STMG: A Machine Learning Microgesture Recognition System for Supporting Thumb-Based VR/AR Input

Kenrick Kin
Reality Labs, Meta
Redmond, WA, USA
kenrickkin@meta.com

Chengde Wan
Reality Labs, Meta
Redmond, WA, USA
vgrasp@meta.com

Ken Koh
Reality Labs, Meta
Burlingame, CA, USA
kenbkoh@meta.com

Andrei Marin
Reality Labs, Meta
Zurich, Switzerland
andreimarin@meta.com

Necati Cihan Camgöz
Reality Labs, Meta
Zurich, Switzerland
neccam@meta.com

Yubo Zhang
Reality Labs, Meta
Burlingame, CA, USA
yubozhang@meta.com

Yujun Cai
Reality Labs, Meta
Redmond, WA, USA
yujuncai@meta.com

Fedor Kovalev
Reality Labs, Meta
Amsterdam, Netherlands
kovalevfm@meta.com

Moshe Ben-Zacharia
Reality Labs, Meta
Seattle, WA, USA
moshebenzacharia@gmail.com

Shannon Hoople
Reality Labs, Meta
Seattle, WA, USA
shannonhoople@meta.com

Marcos Nunes-Ueno
Reality Labs, Meta
Redmond, WA, USA
marcosnu@meta.com

Mariel Sanchez-Rodriguez
Reality Labs, Meta
Redmond, WA, USA
mariel.sr.003@gmail.com

Ayush Bhargava
Reality Labs, Meta
Burlingame, CA, USA
ayush3@meta.com

Robert Wang
Reality Labs, Meta
Redmond, WA, USA
rywang@meta.com

Eric Sauser
Reality Labs, Meta
Zurich, Switzerland
esauser@meta.com

Shugao Ma
Reality Labs, Meta
Redmond, WA, USA
shugao@meta.com

ABSTRACT

AR/VR devices have started to adopt hand tracking, in lieu of controllers, to support user interaction. However, today's hand input rely primarily on one gesture: pinch. Moreover, current mappings of hand motion to use cases like VR locomotion and content scrolling involve more complex and larger arm motions than joystick or trackpad usage. STMG increases the gesture space by recognizing additional small thumb-based microgestures from skeletal tracking running on a headset. We take a machine learning approach and achieve a 95.1% recognition accuracy across seven thumb gestures performed on the index finger surface: four directional thumb swipes (left, right, forward, backward), thumb tap, and fingertip pinch start and pinch end. We detail the components to our machine learning pipeline and highlight our design decisions and lessons learned in producing a well generalized model. We then demonstrate how these microgestures simplify and reduce arm motions for hand-based locomotion and scrolling interactions.

CCS CONCEPTS

• **Human-centered computing** → **Gestural input**; *Virtual reality*; • **Computing methodologies** → *Neural networks.*

KEYWORDS

microgestures, neural networks, machine learning, augmented reality, virtual reality

ACM Reference Format:
Kenrick Kin, Chengde Wan, Ken Koh, Andrei Marin, Necati Cihan Camgöz, Yubo Zhang, Yujun Cai, Fedor Kovalev, Moshe Ben-Zacharia, Shannon Hoople, Marcos Nunes-Ueno, Mariel Sanchez-Rodriguez, Ayush Bhargava, Robert Wang, Eric Sauser, and Shugao Ma. 2024. STMG: A Machine Learning Microgesture Recognition System for Supporting Thumb-Based VR/AR Input . In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3613904.3642702

CHI '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0330-0/24/05
https://doi.org/10.1145/3613904.3642702

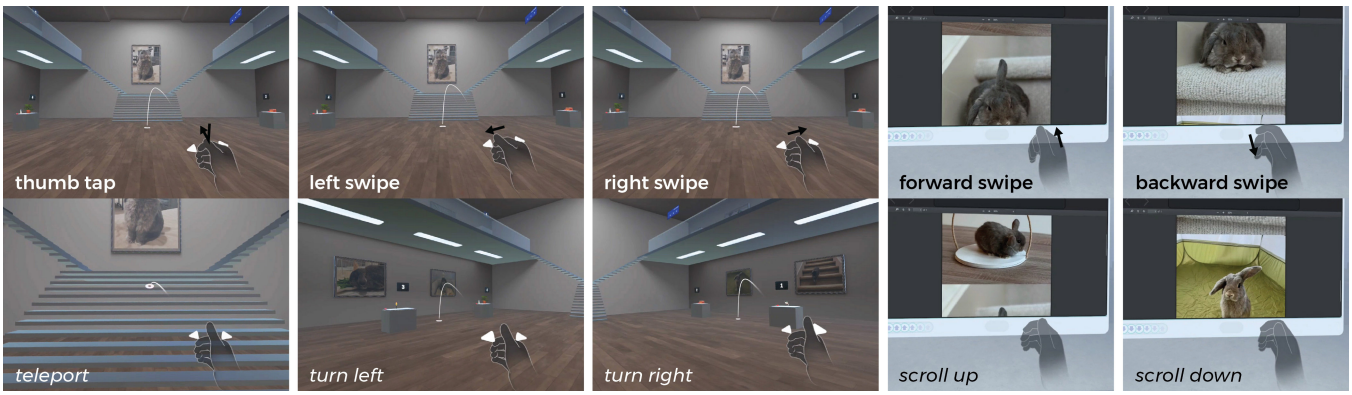

Figure 1: STMG microgestures are thumb taps and swipes performed on the radial side of the index finger. We map them to locomotion commands: thumb tap to teleport, left swipe to snap turn left, right swipe to snap turn right. We also map them to vertical scrolling: forward swipe to scroll up, backward swipe to scroll down.

1 INTRODUCTION

Vision-based hand tracking provides a convenient input modality for augmented and virtual reality. Without the need of controllers or peripheral devices, users can interact with virtual objects directly with their hands. For interactions at a distance, such as indirect or far-field interactions, current headset devices provide thumb and index fingertip pinch-based input [1, 24, 48, 49]. Pinches are simple gestures: by just contacting the thumb and index fingertip together, the user can generate a clear signal of intent to the gesture recognizer. However, when pinch is the only gesture used, developers need to incorporate additional hand and arm movements to differentiate between different pinch interactions.

With skeletal tracking microgestures (STMG), we extend the pinch gesture set to include more one-handed thumb-to-index microgestures - thumb tap and four directional thumb swipes (left, right, forward, backward) performed on the radial side of the index finger. These additional low-calorie gestures are distinct from pinch and can be mapped to other commands in an eyes-free manner. Due to their similarity to smartphone interactions, these gestures are easy to learn. The self-haptic feedback afforded by the interaction between the thumb and index finger reinforces the gesture motions.

Recognizing microgestures is a challenging task. The power of the gestures is due in part to their small motion, i.e., 2-3 cm swipes or taps of the thumb, which are difficult to differentiate from hand tracking jitter or noise. Though the motions are easy to describe, there is also significant variation across users, which makes it difficult to recognize the gesture motions with simple heuristics. We take a machine learning approach, applying temporal neural networks to skeletal hand motion generated from Meta Quest's egocentric monochrome cameras, without the need of additional wearable sensors. We achieve 95.1% F1 score across seven gestures (thumb tap, four thumb swipes, pinch start, pinch end) on a challenging offline testing dataset of 20 users.

Using our STMG microgesture model we demonstrate that these thumb-based gestures are a viable input method for VR/AR uses cases (Figure 1). Thumb tap and swipes can be used for locomotion control in VR (e.g., teleportation and snap turning). We present a locomotion study where we found 17 of 24 users preferred microgestures to VR controllers and 22 of 24 users preferred microgestures to a pinch-based locomotion technique. Additionally, we present a scrolling design (applicable to VR and AR) that maps the discrete microgesture signals to scrolling for both short and long distances.

Our contributions are as follows:

- A detailed guide for machine learning model training (gesture design, closed-loop data collection, data annotation, model architecture, and evaluation) that achieves high recognition accuracy.
- A set of design guidelines based on lessons we learned to aid developers and researchers in building their own gesture recognition system.
- A pair of explorations on how we apply thumb-based microgestures to VR/AR use cases: locomotion and scrolling.

2 RELATED WORK

2.1 Microgestures

The design space of single-handed microgestures have been mapped out extensively [7, 9, 28, 38]. Thumb-to-finger microgestures primarily consist of thumb taps and swipes on the volar (palm) side and radial (thumb) side of each opposing finger. To recognize thumb-to-finger microgestures, researchers have considered wearable or acoustic sensors [10, 13, 57, 71, 72, 75], short-range radar [41] and infrared [16] sensing, as well as vision-based approaches using camera rings [40, 65], wrist cameras [34, 58], chest camera [42], shoulder-mounted depth sensor [63].

The thumb-to-volar (palm) side interactions are commonly used for text entry as the palm represents a larger surface area and all finger segments are visible tap landmarks [25, 58, 65, 72, 73, 75]. However, we focus our attention on thumb-to-index microgestures where the radial (thumb) side of the index finger acts as an interactive touch surface [57, 66, 71] including for typing [39]. The ring is a common form factor for sensing thumb-to-index microgestures. DualRing [40] demonstrates small discrete thumb swipes on the index finger distal segment, using finger-worn IMU sensors. TipTap [29] uses wearable sensors to detect taps between thumb and

index finger at precise locations. Recent single rings EFRing [11] and OptiRing [68] use electric-field sensing and camera-based sensing to recognize similar thumb swipes and taps as STMG as well as continuous sliding.

Our gesture recognition works on top of hand skeletal poses, which is extracted from monochrome camera images found on current VR headsets and does not require the user to wear additional wrist or finger sensors. Since vision-tracking at a distance is less precise, we chose microgestures that are discrete and slightly larger in size and performed primarily on the radial side of the middle segment of the index finger. Despite these differences, our gestures still allow the hand to be held in a relaxed posture and in view of the cameras, much like holding a game controller. EFRing achieved cross-user 85.2% accuracy and OptiRing achieved 93.1% accuracy (99.8% for stateful pinch) using a heuristic/ML hybrid approach. Our 95.1% F1 score is competitive against finger-worn sensing approaches.

