# Creation of the SHREC track: Recognition Of Dynamic Hand Motions Molding Clay

Master Thesis Computing Science

**Ben Veldhuijzen**

Student Number: 5981778
Supervisor and first examiner: Remco Veltkamp
Second examiner: Michael Behrisch

Graduate School of Natural Sciences
Utrecht University
The Netherlands

## Preface

This is a master thesis created by Ben Veldhuijzen, supervised by Remco Veltkamp with Michael Behrisch as my 2nd examiner. This master thesis is made to meet the graduation requirements Of the Computing Science master course at Utrecht University.

In this master thesis I have learned to work with for me new technologies like the Vicon Shogun System in The Motion Capture and Virtual Reality Lab of Utrecht University. I Learned more about blender and how to create working neural networks. I also got more insight on 3D Shape Retrieval techniques, The SHREC challenges, and how to retrieve useful information from hand motions for analysis to create your own benchmark.

Finally, I want to thank my supervisor Remco Veltkamp for his guidance and help during this master thesis. I would also like to thank the participants of the SHREC 2024 track: Recognition Of Dynamic Hand Motions Molding Clay, for participating. I would also like to extend my sincere appreciation to Kees Agterberg for being my model for motion capture. Your skills in pottery are truly amazing. Organizing the SHREC challenge was enjoyable, and the skills and knowledge I gained during this research period have significantly enhanced my understanding of the shape retrieval field.

## Contents

**Abstract**

Gesture recognition is a tool that enables intuitive Human Computer interactions (HCI) for techniques and applications in the fields of Extended Reality (XR). In this master thesis we present the steps we took to create a new SHREC Track for the hand gesture category. Namely the track Recognition Of Dynamic Hand Motions Molding Clay. The task is the recognition of 7 motion classes given their spatial coordinates in a frame by frame motion. Our novel dataset has been captured using a Vicon system and has been pre-processed using blender and Vicon Shogun Post. This paper presents the creation method of the novel dataset used in our challenge combined with the methodology and results of the baseline method.

## 1 Introduction

Skeleton based action recognition is becoming more widely used in Human-computer Interaction (HCI) due to its unique and intuitive way of controlling applications in various fields. This research project was designed around creating a new shape retrieval challenge (SHREC) [1] for the hand gestures category. SHREC general objective is to create 3D Shape Retrieval Challenge to evaluate 3D shape retrieval systems in hope of finding new ways on how we can improve these retrieval systems.

In this thesis we will address the challenges of creating a dataset of hand motions molding clay captured using a Vicon system [2]. The Vicon system uses 14 Vantage Cameras that will track reflective markers to create a desirable human skeleton. We explain all steps involved to create the skeletal coordinate system from the Vicon data in section 4. We also present the steps we took to create a new SHREC Track for the hand gesture category. Namely the track Recognition Of Dynamic Hand Motions Molding Clay in section 5. We also explain in detail the creation and evaluation of our skeleton based recognition system on our novel dataset.

A well functioning Skeleton based recognition system needs to work efficient and be adaptable to multiple forms of data. Our retrieval system will run using only the global coordinate system of the skeletal hand structure. In order for a retrieval system to function well on different types of data, it should be location-viewpoint invariant, while containing global motion information as well as a feature for the hand shape. We trained a convolutional neural network (CNN) that makes use of a two-scale global motion feature and a highly efficient Joint collection Distances feature (JCD) [3] on a novel dataset created by a Vicon system.

The rest of the paper is organized as follows: Section 2 presents the related work, section 4 presents our novel dataset and the collection steps used, section 5 goes into details of the proposed SHREC challenge, section 6 presents our implementation, and section 7 presents the results and evaluation of our retrieval system on different configuration settings.

## 2 Related Work

### 2.1 SHREC

The main objective of the 3D Shape Retrieval Contest (SHREC) [4] is to evaluate the effectiveness of 3D-shape retrieval algorithms. They provide resources in the form of challenges and benchmarks to compare and evaluate existing or new 3D retrieval methods for the last 18 years. Participants can join these challenge to compete and work together on a track. Where the final goal is to create a collective research paper with the results and methods of all participants. This paper will be published in the proceedings of the Eurographics Workshop 3D Object Retrieval. Every year there is a workshop where at least one author per track can register for a time slot to present their results.

As of writing this Thesis there have been a total of 103 SHREC challenges held, 5 of which have been on the hand gesture track which will be discussed in section 2.2.1.

## 2.2 Hand Gesture Recognition

Hand gesture recognition is becoming more and more important due to the rise in Augmented and virtual reality [5]. Hand gesture recognition tries to classify hand gestures. There are 2 categories of hand gesture retrieval: static and dynamic. Static hand gesture retrieval focuses on hand gestures from a single still image while dynamic gesture retrieval sequence of data. This sequence of data can be a sequence of images of the hand, frame by frame skeletal structural data or just the global coordinate system of the skeleton.

Recent advancements in hand gesture recognition [6] [7] allows for real time generation of hand skeletons and even gesture recognition. The current leading technique shown by Andrea Giachetti et al. [8] [9]. is to create a hand skeletal coordinate structure and utilize features on these coordinates.

Hand gesture recognition can be used to control certain operations in human-machine applications or human–computer interaction. They can replace physical hardware which for some of these applications might be undesirable. It is shown that gestures can be recognized accurately and the execution of gesture commands can be done on an interactive level [10].

### 2.2.1 Previous Hand Gesture Tracks

Hand gesture recognition has been a consistent research field where several benchmarks have been created over the years. In order for us to create our own hand gesture track we looked at the previously organized hand gesture SHREC tracks.

The SHREC'17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset [11], is the most cited track and the most used benchmark of the 5 SHREC tracks. This track features dynamic gestures that are created with the HCI viewpoint in mind, meaning that they are mostly gestures that can be used for interactive applications.

The SHREC'19 track on online gesture detection [8] challenged methods with a new task by creating gesture sequences. These gesture sequences where created to challenge gesture recognition systems to reduce the amount of false positives in their retrievals. This benchmark consisted of 5 gesture classes where the gestures where performed by either their index fingertip or their hand palm. In the SHREC 2020 Track called recognition of sequences of gestures from fingers' trajectories [9] they improved on their previous 2019 track by increasing the amount of gesture classes from to 13 while also including the entire shape of the hand.

The SHREC'21 track Skeleton-based Hand Gesture Recognition in the Wild [12] was created to test more complex gestures in the form of XR interactions. The SHREC 2022 track on online detection of heterogeneous gestures [13] was the continuation of their previous 2021 track, the authors removed ambiguous classes and avoided annotation issues affecting the previous SHREC 2021 benchmark.

Looking at these previous hand gesture tracks we notice certain important aspects on their benchmarks. First of all we notice that skeletal coordinate data is the preferred way of storing hand motions. Secondly we see that in order to create a good hand gesture track, your track should either challenge retrieval systems in a new way, or provide a new benchmark that exists of more detailed motions increasing the difficulty of retrieval.

### 2.2.2 Gesture Recognition Systems

As mentioned in section 2.2.1 hand recognition has been a consistent research field, in this section we will look into a few state of the art recognition systems that we have considered as potential candidates to adjust, or just utilize certain features from.

Graph Neural Networks (GNN's) are a highly-scalable class of models that can "learn" on graph structured data. The idea about a GNN is that: If you can define a correct graph the GNN can retrieve features on this graph. The Graph is denoted as $G(V, E, X_V, X_E)$ where V stands for Vertices, E stands for edges and $X_V$ and $X_E$ for attributes in the vertices and edges respectively. A special type of graph neural network is the Spatio-Temporal Graph. The Spatio-Temporal graph tries to deal with time varying features, in this case its feasible to create a graph where the features $X_V$ and $X_E$ are sampled over time. Resulting in the creation of a sequential graph that reflects the real world.

In the case of ST-GCN [14–16] they used the idea of this Spatio-Temporal Graph in combination with a convolutional graph networks (CNNs) to create a ST-GCN. The ST-GCN has demonstrated remarkable capabilities in learning temporal relationships. It enables identification of complex patterns in sequential data by creating a graph network that connects the body joints based on natural connections in the human body. Afterwards Hazim wanous et al, [15] used these principles to create a ST-GCN based on the skeleton of the human hand. For our proposed dataset (more in section 4) the proposed graph could be split into a separate graph for each hand, seen in figure 1.

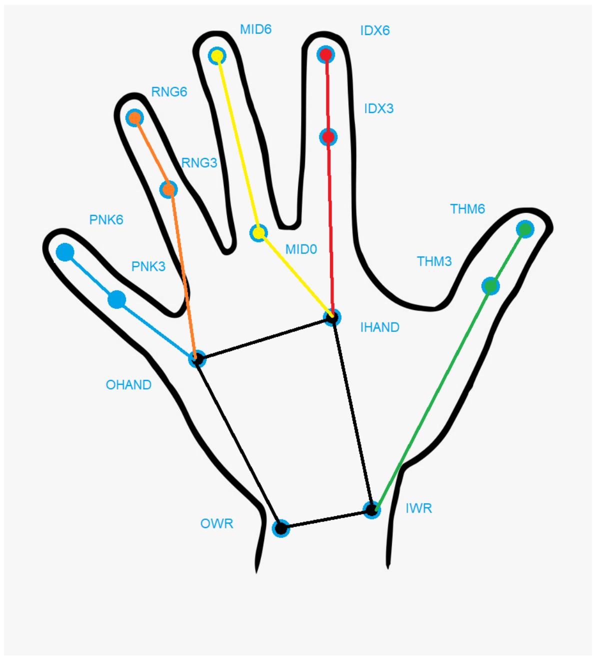


Figure 1: Proposed graph connectivity on a singular hand based on our marker locations.

Media pipe is a customizable machine learning solutions framework developed by Google, Media pipe uses pre-trained machine learning solutions such as object detection, face detection and also hand recognition. The biggest strength of media pipe is it's pre-trained models by google. For example you can detect hand key points from your camera in real time. This key points detection of the hand will create the hand skeleton from your video data in real time.

There are many online papers [17] [18] [19] and video tutorials on how to use media pipe for hand gesture recognition, some of these tutorials even show how to train media pipe to learn new gestures. The issue however is that media pipe is a gesture recognition system that is trained for video data and recognition in real time. Since in our challenge we will not be releasing video data but instead the skeletal coordinate data, there is no real benefit in using google's pre-trained models to detect hand key points.

For our retrieval system we will be using the principle of a feature from Fan Yang et al. [3] the Joint collection feature (JCD). We chose this feature due to it's high performance on the SHREC-17 hand gesture dataset from Quentin De Smedt et al [11], and the fact that it's a lightweight feature that is both location viewpoint invariant and easily implementable for any network. These traits make the JCD feature the perfect candidate for our baseline method.

## 2.3 Neural Networks

Neural networks also known as artificial neural networks (ANNs) have shown great results in modeling human motions recently. ANNs are comprised of node layers, containing input output and hidden layers. each node connects to another node with an associated weight and threshold. The main advantage of using deep learning approaches is their flexibility in designing the architecture [20]

For human motion analysis there are tHree special types of neural networks that are interesting. These are the recurrent neural network (RNNs), The graph neural networks that where mentioned in section 2.2.2, and the convolutional neural networks (CNNs).

RNNs make use of sequential data or time series data. These neural networks have utilize their training data to learn rather than see every input and output as independent. Meaning that the prior inputs of a sequence are dependent to the output of the RNNs.
CNNs are a type of feed-forward neural network majorly used for image recognition, pattern recognition and computer vision [15]. The biggest advantage of using CNNs is that you don't need to do a lot of pre-processing on your data. Another advantage of the CNN is that it uses convolutions by leveraging principles from linear algebra instead of doing matrix multiplications. This speeds up the computation time drastically.
For this project we decided to use CNNs. We value the advantages of the CNN on spatial data as well as the performance increase we get from the convolution method.

