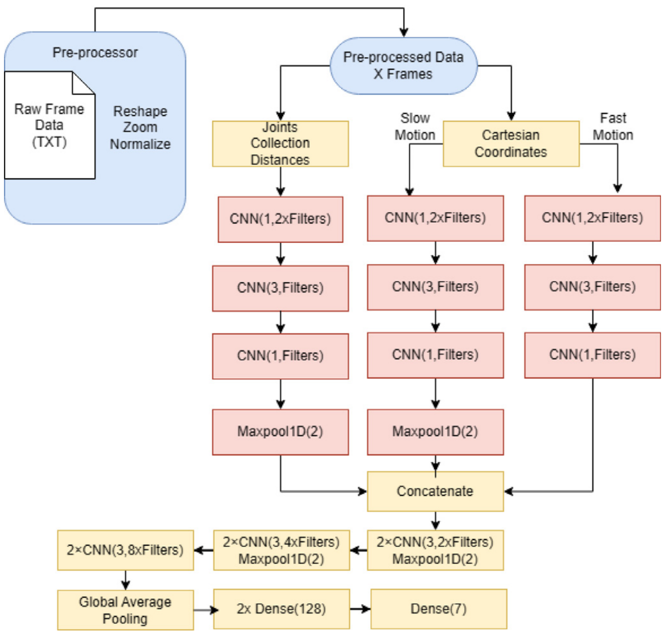

Fig. 3. Overview of the SHREC2024 DD-net framework.

- *Group 5: SE(3)-equivariant Graph Convolutional Network*
  Martin Hanik, Esfandiar Nava-Yazdani and Christoph von Tycowicz.

5.1. Baseline: Modified DD-net

For the baseline method, we customized an algorithm from Fan Yang et al. that showed high results in a previous SHREC hand gesture track [1], namely the Double-feature Double-motion Network (DD-Net) [6] available at GitHub.[3] This network uses simple 1D convolutional operations to classify motions using Cartesian coordinate features. The network architecture can be found in Fig. 3.

5.1.1. Feature extraction

DD-net derives three features from the hand joint stream, namely: the Joint Collection Distances (JCD), the short-term slow motion M_{slow}, and the short-term fast motion M_{fast}. The JCD is a location-viewpoint-invariant feature that calculates the Euclidean distances between a pair of collective joints on all frames, a feature that characterizes the hand pose. The slow motion and fast motion features calculate the temporal difference of the Cartesian coordinate feature to obtain the global motions of the hand skeleton. The slow motion calculates this for every frame, while the fast motion calculates this for every other frame.

5.1.2. Changes for SHREC2024

For the customization of the algorithm, we first had to make sure that all the data from the SHREC2024 track would be loaded correctly into the algorithm. We transitioned from 22 joints in the SHREC17 [1] track to 28 joints using both hands. Furthermore, we immediately realized the difference in frame lengths of the SHREC17 skeletal coordinate data compared to our SHREC24 data. After parameter tuning, we realized that increasing the frame length from 32 frames to 64 frames, 128 frames or even 256 frames did not increase the global accuracy of the method. It did however slow down the performance significantly. In the end, we decided to use a frame length of 32 frames for the zoom function. Afterwards, we decided to upscale the filters from the

[3] GitHub-Link-Baseline-group.

previous 64 filters to 256 filters. This increase in the number of filters improved the accuracy of the classification in the test set on motion classes like "Centering" and "MakingHole".

5.1.3. Implementation details

We trained each model on DD-net for 500 epochs using an Adam optimizer. With an annealing learning rate that drops from 1×10^{-4} to 5×10^{-6}. We did not apply pre-trained weights. We experimented with different frame lengths and filter sizes to see the effect on the global accuracy of the model. All models are implemented in TensorFlow. We trained the system on an Intel(R) Core(TM) i7-7700K CPU @ 4.2 GHz with 32 GB 3000 MHz DDR4 RAM.

5.2. Group 1: SkelMAE: Skeleton-based MAE and STGCN

5.2.1. Method description

Group 1 proposed an innovative approach to improve skeleton-based hand gesture recognition by integrating self-supervised learning, a promising technique for acquiring distinctive representations directly from unlabeled data and showed to be useful in case of limited annotated data [7,8]. The proposed method takes advantage of prior knowledge of hand topology, combining topology-aware self-supervised learning with a customized skeleton-based architecture to derive meaningful representations from skeleton data under different hand poses.

The proposed Mask Auto-Encoder (MAE) [9] is based on a Vision Transformer (ViT) [10] architecture adapted for skeletal data processing, with some novelties including: (1) Integration of Fourier feature mapping, showed to outperform linear mapping in capturing spatial relationships [7,11]. (2) A modified attention mechanism formula that incorporates adjacency information, enhancing joints spatial connectivity encoding. Code and trained models are available at GitHub[4]

5.2.2. Masking strategy

We propose to use a widely adopted technique involving randomly masking a number of joints in the hand skeleton [8,9]. We adapt this method to randomly mask a given ratio of joints in each hand (see Fig. 4).

5.2.3. Model architecture

The architecture of the MAE is designed to process skeletal data. It is based on an asymmetric encoder–decoder architecture, both built upon the ViT model as illustrated in Fig. 4.

Encoder. Based on ViT model, we design our encoder to process skeleton data. Given the non-masked joint-level coordinates v of a hand skeleton, the encoder employs a Fourier feature mapping $\gamma(v)$ [11] to project spatial coordinates into a higher-dimensional space using sine and cosine functions of different frequencies. Fourier features embedding enhances the model's ability to capture spatial relationships in the skeleton data. By representing joint movements and interdependencies as frequencies, the model gains a more comprehensive understanding of the nuanced patterns in skeletal structures.

The Fourier feature mapping is employed to embed the 3D coordinates (x, y, z) into a 256-dimensional vectors. They are then fed into a series of ViT blocks including a self-attention mechanism and feed-forward layers to learn distinctive features in latent space for each hand pose. This architecture allows the encoder to capture complex relationships and dependencies between skeleton joints.

