Learning on the Rings: Self-Supervised 3D Finger Motion Tracking Using Wearable Sensors

HAO ZHOU*, The Pennsylvania State University, USA
TAITING LU*, The Pennsylvania State University, USA
YILIN LIU, The Pennsylvania State University, USA
SHIJIA ZHANG, The Pennsylvania State University, USA
MAHANTH GOWDA, The Pennsylvania State University, USA

This paper presents *ssLOTR* (self-supervised learning on the rings), a system that shows the feasibility of designing self-supervised learning based techniques for 3D finger motion tracking using a custom-designed wearable inertial measurement unit (IMU) sensor with a minimal overhead of labeled training data. Ubiquitous finger motion tracking enables a number of applications in augmented and virtual reality, sign language recognition, rehabilitation healthcare, sports analytics, etc. However, unlike vision, there are no large-scale training datasets for developing robust machine learning (ML) models on wearable devices. *ssLOTR* designs ML models based on data augmentation and self-supervised learning to first extract efficient representations from raw IMU data without the need for any training labels. The extracted representations are further trained with small-scale labeled training data. In comparison to fully supervised learning, we show that only 15% of labeled training data is sufficient with self-supervised learning to achieve similar accuracy. Our sensor device is designed using a two-layer printed circuit board (PCB) to minimize the footprint and uses a combination of Polylactic acid (PLA) and Thermoplastic polyurethane (TPU) as housing materials for sturdiness and flexibility. It incorporates a *system-on-chip* (SoC) microcontroller with integrated WiFi/Bluetooth Low Energy (BLE) modules for real-time wireless communication, portability, and ubiquity. In contrast to gloves, our device is worn like rings on fingers, and therefore, does not impede dexterous finger motion. Extensive evaluation with 12 users depicts a 3D joint angle tracking accuracy of 9.07° (joint position accuracy of 6.55*mm*) with robustness to natural variation in sensor positions, wrist motion, etc, with low overhead in latency and power consumption on embedded platforms.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile devices**.

Additional Key Words and Phrases: IoT, Wearable, Finger motion tracking, Self-supervised learning

ACM Reference Format:
Hao Zhou, Taiting Lu, Yilin Liu, Shijia Zhang, and Mahanth Gowda. 2022. *Learning on the Rings:* Self-Supervised 3D Finger Motion Tracking Using Wearable Sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 2, Article 90 (June 2022), 31 pages. https://doi.org/10.1145/3534587

*Co-primary authors

Authors' addresses: Hao Zhou, hfz5190@psu.edu, The Pennsylvania State University, USA; Taiting Lu, txl5518@psu.edu, The Pennsylvania State University, USA; Yilin Liu, yzl470@psu.edu, The Pennsylvania State University, USA; Shijia Zhang, scarlettzhang27@psu.edu, The Pennsylvania State University, USA; Mahanth Gowda, mahanth.gowda@psu.edu, The Pennsylvania State University, USA.

2474-9567/2022/6-ART90 $15.00
https://doi.org/10.1145/3534587

1 INTRODUCTION

3D finger motion tracking enables several exciting applications in the areas of augmented and virtual reality (AR and VR) [43, 77], sports analytics [21, 37], sign languages [4], smart healthcare and rehabilitation, etc [39, 119]. For example, the finger motion patterns of a baseball pitcher can be analyzed for providing feedback about pitching an effective knuckleball [24]. Flight simulation, cooking, or artwork lessons require precise motor skills of the hand [47]. Such lessons can be delivered through VR classrooms which benefit from accurate finger motion tracking. Finger motion patterns have also been shown to be predictive of motor neuron diseases [3, 18], and detecting them early can facilitate preventive measures.

Motivated by the above applications, many recent works [13, 46, 86] show the ability to track 3D finger motion using a simple monocular camera. Powered by the latest advances in machine learning (ML) combined with the availability of large-scale labeled training data, precise finger motion tracking is possible. Even though the depth information is unavailable in monocular cameras, the ML algorithms can learn the constraints between locations of different finger joints and extract the 3D pose of finger motion with high accuracy. While cameras have the above benefits, they can be susceptible to occlusions, lighting, and resolution. The tracking is also limited to the line of sight distance of the camera. In contrast, tracking with wearable devices is independent of lighting and occlusions, while providing completely ubiquitous and portable tracking anywhere and anytime. Unlike vision where there is a huge number of high-quality training datasets for developing robust ML algorithms, the wearable devices based applications lack such large-scale training data because of challenges in annotation and labeling [38, 58, 104]. In this paper, we propose a self-supervised learning based system, *ssLOTR* (self-supervised learning on the rings), to dramatically cut down the cost of training overhead for wearable devices.