### 2.4 Motion Capture Systems to record the data

There have been multiple studies that showed that Motion Capture system can be used as an accurate and robust option to create benchmarks for Human Movement Analysis and gesture recognition [21] [22]. These papers have shown their steps taken to create a professional benchmark from scratch, as well as assuring the quality of the data. This has been our guide to ensure that we are ready for all the steps to create our own benchmark.

The Vicon Motion System is the system we decided to utilize for the creation of our benchmark. The vicon system uses 14 Vantage Cameras to capture lightweight markers placed on various locations on the subject in 3D space. Jonathan et al. [23] showed in his work that the Vicon System can be used for accurate real-time hand posture tracking.

Our project used the Motion Capture and Virtual Reality Lab of Utrecht University to record our hand motion data, processed and later used in the benchmark for our SHREC track. See section 4 where we explain in detail on how we used the the Motion Capture Lab at Utrecht University to create our benchmark.

## 3 Study Goal

Our goal is to propose, implement and evaluate a new SHREC track for the hand gesture category. In order to create this track we will create a novel benchmark that will be the foundation of our new SHREC hand gesture track. This benchmark needs to either challenge retrieval techniques in a new way, or provide a continuation on an already existing track by improving on their benchmark (section 2.2.1).

**The main research goal is as follows:** *Propose, participate and evaluate our SHREC track on Recognition Of Dynamic Hand Motions Molding Clay.*

In order to reach this goal we need to complete the following sub-tasks.

1. Define the objective, task and evaluation for the participants.
2. Create the benchmark for the track.
3. Find Participants for the track.
4. Participate with a baseline retrieval method.
5. Evaluate all submitted results.
6. Follow the SHREC procedure to submit the track.

After we have finished these tasks we can finally give an answer to our research question: "*How do we propose, participate and evaluate our SHREC hand gesture track on Recognition Of Dynamic Hand Motions Molding Clay?*".

The procedure to submit a SHREC track is as follows: First the organizer proposes their track by describing their task, objective and evaluation method. They also send in their benchmark together with a rough expected amount of participants. After acceptance of their proposal, the track will be listed on the SHREC web-page. You can find our SHREC web-page at the footnote[1].
On this page participants can register by contacting the organizers of their desired to join SHREC track. These tracks each have their own time schedule for the participants, however they do have certain deadlines to meet. These deadlines are for the joint research paper submissions as well as the deadlines for their peer reviewed revisions on this paper.
The steps that we took to create our hand gesture track can be read in section 5. You can read our current version of the SHREC 2024: Recognition Of Dynamic Hand Motions Molding Clay paper, that is currently in it's second stage of peer-review at Appendix A.

## 4 Data Set

Our goal is to create a dataset of hand motions that are highly similar and contain precise but small motions using both hands. We hope to challenge participants in a new way by creating this novel benchmark. Our benchmark focuses on retrieval of hand movements on mold-able objects. This benchmark will be the first SHREC benchmark to provide hand motions using both hands while the motions themselves are precise and highly similar.

The previously mentioned benchmarks in section 2.2.1 have certain weaknesses that we would like to overcome during this track. The first weakness that we found is that they are highly focused on their global motion due to the fact of them being applicable for HCI. For example the benchmark of the SHREC-17 track from Quentin De Smedt et al. [11] has 11 out of 14 classes that simply can be solved by looking at the global motion regardless of the hand shape. These classes all fall under the category rotations or swipes.

The second issue we found is that the benchmarks do not contain similar or highly detailed motion classes. We are interested to see if hand recognition systems can differentiate between highly similar motions. In our case of molding clay all actions performed by the potter are

[1]https://www.shrec.net/SHREC-2024-hand-motion/

precise, similar motions.

Lastly all these previously mentioned benchmarks consist of data using a singular hand. Some gestures might be performed using both hands. We will provide a benchmark that tackles these found weaknesses, this benchmark will be the foundation for a new SHREC track to challenge 3D hand gesture recognition systems.

## 4.1 Data Collection

Hand-motion data has been captured using many different technologies so far. Previous papers have used a Leap Motion Device, Hololens 2 finger tracking headsets, other VR/AR technologies, and short range depth cameras. This data is later processed into skeletal data for hand-movements.

We recorded hand motions of an experienced potter who sculpted the same pot with and without clay. For this project we are using the motion capture lab at Utrecht University [24]. 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 reflective Soft-base markers on the potter's hands and body, see figure 2. This way we can do full body and hand tracking in real time while the potter is at work. After recording we use post-processing software namely Blender and Vicon Shogun Post to extract the coordinates information of the hand skeleton on a frame by frame basis.

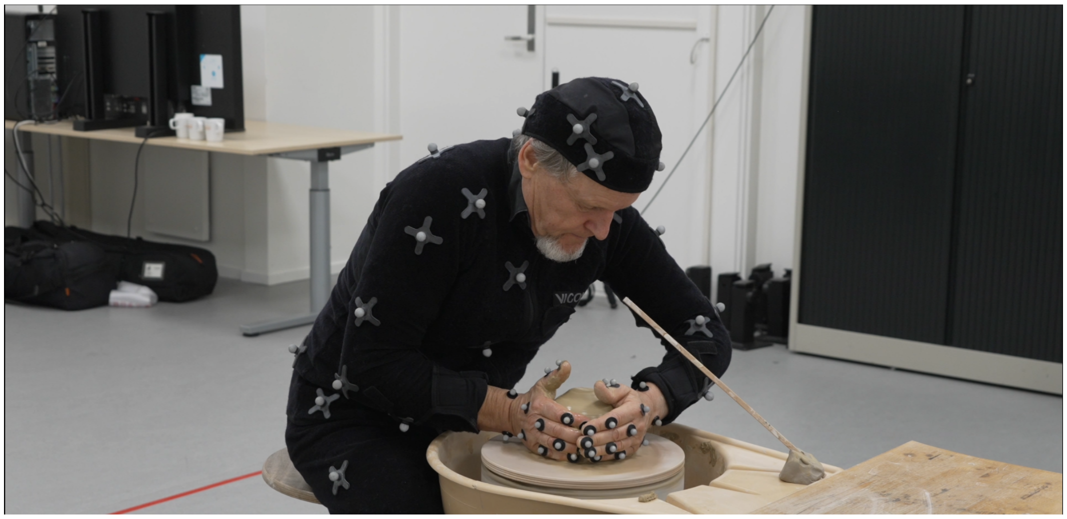


Figure 2: Recording of Kees Agterberg using clay to create a vase

### 4.1.1 Motion Capture Lab

The Motion Capture and Virtual Reality Lab of Utrecht University [24] is one of the only Motion capture lab's in the Netherlands at this scale that allows detailed captures of multiple actors in a multi-modal manner. The MoCapLab includes captures of facial expressions, body movements, finger tracking and audio which can be captured in a fully immersive VR experience.

The motion capture lab has a library of 3D Software that is available for use within the lab. Vicon Shogun and Shogun Post, Dynamixyz Grabber and Performer, Unreal Engine, Motion Builder and Maya. For our project are using Vicon Shogun and Vicon Shogun Post.

The Body Motion Capture is done by the Vicon System. This system makes use of 14 Vantage Cameras. These cameras need to be calibrated in a specific way using the Vicon Shogun software [2]. There are some fairly important things to notice before you can use this system to record subjects.

**Calibration of the Cameras**: Before you can calibrate the cameras you have to wait between 45 minutes to 1 hour for the cameras to warm up. The Vantage camera has an on-board sensors that measures the camera position and temperature to ensure optimal performance. The status of these cameras is visible in the Vicon Shogun software. After the cameras are finished warming up. Simply go the Camera Calibration tab on the top right and press the Activate Video Calibration button. Now we have to complete the following steps: The first step is doing a wave task by waving to all the cameras using a Calibration Wand. After that put the wand in the middle of the room to set the Origin and the Floor Plane. Now we remove the Calibration wand from the scene, setup the following items needed for the recording and use the Mask option to remove any unwanted reflections seen by the motion capture cameras.

**Creating the subject** We create a subject for recording. To achieve accuracte hand motion capture we use the FrontWaist10Fingers Marker set. This set will create a skeleton that contains 10 fingers. For this project we used Soft-base markers in combination with the full-body suit. The Soft-base markers need to be placed in precise locations on the subjects hand in order to track the hand motions correctly. The location of these markers can be seen in figure 3. We did not use a glove but instead used tape to attach all the markers at the correct location. We did this to ensure better stability on the markers when they come into contact with the clay of the potter. Since the force might shift the glove making the captures less accurate.

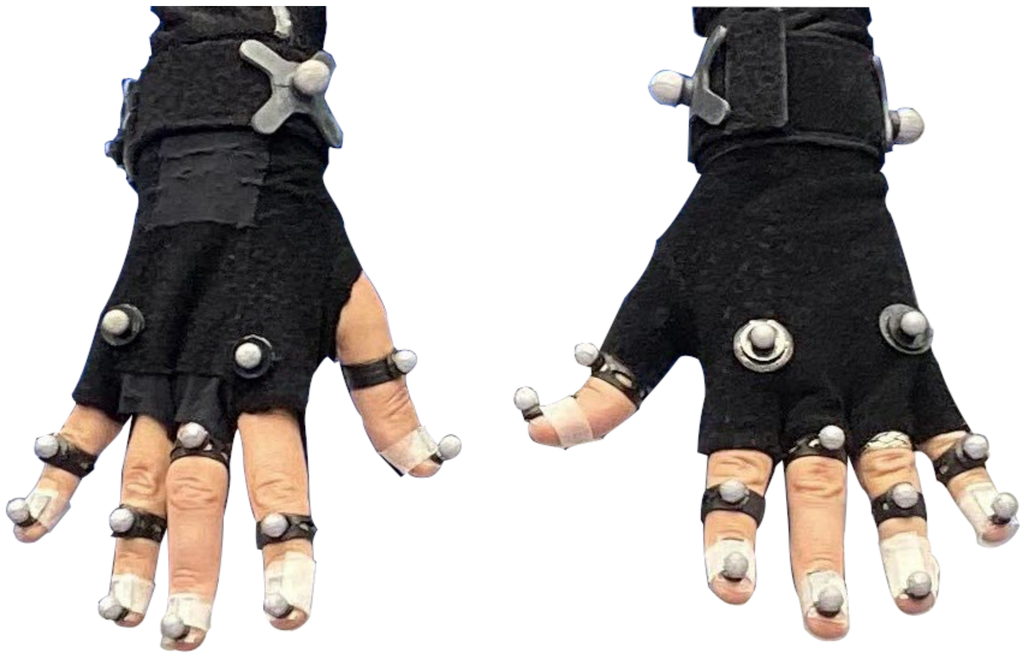


Figure 3: Placement for all the finger markers for the 10 finger marker set as depicted in Vicon documentation [25]

Now that the subject is wearing the full bodysuit and the markers are in place it's time to calibrate and create the subjects skeleton. In the Tracking tab of the Vicon shogun software we choose the FrontWaist10Fingers Template, remove the Prop option and click Create. Now we use the calibration option of the software. Before continuing we do a manual check to see if the skeletal structure looks correct, and when needed we adjusted markers accordingly.

**Calibrating the Subject** After clicking the Accept A-pose button, the Subject for calibration needs to do a Range Of Motions. These include moving your arms and legs. Since we are interested in hand motions we made the subject do a special range of motions to inspect the finger movements. This ROM exists of the following motions for both hands:

1. Make a fist
2. Wiggle all fingers
3. Make the tip of your thumb touch all fingers of the same hand.

After the ROM is completed and we examined the correctness of the motions, we stopped calibrating which will finalize the creation of our subject. We then did a quick check to see if the markers on the hands are labeled correctly, After this inspection we were ready to record.

### 4.1.2 The Recording day of the Experienced Potter

We calibrated the camera's then place all necessary equipment in the correct locations and masked the room to be ready for recordings. As noted above the cameras take about one hour to warm up so this was all done ahead of time. Before we attach the markers to the potters hands we will give him time to create 1 pot without recording him. This is so he can get a feeling for what type of pot he wants to create. The potter created the same pot a total of 4 times, twice without clay and twice with clay. He also created the end of the pot twice without clay using a video for guidance that was recorded beforehand. This gives us a total of 6 recordings.