2.2 Vision-Based Hand Gesture Recognition

Hand gesture recognition has been studied by the vision community for the last three decades [32, 50, 74]. Similar to action [45, 60] and sign language recognition [36], gesture recognition is a spatio-temporal classification task. Classical approaches divide this problem into two steps, first extracting spatial representations from frames, then learning a discriminator between gesture classes.

Based on their input modality, researchers devised and utilized various spatial feature representations [55, 59]. Early approaches used edge descriptors to localize the hand and describe its shape from RGB images [3, 64]. Availability of consumer depth cameras, such as Kinect [76] and Leap Motion [19], and the subsequent depth-based pose estimation methods [31, 62] addressed the problem of hand localization. Accurate hand pose estimation from depth images further encouraged the development of skeleton-based gesture recognition datasets [15, 18] and methods [43, 46, 54]. The recent advances in deep learning alleviated the requirement of depth cameras for accurate pose estimation, enabling the use of RGB[6, 44] or monochrome[20, 22] cameras for skeletal-based gesture recognition. Compared to visual features [52], skeletal representation has the benefit of removing the redundant appearance information, such as background, clothing and skin tone variants, allowing the models to generalize unseen environments and demographics [26]. However, use of skeletal input bounds the performance of the gesture recognition system with the accuracy of the utilized pose estimation method [51]. Taking these trade-offs into consideration, in this work we utilize a monochrome-based pose estimation method [22] and train our models using skeletal inputs to leverage their generalization capabilities.

Once the spatial representations are extracted, the next step is to learn a discriminator between gesture classes. Earlier approaches utilized a variety of classification strategies, such as building a bag of features [69] or creating templates [4], and using classical classifiers, i.e., SVM [12] and Random Forest [27], or more sophisticated temporal modeling approaches such as HMMs [30] or CRFs [70].

More recently, modern deep learning-based temporal models have shown significant improvements over the classical classifying techniques [52]. These approaches can be gathered under three categories, namely Recurrent Neural Networks (RNNs), Temporal Convolutional Networks (TCNs), and Transformers. RNNs have been utilized for gesture recognition in combination with convolutional networks to jointly learn spatio-temporal feature in an end-to-end manner [8]. However, RNNs and its variants take longer to train as they can only process inputs sequentially and their receptive field cannot be explicitly controlled. To address these issues researchers proposed TCNs [37], which learn discriminative features from the changes in the spatial representations via convolution operations over an architecturally defined receptive field, which has been subsequently applied to gesture recognition [23]. Transformers [67] further improved the temporal modeling capabilities of gesture recognition models [5, 14]. Although transformers are the state-of-the-art temporal modeling technique [33] and our initial experiments with an optimized transformer architecture [61] gave small improvements over our TCN model (96.0% F1 score on the test set described in Section 4.4.1), current headsets have not been optimized to run the computationally heavy multi-head self-attention module and thus transformers are harder to run in real time. Hence, in this work we chose to present TCNs as our temporal modeling approach to classify microgestures from skeletal inputs.

3 MICROGESTURE DESIGN

Existing hand input in products generally use pinch for selection in indirect interactions (e.g., Meta Quest headsets [48], HTC Vive [24], Microsoft Hololens 2 [49], Apple Vision Pro [1]). To distinguish between different pinch-based interactions, gestures need to be designed with an additional hand or arm movement, e.g., pinch and move up/down to scroll. The thumb-based microgestures (Figure 2) supported by STMG increase the gesture space with smaller low-calorie thumb movements where the hand can be stationary and the arm and shoulder relaxed. The hand starts in a handshake orientation with the fingers slightly curled as if holding an imaginary controller. The radial side of the index finger is the interaction surface for the thumb to swipe against, much like the surface of a smartphone.

These thumb swipes map naturally to the four cardinal directions; we will refer to them as left, right, forward, and backward. These swipes are akin to familiar inputs including gaming directional pads or joysticks, television remotes, arrows on a keyboard and swipes on a phone. A swipe is one fluid motion that begins with the thumb making contact with the side of the index finger, quickly sliding in one of the four directions, and then lifting. For the forward and backward swipes, the index finger may move in the opposing direction of the thumb to more easily generate a larger relative motion between the thumb and index fingers. Note we initially referred to the forward and backward swipes as up and down swipes; however, we found that gave some users the misconception that the swipes were performed vertically like an up swipe being a thumbs up gesture or a down swipe means tucking the thumb down into their palm.

Additionally, we support a thumb tap, which consists of the thumb contacting the side of the index finger and then lifting in one motion. We opted not to break the thumb taps into separate thumb contact and thumb lift events for two reasons. 1) On the modeling side, recognizing a full contact and lift motion is a more

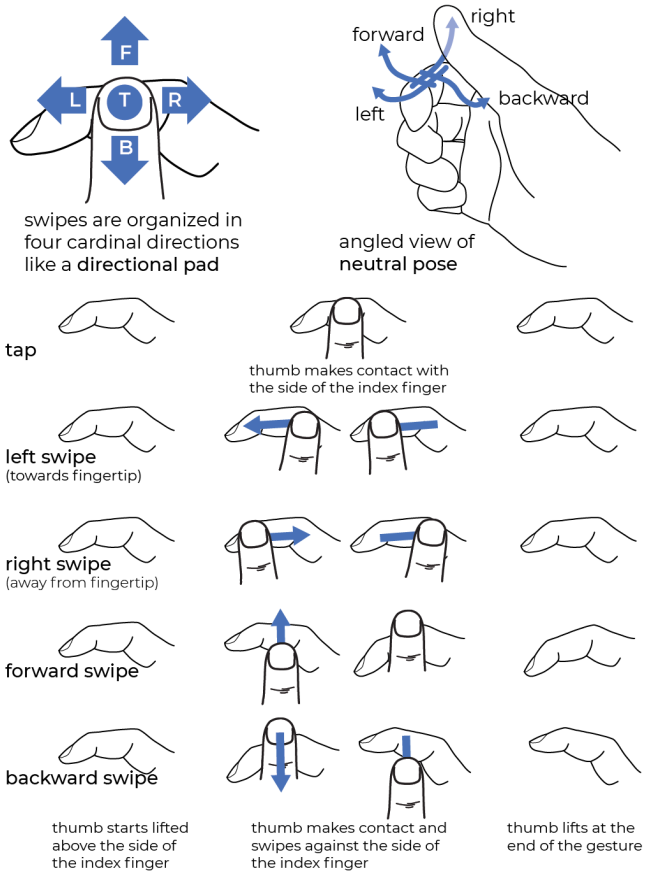

Figure 2: The STMG microgesture set consists of thumb motions performed against the radial side of the index finger. They include thumb tap and four directional swipes in the left, right, forward, and backward directions.

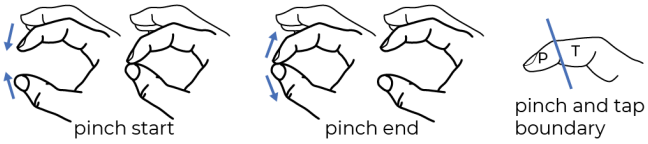

Figure 3: The STMG gesture set also contains pinch start and pinch end. The boundary that determines whether the thumb contacting and lifting against the index finger is a pinch or a tap is the index distal joint.

robust signal compared to separate contact and lift motion signals and 2) on the user side, triggering events on a thumb contact means a user cannot rest their thumb for fear of inadvertently triggering a command.

This microgesture set also benefits from self haptics. The thumb makes contact with the index finger and provides the user kinesthetic feedback to help guide the thumb's motion. We should note that in our vision-based approach, the amount of pressure the thumb applies against the index finger does not matter; they just need to visually make contact.

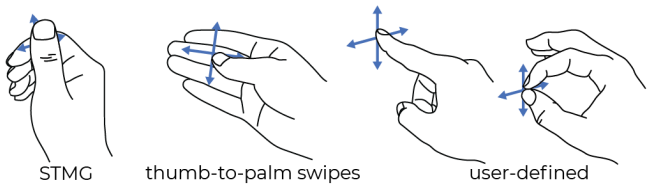

Figure 4: We compared STMG gestures to thumb-to-palm swipes to user-defined gestures. Some user-defined gestures include in-air finger swipes and pinch and drags.

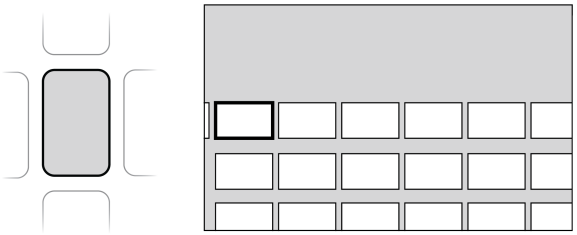

Figure 5: Participants could try navigating 2D grid interfaces using different gestures with Wizard of Oz. Left: Participants could swipe through content like on a phone. Right: Participants could traverse a grid via directional gestures.

Also, to more easily integrate microgestures with pinch-based input, we wanted the model to learn the difference between swipes and pinch. Without a model that understands both types of gestures, interference between pinches and swipes are likely to occur. For example, it is easy for the thumb to touch the index fingertip at the start of a right-handed right swipe. A unified model would avoid the false recognition of a pinch. To accommodate users who like to pinch with their thumb hitting the side if the index fingernail rather than the index fingerpad, we use the distal joint as the boundary between a pinch and a tap (Figure 3).

In short, the STMG microgesture set benefits from:

- Small motions - The motion required for the thumb to swipe or tap against the side of the index finger is much smaller than whole hand or arm motions, but still salient enough to outweigh hand tracking noise.
- Eyes-free - Directional swipes can be performed without looking.
- Self haptics - Self contact between the thumb and index finger gives the user and model a clear signal of intent and reinforces the thumb motion.
- One-handed - The other hand is free or both hands can independently perform gestures.
- Familiar hand pose - The neutral hand pose is similar to commonly performed hand poses such as when shaking hands or holding a game controller.

3.1 STMG vs Palm Gestures vs User-Defined Gestures

To vet our STMG gestures, we compared them against thumb swipes on the palm and user-defined gestures (Figure 4). We had 13 participants first define their own gestures for navigating 2D grids (e.g., Instagram and Netflix-style interfaces shown in Figure 5). We then used a Wizard of Oz approach where the participants would perform each set of gestures and the moderator would act as the "wizard" and manually navigate the UI in response to the participants' gestures. This allowed the participants to experience how each gesture set would work in real time.