The MAE encoder is implemented based on a ViT of depth 6, with attention mechanisms in each layer with 8 heads for multi-head attention and incorporates feed-forward networks with a dimension of 512. The embedding dimension is set to 256.

Decoder. The decoder is designed to reconstruct the masked joints in the skeleton data. It operates identically to the encoder, but with a different set of parameters. It first adds positional embeddings specific to the decoder. Then it concatenates the masked tokens, represented by a learnable mask token, with the encoded non-masked joints tokens. Subsequently, the decoder attends to this combined sequence using a ViT transformer. Finally, the model predicts the missing joints' coordinates.

The MAE decoder is built as a counterpart to the encoder, adopting the ViT architecture with a depth of 6 and an 8-head attention mechanism for multi-head attention.

Enhancing spatial connectivity. Spatial connectivity between hand joints is crucial for accurate recognition of hand gestures. While ViT models intrinsically capture a certain level of spatial relationships in their attention mechanisms, the anatomical constraints of the hand skeleton can benefit from the explicit integration of adjacency matrices. In our approach, we incorporate adjacency matrices during both the encoding and decoding.

The inclusion of adjacency matrices improves spatial modeling, enabling the attention mechanism to explicitly take into account the spatial layout of hand joints. The modified attention mechanism formula is provided in Eq. (4).

$$\text{Attention}(Q, K, V, \mathbf{A}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} \odot \mathbf{A}\right)V \tag{4}$$

In this context, Q, K, and V represent the query, key, and value components of the original attention mechanism [12]. While A denotes the adjacency matrix, embedding spatial connectivity between hands joints.

For the encoder, we only consider the connectivity between the non-masked joints (masked adjacency matrix), while for the decoder, the connectivity between all joints is considered (complete adjacency matrix). The proposed adjacency matrix is illustrated in Fig. 5.

We employ the Mean Squared Error (MSE) as the main loss function for the MAE.

5.2.4. Fine-tuning for dynamic hand gesture recognition

To assess the ability of the MAE model to acquire discriminative representations of the hands at various poses, we rely on the Space–Time Graph Convolutional Network (STGCN) [13] as the backbone architecture for skeleton sequence classification. The STGCN has demonstrated remarkable capabilities in learning temporal relationships, enabling it to identify complex patterns in sequential data.

Given a sequence of 3D hand joints, we use the MAE pre-trained encoder to acquire the learned representations (latent space), which then serve as the basis for training the STGCN.

5.2.5. Implementation details

For MAE training, we selected the AdamW optimizer with a learning rate of 2×10^{-4} and a weight decay of 5×10^{-2}. The learning rate is gradually reduced during training using Cosine Annealing scheduler [14]. The masking ratio is set to 0.7, meaning that 70% of the joints are randomly masked in each hand.

For the STGCN, we set a sequence length of 3000 frames, either padding or truncating sequences accordingly. For training we employed the AdamW optimizer with a learning rate of 1×10^{-3} and a weight decay of 5×10^{-4}. The learning rate is gradually reduced using the same scheduler as for MAE. We adopt the cross-entropy loss with label smoothing of as the fine-tuning loss with a smoothing rate of 0.1.

Pre-training spans 100 epochs with a batch size of 64, while fine-tuning spans 30 epochs with a batch size of 2. All models are implemented in PyTorch.

[4] GitHub-Link-group1-SkelMAE.

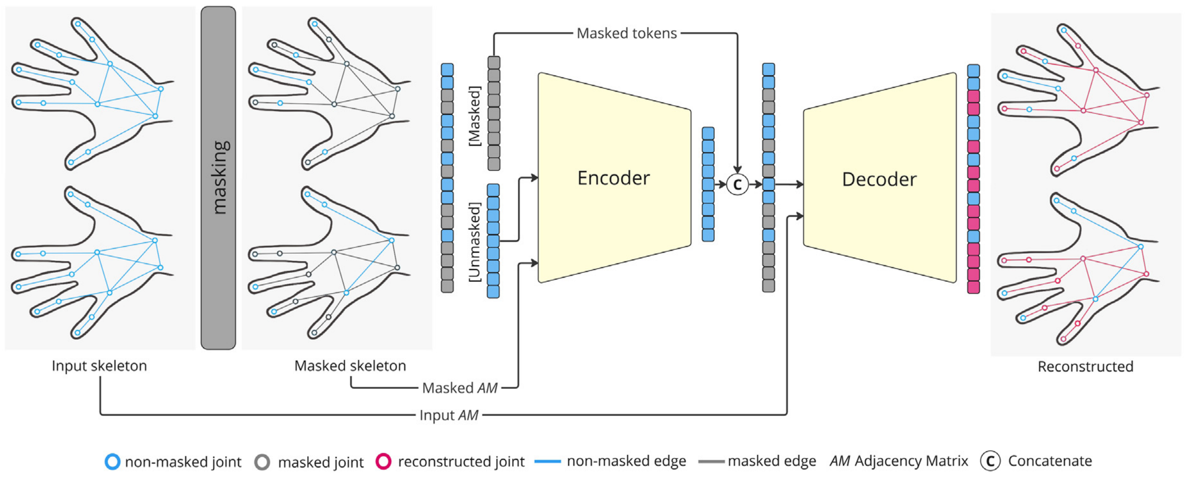

Fig. 4. Proposed MAE for hands skeleton reconstruction. We mask a given ratio of joints in each hand, the unmasked joints are encoded by the encoder while taking into account their connectivity given by the masked adjacency matrix. The encoded joints are then concatenated with the masked tokens and passed through the decoder along with the connectivity matrix to reconstruct the masked joints.