A recording will be from the start to finish of creating one singular vase, as said before we will make the potter create the exact same vase 6 times. The reasoning for not using clay at the start is to not make the markers filthy or potentially fall off the hand.

At the end of the 4 recordings without clay tried to do recordings with clay. After the first recording we quickly realized that the clay would cover the markers. We anticipated that this could be an issue however we did not realize that it would make the tracking break to this extent. Even the slightest bit of clay would make markers function significantly less or even hide markers completely from the vicon system. This is visible on the recorded data and can been seen on figure 4. This resulted in us losing most of the later data of the first recording using clay. During the second recording we made sure to clean the markers more often with a towel. This resulted in a better recording.

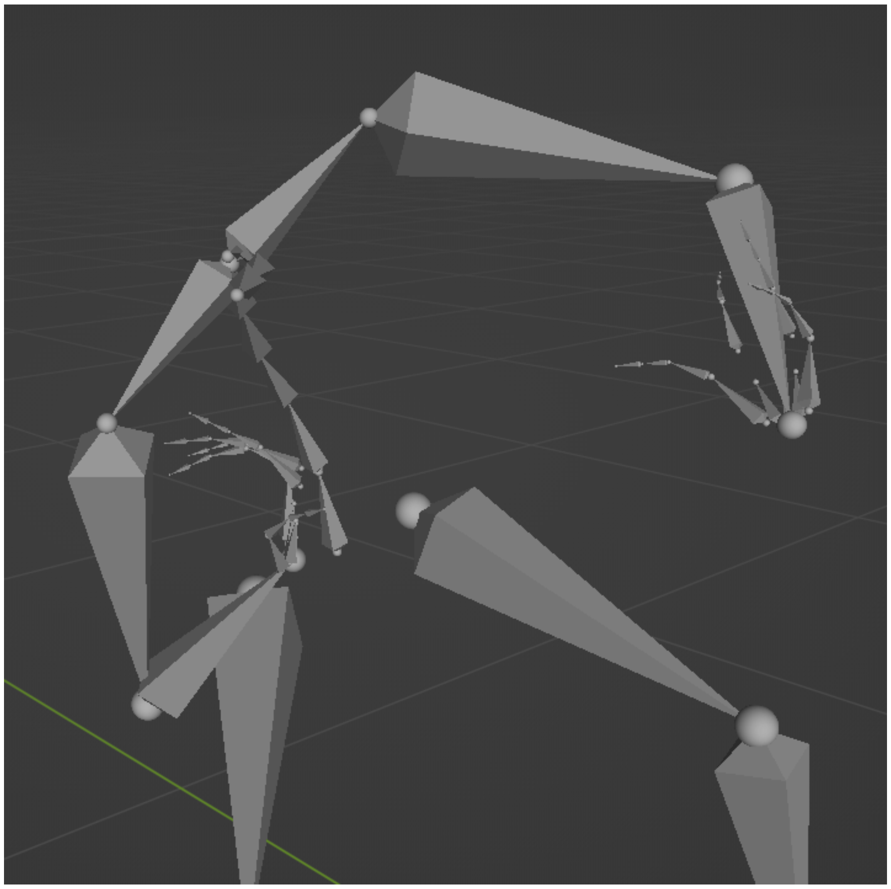


Figure 4: Screenshot of Hand tracking issues due to markers not responding well while being covered with clay

### 4.1.3 Post-Processing the Recordings

After the recording we used Vicon Shogun Post, automated post processing functions to smooth the stuttering from the trackers. We solved finger joint issues and automatically labeled the markers. We exported the MCP file format to FBX (Filmbox) format. We can now open these FBX files using Blender 3.6 and load in the armatures, see figure 5 for an example of an armature in blender.

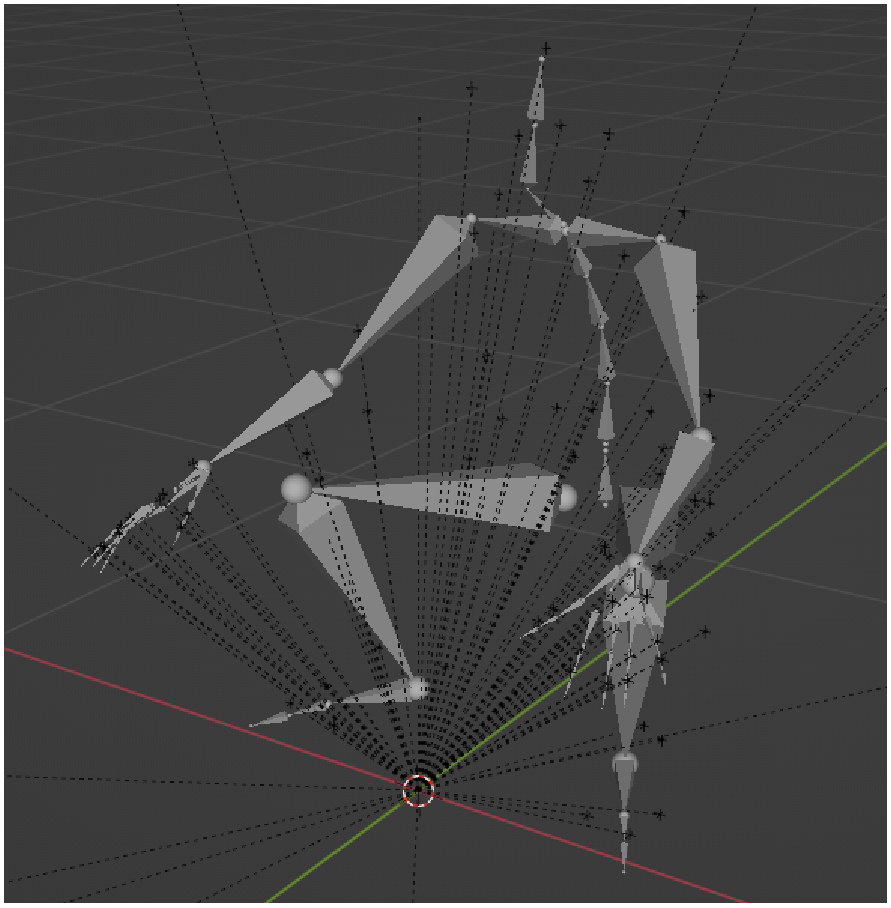
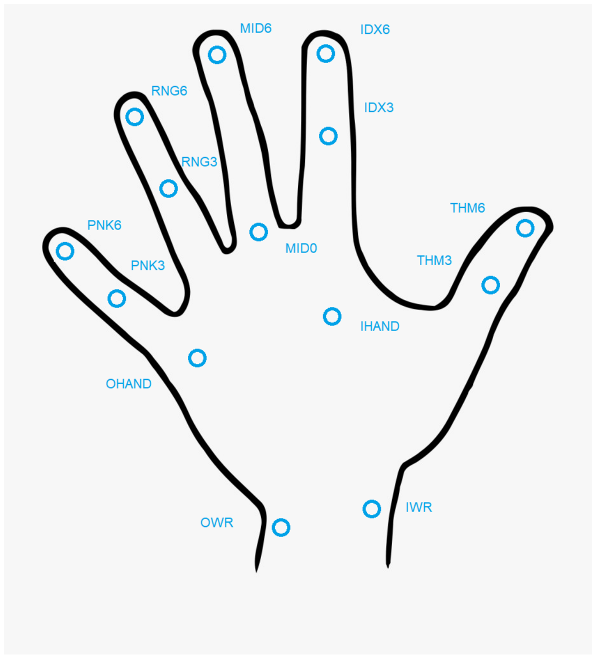


Figure 5: Example of an FBX file opened in Blender 3.6

The markers are as mentioned before automatically labeled by Vicon shogun post. The labeling can be seen in table 1. These labels begin with an L or an R at the front standing for left or right hand. The number at the end of the label is 0 if the marker is at the metacarpal of the finger, 3 if it is at the start of the phalanx of the finger, and 6 if it's at the end of the phalanx.



| | | |
|---|---|---|
| IWR | OWR | IHAND |
| OHAND | THM3 | THM6 |
| IDX3 | IDX6 | MID0 |
| MID6 | RNG3 | RNG6 |
| PNK3 | PNK6 | |

Table 1: Location of markers on the left hand

After our files are converted in FBX format, we open them in blender 3.6 and manually find hand motions by watching the recordings. After seeing a motion we note down the motion class combined with the start and end frame. We then use a custom made blender script that requires as inputs the FBX file combined with the start and end frame to extract the skeletal coordinate system for all frames given in the input. It does this by looking at the XYZ coordinates from the markers and creating a singular line per frame in the following format:

*[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)]*

After continuing this XYZ extraction for every frame on a motion, we acquired a list of XYZ coordinates that will be written in a singular TXT file. We give the TXT file a name based on it's motion class, label it with a number, and put it in the correct folder. As mentioned in section 2.2.1 participants are familiar with this specific skeletal coordinate framework. After extracting all the movements we separated the movements in a 80-20 distribution for the train and test set.

#### 4.1.4 The Finalized Dataset

The novel dataset is composed of 62 motions (split into a 80/20 train/test split), these motions are quite long with frame lengths between 29 frames to 3721 frames averaging around 990 frames. The motions are of high quality, showing highly similar, precise motions using both hands on the subject of molding clay. These frame lengths are highly different from previous SHREC tracks mentioned in section 2.2.1. Since previous SHREC tracks had frame lengths averaging around 30 frames per motion. The data consists of 7 motion classes, these 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
- **Smoothing** the walls
- Using the **sponge** to make the clay more moist.

This dataset is fairly small, however due to the uniqueness of the data recorded we still value this dataset highly. The data has been made available on our SHREC track web-page[2].

## 5 My Hand Gesture SHREC Track

In section 3 we defined the goals for our thesis to hold our own SHREC challenge. Now that we have created a novel benchmark it is time for us to lay the foundation for our shape retrieval challenge. This section will show the steps taken by us to organize the SHREC 2024: Recognition Of Dynamic Hand Motions Molding Clay challenge found at Appendix A. This section will also go into detail on the SHREC procedure to submit the track.

### 5.1 Defining the Track

This contest will focus on trying to recognize different highly similar hand motions from a professional potter. Participants will try to recognize the hand motions from the given skeletal coordinate data of both potters hands. The small size of train set, the motion of two hands simultaneously, and the precise and highly similar motions of the professional potter hands makes it a novel recognition task. This track is created to evaluate the current state of the art gesture recognition systems by challenging them with new highly similar skeletal data using two hands.

The task for the participants is to develop methods that can detect and classify the hand movements based on the classes given in section 4.1.4. Their results should consist of a singular txt file, were a singular row would represent a singular motion. The format of each row should be: The number of motion in the test dataset, followed by the respective motion class. We decided to look at the accuracy per class, combined with the total accuracy over the entire test set for the evaluation method. We will create a confusion matrix to gather insight in the algorithms. The participants where asked to make their algorithm available for download.

[2]https://www.shrec.net/SHREC-2024-hand-motion/

This is to comply with the rules of the Graphics Replicability Stamp Initiative [26].
For the submission we asked the participants the following 4 items:

A **Readme** file containing information on how to compile and run their code, combined with required parameters and dependencies to reproduce their results.
Their **Results file** in the exact format mentioned before.
Their **executable code**, or the link where the code could be downloaded from.
Their **Method description**, consisting of 1-2 pages in Latex format. Containing the Important implementation details of the classification system. We also provided the participants with a link to the Computer and Graphics (C&G) LaTex template[3]. We made the dataset available for download, combined with a minor description of the dataset to inform the participants.