Of the 13 participants, seven preferred STMG gestures. Reasons included it feeling most similar to using a controller. Four preferred their own gestures, three of which were in-air index finger swipes and one who imagined a pinch and drag. Without a "surface" to indicate intention, we consider in-air swipes to be challenging to recognize reliably. Finally, two preferred swiping on palm. Thumb swiping on the palm was considered more straining than swiping on the side of the index finger. This is consistent with Tip-Tap's reachability study [29] that found a wider comfort interaction area on the side of the index finger rather than on the palm side. Given the participant feedback, we felt confident that our STMG gestures were a suitable set of discrete, directional gestures.

3.2 Heuristic vs ML Recognition

An effective recognizer should generalize to all users. This is challenging as for a single gesture there may be a wide range of user variation (Figure 6). A recognizer not only needs to distinguish between several different gestures but also capture all the variations of a single gesture across users.

Although a thumb swipe may sound simple, we found significant variation in how users coordinate their thumb and index finger motions. A heuristic recognizer would need to account for where the thumb contacts the index finger, how high the thumb lifts, the angle in which the thumb moves, and the reactive motion of the index finger. Additionally, the skeletal pose provided by hand tracking is not perfect, and the recognizer would need to be adapted to the different noise and accuracy profiles of different hand trackers. We initially formulated a heuristic recognizer based on hand mesh collision detection, but found that tuning thresholds for each gesture per hand tracker was error prone and unsustainable. See Section 4.4.4 for results comparing our heuristic and machine learning approaches.

Hence, we take a machine learning approach. By collecting training data from a wide range of participants, we can train a model to learn the variations of a gesture and distinguish them from other gestures in the set. The model can also learn what motions are not gestures. If the hand tracker changes or the source of the input features changes, we can simply retrain the network on the recomputed hand skeletons or input features.

4 MACHINE LEARNING PIPELINE

Training our machine learning model[1] involved collecting a dataset with ground truth annotations, designing the machine learning model, including network architecture and loss functions, and evaluating the model via offline metrics. In this section we summarize each of these steps, report quantitative results, and share design guidelines and lessons learned.

[1]Visit https://github.com/facebookresearch/stmg for information on how to try our model on Meta Quest.

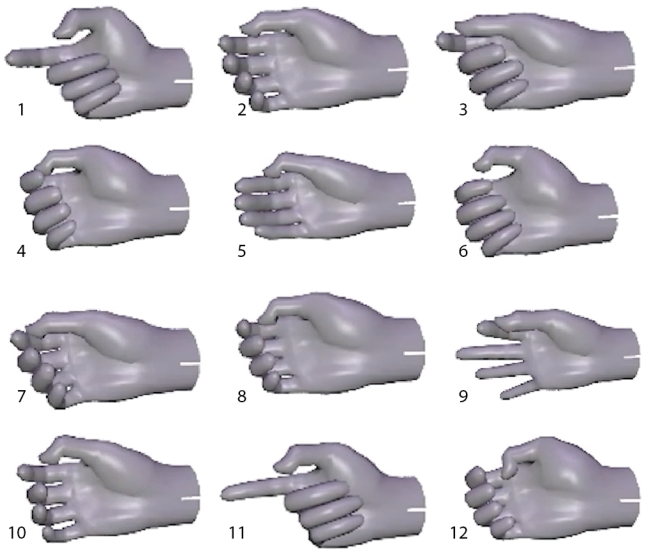

Figure 6: These examples are the hand pose at the peak of the backward thumb swipe motion from 12 different users. They illustrate the amount of variation a recognizer has to handle. For example at the end of the swipe, user 1, 5, and 11 extend their index finger a great amount, while user 6 keeps their index finger more curled. User 12 dips their thumb more. Users 4, 7, and 9 have smaller swipes.

4.1 Data Collection

We collected 228 participants 208 for training and 20 for testing, with a 51%/49% female/male split and a 86%/14% right/left hand split. On average we collected 40 minutes of gesture sequences per participant. This amounted to 100 gestures sequences each containing 8-12 gestures. We also collected 20 hard negative sequences each containing zero gestures. We only collected motions of their dominant hand. Participants were asked to hold their hand between their chest and lower abdomen heights. Some tasks also require the participant to aim a virtual ray from their hand to control a cursor on the 2d monitor. Because camera viewing angle can affect the characteristics of the skeletal tracking while performing gestures, having the participant place their hand in different locations adds camera-angle tracking variation to our training data. Our training and testing splits are outlined in the Results section.

Our data collection setup includes a motion capture system consisting of 37 OptiTrack [53] cameras (29 Prime 17W and eight Prime 13W) and four head-mounted fisheye monochrome 2D cameras. The head mount does not contain a display; instead, the participant views the collection tasks on a 2D monitor positioned directly in front of them. Figure 7 shows the capture setup.

4.1.1 Collection tasks. We collect realistic training data by asking participants to perform tasks with the STMG gestures. We provide

Figure 7: We use motion capture and a heuristic-based recognizer to provide real time feedback to the participant. Markers are attached directly to the skin to allow free movement of the hand. We also calibrate the space so that for certain tasks a virtual ray from the participant's hand position can control a cursor on the 2D monitor.

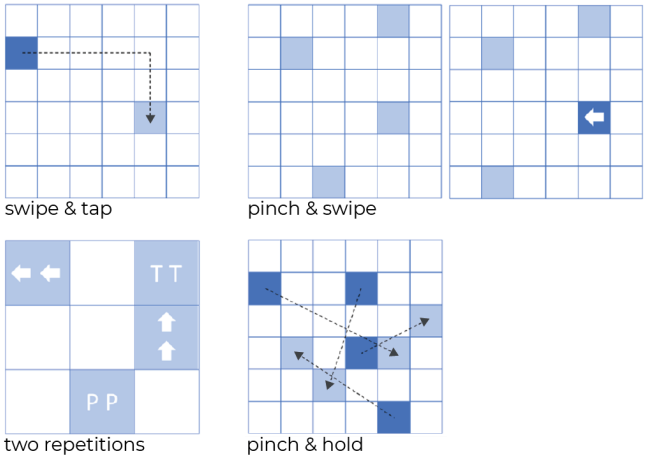

Figure 8: Our task designs for capturing sequences consisting of 1) swipes and taps 2) pinch and swipes 3) gesture repetition 4) pinch and hold. Block arrows indicate swipe direction, T denotes thumb tap, P denotes pinch (start and end).

the participants with real-time feedback in an interactive system, i.e., closed-loop data collection. There are two components that allow us to provide real-time feedback without a complete working and fully generalized ML model.

First, our tasks are designed to prompt the participant to perform a predetermined sequence of gestures, so we know the expected gesture sequence. Our four collection tasks (Figure 8) are as follows:

- *swipe & tap* - the user performs swipes to traverse a grid via a predetermined path and finishes with a thumb tap.
- *pinch & swipe* - the user aims the ray at a blue square and pinches, which then reveals a subsequent swipe to perform. Thus, we expect a pinch then the prompted swipe, etc.
- *two repetitions* - the user aims the ray at a square and performs the corresponding gesture twice. Since we know where the ray is pointing, we know which gesture pair the user should be performing.
- *pinch & hold* - the user drags one of four source squares to one of the four target squares until no squares are left. Pinch and holds create sequences where pinch start and pinch end events are spaced further apart in time.

Second, we use heuristic-based microgesture recognition from an additional motion capture hand tracking system. We have participants wear motion capture markers (small 2 mm retroreflective markers adhered directly to the skin) in order to run a high quality hand tracker [21]. We then detect gestures using a heuristic-based detection method (i.e., programmatically checking the position of the finger joints to see if we think a gesture has occurred). Because we know the next gesture the participant is being prompted to perform, we can be more generous with our heuristic recognition, i.e., the recognizer can "cheat" by accepting an ambiguous gesture if it is close enough to the prompted gesture, without concern for false positives. By being generous (or loose) with the recognition at this stage, we encourage participants to perform gestures naturally.

To balance eliciting user variation and gesture correctness, we rely on moderators to monitor the participants and the mocap heuristic to enforce certain criteria, such as requiring that swipes are performed on the side of the index finger, that swipes are at least 10% the length of the sum of the index and thumb bone segments, and that the thumb lifts at least 5 mm at the end of a gesture. Otherwise, the participant is free to perform the gestures as they feel most comfortable.

4.1.2 Hard negatives. Just as important as collecting samples of participants performing microgestures is the collection of sequences where participants perform hard negative sequences. We collect sequences of participants performing "almost microgestures." This is to carve out the negative space of gestures so the network can better learn where the boundary is between a gesture and non gesture. An example of a negative motion is asking the participant to slide their finger back and forth against the index finger without lifting, or conversely, waving their thumb back and forth above the index finger without touching. This is to contrast against the requirement that our microgesture swipes require a thumb contact to begin the swipe and a lift at the end. Another example is collecting tap and hold sequences as hard negatives so that the network can learn that a thumb tap requires both the thumb down and lift motion and *not* just the thumb lift motion (that would occur at the end of a thumb tap and hold).

4.1.3 Annotation (ground truth labeling) of microgesture data. We use our motion capture heuristic recognizer as a first pass for generating ground truth framewise labeling. If the ordering of gestures recognized matches the prompted gesture ordering, then we consider the annotation correct. However, participants can make mistakes even in a prompted setting and motion capture hand tracking system can introduce errors. Hence, we introduce manual inspection checks focused on areas where the motion capture heuristic recognizer disagrees with the expected gesture ordering.

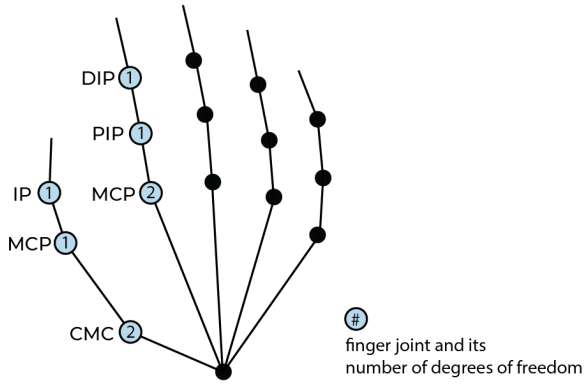

Figure 9: Our model input features are eight joint angles, highlighted here in blue. Four from the thumb: interphalangeal, metacarpophalangeal, carpometacarpal (flexion), carpometacarpal (abduction) and four from the index finger: distal interphalangeal, proximal interphalangeal, metacarpophalangeal (flexion), metacarpophalangeal (abduction).