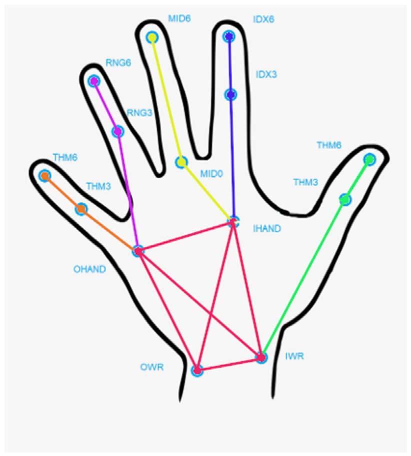

Fig. 5. SkelMAE: proposed hand adjacency connections.

5.3. Group 2: Windowed multi view

5.3.1. Method description

Group 2 leveraged a method successfully applied on a continuous gesture recognition task, On-Off deep Multi-View Multi-Task [15], adapted for this specific action recognition problem. Starting from the OO-dMVMT code,[5] they adjusted it by eliminating the multi-task component. As in their original work, providing state-of-the-art performances on continuous gesture recognition benchmarks [2,5].

The network used for the windows' classification and the input features used to feed it are derived from the Double-feature Double-motion Network (DD-Net) [6] framework. The network is based on a simple 1D convolutional neural network and provides a good classification of segmented gestures. The network is trained with feature arrays that are derived by the original hand joints stream, for each input sequence are extracted three features:

- *Joint Collection Distances* (JCD): Represents the Euclidean distances between pairs of collective joint features, invariant to location and viewpoint.
- *Short-term slow motion* M_{slow}: Calculates the 1-frame linear velocity for every individual joint across all joints.
- *Short-term fast motion* M_{fast}: Similar to short-term slow motion, but linear velocity is computed every other frame, skipping the ones in between.

In practical terms, $\mathrm{M}_{\mathrm{slow}}$ and $\mathrm{M}_{\mathrm{fast}}$ model the short-term global motion of the skeleton in terms of speed, while JCD characterizes the hand pose.

Group 2 trained the network to classify fixed-sized windows of the hand pose stream. In the testing phase, it predicts a label for a set of windows of the same size sampled in the processed action stream. The action label of the test sequence is then obtained with a majority voting over the window set.

5.3.2. Sliding-window approach

We incorporated the sliding-window approach proposed in the gesture recognizer On-Off deep Multi-View Multi-Task paradigm (OO-dMVMT) [15]. Rather than feeding the entire sequence directly into the network, we decompose the sequence into smaller windows. We extract DD-Net features for each of these windows. Subsequently, we assign the corresponding action label to each window. All the windows extracted in this manner collectively form the input dataset for the network.

5.3.3. Fine-tuning of parameters

The network underwent testing with window sizes 16, 50, and 100 frames. We maintained a consistent 10% shift between windows, resulting in 1, 5, and 10 frames distance between the center of one window and the next. Upon analysis of our graphs all window sizes exhibit strong performance during training phase. However the 100-frame windows achieved the best results in classifying.

5.3.4. Train and test

To assess the method's effectiveness, we partitioned the Training-set, allocating approximately 75% for training and the remainder for validation. During the dataset loading phase, each sequence is segmented into windows of 100 frames, with a step size of 10 frames between the center of one window and the next. These windows, created through this process, are utilized for training the network. In each epoch, the network undergoes testing on the validation dataset. If the network achieves a superior result compared to the previously saved one, the network parameters are then saved.

After completing 100 training epochs, the test set sequences are sequentially segmented into windows and provided as input to the network to evaluate its effectiveness.

[5] GitHub-Link-group2-Windowed-Multi-View.

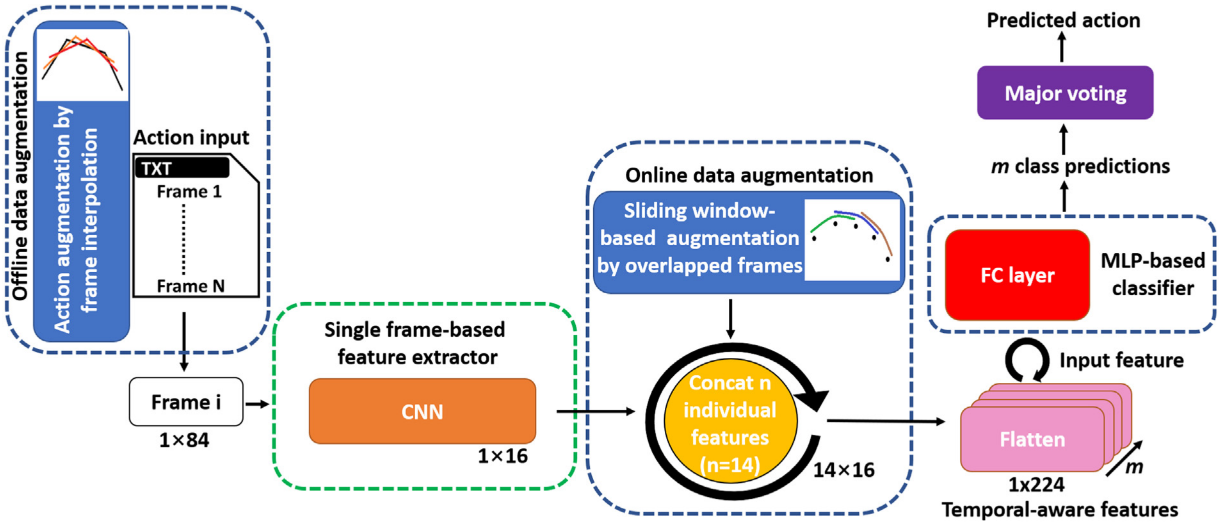

Fig. 6. Overview of DET-ACTIONS bottom-up deep learning-based analysis framework for hand motion recognition.