## 5.2 Creating a Schedule

Now that we have our track defined it's time to create a time schedule for the track. Since our SHREC track is part of the Computer and Graphics Journal (Elsevier) for the 3D Object retrieval (3DOR) section we are bound to their timeline. There are a few important dates for our track:

- **February 16** - Data made available.
- **March 8** - Registration deadline.
- **March 29** - Submission deadline of the results for participants.
- **April 12** - Track report submission to Computers  Graphics for review.
- **May 31** - First revision due.
- **July 4** - Final version submission.
- **August 26-27** - Presentation at the Eurographics 2024 Symposium on 3D Object Retrieval

We have decided to close the registration for participants on March 8th. This is not necessarily a hard deadline but more a deadline for us to estimate how many participants we will have.

Since the first paper track submission is due at April 12th we decided that I require 2 full weeks to work through the participants submissions. This would require me to compile their code to check their reliability and validity in comparison to their results. It requires me to read through their method descriptions and do minor to major adjustments to their Latex files. This would also ensure me enough time to write the results, discussion and conclusion sections for the paper. For these reasons we decided to set the submission deadline for the participants on March 29th.

For the deadlines for the first revision we awaited the reviewers response to our paper, and

[3]https://legacyfileshare.elsevier.com/promis_misc/elsarticle-CAG-template.zip

applied the revisions ourselves. After the revisions where finished we contacted the participants of our track, notifying them about the revision of the paper and asking them about their feedback. Then after utilizing their feedback we would send them the final version of the revised paper. We would handle this similarly for the second revision, which is due at July 4th.

## 5.3 Finding Participants

The preperations are finished, we have defined our track and created a schedule and have a benchmark ready, now we have to find participants for our challenge. First of all we made a promotion web-page for our SHREC track[4]. This page provides the definition of the track, the information needed for participants to participate in the track and our time schedule for the track.

Secondly we decided to email all 37 participants of previous hand recognition SHREC tracks on the 22nd of February. Stating that we were looking for participants for the new hand recognition track. In total we have 5 groups participating, 1 group participated using 2 different retrieval systems.

## 5.4 Creation of the SHREC Paper

Now that we have our participants and our track is ongoing, it is time to look at the joint paper creation. Before obtaining the submissions and results of our participants we can already do a fair amount of work for our paper. We could already write the Introduction, Related Work, the Dataset, the Task and Evaluation Section, and the baseline retrieval method section for the paper.

Then after confirming every groups results by running their retrieval system(s) and reading their Method description it was time to implement their method descriptions in the paper. Most method descriptions only required minor changes to be implemented in the SHREC2024 paper, while there was also a group (Windowed Multi View) that gave me their previous 2023 paper [27] with a list of changes made for the SHREC2024 challenge. This required me to rewrite their method description in a way that fits in the style of the short paper. After I obtained all the groups submissions I was able to write the results, discussion, and conclusion sections to finish the paper.

After the paper was finished, we emailed all participants asking them for their feedback on the paper. After implementing their feedback we submitted the paper to the editorial manager from Computer and Graphics[5]. You can read our current version of the SHREC 2024: Recognition Of Dynamic Hand Motions Molding Clay paper at Appendix A.

[4]https://www.shrec.net/SHREC-2024-hand-motion/
[5]https://www.editorialmanager.com/cag/default.aspx

# 6 Methodology

We created a CNN hand motion recognition system by customizing an existing algorithm from Fan Yang et al available at GitHub[6]. Called the Double-feature Double-motion Network (DD-Net) [3]. DD-Net uses local combined with global motion features that are location-viewpoint invariant. DD-Net utilizes simple 1D convolutional operations to classify motions, in this case the skeletal coordinate data which is converted into 1 dimensional arrays of the skeletal coordinate system. The network architecture of our modified DD-Net can be found at Fig. 6.

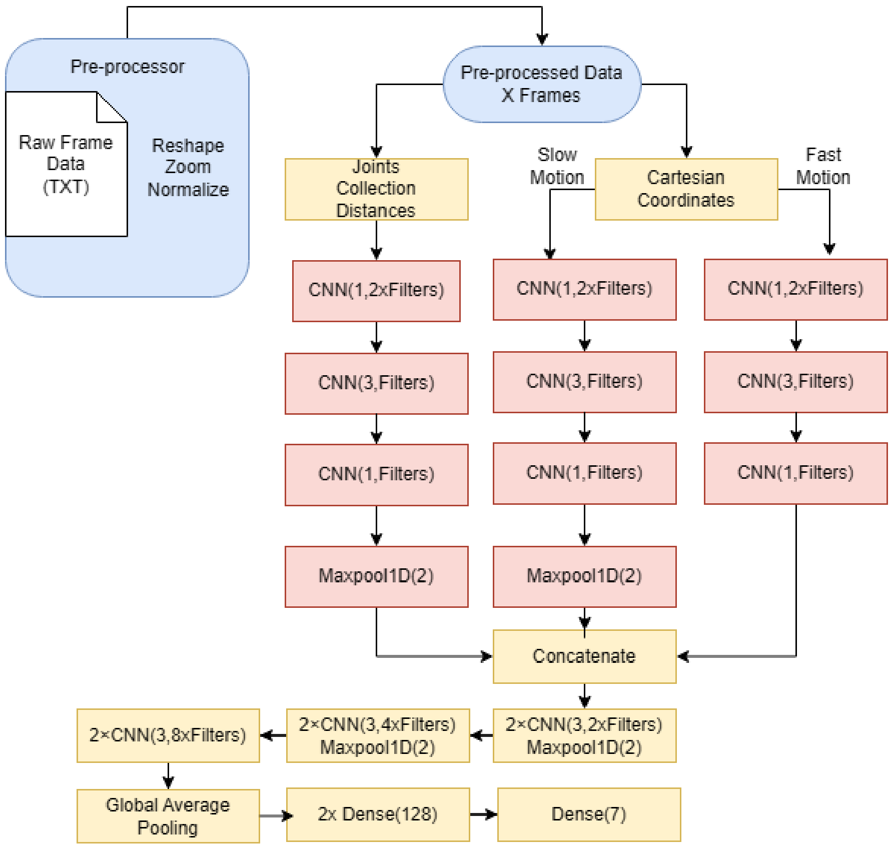


Figure 6: Overview of the baseline method for SHREC2024 modified DD-net framework

As seen in Fig. 6 we start by pre-processing the data which will be explained in section 6.1, followed by 3 main feature arrays. The 3 main feature arrays used in our retrieval system are: The Joints Collection Distances (JCD) feature explained in section 6.2, the Slow Motion ($M_{slow}$) feature, and the Fast Motion feature ($M_{fast}$)both explained in section 6.3.

## 6.1 Pre-processor and variable tuning

For the pre-processing steps we load in all the raw TXT files containing all the skeletal coordinate data from all the motions. This data is preemptively split in a train and test folder.

[6]https://github.com/bennie010697/DD-Net-SHREC-2024

We created some minor automatic data loading functions to load the data more smoothly. for example a function that will automatically extract the motion class labels. Then we apply 3 major pre-processing steps to this data: A Reshape, a Zoom function, and a Normalization function.

First we call a simple NumPy reshape function to reshape the NumPy arrays to a 28x3 size. This size is favorable to read since this reshapes the array to a size of Nx28x3, where we have 28 markers with 3 coordinates (x,y,z) per marker. This is not necessary for the system to work, but it makes the array way more readable for debugging, programming, and testing purposes.

After we reshaped the data, we call a zoom function. The Zoom function will "zoom" the data. This means that by using a medfilt array function from the scipy library, we can scale our frame length to a desirable number. This is a very important step for our CNN since all input data is required to be the same size. The medfilt function is called with a kernel size of 3 meaning it will look at it's neighbouring frames. Medfilt will automatically use the zero-padding strategy when needed. This is again desirable for our retrieval system since it will not only zoom out to decrease the frame length of certain motions that are too long, but with padding can also zoom in to increase the length of short motions.
Since our recorded motions have a larger frame-length than previous gesture recognition tracks (see section 4.1.4) we decided to test the zoom function for different values to see if it has an impact on our performance. We tested the retrieval system on 32, 128 and 256 frames. In the end we did not notice a performance increase on the accuracy by increasing the frame length. We did however just to be sure train on a higher frame length of 256 frames to test the effect of filters on the accuracy. After we found our desired filter size, we again trained on a lower frame length of 32, again not noticing a difference in the accuracy of our network. More about how we trained the network can be found in section 7.

Lastly we normalized the data using a Normalization function. This normalization function subtract the mean values from the skeletal coordinate data, resulting in a nicely centered skeletal coordinate framework.

## 6.2 Joint Collection Distances Feature

The issue of Cartesian coordinate features are that they are highly variable on the location or viewpoint of the skeleton. The Joint Collection Distance feature is a location viewpoint invariant feature that calculates the Euclidean distances between a pair of collective joints to obtain a symmetric matrix. This feature roughly translates the shape of the hand. Then to reduce the amount of redundant data we use the NumPy triu_indices function with a diagonal offset of 1 to return only the indices of the upper triangle.

We assume that the total number of frames is $T$ (of size 32 in our case) and the set of all joints $N$ (in our case a total of 28). We represent the Cartesian coordinates of joint $n$ at frame $t$ as $J_n^t = (x, y, z)$. By combining all joints into a set we have $S^t = \{J_1^t, J_2^t, ..., J_N^t\}$. Now calculating this JCD feature for $S$ on frame $t$ we get:

$$JCD^t = \begin{bmatrix} \left\|\overrightarrow{J_2^t J_1^t}\right\| & & & \\ \vdots & \ddots & & \\ \vdots & \dots & \ddots & \\ \left\|\overrightarrow{J_N^t J_1^t}\right\| & \dots & \dots & \left\|\overrightarrow{J_N^t J_N - 1^t}\right\| \end{bmatrix};$$

Here $\left\|\overrightarrow{J_i^t J_j^t}\right\|$ $(i \neq j)$ denotes the euclidean distance between $J_i^t$ and $J_j^t$ on frame $k$. As noted in our pre-processing steps section 6.1. We transform this matrix to a 1D vector by appending it to a list in order for the data to be accepted in our convolutional neural network.

## 6.3 Global Motion Features

While the JCD is location viewpoint invariant it does not contain information about the global motion of the hands. We calculate the speed by looking at the temporal differences between frames, this is a location viewpoint invariant feature. The *Short-term slow motion* $\mathrm{M_{slow}}$ computes the linear velocity of every single joint for all joints on every frame. The *Short-term fast motion* $\mathrm{M_{fast}}$ is similar to $\mathrm{M_{slow}}$ but the linear velocity gets computed every other frame. In practice, $\mathrm{M_{slow}}$ and $\mathrm{M_{fast}}$ model the short-term global motion of the skeleton in terms of speed.

## 6.4 Combining the Features

DD-Net embeds the 3 features: $\mathrm{M_{slow}}$, $\mathrm{M_{fast}}$, and JCD into latent vectors at each frame. This embedding process automatically learns the network the correlation between markers on the hand. The embedding process can be seen in figure 6. Here we can see the embedding steps colored in red. Now that we have our 3 latent feature vectors we concatenate them into our network. Since $\mathrm{M_{fast}}$ only calculated the spatial differences for every other frame. We require to call the MaxPooling1D function in the embedding process to down samples the output of the $\mathrm{M_{slow}}$ and JCD tracks.
Afterwards we call more 1D convolutional operations with increasing filter size while maintaining a kernel size of 3, this is a general best practice for convolutional networks and helps the network learn the temporal information. Afterwards we call the GlobalMaxPool1D function to extract the maximum values over the time dimension.
The major difference between the Maxpool and the GlobalMaxPool is that the GlobalMaxPool takes the max vector over the steps dimension instead of just over the stride. Meaning that we take the max values of all frames. For example imagine we have a shape of (batch_size, steps, hidden_size) then after the function GlobalMaxPool we will get the shape (batch_size, hidden_size) by pooling over the steps. Afterwards we create 2x 128 units fully connected dense layers. Then in the end we create a fully connected Dense layer with 7 units, each representing one of the 7 motion classes.