4.2 Machine Learning Model

In this section we introduce the machine learning model we developed to recognize microgestures. Our objective is to learn the conditional probability $p(\mathcal{G}|\mathcal{S})$ of predicting a gesture class for each frame $\mathcal{G} = (g_1, ..., g_T)$ given the corresponding skeletal inputs $\mathcal{S} = (s_1, ..., s_T)$.

4.2.1 Skeletal Inputs. As our gestures only rely on the motions of the thumb and index fingers within the hand space, we use the joint angles of thumb and index fingers as the input features, s_t. Each finger has four joints leading to an eight-dimensional feature vector (Figure 9), i.e., $\mathcal{S} \in \mathbb{R}^{Tx8}$. The joints were generated using UmeTrack [22] on the collected monochrome images, though hand skeletal joints could come from other sources or sensors. Joints were updated at a capture rate of 30 frames per second. With respect to handedness, if the same joint angles are applied the left and right hands, their poses will be mirrors of each other. For example a left-handed right swipe (towards the index fingertip) is the same as a right-handed left swipe (towards the index fingertip). So to train a one-handed model we simply convert left-handed data into right-handed data by swapping the left and right swipe labels. This creates a model that can be run on either hand. Note when running the model on the left hand, the left and right swipe outputs will need to be swapped again.

4.2.2 Recognition Network. We formulate estimating $p(\mathcal{G}|\mathcal{S})$ as a classification problem and tackle it using a Temporal Convolutional Network (TCN) [2] to best capture the temporal nature of the thumb and index finger movements, $\mathcal{S}$. Given N gestures to be recognized, the network performs classification with $N + 1$ classes, with the additional "idle/no gesture" class.

Our TCN is composed of 10 temporal convolution modules, each built with two consecutive residual 1D convolution layers. It works in a sliding window manner, and takes the history from previous frames as input $\mathcal{S}_{t-r:t}$ to estimate gesture of current frame g_t, where r represents the receptive field size of our network. To make the network see a longer history, which helps estimation, we use dilated convolutions to increase the receptive field size.

4.2.3 Loss functions. Our loss function is a combination of three loss functions: Connectionist Temporal Classification (CTC) loss ($\mathcal{L}_{CTC}$) [17] (for sequences containing positive samples), Cross Entropy loss ($\mathcal{L}_{CE}$) (for fully negative samples), and a Temporal Alignment loss ($\mathcal{L}_{TA}$) (specifically for pinch start and pinch end, where frame alignment is important).

Due to the sequence-to-sequence nature of $\mathcal{L}_{CTC}$, it does not require framewise gesture labels. By providing just the ground truth sequence of labels, $\mathcal{L}_{CTC}$ will supervise the ground truth gesture order. And by providing enough variety of gesture sequence orderings, whereby it tries to output gestures in the same order as ground truth, $\mathcal{L}_{CTC}$ will by necessity learn the gesture alignment and predict the frame where the gesture has occurred. As it can be somewhat unclear precisely which frame a gesture motion has "completed," we let the model decide via $\mathcal{L}_{CTC}$. Note this can lead to the feeling of recognition latency, if the network predicts the gesture a few frames later than a user thinks they completed it.

We use $\mathcal{L}_{CE}$ loss for fully negative sequences, i.e., hard negative sequences, as $\mathcal{L}_{CTC}$ requires sequences to contain positive samples. Since our negative sequences contain no positive gesture samples, we supervise our network with CE loss to predict "idle/no gesture" class for every frame of these sequences.

$\mathcal{L}_{TA}$ loss penalizes pinch predictions that are far away (or nonexistent) from the ground truth framewise label. For each prediction, we add a penalty if there is no matching ground truth label within j frames behind or k frames ahead. Symmetrically, for each ground truth label, we add a penalty if there is no matching prediction within k frames behind or j frames ahead. No penalty is added if a match is found. We use j=4 and k=2; we use a larger j than k to be more tolerant to predictions that occur after ground truth.

We use $\mathcal{L}_{TA}$ (in addition to $\mathcal{L}_{CTC}$) on pinch gestures for two reasons. First is we want our model to predict pinch start and end events as close to ground truth pinch start and end frames as possible, because when a pinch starts and ends is well defined: pinch start occurs on the frame when the fingertips first touch and pinch end occurs on the frame when the fingertips separate. Note that it can be hard to detect the first frame of touch and separation due to hand tracking noise, so $\mathcal{L}_{TA}$ accommodates this by allowing a small window in which no penalty is incurred. The second reason for needing $\mathcal{L}_{TA}$ is $\mathcal{L}_{CTC}$ has the problem of prediction collapse. Because pinch end always follows pinch start, $\mathcal{L}_{CTC}$ can be "lazy" and collapse the two gestures into adjacent frames, since from its perspective it got the gesture order correct. Figure 10 illustrates the difference between $\mathcal{L}_{CTC}$ and $\mathcal{L}_{TA}$.

We train our networks by minimizing the joint loss term $\mathcal{L}$, which is a weighted sum of $\mathcal{L}_{CTC}$, $\mathcal{L}_{CE}$, and $\mathcal{L}_{TA}$:

$$\mathcal{L} = \lambda_{CTC}\mathcal{L}_{CTC} + \lambda_{CE}\mathcal{L}_{CE} + \lambda_{TA}\mathcal{L}_{TA} \quad (1)$$

where λ_{CTC}, λ_{CE}, and λ_{TA} are empirically set hyper-parameters, which decide the importance of each loss function.

4.3 Implementation and Evaluation Details

4.3.1 Training. We implemented our models using PyTorch framework [56]. We use Adam [35] optimizers with a batch size of 128,

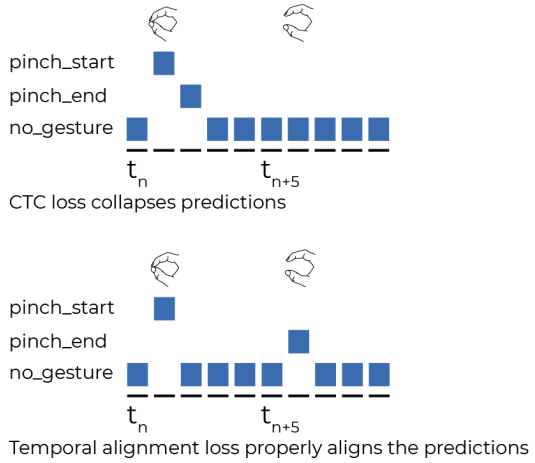

Figure 10: To prevent CTC loss prediction collapse for pinches, we rely on framewise ground truth labels and temporal alignment loss to force the pinch start and pinch end predictions to their respective temporal locations. The blue rectangles indicate the gesture with the highest probability at time *t*. (Probabilities of other gestures are not shown.)

a learning rate of 10^{-3} (β_1=0.9, β_2=0.999), and a weight decay of 10^{-4}. We utilize a multi-step learning rate scheduling, which lowers the learning rate every 200 epochs by a factor of 0.75, and optimize our networks for 1200 epochs. We also apply dropout with 0.2 drop rate on temporal convolution modules.

4.3.2 Network Details. Our network is composed of temporal convolution layers with kernel size of three. Our 10 temporal convolution modules are built with (32, 32, 32, 64, 64, 64, 64, 64, 64, 64) feature channels and dilation sizes of (1, 2, 4, 6, 8, 1, 2, 4, 6, 8). Loss weights λ_{CTC}, λ_{CE}, and λ_{TA}, are set to 10, 1, and 1, respectively.

4.3.3 Data Augmentation. To increase the variety of our training set we employ data augmentation. During each epoch there's a 50% chance we apply a speed augmentation to a sequence and an independent 50% chance we apply a index finger curl augmentation to a sequence if it does not contain any pinches. For a speed augmentation, we speed up or slow down the sequence by up to 20% by interpolating the joint angles. For the index finger curl augmentation, we curl or uncurl the index finger joint angles by up to 10%. Changing the amount of index finger curl adds variety to where the thumb makes contact on the index finger during swipes and taps. We purposely avoid using the curl augmentation on pinch sequences as that may push fingertips through each other or create separation between fingertips when they are supposed to be touching.

4.3.4 Performance Metrics. We use our ground truth framewise labels to count true positives (TP), false positives (FP), and false negatives (FN). We consider a prediction a true positive if it matches a ground truth label within 30 frames (1 second). This window is sufficiently large to accommodate fuzziness in the ground truth label as well as the prediction. Otherwise, a made prediction without a match is a false positive and a ground truth label without a match is a false negative. We use the standard formulas to calculate precision, recall, and F1 scores for each gesture class. The overall scores we report are class normalized, i.e., we average the gesture class scores.

$$\text{precision} = \frac{TP}{TP + FP} \quad \text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Although we have ground truth framewise labels, it is worth discussing how to approximate these metrics without the need for framewise labels, as training with CTC loss does not require framewise labels either. We can approximate the number of prediction errors by comparing the ground truth gesture order and the predicted gesture order. If we think of each gesture as a character and the gesture order as a string, we can compute its Levenshtein distance: the number of insertion, deletion, and substitution operations to convert one string to another string. Thus, we would get the number of prediction mistakes compared to the ground truth. Then, if we backtrace the dynamic programming operation used to compute the Levenshtein distance, we can determine the alignment between strings, which tells us the insertion, deletion, and substitution operations for the alignment. An insertion means the prediction missed a gesture: a false negative. A deletion means the prediction incorrectly predicted a gesture: a false positive. And, a substitution means one gesture was confused for another gesture: false negative of the missed gesture and a false positive of the predicted gesture. Characters aligned correctly are true positives.

There are two caveats with this approach. First, there can be multiple sets of operations, or alignments, that achieve the same Levenshtein distance and second, there is no guarantee the alignment of the characters in the two strings align temporally. For the former an option is to compute all possible minimum alignments and average the count of each unique alignment operation by the number of possible alignments. For the latter, ultimately framewise ground truth labels are needed to get an exact error count. In practice we found the single path backtracing approach gave an error count very close to using the error count from our ground truth framewise labels. In other words, if ground truth framewise labels are not readily available or annotation is difficult, the backtracing error approximation provides a close representation of the true error and can be used to guide model improvement strategies.