5.4. Group 3: DET-ACTIONS: DEep-based technique for ACTion identification operations from haNd-derived skeletons

5.4.1. Method description

Group 3 proposed a deep learning based framework for action recognition illustrated in Fig. 6 available at Google Colab.[6] First, an action augmentation stage is operated over the imbalanced action data through an offline stage. Our augmentation aims to produce the same number of action files per action category. To perform this augmentation, we apply an ordered interpolation (e.g. a variant of [16]) over frame coordinates of an action file in order to generate a new one. This interpolation acts as an action motion translation and is guided by an alpha parameter which regulates the translation steps for generating a number of actions which is equal to the maximal number of actions contained in a class category amongst the original dataset. Once the number of action files balanced for each class, we apply a feature extractor over each single frame contained in an action file in order to get a feature vector of dimension 16. Then an online augmentation stage is performed on the transformed action files by using a sliding window-based strategy [17].

To this end, a set of n successive frames is considered (n = 14) in order to take into account the temporal dimension. This operation which is repeated with an overlapping step equal to one permits to produce m sequences of temporal-aware features (each one of dimension 224). Then, a MLP-based classifier is employed to predict a membership class to an action from a temporal-aware feature. The prediction of the class of the input action file is finally calculated by applying a major voting over the output class predictions obtained from the m temporal-aware features. Our analysis framework operates through a bottom-up strategy in the sense that frames are first individually characterized by considering that each frame represents a pattern of an action motion. Then characterized frames are aggregated for being processed by sequences in order to embed the temporal dimension. Additionally, two successive augmentations are applied towards improving the classification performances. The core component of our framework which is a CNN-based feature extractor is described hereafter.

5.4.2. Single frame-based feature extractor

To build our feature extractor from a single motion frame, we designed a CNN-based architecture [18] which is illustrated in Fig. 7. The architecture is composed of two successive processing backbones namely image generator and classifier. The image generator takes in input f_i vector corresponding to the 28 3D raw coordinates of the hand markers (LIWR(x;y;z), etc.) and transforms it into a new frame representation, namely a feature map M_i (28×28). The feature map is then injected into the classifier backbone to predict its action class C_i. The final features are extracted from the intermediate global average pooling layer preceding the output layer and corresponding to a vector of dimension 16. It is worth mentioning that the whole architecture is trained on the dataset including augmented actions.

5.4.3. Implementation details

Our method is implemented in Google Colab, Colab GPU runtime comes with an Intel Xeon CPU @2.20 GHz, 13 GB RAM, a Tesla K80 accelerator, and 12 GB GDDR5 VRAM. The training to create our two pre-trained files: The MLP classifier and the CNN feature extractor took around 30 min.

5.5. Group 4 Run1: HMM-based classification

5.5.1. HMM-based classification

The proposed method utilizes an array of Hidden Markov Models (HMMs) with Gaussian mixture emissions. HMMs are known to be particularly well suited for modeling and classifying signals that demonstrate intrinsic temporality, like human speech [19] and movement [20]. This makes them a promising choice for the present task of hand action recognition. Our code is available at GitHub[7]

5.5.2. Architecture

The basic architecture of the proposed solution is illustrated in Fig. 8 Each input sequence is initially filtered, processed and flattened to a single vector (h), which is then fed to N distinct HMMs. Each HMM models one of the observed actions (classes) and, using a scoring function, evaluates the (log) probability of the given input sequence. The most likely match can then be extracted using a simple voting system based on the generated probabilities.

Each processing step is described in detail in the following paragraphs.

[6] Colab-Link-group3-DET-ACTIONS.

[7] GitHub-Link-group4-1-HMM.

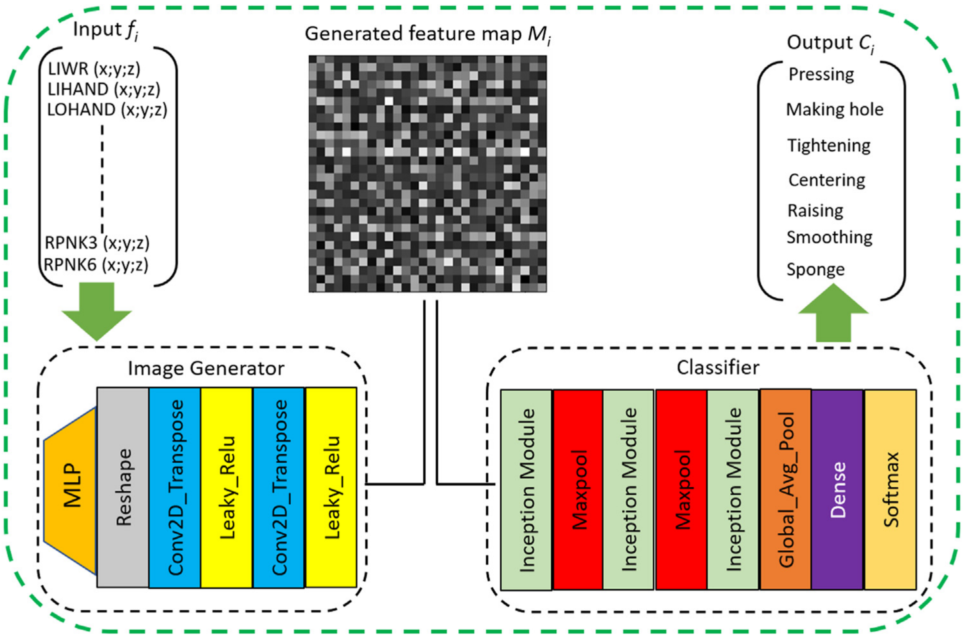

Fig. 7. CNN-based architecture for feature extraction from a single motion frame.