# 7 Results and Evaluation

In this section we present about our results on the SHREC-2024 novel dataset mentioned in section 4. We trained our retrieval method using 6 different parameter settings, we changed

the frame length of the zoom function and the amount of filters used in the layers of our network.
In this section we will look at effect on the training time and accuracy when changing these parameters. We also calculate 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 for our network on all our tested parameter settings.

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

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

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

Table 2 Shows the effect of changing the filters parameter in the network. By changing the filters we can see that both the Total parameters of the network as well as the training time increase in exponentially. Note that the training time is in seconds, and that we trained the system for 3000 epochs on a frame length of 256 frames. Looking at this table we can see that when we train the network with 512 filters for the neural layers that we train for roughly 11 hours.

| **Filters** | **Total Params** | **Training Time (seconds)** |
|---|---|---|
| 128 | 7.043.463 (26.87 MB) | 2.768 |
| 256 | 27.536.263 (105.04 MB) | 11.523 |
| 512 | 108.924.807 (415.52 MB) | 40.548 |

Table 2: Training time in seconds and total parameters of the network, based on the filter parameter used in the configuration. Trained on 3000 epochs with a frame length of 256 frames.

In table 3 the effect of changing the Frame_length parameter is visible on the training time. Lowering this parameter will decrease the amount of frames loaded into the CNN. This is done by the zoom function explained in section 6.1. We trained all the networks shown in the table on 3000 epochs with a filter size 128 filters.

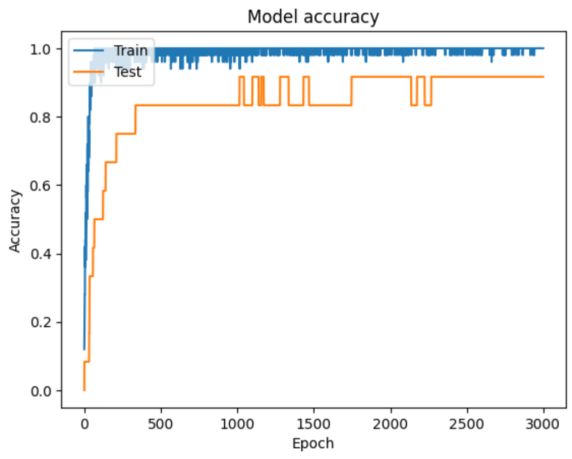
| **Frame_length** | **Training Time (seconds)** |
|---|---|
| 32 | 1.456 |
| 128 | 2.220 |
| 256 | 2.768 |

Table 3: The effect of changing the Frame_length on the training time.

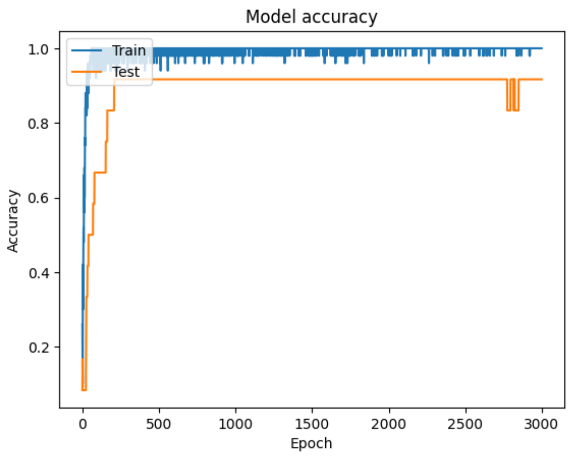
We then combined table 2 and table 3 to create table 4, we also added the global accuracy to this table. As shown In table 4 the network with similar filter sizes perform the same on the global accuracy of the test set. The issue however is that the increase in filter size slows the training time significantly. The network with 256 filters trains quicker by a factor of roughly 4, compared to our network when using 512 filters. Since all methods with 256 or more filters got the same global accuracy, we decided to crown the method with the lowest training time the best performer. Resulting in us using a filter size of 256 and a frame_length of 32 as our baseline configuration for our method.

| **Filters** | **Frame_length** | **Total Params** | **Training Time** | **Accuracy** |
|---|---|---|---|---|
| 128 | 32 | 7.043.463 (26.87 MB) | 1.456 | 0.83 |
| 128 | 128 | 7.043.463 (26.87 MB) | 2.220 | 0.83 |
| 128 | 256 | 7.043.463 (26.87 MB) | 2.768 | 0.83 |
| **256** | **32** | **27.536.263 (105.04 MB)** | **7.257** | **0.92** |
| 256 | 256 | 27.536.263 (105.04 MB) | 11.523 | 0.92 |
| 512 | 256 | 108.924.807 (415.52 MB) | 40.548 | 0.92 |

Table 4: Combined table of table 2 and table 3 and the accuracy of the network.



256filters 256frames



256filters 32frames

Figure 7: Training Graph, number of epochs on the x-axis, the corresponding accuracy percentages for both train and test datasets on the y-axis.

As mentioned above all networks are trained on 3000 epochs. Figure 7 shows the model accuracy of our method on 256 filters and a frame length of 256 and 32 frames over the training period. Figure 7 shows that the network converges at around 300 epochs on the training set. Normally a network is done training slightly after it converges. However due to the small data set it is hard to automatically detect if a network is converged.
Which is why in our case we manually chose to run the network for 3000 epochs, to make sure the network is fully trained. We can also see that our network increases it's performance on the test set, way after the training convergence point. We are also able to spot a difference in the training graphs at 256 and 32 frames. Namely that the network

using 32 frames shows better results earlier in the training stages on the test set, compare to the model using 256 frames. When using 32 frames we also gain a more stable performance on the test set.
Normally computation speed is an importance factor for your program, our training could be stopped at around 400-500 epochs. However since we are dealing with a retrieval system where we can create a pre-trained model, we do not care about the metric of Frames Per Seconds (FPS) or total training time as much as the metric of accuracy. Predicting a motion will be just as quick on a model that's trained for only 400 epochs as for a model that is trained for 3000 epochs.
However if we do care about the performance in terms of training time we could always train the neural network for 300 epochs. By setting the configuration of the parameter settings to 256 filters and 32 frames we are able to obtain an accuracy of 92% when training for 300 epochs. This would still give us the best found results in terms of global accuracy on our test set, while only requiring 700 seconds to train.
The correlation between accuracy and parameters might change when using a larger test and or training set, where perhaps having a higher frame length might result in a better global accuracy score.

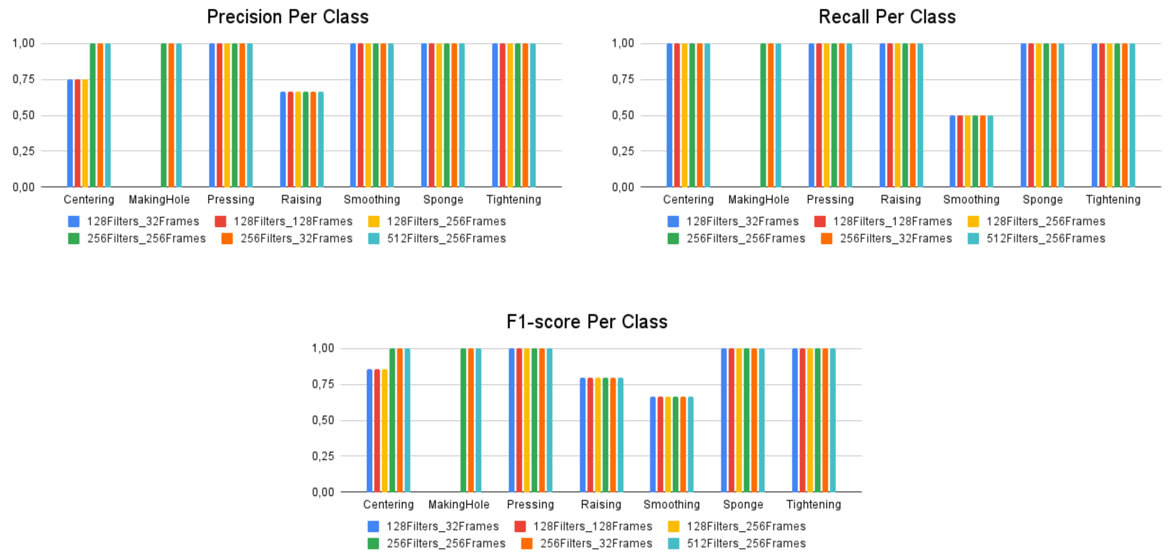


Figure 8: Performance metrics per motion class on the different parameters

The bar charts in figure 8 show the per-class precision, recall and F1 scores of the retrieval system on different configuration settings. The confusion matrix of the retrieval system can be seen in figure 9.
First of all we notice that the frame lengths that we used for training do not impact the retrieval accuracy on our dataset. All three configurations using 128 filters, and all three configurations using 256 or 512 filters gave similar results in both Precision, Recall and F1-score as well as similar confusion matrices.
Looking at these 2 figures, we can see that no matter which parameters we use we will always make a prediction error in the 'Smoothing' and 'Raising' classes. Where the true

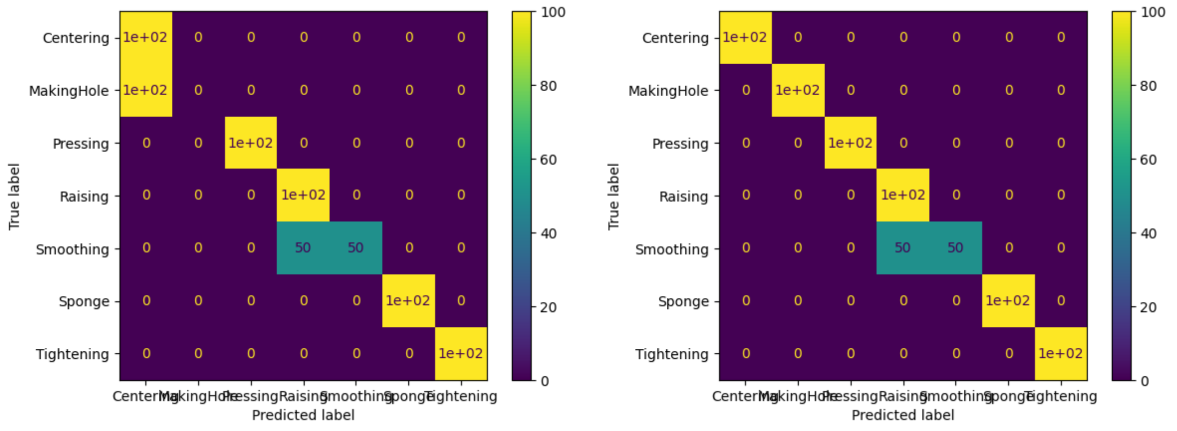


Figure 9: Confusion matrix of 128 filters on the left and 256 & 512 filters on the right

label is 'Smoothing' but we predict 'Raising'. These motions are highly similar due to the fact that the potter's left hand makes the exact same upwards motion to stabilize the inside of the pot. While the potter's right hand makes a similar upwards global motion. There is however a difference in these motions, namely in the fact that the right hand is angled slightly differently. Both these motions can be seen in figure 10.
We also notice a change on the confusion matrix in the 128 filters compared to the 256 & 512 filters configuration setting. Where we predict the 'Centering' class when the true label is 'MakingHole' when using the configuration of a 128 filters. Showing that the global accuracy of the model is increased whenever we upscale the filter size to 256 or 512 filters.