Fig. 5 shows the wearable device designed for usage in *ssLOTR*. The device consists of five inertial measurement unit (IMU) sensors worn on the fingers like rings, and another IMU worn on the wrist like a smartwatch. Detailed in Section 4, the design is based on the two-layer printed circuit board (PCB) that minimizes the overall size of the electronics. For sturdiness and flexibility in packaging the electronics and interfacing with the hand, we use a combination of Polylactic acid (PLA) and Thermoplastic polyurethane (TPU) materials [98, 125] . Finally, a *system-on-chip* (SoC) with microcontroller is incorporated with integrated WiFi/Bluetooth Low Energy (BLE) [11] for real-time streaming of sensor data. Therefore, we believe the device is portable and comfortable for wearability and usage. Furthermore, there is a graceful degradation in motion tracking accuracy with the number of sensors used, thus providing further opportunities for miniaturizing the sensing device (Section 6). The overall form factor is similar to the commercially available Tapstrap2 [122] device [9, 123, 130] for applications in typing-based and gesture-based interaction with Internet-of-Things (IoT) devices. However, in contrast to Tapstrap2 which focuses on typing and predefined gesture classification, *ssLOTR* performs continuous 3D finger motion tracking for completely free form and arbitrary motions. A sample demo has been published with this paper. In addition, while Tapstrap2's technical details are closed due to the commercial nature of the device, we outline the technical details in developing the device including plans to opensource the hardware platform. Finally, while Tapstrap2 only provides accelerometer data, our device provides 9-axis IMU data (accelerometer, magnetometer, and gyroscope) for fine-grained sensing and precise orientation tracking. The 9-axis IMU data from the 5 fingers and wrist are used for performing 3D finger motion tracking which can enable several applications in AR/VR, sports analytics, smart health, etc.

Other than the commercially available Tapstrap2 sensor discussed above, prior works mainly from academia include data gloves [4], wrist-based sensing using capacitive sensing [129] for classifying 15 gestures, electromyography (EMG) [73, 74], thermal cameras [40], etc. While data gloves can restrict dexterous motion of the fingers thereby compromising usability [103], the thermal cameras can be susceptible to interference from background temperature (sun, heater, etc) and wrist motion [40]. EMG sensors on the other hand need calibration and warming of the skin to be in proper contact with the electrodes which can even take up to 5 minutes during

each instance of wearing, causing usability issues [87, 128, 136]. In contrast to these works, *ssLOTR* differs in the following ways: (i) Performs continuous 3D finger motion tracking with completely arbitrary motion instead of predefined gesture recognition. (ii) Robust to background lighting, occlusions, ambient temperature, etc. There is no need for a separate calibration. (iii) Requires substantially lower overhead of labeled training data.

Such tracking of 3D finger motion with IMU is challenging for many reasons: (i) Unlike vision and speech domains, there is a lack of large-scale labeled training datasets to develop robust ML models with IMU data. (ii) The ML model has to be robust to diversity across users, gender, body masses, motion patterns, etc. (iii) From the usability perspective, the sensor form factor must be convenient to wear and perform natural hand gestures.

ssLOTR exploits several opportunities to address these challenges. (i) *ssLOTR* designs the ML architecture based on self-supervised learning. *ssLOTR* first develops efficient representations from sensor data with only unlabeled training data. As the users continuously use the sensor for applications like AR/VR, generating unlabeled training data is easier with no overhead of annotation and labeling. *ssLOTR* then uses a small amount of labeled training data (≈ 15% of fully supervised training) to learn from the representations and achieves an accuracy closer to fully supervised learning. This achieves a sweet spot in the trade-off between training overhead and accuracy. (ii) The contrastive loss function used for self-supervised learning extracts representations that are robust despite perturbations introduced via data augmentation techniques. This facilitates the ML model to achieve inherent robustness to diversity across users. In addition, domain adaptation techniques to customize the model developed from one user for performing inferences on another user can help handle diversity. (iii) *ssLOTR* designs the hardware platform by exploiting advances in a 2-layer PCB, 3D printing materials, SoC and wireless communication, etc., to minimize the size of the device and maximize portability and ubiquity of the device.