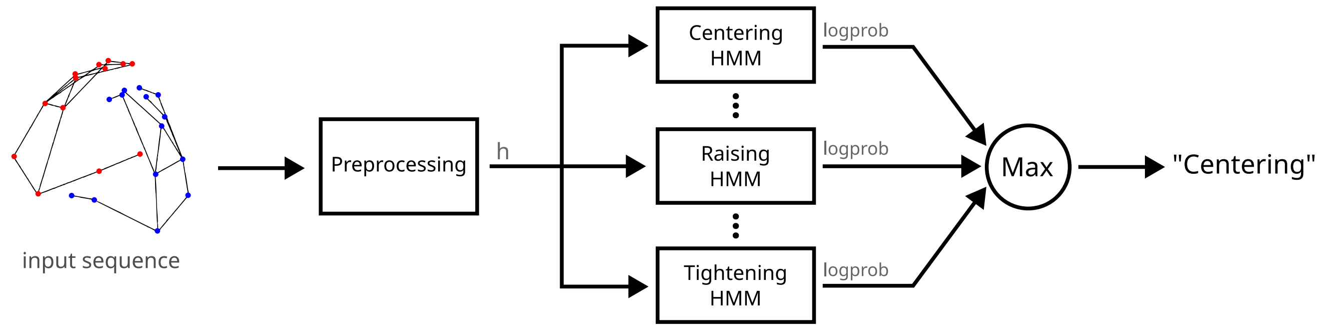

Fig. 8. Architecture of the proposed HMM-based method.

5.5.3. *Preprocessing*

Each of the provided examples is compromised of a sequence of frames, with each frame containing the coordinates of each marker. In order to train the HMMs, each sequence has to be converted to a single vector. Different ways of generating this representation were tested and compared, with the most efficient ultimately being interlacing the position data with estimated velocity data:

$$h_t = [x_1\ y_1\ z_1\ \Delta x_1\ \Delta y_1\ \Delta z_1\ x_2\ y_2\ z_2\ \Delta x_2\ \Delta y_2\ \Delta z_2\ \cdots\ ] \tag{5}$$

$$h = [h_0\ h_1\ h_2\ \cdots\ ] \tag{6}$$

The velocity of each marker is estimated as the difference between the current coordinates of the marker and those of the previous frame.

Another useful preprocessing step identified during testing was filtering the data by keeping only markers placed on the subjects fingertips (*THM, IDX, MID, RNG* and *PNK*), wrist (*IWR* and *OWR*) and center of the hand (*IHAND*). This improves training speed without affecting the models performance, as the positions of the other markers seem to provide mostly redundant information.

Finally, each sequence can be downsampled by only keeping every n frames. This improves training speed and, in some cases, also improves performance as the delta values become more intensified.

5.5.4. *HMMs*

One fully connected first order HMM is fitted to model the provided training examples of each separate class using the Expectation–Maximization (EM) algorithm [21]. The observations for each state are modeled using a Gaussian Mixture Model (GMM) with a full covariance matrix. The number of states of each HMM, as well as the number of states of each GMM are considered free variables.

The implementation of HMMs used was provided by the hmmlearn[8] python library, while hyperparameter optimization was performed based on leave-one-out cross validation (LOOCV) manually and automatically using Optuna [22].

5.6. *Group 4 Run2: RNN-based approach*

5.6.1. *RNN-based approach*

The data was provided as sequences of frames requiring classification. This made Recurrent Neural Networks (RNN) perfect for the task. A bidirectional Long Short-Term Memory (bi-LSTM) layer was used as the RNN layer, in order to extract the features from the data, preserving temporal relations. The features were subsequently fed into a linear layer with one output per class in the dataset, representing the score for that particular class. Our code is available at GitHub.[9]

[8] https://github.com/hmmlearn/hmmlearn.
[9] GitHub-Link-group4-2-RNN.

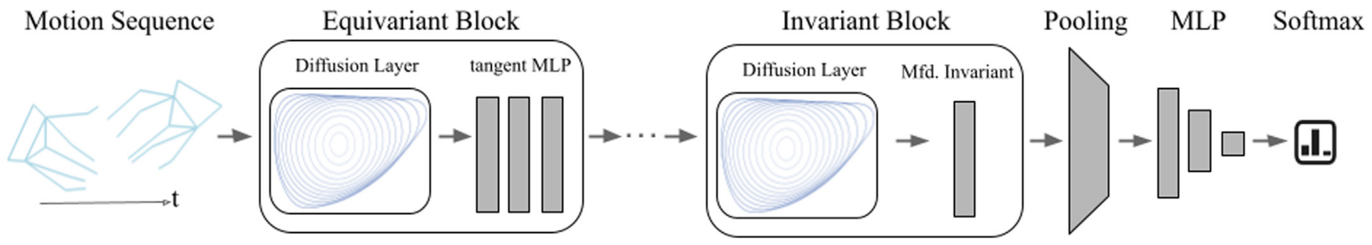

Fig. 9. Schematic of the proposed manifold GCN architecture.

The dataset featured a few interesting challenges. Its rather small size would give most neural networks a tough time learning meaningful properties while avoiding overfitting to the exact input. To combat the aforementioned issue, we designed our LSTM to be relatively small in size, only including one hidden layer of 128 neurons. Additionally, the provided dataset was heavily imbalanced; a weighted cross-entropy loss criterion, whose weights reflected this imbalance, was used in the training loop. The possibility of using focal loss [23] was investigated, but no noticeable improvement during training was observed.

The coordinates of the data were centered around (0, 0, 0) and normalized to lay within the range $[-1, 1]$, keeping the aspect ratio intact. Xavier initialization [24] was used to initialize the trainable parameters of the network and the Adam optimizer with a learning rate of 0.001 was used in the training process. The data was not fed in batches into the network. We experimented using batches and padding the samples to include the same number of frames but, probably due to the vastly different number of frames between each sample, the results were significantly worse.

5.6.2. *Implementation details*

The training took place for just under 2 h on our NVIDIA RTX™ 2060 SUPER GPU with 8 GB of video memory, over 3000 epochs (the dataset was kept loaded in RAM). Early results were promising, regularly managing higher than 50% accuracy on both the training and test datasets. The test dataset accuracy, specifically, was closely monitored throughout the training process. With no regularization means (other than the small network size), we had to ensure that the quick drop in training loss and increase in the training set accuracy was not a product of overfitting and that the accuracy of the test set remained close to that of the training set.