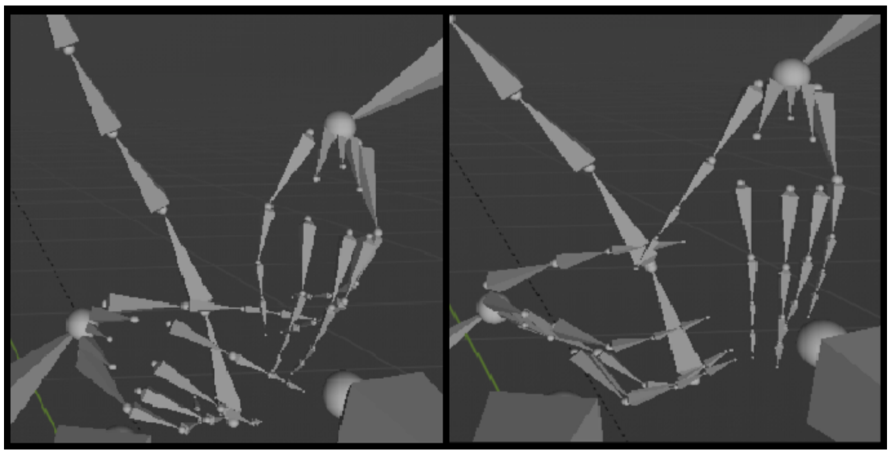


Figure 10: Static screen-capture in blender of the 'Smoothing' motion (left) where the potter uses his phalanges and the 'Raising' motion (right) where the potter uses the tips of his fingers

Our results are compared to other retrieval methods in the SHREC 2024: Recognition Of Dynamic Hand Motions Molding Clay paper at Appendix A. In this paper we used the neural network with 256 filters and frame lengths of 32 frames as the baseline method.

## 8 Conclusion

In this paper we have presented the steps taken to organize the SHREC 2024 track: Recognition Of Dynamic Hand Motions Molding Clay. We reported the creation steps of the novel dataset and the baseline method. We then explained our evaluation method and evaluated our results on our novel benchmark. The evaluation of our benchmark shows that due to the small size of the dataset it's difficult to tune our neural network properly. Due to the small test set, we believe that it is necessary to continue the study on a larger test set to get a more in depth evaluation. We do however believe that we have shown the valuable steps on how we created our benchmark and our retrieval method. Just as the steps we took on how to successfully hold your own SHREC track. We also believe that our dataset even in it's small scale is a valuable asset to the field of hand recognition, due to it's uniqueness and high quality. As of writing this paper there are little to non benchmarks available of hand motions on mold-able objects. Our benchmark also contains highly similar, and precise motions using two hands, making it a new type of benchmark for gesture recognition.

## 9 Future Work

The current research has a limited scope, due to the rather small dataset which is meant to introduce the concept of retrieving highly similar motions on objects while using both hands. Follow-up research could expand on the dataset by generating more data by using the same or other systems and techniques to generate data on mold-able objects. We could expand the data by having different subjects recorded while also looking at the possibility of increasing the amount of gesture classes. Having this larger dataset would improve the quality of our results and will make it easier to see differences between the parameters as well as select the best parameters for our retrieval method.

A supplementary study could be performed that looks at different retrieval techniques or different neural networks and compares them. As mentioned in the paper there are multiple neural networks that each have their own benefits and reasons to be used for gesture retrieval. Exploring these in more depth could lead to a better retrieval system.

While this is just a new way to look at skeletal based recognition there are numerous new studies, datasets, techniques and gestures that could be added within the SHREC track of hand gesture recognition and increase the effectiveness of skeletal based recognition. Skeletal based recognition is by far not finished and more studies always help to improve retrieval systems.

## References

[1] Shrec. https://www.shrec.net/.

[2] Vicon shogun homepage. https://www.vicon.com/software/shogun/.

[3] Fan Yang, Yang Wu, Sakriani Sakti, and Satoshi Nakamura. Make skeleton-based action recognition model smaller, faster and better. In *Proceedings of the ACM multimedia asia*, pages 1–6. 2019.

[4] Shrec home page 2023. https://www.shrec.net/.

[5] Sarthak Gupta, Siddhant Bagga, and Deepak Kumar Sharma. Hand gesture recognition for human computer interaction and its applications in virtual reality. *Advanced Computational Intelligence Techniques for Virtual Reality in Healthcare*, pages 85–105, 2020.

[6] Google media pipe. https://developers.google.com/mediapipe/solutions/vision/gesture$_recognizer$.

[7] Google tensor flow. https://www.tensorflow.org/.

[8] Andrea Giachetti et al. Fabio Marco Caputo. Shrec 2019 track: Online gesture recognition. 2019.

[9] Marco Pegoraro et al. Ariel Caputo, Andrea Giachetti. Shrec 2020 track: Recognition of gestures sequences. 2020.

[10] Chin K. Y. Richard Lim S. F. Jain Pushpdant Chua, S. N. David. Hand gesture control for human–computer interaction with deep learning. *Journal of Electrical Engineering Technology*, 2022.

[11] Jean-Phillipe Vandeborre et al. Quentin De Smedt, Hazem Wannous. Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset. 2017.

[12] Andrea Giachetti et al. Fabio Marco Caputo. Shrec 2021: Skeleton-based hand gesture recognition in the wild. 2021.

[13] Ariel Caputo Marco Cristani Andrea Giachetti et al. Marco Emporio, Anton Pirtac. Shrec 2022 track on online detection of heterogeneous gestures. 2022.

[14] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[15] Rim Slama, Wael Rabah, and Hazem Wannous. Str-gcn: Dual spatial graph convolutional network and transformer graph encoder for 3d hand gesture recognition. In *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*, pages 1–6, 2023.

[16] Yong Li, Zihang He, Xiang Ye, Zuguo He, and Kangrong Han. Spatial temporal graph convolutional networks for skeleton-based dynamic hand gesture recognition. *EURASIP Journal on Image and Video Processing*, 2019:1–7, 2019.

[17] Indriani, Moh. Harris, and Ali Suryaperdana Agoes. Applying hand gesture recognition for user guide application using mediapipe. In *Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021)*, pages 101–108. Atlantis Press, 2021.

[18] Prachetas Padhi and Mousumi Das. Hand gesture recognition using densenet201-mediapipe hybrid modelling. In *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, pages 995–999, 2022.

[19] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[20] Brenda Elizabeth Olivas-Padilla and Sotiris Manitsaris. Deep state-space modeling for explainable representation, analysis, and generation of professional human poses. *arXiv preprint arXiv:2304.14502*, 2023.

[21] Brenda Elizabeth Olivas-Padilla, Alina Glushkova, and Sotiris Manitsaris. Motion capture benchmark of real industrial tasks and traditional crafts for human movement analysis. *IEEE Access*, 11:40075–40092, 2023.

[22] Asha Kapur, Ajay Kapur, Naznin Virji-Babul, George Tzanetakis, and Peter F. Driessen. Gesture-based affective computing on motion capture data. In Jianhua Tao, Tieniu Tan, and Rosalind W. Picard, editors, *Affective Computing and Intelligent Interaction*, pages 1–7, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[23] Jonathan Maycock, Jan Steffen, Robert Haschke, and Helge Ritter. Robust tracking of human hand postures for robot teaching. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2947–2952, 2011.

[24] Motion capture and virtual reality lab utrecht. https://www.uu.nl/en/research/motion-capture-and-virtual-reality-lab.

[25] Vicon shogun documentation. https://docs.vicon.com/display/Shogun17/Shogun+Documentation.

[26] Graphics replicability stamp initiative web page. https://www.replicabilitystamp.org/.

[27] Federico Cunico, Federico Girella, Andrea Avogaro, Marco Emporio, Andrea Giachetti, and Marco Cristani. Oo-dmvmt: 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*, pages 2745–2754, 2023.

# Appendix

## A SHREC 2024 Short paper: Recognition Of Dynamic Hand Motions Molding Clay

Contents lists available at ScienceDirect

Computers & Graphics

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

# 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, Florida, 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

*Article history*:
Received April 12, 2024
Received in final form July 4, 2024 *
Accepted July 7, 2024 *
Available online August 2024 *

*Keywords:* Gesture recognition, Hand gestures, Hand skeleton, SHREC, 3D Shape Retrieval Challenge, Vicon Shogun, Motion capture, Neural Networks, Classification

## 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 skeletons it's still unclear how well state-of-the-art techniques perform in a scenario using precise motions with 2 hands. This paper presents the results of the SHREC 2024 contest organized to evaluate methods for recognition of highly similar hand motions using the skeletal spatial coordinates data of both hands. The task is the recognition of 7 motion classes given their spatial coordinates in a frame by frame motion. The data have been captured using the 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 duration in frames. This paper shows the results of the contest, showing the techniques created by the 5 research groups on this challenging task and compare them to our baseline method.

*Corresponding author:
*e-mail:* 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. Tycowicz)

## 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, 2, 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 which provides sequences of hand skeletal data using two hands on 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 3 presents the novel dataset, Section 4 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 that 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 amount 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 off 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 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 into two subsets using a (80/20) training/testing split. The motions 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
- **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. Molding clay is a precise and delicate act where the potter makes small and precise hand movements using both hands. The movements the potter creates are perfect for our recognition benchmark since it fits all the aspects to overcome the weakness mentioned earlier. Due to the potters movements being so precise our motions are variable in frame length and quite long. resulting in frame lengths between 29-3721 frames with an average of 990 frames, compared to previous benchmarks where motions had a frame length of 15-50 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.

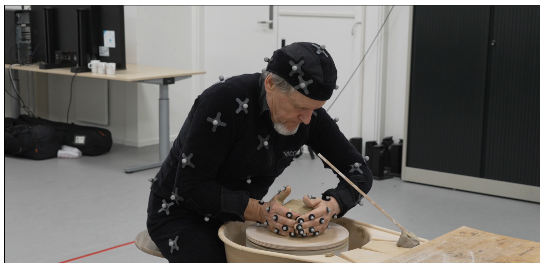


**Fig. 1. Recording of hand motions using clay to create a vase**

We use Vicon Shogun post to remove any stuttering found during the recording and to export armature of the potter

[1] `uu.nl/en/research/motion-capture-and-virtual-reality-lab`
[2] `vicon.com/software/shogun`

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 in a text file where each row represents the data of a specific frame followed by the 28x3 coordinate floats (14 per hand see Fig. 2) (x;y;z) positions of the markers.

The structure of the coordinate system is as followed where the L and R stand for Left and Right hand respectively:

*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);*

see Fig. 2 for the location of the markers on the left hand (marker locations on the right hand are on similar locations).

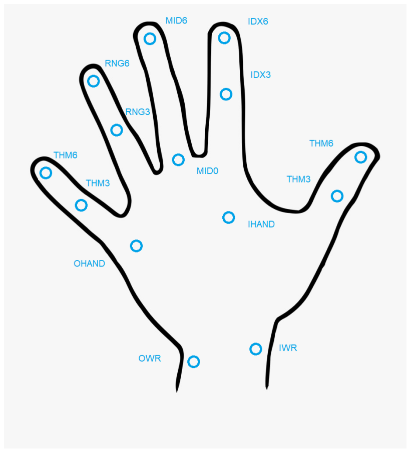


**Fig. 2. Location of markers used on the left hand during recording**

## 4. Task and evaluation

Participants where 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 train set, the motion of two hands simultaneously, and the precise and highly similar motions of the potter hands makes it a novel recognition task. That requires methods to look into motion details as well as creating a 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 combined with 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 out off 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 6 results files 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.
- *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 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 at Fig. 3.

[3]https://github.com/bennie010697/DD-Net-SHREC-2024

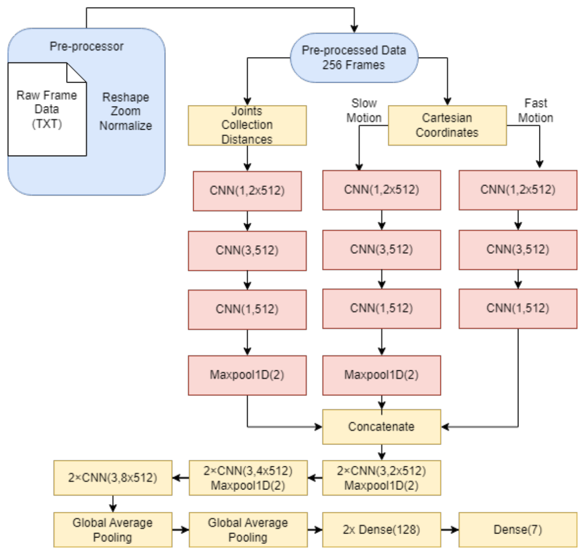


**Fig. 3. Overview of the SHREC2024 DD-net framework**

### 5.1.1. *Feature extraction*

DD-net derives 3 features from the hand joints 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 Characterizing the hand pose. The slow motion and fast motion features calculate the temporal difference of the Cartesian coordinate feature to obtain 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 of the SHREC2024 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 realised the difference in frame lengths off the SHREC17 track compared to our SHREC24 data. Resulting in the removal of too much useful coordinate information in the zoom function from DD-net. The previously used target frame length of 32 frames in the zoom function was increased to 256. During testing we realised that increasing the frame length further would impact too heavily on performance, while not increasing classification accuracy to the same degree. Due to this increase in frame length we decided to up the filters from the previous 64 to the same degree of 512 to keep the same level of detail per frame. This increase in frame length and filter amount improved the accuracy of the classification in the test-set on highly similar motions like ”Centering” and ”MakingHole”.