4.4 Quantitative Experiments

In this section we share our quantitative experiments. We first introduce the attribute balanced test set we've created and compare our models performance on this test set against a cross-validation setup. We then ablate the use of data augmentation, and evaluate its effects on our test set performance. We also compare our model against a heuristic baseline to display the benefits of our ML approach. Finally, we set a virtual upper bound for our model by training it using motion capture hand tracking, thus disentangling the effectiveness our approach and the performance of monochrome based hand tracking.

4.4.1 Test Set Creation and Performance: We evaluate the performance of our ML model via two approaches, using a test set and

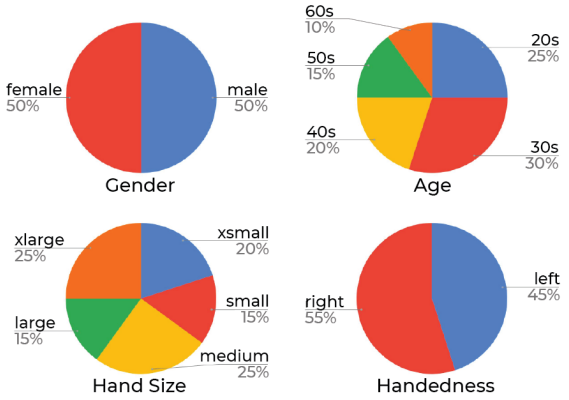

Figure 11: Distribution of our test set over gender, age, hand size, and handedness.

via cross validation. We created a 20-user test set from our 228 participant dataset (Described in Section 4.1). To capture a diversity in skeletal motions and hand skeletal structure, we balanced our testing set in terms of gender, age, hand size, and handedness (see distributions in Figure 11). We train our model using the 208 participants that are not included in the test set and evaluate the model on the test set.

Table 1: Overall and individual gesture recognition performance on our 20-user test set.

	F1 Score	Precision	Recall
All Classes	95.1	95.0	95.2
Left	94.9	94.1	95.7
Right	95.6	95.5	95.6
Forward	95.9	95.7	96.1
Backward	96.4	97.0	95.9
Tap	90.8	90.5	91.1
Pinch Start	95.5	95.3	95.7
Pinch End	96.5	96.5	96.5

As can be seen in Table 1 our model achieves 95.1% class normalized F1 score on our test set, having a balanced precision and recall rate. We also report per-class recognition performance and show the confusion matrix in Figure 12. All gesture classes reach over 95% F1 score besides Tap, which has 90.8% F1 score. We believe this is due to Taps being most impacted by hand tracking quality. If a true thumb Tap is too small it can be hard to differentiate from hand tracking noise, lowering recall. Similarly, tracking noise can also create the appearance of a false Tap, lowering precision. Tap is the least common class in our training set (12,233 Tap instances vs 22,433 Pinch instances), which also biases our model training. These results will guide the future data collection protocols, where we will include more taps of different sizes, to help the network differentiate these motions from tracking noise and other gestures.

We also share per-user gesture recognition performance of our test set (Table 2). Eleven out of 20 were above 95% F1 Score, some reaching as high as 98.7%, and 19 out of 20 were above 92% F1 score. Our model performed worst on user 05, achieving 80.7% F1

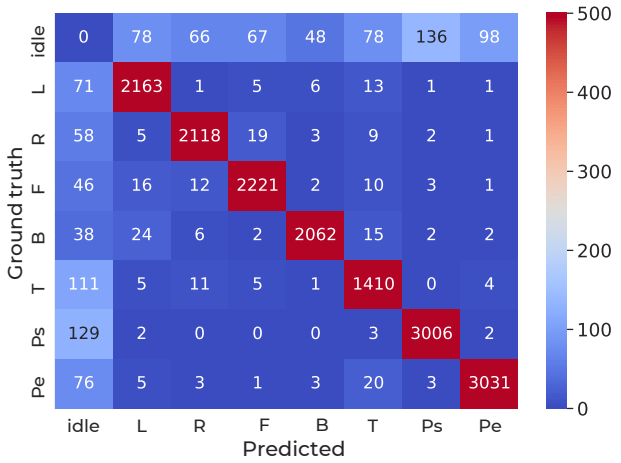

Figure 12: Shown here is the confusion matrix of our model evaluated on our 20-user test set, which corresponds to Table 1.

score. Upon visual inspection of user 05's sequences, we discovered that their gesture style was an outlier in the dataset, producing microgestures which are twitchy and smaller than the monochrome tracker could resolve reliably. As this outlier style is not related to the attributes, we report per attribute results from our test set excluding user 05.

Table 2: Per-user gesture recognition performance of our model.

ID	F1	ID	F1	ID	F1	ID	F1
01	92.6	06	**96.1**	11	**97.9**	16	**97.5**
02	93.0	07	**96.1**	12	94.2	17	92.0
03	94.6	08	**97.1**	13	94.7	18	**98.3**
04	94.2	09	**97.7**	14	**98.1**	19	**95.1**
05	80.7	10	93.1	15	**98.7**	20	**95.2**

Table 3: Per-attribute gesture recognition performance of our model.

Gender		Age				
Male	Female	20s	30s	40s	50s	60s
95.6	95.5	95.4	96.5	94.8	94.7	96.7
Handedness		Hand Size				
Left	Right	XS	S	M	L	XL
95.8	95.4	96.0	95.3	94.1	96.1	96.9

As shown in Table 3, our model performs in the range of 94-96% F1 score for all the attributes, closely bounding the 95.1% F1 score of overall gesture recognition performance. These results suggest that our model is able to generalize over different attributes. Finally, the results of user 05 indicates there is further coverage needed over gesture style variance in further data collections to improve the reliance of our model on outliers.

4.4.2 Cross-validation: In Section 4.4.1 we selected our test set based on user traits. To further scrutinize our model's performance, we employed two-fold cross validation. In this experiment we randomly select 50% of participants for training, while evaluating on the remaining participants, and vice versa.

Table 4: Our models' cross validation results compared against the balanced test set.

	#Train	#Test	F1	Prec.	Rec.
Test Set (Sec. 4.4.1)	208	20	95.1	95.0	95.2
Cross Val. - Fold One	114	114	92.7	92.3	93.1
Cross Val. - Fold Two	114	114	92.5	91.3	93.8

Our model performed similarly over the two folds (See Table 4), achieving 92.7% and 92.5% F1 scores, suggesting even with randomly selected splits our model performs well. Compared to our attribute-balanced 20-user test set, cross validation yields 2.4-2.6% lower F1 score. We believe this is due to models being trained on significantly fewer participants (208 → 114), while being evaluated over a larger test set (20 → 114).

4.4.3 Effects of Data Augmentation: We also ablate the effects of data augmentation described in Section 4.3.3. We train models with different augmentation application rate, ranging from 0%, i.e. no augmentation at all, to 90%. As can be seen in Table 5, increasing the augmentation rate consistently improved our models' recall. Although we see a similar trend for precision until 50%, higher augmentation rates, i.e., 90%, causes the precision to degrade. Due to its balanced precision and recall rate, as well as having the overall highest F1 score, we chose 50% augmentation rate in our experiments.

Table 5: Ablation of different data augmentation rates over the gesture recognition performance.

Aug. %	F1 Score	Precision	Recall
0	94.2	94.4	94.1
10	94.7	94.7	94.7
20	94.9	94.7	95.1
50	**95.1**	**95.0**	95.2
90	94.9	94.3	**95.5**

4.4.4 Hand tracking quality and heuristics: We show our heuristic-based recognition F1 scores in Table 6. Our heuristic is based on mesh collision detection between the thumb and index finger. As meshes and hand tracking are imperfect, distance thresholds are needed to decide if the fingers have contacted. We estimate a threshold on a per sequence basis, by finding the minimum distance between fingers (as an upper bound for threshold setting). Using the collision start and end points, we get the trace of the thumb motion against the index finger. A mostly stationary motion is either a tap or pinch depending on location, whereas a longer motion is a swipe. We then determine the direction of the swipe's trace using the thumb and index bones as the axes of a coordinate frame. As expected, this heuristic performed better on motion capture (73.7%) than monochrome tracking (61.0%). We also tried improving the heuristic by using a neural network classifier (trained on previously collected labeled traces) to determine the gesture from the trace; this hybrid approach improved accuracy. However, ultimately the ML models were far more effective at recognizing microgestures than our heuristics and the gap between the monochrome model (95.1%) and motion capture model (97.0%) was narrowed.

Table 6: A comparison of ML and heuristic-based recognizers on motion capture and monochrome hand tracking (F1 scores shown).

	Machine Learning	Heuristic	Hybrid
Mocap	97.0	73.7	90.4
Monochrome	95.1	61.0	81.6

4.5 Design Guidelines and Lessons Learned

Here we summarize what we learned into a set of guidelines:

Design gestures that accommodate the tracking quality. The quality of the input features, such as the accuracy of the hand joint angles, determines which gestures is more likely to be recognized robustly. The design (Section 3) of the gesture motion size should be larger than the expected noise of the tracking system.

Balance user variation with gesture specificity. One of the goals of collecting data is capturing natural user variability. However, there is a balance between what users are allowed to vary (e.g., how curled their fingers are) and what minimum gesture characteristics are required (e.g., thumb must lift between gestures). Defining these characteristics ahead of time and enforcing them during data collection (Section 4.1.1) will lead to more consistent data, and leaving room for user interpretation will help capture natural user variation.

Design collection tasks that make data easy to annotate. We designed our tasks (Section 4.1.1) so we know what the ground truth labels are beforehand. It can be as simple as dictating to participants the sequence of gestures they should perform. Or if the collection is more interactive, design tasks where participants perform "on rails" or follow an expected path to completion.

Collect negative samples that carve out the "negative" space around the gestures. Just as important as collecting positive samples is collecting negative samples (Section 4.1.2). Anticipate motions users may make that would almost look like gestures and could trigger false positives – collect these motions as negative training data. Real-time testing will reveal more negative samples to collect.