5.7. *Group 5: SE(3)-equivariant graph convolutional network*

5.7.1. *Method description*

Group 5 interpreted the hand motions as sequences of point clouds in three-dimensional space. For such data, methods from the field of geometric deep learning have delivered excellent results. Indeed, through the built-in invariance/equivariance under the symmetries of the data, these approaches can learn desired relationships very effectively. Since time series naturally define neighbor relationships, we utilize the power of graph neural networks. The code to reproduce our experiments can be found online.[10]

5.7.2. *Feature design*

We describe the molding motions as sequences of anatomically corresponding landmarks, i.e., labeled points in three-dimensional Euclidean space, as opposed to the common deep learning approach of viewing vectors as collections of independent, one-dimensional features. Taking this viewpoint allows methods from geometric deep learning to be invariant under (three-dimensional) rigid motions, an invariance that the classification function we want to learn should also possess.

[10] GitHub-Link-group5-SE(3)-GCN.

5.7.3. *Network architecture*

An ensemble of ten manifold graph convolutional neural networks performs our prediction. This architecture for graph learning tasks was introduced in [25]; it is particularly suited to exploit the symmetries of the data space. Our manifold GCN receives 28 channels, each operating on a different 3D landmark, and consists of two types of blocks: an “equivariant block” comprising a (convolutional) diffusion layer with sigmoid-type activation followed by a node-wise tangent multilayer perceptron (MLP), and an “invariant block” that combines a diffusion layer with a node-wise manifold invariant layer. Other than permutation invariant networks for point clouds (e.g., deep sets), we employ geometric fully connected layers on the vector-valued channels to exploit the landmark correspondence. Our architecture stacks multiple equivariant blocks before a single invariant one. To obtain a sequence-level output, we then perform a flat pooling, viz. a concatenation of mean and max pooling, and feed the result to a final (vanilla) MLP. Eventually, the softmax function is employed to map the model output to class probabilities. Fig. 9 illustrates the proposed architecture.

With the chosen Euclidean features, the network is invariant under joint rigid motions of a sequence, i.e., when the same rigid motion is applied to each and every frame. This property leads to a reduced number of parameters, which helps cope with the small amount of training data. The final prediction is obtained from ten of these models by taking the geometric median [26] of their predicted class probabilities based on the Fisher–Rao distance [27] and choosing the most likely class.

Through a hyper-parameter search, we found that the following configuration provides the best performance: We only use the invariant block with a diffusion layer that performs one Euler step; the MLP has three layers mapping from 28 to 14 to 7 dimensions. The resulting network has only about 3000 trainable parameters; we believe that this “slim” network performs best because it is less prone to overfitting the small training set.

5.7.4. *Training*

We trained each of the ten models with RMSProp for 300 epochs on a different 3:1 training–validation split of the full training set; the learning rate was 0.001 and the batch size was one. We employed the common weighted cross-entropy loss, with the inverse number of training samples of each class as the class weight. The final model was selected among those with 100% training accuracy as the first with the highest validation score.

6. Results

Table 1 shows the total accuracy per method over the four cross-validation folds created for this challenge. We can see that group 3 DET-ACTIONS: performed the best with a total accuracy of 91.67%. The bar charts in Fig. 10 show the per-class precision, recall, and F1 scores of all the methods.

Looking at the bar charts in Fig. 10, we can see that most issues are made with the highly similar motions ‘Smoothing’ and ‘Raising’ and in the recall of the ‘MakingHole’ class. The retrieval issues in the

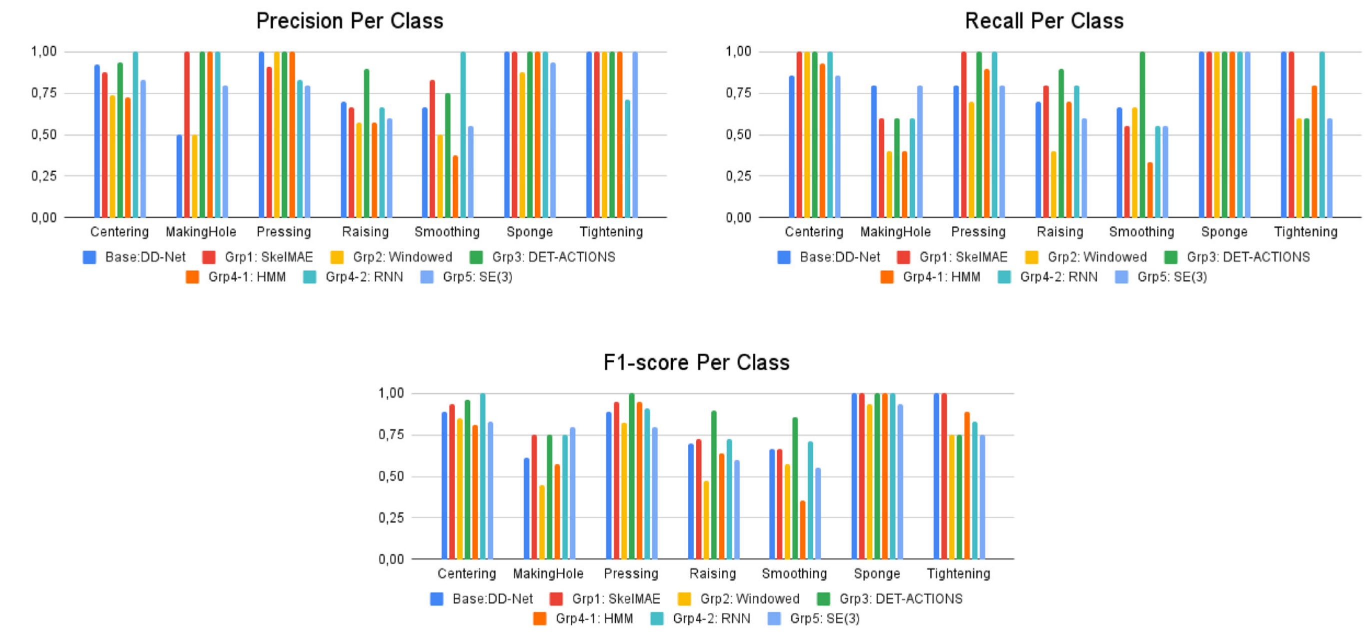

Fig. 10. Performance metrics per motion class.