Fig. 1 summarizes the flow of operations in *ssLOTR*. Unlabeled IMU data is first pre-processed with appropriate filtering, coordinate alignment, and gravity subtraction algorithms. Data augmentation techniques in combination with self-supervised learning help create efficient representations from unlabeled training data. The representations are later fine-tuned with small-scale labeled training data, which yields an ML model capable of accurate prediction of 3D finger motion. The details are elaborated in Section 5.

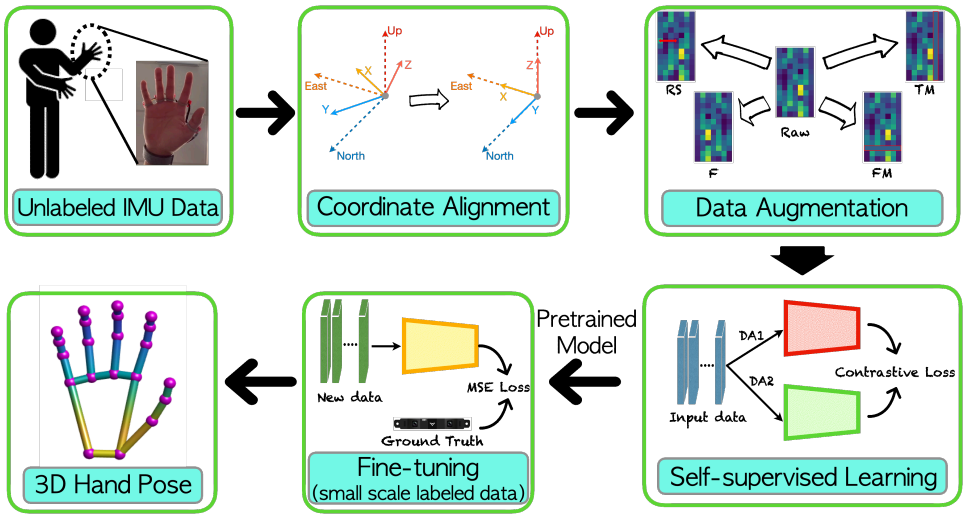


Fig. 1. Overall architecture of *ssLOTR*

Using the custom-built device (Fig. 5) detailed in Section 4, *ssLOTR* extracts 9-axis IMU data from the fingers and wrist, and feeds them to ML models for predicting 3D finger motion. While the ML models for training are

implemented on a Desktop Computer with an NVIDIA GTX 1070 GPU and Pytorch library [97], the inference is done on a smartphone with TensorFlowLite [36], with a latency of 6.8 ms and low power consumption. A systematic study with 12 users achieves a joint angle tracking accuracy of 9.07 degrees and a joint position accuracy of 6.55*mm*. Fig. 2 shows some examples of tracking results. The performance is comparable to fully

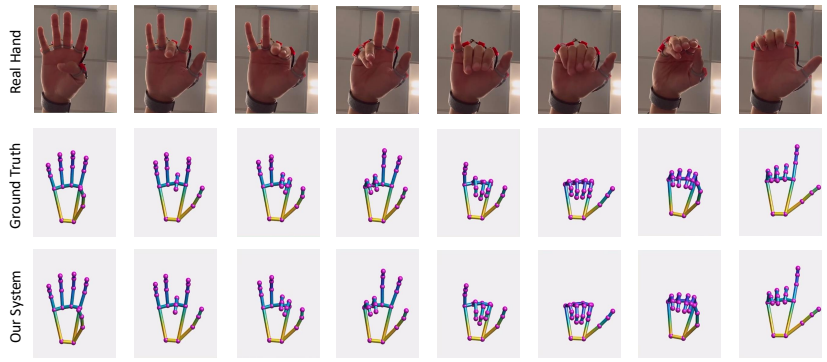


Fig. 2. Comparison of 3D finger tracking results between ground truth and *ssLOTR*

supervised learning but *ssLOTR* only uses 15% of labeled training data in comparison to fully supervised learning. Building on this promise, we believe there is ample scope for extending ideas on self-supervised learning for other aspects of wearable sensing such as body pose tracking. We plan to explore such opportunities in the future.