Leverage CTC loss to minimize the need of framewise labels. Using CTC loss (Section 4.2.3) is a good way to quickly start training models without needing precise ground truth framewise labels, which can be time-consuming or difficult to generate. Accuracy can also be approximated without framewise labels by computing Levenshtein distance and backtracing. After a model is well trained with CTC loss, it can bootstrap a project and be used to quickly test the quality of the gesture set or be used during an initial framewise annotation pass for newly collected data.

Use framewise ground truth labels if CTC loss is collapsing predictions. For gestures that always occur in the same order (e.g.,

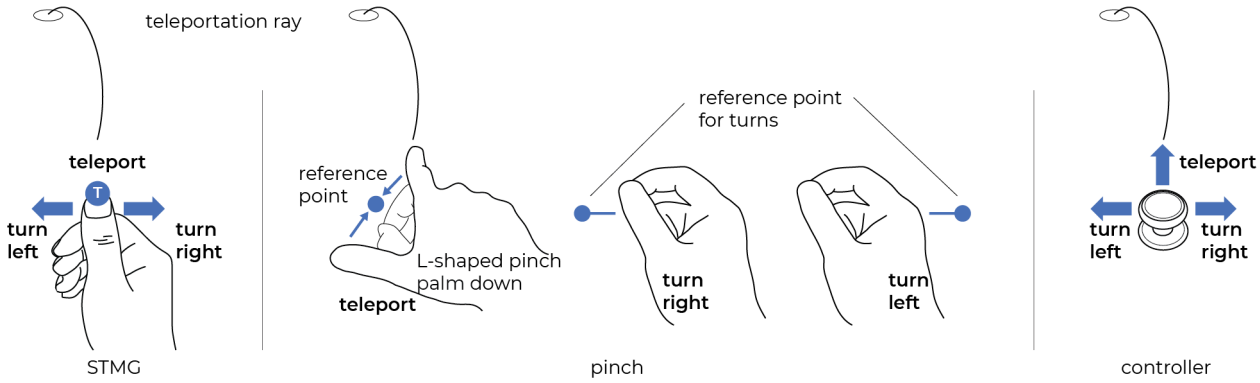

Figure 13: We compared STMG, pinch, and controller locomotion mappings as shown.

pinch start is always followed by pinch end), CTC loss may collapse the predictions (Figure 10). Use a loss function that rewards predictions near a ground truth framewise label or penalizes predictions that are far from a ground truth framewise label.

Add data augmentation to fill gaps in the collected data. Data augmentation (Section 4.3.3) can be used to add variability if collected data is limited. Observe how different people perform the same gesture and what gets varied. Then consider using data augmentation to perturb the existing data in realistic ways to capture a denser range of variability (e.g., full range of gesture speeds).

Use heuristics to filter the model predictions as needed. Sometimes an additional filter may be needed to protect users from triggering gestures from inadvertent motions. For example a heuristic can define a filter or "gate" where the user must hold their hand in a specific orientation (see Locomotion section 5.1) or position to pass the gate and allow a predicted gesture to trigger an operation. A heuristic can also filter out gestures based on attributes such as speed or size. Such filters are likely easier to implement as a heuristic than to recollect data and/or retrain models.

5 APPLICATIONS

Microgestures increase the expressiveness of hand input for VR/AR applications. We showcase two use cases. First is a teleportation-based locomotion technique that allows users to position and orient themselves in a virtual environment with just thumb motions. Second is a design for applying discrete gestures to a continuous scrolling task.

5.1 Locomotion

In VR, a common technique for locomotion is ray-based teleportation and snap turning. With ray-based teleportation, a user can push the joystick forward on a six-degree-of-freedom tracked controller to point to where they want to go and then release the joystick to virtually teleport to that spot. The user can push the joystick to the left or right to snap turn, i.e., rotate at fixed intervals (we use 45 degrees). The purpose of snap turning is to reduce the amount of ocular-vestibular conflict normally exhibited in smooth or continuous joystick turning.

When hands are the primary input, i.e., no controller is present, an existing locomotion method is to combine pinch with hand pose and hand movement to distinguish between each locomotion operation as well as other pinch-based interactions. Specifically, the pinch locomotion method released by Meta in the experience *First Hand* [47], is as follows: while making an 'L'-shaped pose with the thumb and index finger aim the palm facing upwards or downwards (our implementation added downwards for more flexibility) to enter locomotion mode. From there, making a pinch will teleport the user to the ray's target location. If, while maintaining that 'L'-pose, the user rotates the hand such that the palm is facing sideways (i.e., index radial side facing upwards), that will set a reference point in space whereby if the user then moves their hand left of that reference point and pinches, they will rotate left. Similarly, if the user moves their hand right of that reference point and pinches, they will rotate right.

With STMG microgestures we can map a distinct gesture to each locomotion command: thumb tap to teleport to the ray target location, left swipe to snap turn left, right swipe to snap turn right (Figure 1). A simple gate we employed was a user must keep their hand in the handshake orientation, otherwise the microgestures would be filtered out, i.e., ignored. To evaluate our microgesture locomotion design, we compare it to the standard controller method and the pinch-based locomotion as described above. All three locomotion mappings are shown in Figure 13.

5.1.1 Participants and tasks. We recruited 24 participants: 4 females, 20 males and 21 right-handed, 2 left-handed, 1 ambidextrous. Each participant performed a series of tasks using all three conditions using a Meta Quest 2. The order of the conditions was counterbalanced. Before each input condition, participants were shown a video of the locomotion interactions and given verbal descriptions. Then, they put the headset on and had up to two minutes of practice using the respective input. The experiment environment contained a virtual museum with two stories and eight task stations. Participants were asked to move around the virtual space using the three methods and interact with virtual objects and panel interfaces (e.g., manipulating a virtual spray bottle, picking up a torch, rearranging marbles, or navigating a virtual touchscreen). These additional tasks were introduced so participants could experience switching between locomotion and other hand interaction tasks within the same setting.

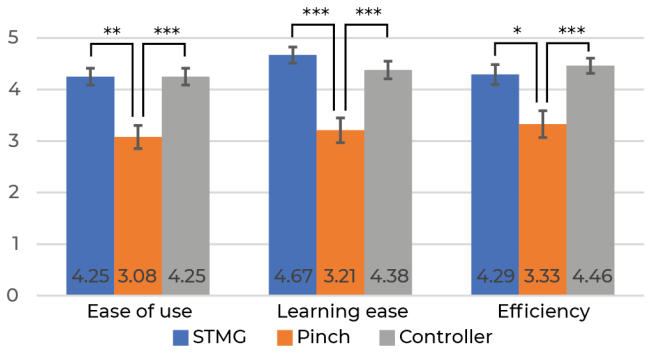

Figure 14: Participants rated each input for its ease of use, learning ease, and efficiency on a scale from 1-5 (with standard error bars). STMG was comparable to controllers and easier to use than the pinch technique.

5.1.2 Results. When asked to rank their preferred input method, 17 of 24 participants preferred STMG microgestures the most. Common reasons from participants included that microgestures was the easiest to learn and felt the most natural. Compared to controllers, microgestures was simpler for switching between locomotion and 3D interactions; it was also more immersive as they could directly use their hands to grab or touch the virtual objects. Six of 24 participants preferred controllers the most as they preferred the reliability of controllers and passive haptic feedback from the joystick. Lastly, the one participant who preferred pinch the most enjoyed that it felt most like a science-fiction interaction.

When only considering the rankings of the hand-based methods, 22 of the 24 participants ranked STMG microgestures above pinch. Participants frequently had trouble remembering how to orient, pose, or move their hand when using pinch, while the simplicity of thumb swipes and tap made them easier to learn and use. This gives us strong indication that expanding the gesture set and mapping the STMG microgestures to appropriate operations will improve the experience of using hand input in VR.

We also asked participants to rate the inputs in terms of ease of use, learning ease, and efficiency (results shown in Figure 14; within-subjects one-way ANOVAs showed input method had a significant effect for each rating comparison: all $p<0.001$). Their responses indicate that STMG was comparable to controllers in terms of ease of use, learning ease, and efficiency (Bonferroni corrected p-values are ≥ 0.74 for all STMG-controller pairwise comparisons), but easier to learn and use and more efficient than the pinch-based method (Bonferroni corrected p-values are ≤ 0.02 for all STMG-pinch pairwise comparisons). We find it encouraging that hand input can match the ease of use of controllers and that a majority of participants even preferred STMG over controllers. We should note that our virtual museum environment was suited more for casual locomotion (e.g., for social VR or virtual tourism), but for fast-twitch scenarios, like intense gaming, controllers may still be warranted.

Through this study we also identified additional ways we can map thumb swipes to locomotion operations. Sometimes participants would teleport too far and wanted a way to step backwards rather than needing to turn all the way around and try again. The forward and backward swipes could map to this forward and backward stepping motion as demonstrated by EFRing [11]. Furthermore, this study focused on one-handed locomotion, but further designs can involve microgestures operated on each hand. For example, one hand can be used to snap turn while the other hand can be used to strafe, i.e., make sideways steps.

5.2 Scrolling

STMG thumb swipes provide an alternative to scrolling than existing pinch-based methods. On Meta Quest, scrolling is performed by using a virtual ray extended from the arm to point and a start pinch to drag the cursor (the ray's endpoint) along the scrolling axis (typically vertically) and then releasing the pinch to end the scroll. For Apple Vision Pro, scrolling is done by making a pinch and flicking the hand vertically. Both require wrist and arm movements.

With thumb swipes, the hand can stay stationary and only thumb and index movements are required. However, STMG only outputs discrete events; discrete thumb swipes match naturally to grid-based navigation, like with the directional pad of television remote control. To investigate microgesture scrolling we conducted two exploratory studies. We started by having ten participants compare STMG to raycast pinch and drag in an Instagram-like application and a browser application. For the Instagram-like application, thumb swipes could navigate both vertically and horizontally. For the browser left and right swipes were shortcuts to move previous and next in browser history. Participants reacted positively when using microgestures to scroll discrete elements like Instagram posts as the could navigate one post at a time. However, a browser is not broken up into regularly spaced grids so we had to map microgestures to a page up or page down operation (i.e., fixed scroll sizes). Participants did not like how page elements could not fit within the window frame, due to the scroll size increments. They did find using a left swipe to go the previous page in history to be effective rather than having to pinch the browser's back button.