Table 1
Total accuracy per group over all the folds.

Method	Accuracy
DD-Net	0.8167
SkelMAE	0.8667
Windowed Multi View	0.7167
DET-ACTIONS	0.9167
HMM-based approach	0.75
RNN-based approach	0.8677
SE(3)-equivariant GCN	0.75

Table 2
Amount of parameters and training time in seconds for a singular fold.

Method	Params	Training time
DD-Net	27.536.263	1.210
SkelMAE	27.767.693	3.180
Windowed Multi View	213.308	817
DET-ACTIONS	190.036	1800
HMM approach	2450	804
RNN approach	220.935	6.600
SE(3)-equivariant GCN	3.003	503

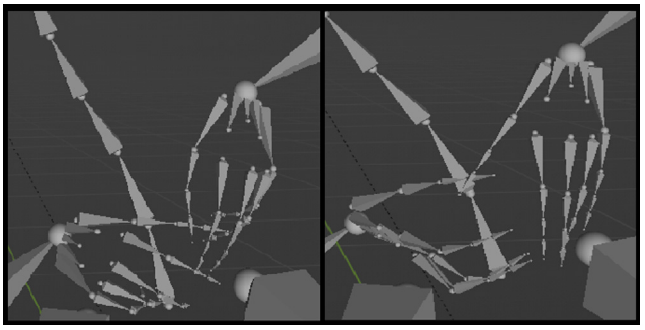

Fig. 11. Static picture of the Smoothing motion (left) where the potter uses his phalanges and the Raising motion (right) where the potter uses the tips of his fingers.

'Smoothing' and 'Raising' classes can be understood by looking at these motions. These motions are highly similar due to the fact that the potter's left hand makes the same motion, while his right hand makes a similar upward motion in both classes. The difference between these two motions is in how the potter's right hand is angled. See Fig. 11 for a better visualization between the two hand motions.

As mentioned before, the recall rate of the 'MakingHole' motion class is struggling. This can be explained due to the cross-validation folds that we have created, in combination with the underrepresentation of the class itself. The 'MakingHole' class has a total of 5 motions captured; 3 of those 5 motions are in the test set for fold 2. This meant that the retrieval techniques only had 2 motions of the class 'MakingHole' to train on for this fold. The 'MakingHole' and 'Centering' dynamic motions both begin with a downward motion from the right hand, which might clarify the confusion of the methods. They also both use the left hand to stabilize and centralize the clay. However, in the 'MakingHole' motion class, the right hand uses the index finger and middle finger to create a hole at the end of the motion instead of them resting on the clay.

Table 2 shows the total number of parameters in the network, and the required time for training in seconds for all methods. We would like to note that for this challenge, we did not tell our participants that we would evaluate them based on their performance in terms of speed. This means that many of our methods are not optimized to perform on this metric. Furthermore, each method has been trained on its own computer specifications. The training specifications can be found in each group's specific method description in Section 5.

7. Discussion

The evaluation outcomes provide insights that state-of-the-art techniques can indeed provide promising scores given the limited data size, large variation in frame lengths, and high detail of the motions. Given the small amount of time available for the contest, we can say that these results are exceptional. We have seen many different methods and network approaches, namely: Convolutional Neural Networks, Recurrent

Neural Networks, the Hidden Markov Models, and Space–Time Graph Convolutional Networks.

The issue that the gesture class ‘MakingHole’ got predicted as the motion class ‘Centering’ by all methods could derive from the limited test and train data. The gesture ‘MakingHole’ exists a total of 5 times in the entire train and test set, while centering exists a total of 16 times. Creating a higher saturation on the ‘MakingHole’ motion classes could have solved this issue. We do however believe that the issues in the ‘Smoothing’ and ‘Raising’ classes do derive because they are highly similar in both the left and right hand.

8. Conclusion

In this paper, we have presented a novel dynamic hand gesture dataset and reported and analyzed the results of the submissions for the SHREC 2024 track on Recognition Of Dynamic Hand Motions Molding Clay. The evaluation of the proposed methods shows that there are many different methods and network approaches that show high results on classifying hand movements using both hands with precise and highly similar motions on a small training set. Due to the small test set, we believe that it is necessary to continue the evaluation on a larger test set to get a more accurate evaluation of the methods. A possible future research direction could be to improve on this dataset, both increasing the number of gestures and the amount of gesture classes. We can achieve this by recording a higher number of subjects and by creating more pottery. This might give us a better insight on how to solve the problem of highly similar motions. We do believe that to bring the field of hand recognition further. There should be a focus on hand gesture recognition for highly detailed, highly similar motions using both hands.

CRediT authorship contribution statement

Ben Veldhuijzen: Writing – review & editing, Writing – original draft, Software, Project administration, Methodology, Investigation, Data curation. **Remco C. Veltkamp:** Supervision. **Omar Ikne:** Software, Methodology. **Benjamin Allaert:** Software, Methodology. **Hazem Wannous:** Software, Methodology. **Marco Emporio:** Software, Methodology. **Andrea Giachetti:** Software, Methodology. **Joseph J. LaViola Jr.:** Software, Methodology. **Ruiwen He:** Software, Methodology. **Halim Benhabiles:** Software, Methodology. **Adnane Cabani:** Software, Methodology. **Anthony Fleury:** Software, Methodology. **Karim Hammoudi:** Software, Methodology. **Konstantinos Gavalas:** Software, Methodology. **Christoforos Vlachos:** Software, Methodology. **Athanasios Papanikolaou:** Software, Methodology. **Ioannis Romanelis:** Software, Methodology. **Vlassis Fotis:** Software, Methodology. **Gerasimos Arvanitis:** Software, Methodology. **Konstantinos Moustakas:** Software, Methodology, Conceptualization. **Martin Hanik:** Software, Methodology. **Esfandiar Nava-Yazdani:** Software, Methodology. **Christoph von Tycowicz:** Software, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Me and the other authors all have agreed to make all data including code used available on GitHub.