Summarizing the above possibilities, we enumerate our main contributions below: (i) The feasibility of self-supervised learning to achieve a performance comparable to fully supervised learning with only $\approx$ 15% labeled training data as fully supervised learning. (ii) Design of a wireless sensor device with embedded IMU sensors while allowing comfortable usage with the ability to perform precise and natural finger motion. (iii) Ubiquitous 3D finger motion tracking (with 24 degrees of freedom, discussed in Section 3) without constraints of calibration, interference from environmental backgrounds, robustness to sensor position, etc. (iv) A systematic user study as well as an end-to-end implementation and evaluation on embedded platforms for validation.

2 RELATED WORK

Vision: Depth cameras including Kinect [53] and leap motion [59] sensors can track finger motion and have revolutionized the gaming industry by gesture interfaces. More recently, even monocular RGB cameras are being able to capture the 3D motion of fingers by exploiting advances in machine learning [13, 46, 86]. Fisheye cameras are combined with deep learning in DeepFisheye [96] for tracking fingertips. While such works are truly transformative in nature, vision-based approaches are non-ubiquitous and can be sensitive to lighting, background, and resolution. Digits [52] uses wrist-mounted infrared cameras for 3D finger pose tracking. Similarly, DorsalNet [137] uses wrist-mounted visual cameras for 3D finger motion tracking. The dorsal hand region including the motion of bones, muscles, and tendons is analyzed with a two-stream convolutional neural network for precise 3D motion tracking. However, the camera needs to sit high enough on the wrist or even reach the palm to capture the full range of finger motion. Recently, FingerTrak [40] has innovatively designed wearable thermal cameras that can track 3D finger motion, but as pointed out by the authors themselves, the system is not robust to background temperatures (sun, heater, etc) as well as changes in sensor position due to wrist motion. In contrast to the above works, *ssLOTR*'s solution is ubiquitous while being robust to environmental conditions (occlusions, lighting) as well as natural variation in sensor positions and wrist motions.

Finger Motion Tracking by Radio Frequency Reflections: WiFi signals have been used to classify discrete gestures [62, 82, 106]. mmWrite [102] performs handwriting recognition using mmWave radars. RFWash [50] detects handwashing hygiene using mmWave radars near the bathroom mirror. SignFi [76] uses channel measurements from WiFi Access Points (APs) for sign language recognition. ExASL [105] uses range-doppler spectrum and angle of arrival spectrum of mmWave reflections from the hand for classifying 23 discrete American Sign Language (ASL) gestures. Google Soli [135] exploits reflections from mmWave signals in combination with deep convolutional and recurrent neural networks to track 11 finger motion gestures. While the above works only perform predefined gesture classification, *ssLOTR* performs continuous 3D motion tracking. Moreover, since the device is worn on the body, it offers a more ubiquitous tracking without limitations on the range of sensing.

IMU, Wrist Bands, and Wearable Sensing: IMU, WiFi, and acoustic signals have also been used for hand gesture recognition [94, 110, 131, 142]. FingerIO [88], FingerPing [139] use acoustic signals for finger gesture detection. uWave[70] uses accelerometers for user authentication and interaction with a mobile device. Capband [129] uses capacitive sensing for the recognition of 15 hand gestures. ElectroRing [51] attaches electrodes on the index finger and combines them with IMU sensors for detecting six different pinch-like finger gestures. ThumbTrak [118] detects 12 finger gestures by placing 9 proximity sensors on the thumb and measuring the distance from thumb to other fingers and palm. ZeroNet [75] extracts training data from videos to classify 50 hand gestures. Specifically, while the above works can only distinguish predefined finger gestures, *ssLOTR* performs free form 3D finger motion tracking for arbitrary motion.