### 5.1.3. *Implementation details*

We trained each model on DD-net for 3000 epochs using a Adam optimizer. With an annealing learning rate that drops from $1\times10^{-4}$ to $5\times10^{-6}$. We did not apply pre-trained weights. We experimented on different frame lengths and filter sizes. All models are implemented in TensorFlow.

## 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 [9, 8]. 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.

[4]`https://github.com/o-ikne/skelmae-shrec24.git`

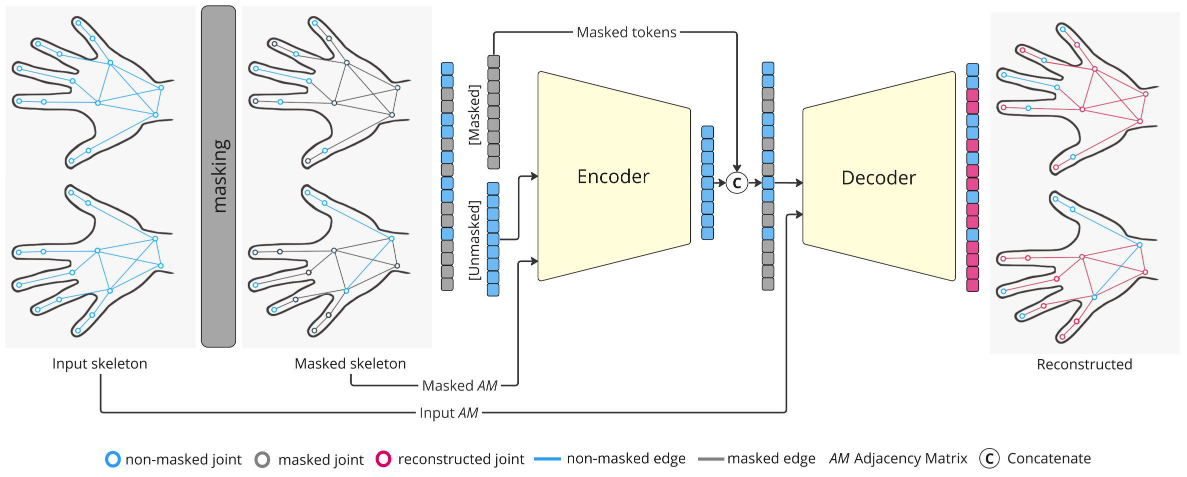


**Fig. 4. Proposed MAE for hands skeleton reconstruction. We mask a given ratio of joints in each hand, the unmasked joints are encoded by the encoder while taking into account their connectivity given by the masked adjacency matrix. The encoded joints are then concatenated with the masked tokens and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

*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.

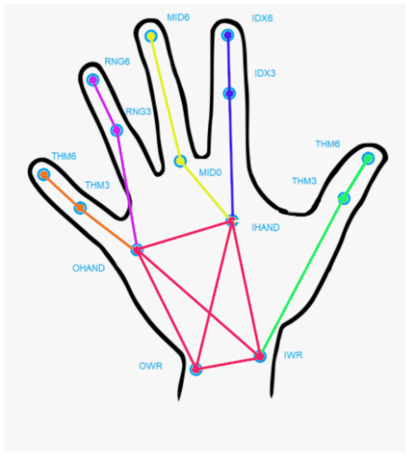


**Fig. 5. SkelMAE: proposed hand adjacency connections.**

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 pretrained 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 $2 \times 10^{-4}$ and a weight decay of $5 \times 10^{-2}$. 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 $1 \times 10^{-3}$ and a weight decay of $5 \times 10^{-4}$. 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.

## *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, 16].

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, $\mathrm{M_{slow}}$ and $\mathrm{M_{fast}}$ 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.

## *5.4. 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 [17]) 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 [18]. To this end, a set of n successive frames is considered (n=14) in order to take into account the temporal dimension. This operation

[5]https://github.com/intelligolabs/OO-dMVMT

[6]https://colab.research.google.com/drive/1fi4PPsHp9K5Ovg9qLBFYWCzk5A5_kPP2?usp=sharing

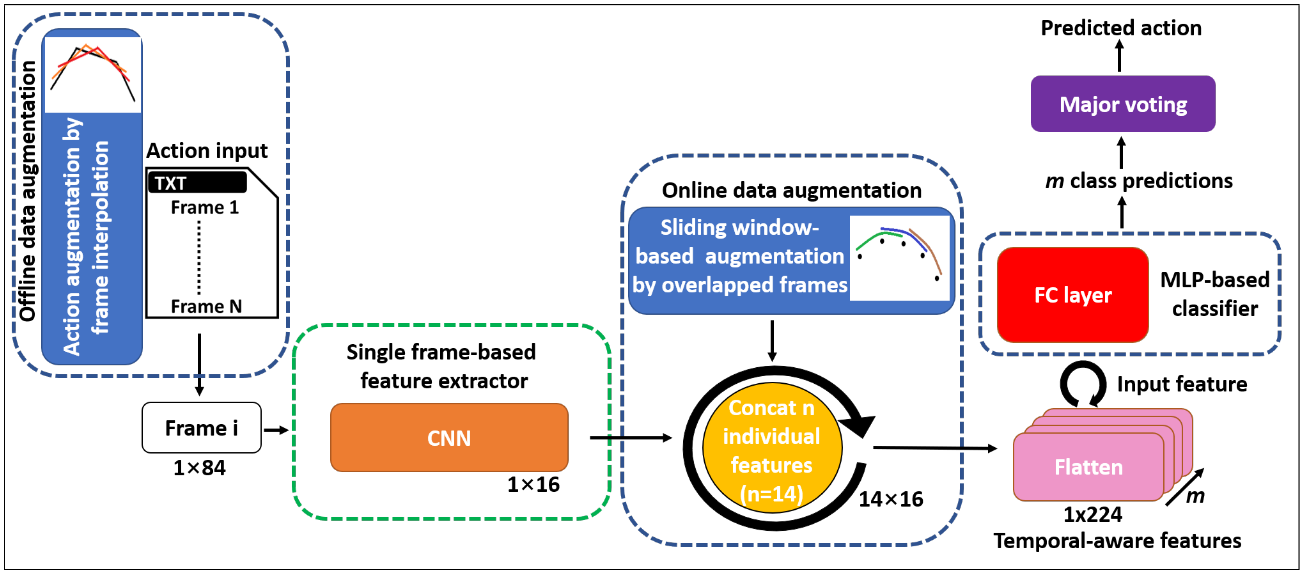


**Fig. 6. Overview of DET-ACTIONS bottom-up deep learning-based analysis framework for hand motion recognition.**

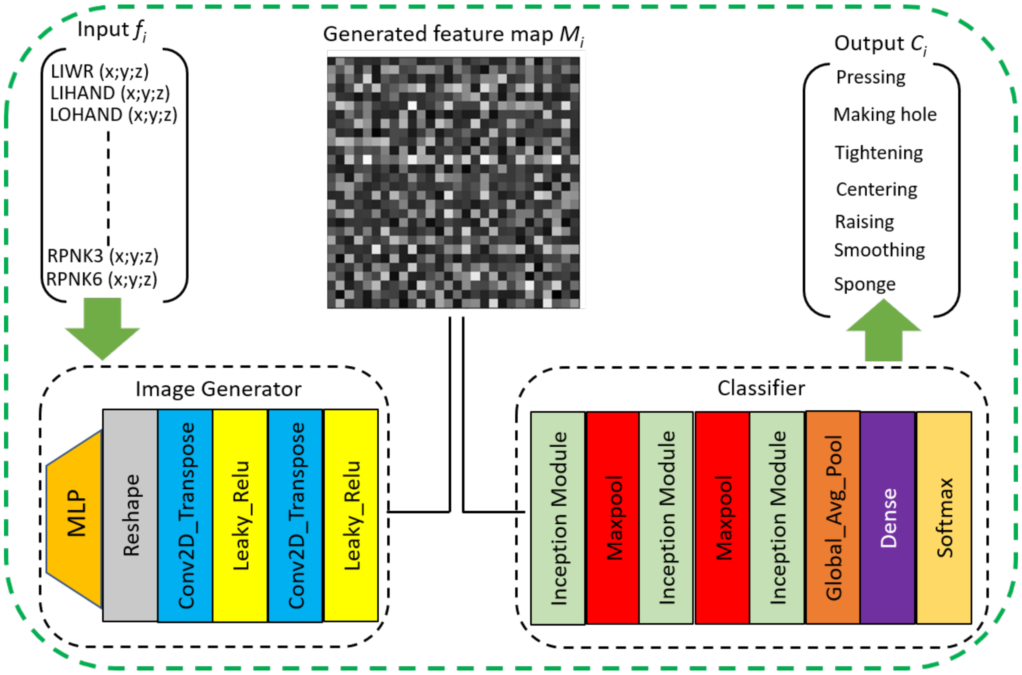


**Fig. 7. CNN-based architecture for feature extraction from a single motion frame.**

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 [19] 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 $M_i$ ($28 \times 28$). 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.

## 5.5. 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 modelling and classifying signals that demonstrate intrinsic temporality, like human speech [20] and movement [21]. 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.

### 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:

$$h_t = [x_1\ y_1\ z_1\ \Delta x_1\ \Delta y_1\ \Delta z_1\ x_2\ y_2\ z_2\ \Delta x_2\ \Delta y_2\ \Delta z_2\ \cdots] \quad (5)$$

$$h = [h_0\ h_1\ h_2\ \cdots] \quad (6)$$

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 [22]. 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 automcatically using Optuna [23].