Thus, in a second exploratory study we investigated an alternative design for scrolling continuous content. To provide the illusion of continuous scrolling, we use the speed of the thumb swipes to impart momentum to scroll. We calculate speed upon the recognition of a swipe by using a four-frame sliding window across 12 frames of history and finding the highest four-frame average of relative speed between the thumb and index fingertips. A slower swipe will then scroll a smaller amount, whereas a faster swipe will scroll a larger amount. However, we discouraged the use of very fast swipes as this can lead to more erratic or uncomfortable swipe motions. Instead, we were inspired by the speed accumulation found on mobile phone swiping, and use consecutive swipes (i.e., when a new swipe occurs before the previous scroll's momentum ends) to build up speed and generate fast scrolling. To prematurely stop a scroll the user can perform a thumb tap.

We asked eight participants to scroll web pages and pdf documents using STMG (Figure 1) and pinch and drag. We allowed participants to invert the STMG scroll direction if preferred. Half of the participants liked to invert the swipes (swipe forward to scroll down, swipe backward to scroll up), which follows the direction of swiping on a phone and is also common on laptop trackpads. With

each input, they were asked to either scroll to particular location in a web page or find a specific numerical step in an instructional document. Different target locations would require short and/or long-distance scrolls. For short-distance scrolls, six of eight participants still preferred pinch and drag as they appreciated the precision dragging afforded and felt more in control. Thumb swipes could overshoot their target if they were not careful. For long-distance scrolling, six of eight preferred STMG is they could generate speed more quickly and was less prone to fatigue from arm movement as compared to repetitive pinch and drags.

Our exploration of using the discrete STMG microgestures for scrolling reveals two paths to expand the capabilities of thumb swipes for scrolling. 1) For discrete scrolling, recognizing swipe and hold gestures would allow users to hold their thumb stationary after a swipe to repeatedly step through discrete elements (similar to holding an arrow button down). 2) For true continuous scrolling, if we can recognize gesture primitives: thumb touch on index finger, thumb move, and thumb lift events, we can provide continuous trackpad-style scrolling using the index finger as a touch surface and give users more control over short-distance scrolling.

6 CONCLUSION

We presented STMG a system that recognizes microgesture thumb swipes, thumb taps, and pinches with an accuracy of 95.1% F1 score. These gestures give VR/AR users more small, thumb-based gestures to use to control their devices. To aid researchers and developers in building models to recognize their own gesture sets, we detailed each component of our machine learning pipeline and summarized the lessons we learned in a set of design guidelines.

ACKNOWLEDGMENTS

We thank Mark Richardson, Steve Miller, Matt Maas, Kaichen Sun, Matthew Prasek, David Dimond, Kevin Harris, Bradford Snow, Steve Olsen, Melissa Wallace, Noah Perez, Stephanie Aida Gallegos Berrocal, Peizhao Zhang, Xiaofang Wang, Muzaffer Akbay, David Cheng, Matthew Langille, Mike Perry, Marta Egorova, Linguang Zhang.

REFERENCES

[1] Apple. 2023. *Design for spatial input.* https://developer.apple.com/videos/play/wwdc2023/10073

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018). https://doi.org/10.48550/arXiv.1803.01271

[3] Richard Bowden and Mansoor Sarhadi. 2002. A non-linear model of shape and motion for tracking finger spelt American sign language. *Image and Vision Computing* 20, 9-10 (2002), 597–607. https://doi.org/10.1016/S0262-8856(02)00049-5

[4] Necati Cihan Camgöz, Ahmet Alp Kindiroglu, and Lale Akarun. 2014. Gesture recognition using template based random forest classifiers. In *European conference on computer vision*. Springer, 579–594. https://doi.org/10.1007/978-3-319-16178-5_41

[5] Congqi Cao, Yifan Zhang, Yi Wu, Hanqing Lu, and Jian Cheng. 2017. Egocentric gesture recognition using recurrent 3d convolutional neural networks with spatiotemporal transformer modules. In *Proceedings of the IEEE international conference on computer vision*. 3763–3771. https://doi.org/10.1109/ICCV.2017.406

[6] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019). https://doi.ieeecomputersociety.org/10.1109/TPAMI.2019.2929257

[7] Adrien Chaffangeon Caillet, Alix Goguey, and Laurence Nigay. 2023. μGlyph: a Microgesture Notation. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–28. https://doi.org/10.1145/3544548.3580693

[8] Xiujuan Chai, Zhipeng Liu, Fang Yin, Zhuang Liu, and Xilin Chen. 2016. Two streams recurrent neural networks for large-scale continuous gesture recognition. In *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 31–36. https://doi.org/10.1109/ICPR.2016.7899603

[9] Edwin Chan, Teddy Seyed, Wolfgang Stuerzlinger, Xing-Dong Yang, and Frank Maurer. 2016. User elicitation on single-hand microgestures. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 3403–3414. https://doi.org/10.1145/2858036.2858589

[10] Liwei Chan, Rong-Hao Liang, Ming-Chang Tsai, Kai-Yin Cheng, Chao-Huai Su, Mike Y Chen, Wen-Huang Cheng, and Bing-Yu Chen. 2013. FingerPad: private and subtle interaction using fingertips. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 255–260. https://doi.org/10.1145/2501988.2502016

[11] Taizhou Chen, Tianpei Li, Xingyu Yang, and Kening Zhu. 2023. EFRing: Enabling Thumb-to-Index-Finger Microgesture Interaction through Electric Field Sensing Using Single Smart Ring. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–31. https://doi.org/10.1145/3569478

[12] Nasser H Dardas and Nicolas D Georganas. 2011. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and measurement* 60, 11 (2011), 3592–3607. https://doi.org/10.1109/TIM.2011.2161140

[13] Artem Dementyev and Joseph A Paradiso. 2014. WristFlex: low-power gesture input with wrist-worn pressure sensors. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 161–166. https://doi.org/10.1145/2642918.2647396

[14] Andrea D'Eusanio, Alessandro Simoni, Stefano Pini, Guido Borghi, Roberto Vezzani, and Rita Cucchiara. 2020. A transformer-based network for dynamic hand gesture recognition. In *2020 International Conference on 3D Vision (3DV)*. IEEE, 623–632. https://doi.org/10.1109/3DV50981.2020.00072

[15] Sergio Escalera, Xavier Baró, Jordi Gonzalez, Miguel A Bautista, Meysam Madadi, Miguel Reyes, Víctor Ponce-López, Hugo J Escalante, Jamie Shotton, and Isabelle Guyon. 2015. Chalearn looking at people challenge 2014: Dataset and results. In *Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part I 13*. Springer, 459–473. https://doi.org/10.1007/978-3-319-16178-5_32

[16] Jun Gong, Yang Zhang, Xia Zhou, and Xing-Dong Yang. 2017. Pyro: Thumb-tip gesture recognition using pyroelectric infrared sensing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 553–563. https://doi.org/10.1145/3126594.3126615

[17] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*. 369–376. https://doi.org/10.1145/1143844.1143891

[18] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Ben Hamner, and Hugo Jair Escalante. 2012. Chalearn gesture challenge: Design and first results. In *2012 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE, 1–6. https://doi.org/10.1109/CVPRW.2012.6239178

[19] Tibor Guzsvinecz, Veronika Szucs, and Cecilia Sik-Lanyi. 2019. Suitability of the Kinect sensor and Leap Motion controller—A literature review. *Sensors* 19, 5 (2019), 1072. https://doi.org/10.3390/s19051072

[20] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, et al. 2020. MEgATrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics (ToG)* 39, 4 (2020), 87–1. https://doi.org/10.1145/3386569.3392452

[21] Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D Twigg, and Kenrick Kin. 2018. Online optical marker-based hand tracking with deep labels. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10. https://doi.org/10.1145/3197517.3201399

[22] Shangchen Han, Po-chen Wu, Yubo Zhang, Beibei Liu, Linguang Zhang, Zheng Wang, Weiguang Si, Peizhao Zhang, Yujun Cai, Tomas Hodan, et al. 2022. UmeTrack: Unified multi-view end-to-end hand tracking for VR. In *SIGGRAPH Asia 2022 Conference Papers*. 1–9. https://doi.org/10.1145/3550469.3555378

[23] Jingxuan Hou, Guijin Wang, Xinghao Chen, Jing-Hao Xue, Rui Zhu, and Huazhong Yang. 2018. Spatial-temporal attention res-TCN for skeleton-based dynamic hand gesture recognition. In *Proceedings of the European conference on computer vision (ECCV) workshops*. 0–0. https://doi.org/10.1007/978-3-030-11024-6_18

[24] HTC. 2023. *Hand tracking.* https://www.vive.com/us/support/focus3/category_howto/hand-tracking.html

[25] Da-Yuan Huang, Liwei Chan, Shuo Yang, Fan Wang, Rong-Hao Liang, De-Nian Yang, Yi-Ping Hung, and Bing-Yu Chen. 2016. Digitspace: Designing thumb-to-fingers touch interfaces for one-handed and eyes-free interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1526–1537. https://doi.org/10.1145/2858036.2858483

[26] Tao Jiang, Necati Cihan Camgoz, and Richard Bowden. 2021. Skeletor: Skeletal Transformers for Robust Body-Pose Estimation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE Computer