Sensor-embedded gloves are popular in applications of gesture recognition. IMU, flex sensors, and capacitive sensors have been embedded in gloves for applications in sign language translation, gaming, user interface, etc [4]. An array of 44 stretch sensors have been used in [35] for finger pose tracking. 17 IMU sensors embedded in gloves are used for hand pose recognition [17, 65]. Commercially available products such as CyberGlove [19], ManusVRGlove [79], 5DT Glove [1] use flex sensors for hand gesture recognition, etc. However, wearing gloves precludes the user from performing activities that require fine precision as studied in recent works [103] because they may hinder natural dexterous hand motion. Accelerometer embedded rings for applications in typing-based interfaces for smartwatches, smart-TV, and other IoT interfaces. Work in [83] evaluates Tapstrap2 [122] and achieves a typing rate of 22.1 words per minute (WPM) [130]. Work in [121] designs accelerometer embedded rings powered by a wrist-worn device using inductive telemetry. It achieves 89.1% accuracy in detecting characters in a typing application. Skinwire [49] shows the feasibility of laying out electronics attached to the skin to connect IMU sensors placed on the fingers to a PCB placed on the wrist. DualRing [63] places IMU sensors on the thumb and index fingers for tracking gestures such as pinching, scrolling, swiping, touching, tapping, etc. However, the sensors are connected by a wire to the data processing device thereby hindering natural hand motion. Oura ring [92] is popular for tracking heart rate, sleep cycle, and human activity. However, it does not provide 3D motion tracking of the fingers. Also, the platform is closed with no access to raw sensor data. The commercially available Tapstrap2 device [122] is perhaps the closest to our hardware, primarily branded as a remote keyboard for interacting with IoT devices. In contrast to such works which focus on typing or recognition of predefined gestures, *ssLOTR* performs 3D finger motion tracking for arbitrary motion, thus being suitable for many applications in AR/VR, Sign Language Recognition (SLR), sports, etc. AuraRing [95], tracks the index finger precisely using a magnetic wristband and a ring on the index finger. In contrast, *ssLOTR* tracks all fingers with the following differences: (i) Even though AuraRing [95] tracks the index finger, the tracking only includes the MCP joint. In contrast, *ssLOTR*'s experiments show the feasibility of tracking the MCP, PIP, and DIP joints of all fingers, thus capturing a total of 24 degrees of freedom including the wrist motion. In addition, we believe the experimental proof of validation of 24 Degrees of Freedom (DoF) finger motion is an important difference. (ii) While AuraRing could be extended to track all fingers, the wristband needs to decouple the interference due to magnetic field generated by individual fingers (potentially by designing protocols based on channel allocation, multiplexing, etc). AuraRing, in its current form does not show the feasibility of resolving such interference.

In contrast, the sensors on various fingers in *ssLOTR* do not interfere with each other. (iii) AuraRing can be susceptible to interference from metallic objects as noted in the paper. In contrast, *ssLOTR* is immune to such interference since *ssLOTR* adopts opportunistic calibration techniques based on A3 [142].

Based on techniques in Convolutional neural network (CNN), Recurrent neural network (RNN), etc, prior works on EMG sensing perform classification of discrete hand poses [22, 26, 100, 101, 113] or track a predefined sequence of hand poses [100, 113]. Work in [93] tracks joint angles using EMG sensors but only for one finger. Using an array of over 50 EMG sensors placed over the entire arm, some works [115] track joint angles for arbitrary finger motion. Recent works [73, 74] show the feasibility of sensing finger motion using EMG armbands. However, EMG sensors need calibration and warming of the skin to be in proper contact with the electrodes which can even take up to 5 minutes during each instance of wearing, leading to usability issues [87, 128, 136]. In contrast, *ssLOTR*'s solution does not need any calibration, and is therefore easy to use.

Self-Supervised Learning: Self-supervised learning is gaining in popularity because of their ability to minimize training overhead as well as the ability to adapt to new tasks with less overhead [7]. One of the popular applications of self-supervised learning includes the development of Bidirectional Encoder Representations from Transformers (BERT) [23] language model, which is developed by masking words in a sentence and teaching the network to predict the masked words, thereby learning efficient representations in the process. The representations can later be used for a number of natural language processing (NLP) tasks in question-answering [99], text-summarizing [140] etc. Similarly, Word2Vec [85] developed continuous representations for words with applications to tasks in NLP. A number of recent works in computer vision exploit self-supervised learning. Rotation of images, change of colorization, etc, have been popularly used for performing self-supervision in computer vision [33, 141]. ImageNet classification accuracy can be boosted [14]. Works in [12] shows the feasibility of estimating human body pose by exploiting consistency across multiple camera views to train a self-supervised representation. *ssLOTR* is inspired by such foundational works on self-supervised learning. However, the model architectures and the fusion with data augmentation techniques have been carefully designed to satisfy the constraints of our problem domain. Self-supervised learning techniques have also been used for wearable human activity recognition. Ten classes including walking, running, sitting, jogging, etc. are classified efficiently in [38, 69, 104]. In contrast, *ssLOTR* focuses on continuous tracking of 3D finger joints involving intricate motion.