## 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.

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 [24] 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 [25] 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 hours on our NVIDIA RTX™ 2060 SUPER GPU with 8GB 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

[7]`https://github.com/ChristoforosVlachos/shrec2024`

[8]`rhttps://github.com/hmmlearn/hmmlearn`

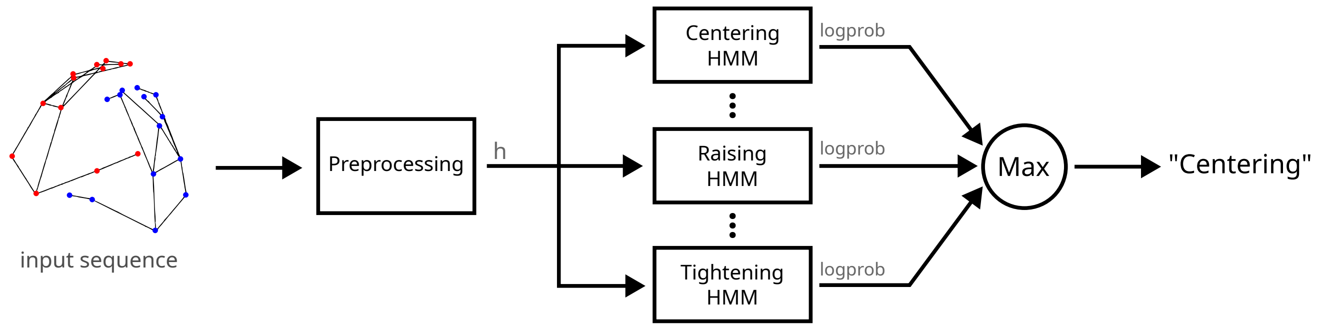


**Fig. 8. Architecture of the proposed HMM-based method.**

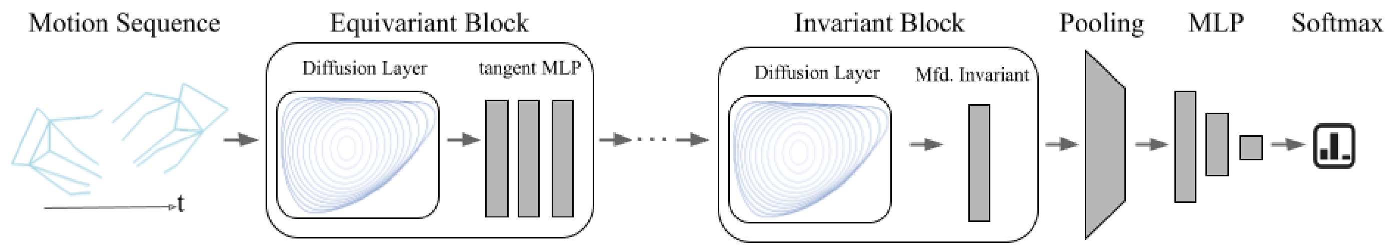


**Fig. 9. Schematic of the proposed manifold GCN architecture.**

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. You can find our code at Github [9]

## *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.

[9] `https://github.com/ChristoforosVlachos/shrec2024`
[10] `https://github.com/morphomatics/SHREC24`

### *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 [26]; 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 [27] of their predicted class probabilities based on the Fisher-Rao distance [28] 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 test set. It’s hard to say which group is the best due to the small Test set. We can however see that 4 groups: *SkelMAE, DET-ACTIONS, RNN-based approach and SE(3)-equivariant GCN* got a perfect score. It however is more interesting to look at the mistakes made by the other methods. The bar charts in Fig. 10 show the per-class precision, recall and F1 scores of all the methods. We also created a confusion matrix of the 3 other runs.

| Method | Accuracy |
|---|---|
| *DD-Net* | 0.92 |
| *SkelMAE* | 1 |
| *Windowed Multi View* | 0.75 |
| *DET-ACTIONS* | 1 |
| *HMM-based approach* | 0.83 |
| *RNN-based approach* | 1 |
| *SE(3)-equivariant GCN* | 1 |

**Table 1. Total Accuracy per group over the test set.**

Looking at the bar-charts in Fig. 10 we can see that most issues are made with the highly similar motions ’Smoothing’ and ’Raising’ namely 3 out off the 6 errors and can be found in 3 different methods namely: DD-Net, Windowed Multi View and HMM-based approach. These motions are highly similar due to the fact that the potter’s left hand makes the exact same motion, while his right hand makes a similar upwards motion in both classes. The difference is in the way that the right hand is angled slightly differently. See Fig. 11 for a better visualization between the 2 hand motions.

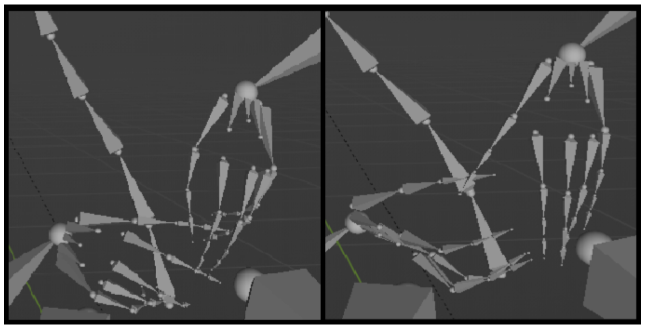


**Fig. 11. Static picture of the Smoothing motion (left) where the potter uses his phalanges and the Raising motion (right) where the potter uses the tips of his fingers**

There are also 2 errors in total in the method of Windowed Multi View and the HMM-based approach where the method predicted the motion class MakingHole as the Centering class. Both the MakingHole and Centering dynamic motions 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 the right hand uses the Index finger and Middle finger to create a hole at the end of the motion.

The last error is found in the Windowed multi view method where the true label was raising and they predicted centering. We do not find many similarities between these 2 motions.

## 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 the 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 gesture class Centering by the methods Windowed Multi View and HMM-based approach could derive from the limited test and train data. The gesture MakingHole exists a total of 5 times in the train and test set while centering exists a total of 16 times. Creating a more evenly distribution of the motion classes could have solved these issues.

We do however believe that the issues in the Smoothing and Raising class do derive because they are highly similar in both left and right hand. One of the goals was to have participants train on a small train set. However due to the small test set, it became hard to find a ”winner” for this challenge.

## 8. Conclusion

In this paper we have presented a novel dynamic hand gesture dataset, and have reported and analyzed the results of the participants submissions for the SHREC 2024 track on Recognition Of Dynamic Hand Motions Molding Clay. The evaluation of

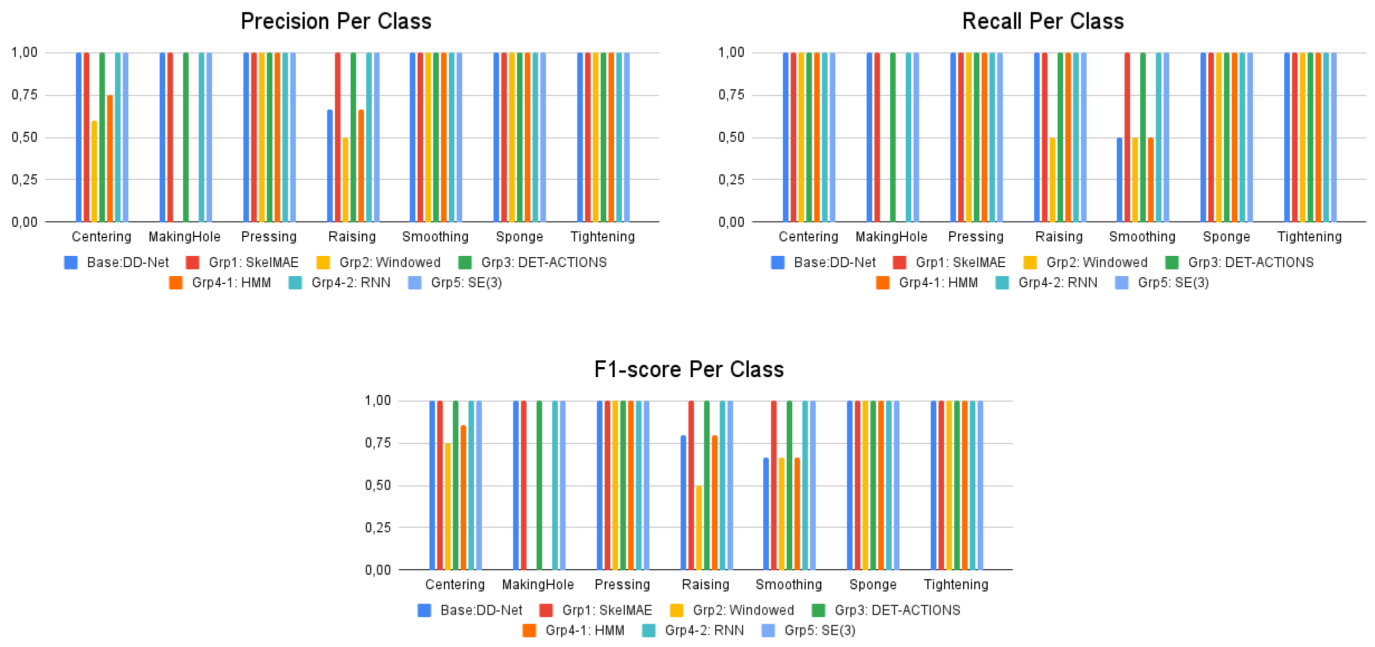


**Fig. 10. Performance metrics per motion class**

the proposed methods show 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 precise evaluation of the methods. A possible future research direction could be to continue with a small train set however expand on the test set, both increasing the amount of gestures and the amount of gesture classes. We can achieve this by recording a higher amount 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 for hand gesture recognition on highly detailed, highly similar motions using both hands.

## References

[1] Quentin De Smedt Hazem Wannous, JPVea. 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, JP, et al. Shrec 2019 track: Online gesture recognition. 2019,URL: `https://api.semanticscholar.org/CorpusID:208524001`.

[3] De Smedt, Q, Wannous, H, Vandeborre, JP. Heterogeneous hand gesture recognition using 3d dynamic skeletal data. Computer Vision and Image Understanding 2019;181:60–72. URL: `https://www.sciencedirect.com/science/article/pii/S1077314219300153`. doi:`https://doi.org/10.1016/j.cviu.2019.01.008`.

[4] Caputo, A, Giachetti, A, Soso, S, Pintani, D, D'Eusanio, A, Pini, S, et al. Shrec 2021: Skeleton-based hand gesture recognition in the wild. Computers Graphics 2021;99:201–211.

[5] Emporio, M, Caputo, A, Giachetti, A, Cristani, M, Borghi, G, D'Eusanio, A, et al. Shrec 2022 track on online detection of heterogeneous gestures. Computers Graphics 2022;107:241–251. URL: `https://www.sciencedirect.com/science/article/pii/S0097849322001388`. doi:`https://doi.org/10.1016/j.cag.2022.07.015`.

[6] Fan Yang Sakriani Sakti, YW, Nakamura, S. 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. arXiv preprint arXiv:230708476 2023;.

[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. 16000–16009.

[10] Dosovitskiy, A, Beyer, L, Kolesnikov, A, Weissenborn, D, Zhai, X, Unterthiner, T, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:201011929 2020;.

[11] Tancik, M, Srinivasan, P, Mildenhall, B, Fridovich-Keil, S, Raghavan, N, Singhal, U, et al. Fourier features let networks learn high frequency functions in low dimensional domains. Advances in Neural Information Processing Systems 2020;33:7537–7547.

[12] Vaswani, A, Shazeer, N, Parmar, N, Uszkoreit, J, Jones, L, Gomez, AN, et al. Attention is all you need. Advances in neural information processing systems 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] Emporio, M, Caputo, A, Giachetti, A, Cristani, M, Borghi, G, D'Eusanio, A, et al. Shrec 2022 track on online detection of heterogeneous gestures. Computers Graphics 2022;107:241–251. URL: `https://www.sciencedirect.com/science/article/pii/S0097849322001388`. doi:`https://doi.org/10.1016/j.cag.2022.07.015`.

[17] Dill, A, Ge, S, Kang, E, Li, CL, Poczos, B. Learned interpolation for 3d generation. 2020. `arXiv:1912.10787`.

[18] 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.

[19] 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. 2818–2826.

[20] Rabiner, L. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 1989;77(2):257–286. doi:10.1109/5.18626.

[21] 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. ISBN 978-3-319-04114-8; 2014, p. 473–483. doi:10.1007/978-3-319-04114-8_40.

[22] Dempster, AP, Laird, NM, Rubin, DB. Maximum Likelihood from Incomplete Data Via the EM Algorithm. Journal of the Royal Statistical Society: Series B (Methodological) 1977;39(1):1–22. doi:10.1111/j.2517-6161.1977.tb01600.x.

[23] Akiba, T, Sano, S, Yanase, T, Ohta, T, Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. 2019. doi:10.48550/arXiv.1907.10902. arXiv:1907.10902.

[24] Lin, TY, 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. 2980–2988.

[25] 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. 249–256.

[26] Hanik, M, Steidl, G, von Tycowicz, C. Manifold GCN: Diffusion-based convolutional neural network for manifold-valued graphs. arXiv preprint arXiv:240114381 2024;.

[27] Fletcher, PT, Venkatasubramanian, S, Joshi, S. The geometric median on Riemannian manifolds with application to robust atlas estimation. NeuroImage 2009;45(1):S143–S152.

[28] 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.