Society, 3389–3397. https://doi.org/10.1109/CVPRW53098.2021.00378
[27] Ajjen Joshi, Camille Monnier, Margrit Betke, and Stan Sclaroff. 2017. Comparing random forest approaches to segmenting and classifying gestures. *Image and Vision Computing* 58 (2017), 86–95. https://doi.org/10.1016/j.imavis.2016.06.001
[28] Nikhita Joshi, Parastoo Abtahi, Raj Sodhi, Nitzan Bartov, Jackson Rushing, Christopher Collins, Daniel Vogel, and Michael Glueck. 2023. Transferable Microgestures Across Hand Posture and Location Constraints: Leveraging the Middle, Ring, and Pinky Fingers. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–17. https://doi.org/10.1145/3586183.3606713
[29] Keiko Katsuragawa, Ju Wang, Ziyang Shan, Ningshan Ouyang, Omid Abari, and Daniel Vogel. 2019. Tip-tap: battery-free discrete 2D fingertip input. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 1045–1057. https://doi.org/10.1145/3332165.3347907
[30] Cem Keskin, Ayse Erkan, and Lale Akarun. 2003. Real time hand tracking and 3d gesture recognition for interactive interfaces using hmm. *Icann/Iconipp* 2003 (2003), 26–29.
[31] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. 2013. Real time hand pose estimation using depth sensors. *Consumer Depth Cameras for Computer Vision: Research Topics and Applications* (2013), 119–137. https://doi.org/10.1007/978-1-4471-4640-7_7
[32] Rafiqul Zaman Khan and Noor Adnan Ibraheem. 2012. Hand gesture recognition: a literature review. *International journal of artificial Intelligence & Applications* 3, 4 (2012), 161. https://doi.org/10.5121/ijaia.2012.3412
[33] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. Transformers in vision: A survey. *ACM computing surveys (CSUR)* 54, 10s (2022), 1–41. https://doi.org/10.1145/3505244
[34] David Kim, Otmar Hilliges, Shahram Izadi, Alex D Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 167–176. https://doi.org/10.1145/2380116.2380139
[35] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). https://doi.org/10.48550/arXiv.1412.6980
[36] Oscar Koller. 2020. Quantitative survey of the state of the art in sign language recognition. *arXiv preprint arXiv:2008.09918* (2020). https://doi.org/10.48550/arXiv.2008.09918
[37] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. 2017. Temporal Convolutional Networks for Action Segmentation and Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.113
[38] Guangchuan Li, David Rempel, Yue Liu, Weitao Song, and Carisa Harris Adamson. 2021. Design of 3d microgestures for commands in virtual reality or augmented reality. *Applied Sciences* 11, 14 (2021), 6375. https://doi.org/10.3390/app11146375
[39] Chen Liang, Chi Hsia, Chun Yu, Yukang Yan, Yuntao Wang, and Yuanchun Shi. 2023. DRG-Keyboard: Enabling Subtle Gesture Typing on the Fingertip with Dual IMU Rings. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–30. https://doi.org/10.1145/3569463
[40] Chen Liang, Chun Yu, Yue Qin, Yuntao Wang, and Yuanchun Shi. 2021. DualRing: Enabling subtle and expressive hand interaction with dual IMU rings. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–27. https://doi.org/10.1145/3478114
[41] Jaime Lien, Nicholas Gillian, M Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. 2016. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–19. https://doi.org/10.1145/2897824.2925953
[42] Christian Loclair, Sean Gustafson, and Patrick Baudisch. 2010. PinchWatch: a wearable device for one-handed microinteractions. In *Proc. MobileHCI*, Vol. 10.
[43] Wei Lu, Zheng Tong, and Jinghui Chu. 2016. Dynamic hand gesture recognition with leap motion controller. *IEEE Signal Processing Letters* 23, 9 (2016), 1188–1192. https://doi.org/10.1109/LSP.2016.2590470
[44] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. 2019. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172* (2019). https://doi.org/10.48550/arXiv.1906.08172
[45] Shugao Ma, Jianming Zhang, Nazli Ikizler-Cinbis, and Stan Sclaroff. 2013. Action recognition and localization by hierarchical space-time segments. In *Proceedings of the IEEE international conference on computer vision*. 2744–2751. https://doi.org/10.1109/ICCV.2013.341
[46] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. 2014. Hand gesture recognition with leap motion and kinect devices. In *2014 IEEE International conference on image processing (ICIP)*. IEEE, 1565–1569. https://doi.org/10.1109/ICIP.2014.7025313
[47] Meta. 2022. *First Hand.* https://www.meta.com/experiences/5030224183773255/
[48] Meta. 2023. *Getting started with Hand Tracking on Meta Quest headsets.* https://www.meta.com/help/quest/articles/headsets-and-accessories/controllers-and-hand-tracking/hand-tracking/
[49] Microsoft. 2022. *Point and commit with hands.* https://learn.microsoft.com/en-us/windows/mixed-reality/design/point-and-commit
[50] Sushmita Mitra and Tinku Acharya. 2007. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37, 3 (2007), 311–324. https://doi.org/10.1109/TSMCC.2007.893280
[51] Amit Moryossef, Ioannis Tsochantaridis, Joe Dinn, Necati Cihan Camgoz, Richard Bowden, Tao Jiang, Annette Rios, Mathias Muller, and Sarah Ebling. 2021. Evaluating the immediate applicability of pose estimation for sign language recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 3434–3440. https://doi.org/10.48550/arXiv.2104.10166
[52] Pradyumna Narayana, Ross Beveridge, and Bruce A Draper. 2018. Gesture recognition: Focus on the hands. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5235–5244. https://doi.org/10.1109/CVPR.2018.00549
[53] NaturalPoint. 2023. *OptiTrack - Cameras.* https://optitrack.com/cameras/
[54] Natalia Neverova, Christian Wolf, Graham W Taylor, and Florian Nebout. 2014. Multi-scale deep learning for gesture detection and localization. In *ECCV Workshop on Looking at People*. https://doi.org/10.1007/978-3-319-16178-5_33
[55] Munir Oudah, Ali Al-Naji, and Javaan Chahl. 2020. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. *Journal of Imaging* 6, 8 (2020). https://doi.org/10.3390/jimaging6080073
[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019). https://doi.org/10.48550/arXiv.1912.01703
[57] Anna Peshock, Julia Duvall, and Lucy E Dunne. 2014. Argot: A wearable one-handed keyboard glove. In *Proceedings of the 2014 ACM international symposium on wearable computers: adjunct program*. 87–92. https://doi.org/10.1145/2641248.2641266
[58] Manuel Prätorius, Dimitar Valkov, Ulrich Burgbacher, and Klaus Hinrichs. 2014. DigiTap: an eyes-free VR/AR symbolic input device. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. 9–18. https://doi.org/10.1145/2671015.2671029
[59] Jing Qi, Li Ma, Zhenchao Cui, and Yushu Yu. 2023. Computer vision-based hand gesture recognition for human-robot interaction: a review. *Complex & Intelligent Systems* (2023), 1–26. https://doi.org/10.1007/s40747-023-01173-6
[60] Mark Richardson, Matt Durasoff, and Robert Wang. 2020. Decoding surface touch typing from hand-tracking. In *Proceedings of the 33rd annual ACM symposium on user interface software and technology*. 686–696. https://doi.org/10.1145/3379337.3415816
[61] Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching-Feng Yeh, Julian Chan, Frank Zhang, Duc Le, and Mike Seltzer. 2021. Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6783–6787. https://doi.org/10.1109/ICASSP39728.2021.9414560
[62] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. 2011. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*. Ieee, 1297–1304. https://doi.org/10.1109/CVPR.2011.5995316
[63] Mohamed Soliman, Franziska Mueller, Lena Hegemann, Joan Sol Roo, Christian Theobalt, and Jürgen Steimle. 2018. Fingerinput: Capturing expressive single-hand thumb-to-finger microgestures. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. 177–187. https://doi.org/10.1145/3279778.3279799
[64] Josephine Sullivan and Stefan Carlsson. 2002. Recognizing and tracking human action. In *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7*. Springer, 629–644. https://doi.org/10.1007/3-540-47969-4_42
[65] Hsin-Ruey Tsai, Cheng-Yuan Wu, Lee-Ting Huang, and Yi-Ping Hung. 2016. ThumbRing: private interactions using one-handed thumb motion input on finger segments. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. 791–798. https://doi.org/10.1145/2957265.2961859
[66] Hsin-Ruey Tsai, Te-Yen Wu, Da-Yuan Huang, Min-Chieh Hsiu, Jui-Chun Hsiao, Yi-Ping Hung, Mike Y Chen, and Bing-Yu Chen. 2017. Segtouch: Enhancing touch input while providing touch gestures on screens using thumb-to-index-finger gestures. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2164–2171. https://doi.org/10.1145/3027063.3053109
[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017). https://doi.org/10.48550/arXiv.1706.03762

[68] Anandghan Waghmare, Roger Boldu, Eric Whitmire, and Wolf Kienzle. 2023. OptiRing: Low-Resolution Optical Sensing for Subtle Thumb-to-Index Micro-Interactions. In *Proceedings of the 2023 ACM Symposium on Spatial User Interaction*. 1–13. https://doi.org/10.1145/3607822.3614538

[69] Jun Wan, Qiuqi Ruan, Wei Li, and Shuang Deng. 2013. One-shot learning gesture recognition from RGB-D data using bag of features. *The Journal of Machine Learning Research* 14, 1 (2013), 2549–2582. https://doi.org/10.1007/978-3-319-57021-1_11

[70] Sy Bor Wang, Ariadna Quattoni, L-P Morency, David Demirdjian, and Trevor Darrell. 2006. Hidden conditional random fields for gesture recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 1521–1527. https://doi.org/10.1109/CVPR.2006.132

[71] Martin Weigel, Aditya Shekhar Nittala, Alex Olwal, and Jürgen Steimle. 2017. Skinmarks: Enabling interactions on body landmarks using conformal skin electronics. In *proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3095–3105. https://doi.org/10.1145/3025453.3025704

[72] Eric Whitmire, Mohit Jain, Divye Jain, Greg Nelson, Ravi Karkar, Shwetak Patel, and Mayank Goel. 2017. Digitouch: Reconfigurable thumb-to-finger input and text entry on head-mounted displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–21. https://doi.org/10.1145/3130978

[73] Pui Chung Wong, Kening Zhu, and Hongbo Fu. 2018. Fingert9: Leveraging thumb-to-finger interaction for same-side-hand text entry on smartwatches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–10. https://doi.org/10.1145/3173574.3173752

[74] Jie Yang, Yangsheng Xu, et al. 1994. *Hidden markov model for gesture recognition*. Carnegie Mellon University, the Robotics Institute.

[75] Cheng Zhang, Anandghan Waghmare, Pranav Kundra, Yiming Pu, Scott Gilliland, Thomas Ploetz, Thad E Starner, Omer T Inan, and Gregory D Abowd. 2017. FingerSound: Recognizing unistroke thumb gestures using a ring. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–19. https://doi.org/10.1145/3130985

[76] Zhengyou Zhang. 2012. Microsoft kinect sensor and its effect. *IEEE multimedia* 19, 2 (2012), 4–10. https://doi.org/10.1109/MMUL.2012.24