Acknowledgments

Capturing the hand movements of modeling clay was initiated by artist Isabel Ferrand, who together with Remco Veltkamp received the PUG prize from Provinciaals Utrechts Genootschap Van Kunsten En Wetenschappen. Thanks to potter Kees Agterberg for having his hand movements recorded. Thanks to Tariq Bakhtali for helping in the motion capture sessions.

References

[1] Quentin De Smedt J-PV, et al. SHREC’17 track: 3D hand gesture recognition using a depth and skeletal dataset. 2017.

[2] Caputo FM, Burato S, Pavan G, Voillemin T, Wannous H, Vandeborre J-P, Maghoumi M, Taranta EM, Razmjoo A, Laviola JJ, Manganaro F, Pini S, Borghi G, Vezzani R, Cucchiara R, Nguyen H, Tran M-T, Giachetti A. SHREC 2019 track: Online gesture recognition. 2019, URL: https://api.semanticscholar.org/CorpusID:208524001.

[3] De Smedt Q, Wannous H, Vandeborre J-P. Heterogeneous hand gesture recognition using 3D dynamic skeletal data. Comput Vis Image Underst 2019;181:60–72. http://dx.doi.org/10.1016/j.cviu.2019.01.008, URL: https://www.sciencedirect.com/science/article/pii/S1077314219300153.

[4] Caputo A, Giachetti A, Soso S, Pintani D, D’Eusanio A, Pini S, Borghi G, Simoni A, Vezzani R, Cucchiara R, Ranieri A, Giannini F, Lupinetti K, Monti M, Maghoumi M, LaViola Jr JJ, Le M-Q, Nguyen H-D, Tran M-T. SHREC 2021: Skeleton-based hand gesture recognition in the wild. Comput Graph 2021;99:201–11.

[5] Emporio M, Caputo A, Giachetti A, Cristani M, Borghi G, D’Eusanio A, Le M-Q, Nguyen H-D, Tran M-T, Ambellan F, Hanik M, Nava-Yazdani E, von Tycowicz C. SHREC 2022 track on online detection of heterogeneous gestures. Comput Graph 2022;107:241–51. http://dx.doi.org/10.1016/j.cag.2022.07.015, URL: https://www.sciencedirect.com/science/article/pii/S0097849322001388.

[6] Fan Yang, Sakriani Sakti, Yang Wu, Satoshi Nakamura. Make skeleton-based action recognition model smaller, faster and better. 2019.

[7] Ikne B, Allaert H, Wannous. Skeleton-based self-supervised feature extraction for improved dynamic hand gesture recognition. In: Accepted at IEEE FG 2024. 2024.

[8] Yan H, Liu Y, Wei Y, Li Z, Li G, Lin L. SkeletonMAE: Graph-based masked autoencoder for skeleton sequence pre-training. 2023, arXiv preprint arXiv:2307.08476.

[9] He K, Chen X, Xie S, Li Y, Dollár P, Girshick R. Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 16000–9.

[10] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al. An image is worth 16x16 words: Transformers for image recognition at scale. 2020, arXiv preprint arXiv:2010.11929.

[11] Tancik M, Srinivasan P, Mildenhall B, Fridovich-Keil S, Raghavan N, Singhal U, Ramamoorthi R, Barron J, Ng R. Fourier features let networks learn high frequency functions in low dimensional domains. Adv Neural Inf Process Syst 2020;33:7537–47.

[12] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Adv Neural Inf Process Syst 2017;30.

[13] Yan S, Xiong Y, Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 32, 2018.

[14] Loshchilov I, Hutter F. SGDR: Stochastic gradient descent with warm restarts. 2017, arXiv:1608.03983.

[15] Cunico F, Girella F, Avogaro A, Emporio M, Giachetti A, Cristani M. OO-dMVMT: A deep multi-view multi-task classification framework for real-time 3D hand gesture classification and segmentation. 2023, arXiv:2304.05956.

[16] Dill A, Ge S, Kang E, Li C-L, Poczos B. Learned interpolation for 3D generation. 2020, arXiv:1912.10787.

[17] Hendriks J, Dumond P. Exploring the relationship between preprocessing and hyperparameter tuning for vibration-based machine fault diagnosis using CNNs. Vibration 2021;4(2):284–309.

[18] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 2818–26.

[19] Rabiner L. A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 1989;77(2):257–86. http://dx.doi.org/10.1109/5.18626.

[20] Papadopoulos GT, Axenopoulos A, Daras P. Real-time skeleton-tracking-based human action recognition using kinect data. In: Gurrin C, Hopfgartner F, Hurst W, Johansen H, Lee H, O'Connor N, editors. MultiMedia modeling. Cham: Springer International Publishing; 2014, p. 473–83. http://dx.doi.org/10.1007/978-3-319-04114-8_40.

[21] Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B Stat Methodol 1977;39(1):1–22. http://dx.doi.org/10.1111/j.2517-6161.1977.tb01600.x.

[22] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: a next-generation hyperparameter optimization framework. 2019, http://dx.doi.org/10.48550/arXiv.1907.10902, arXiv:1907.10902.

[23] Lin T-Y, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 2980–8.

[24] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings; 2010, p. 249–56.

[25] Hanik M, Steidl G, von Tycowicz C. Manifold GCN: Diffusion-based convolutional neural network for manifold-valued graphs. 2024, arXiv preprint arXiv:2401.14381.

[26] Fletcher PT, Venkatasubramanian S, Joshi S. The geometric median on Riemannian manifolds with application to robust atlas estimation. NeuroImage 2009;45(1):S143–52.

[27] Srivastava A, Jermyn I, Joshi S. Riemannian analysis of probability density functions with applications in vision. In: 2007 IEEE conference on computer vision and pattern recognition. IEEE; 2007, p. 1–8.