Table 1. Summary of Main Related Work. For brevity, several other works not in the table are discussed in Section 2.

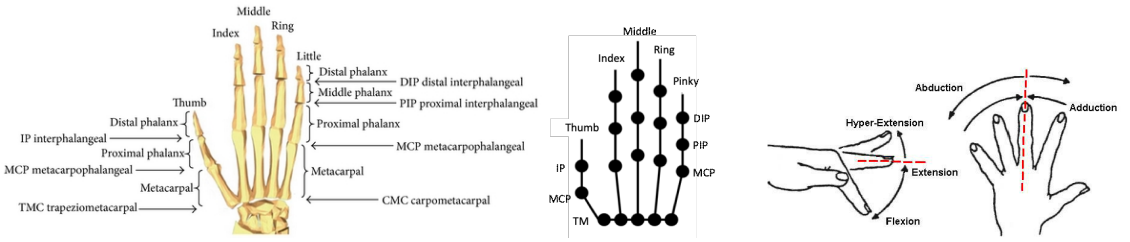
System	Sensor	24 DoF Finger Tracking	Robustness to Ambience	Supports Free Finger Motion	Portability	Training Overhead
TapStrap [122]	Accelerometer	✗	✓	✓	✓	N/A
FingerTrak [40]	Thermal Camera	✓	✗	✗	✓	Fully-supervised
Capband [129]	Capacitive	✗	✓	✓	✓	Fully-supervised
AuraRing [95]	Magnetic	✗	✗	✓	✓	Fully-supervised
Gloves [60]	IMU	✓	✓	✗	✓	Fully-supervised
Vision [20, 86, 96]	Camera	✓	✗	✓	✗	Fully-, Self- supervised
Soli [135]	mmWave	✗	✓	✓	✗	Fully-supervised
SignFi [76]	WiFi	✗	✓	✓	✗	Fully-supervised
FingerIO [88]	Acoustic	✗	✓	✓	✓	N/A
***ssLOTR* (Ours)**	**IMU**	✓	✓	✓	✓	**Self-supervised**

3 BACKGROUND

The human hand consists of complicated anatomical structures including muscles, bones, skin, tendons, and complex relationships between them [57]. This facilitates a high degree of articulation. In this section, we provide a brief background of the human hand, the degrees of freedom for motion, and the anatomical structure.

Finger skeletal model: Human hands have four fingers and a thumb with a complex degree of articulation. Fig. 3a depicts the skeletal structure of the hand while Fig. 3b shows a simplified kinematic view. Each finger,

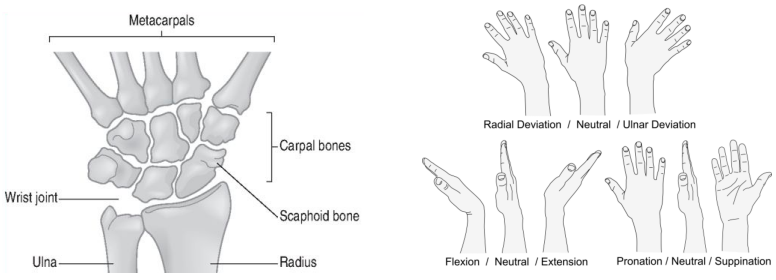
starting with the one closest to the TM (trapeziometacarpal) joint, has a colloquial name of i) Thumb, ii) Index finger, iii) Middle finger, iv) Ring finger, and v) Little finger. The four fingers (except for the thumb) consist of the distal, middle, and proximal phalanges bones. These bones support the following joints: the distal interphalangeal (DIP), proximal interphalangeal (PIP), and metacarpal phalangeal (MCP) joints [89]. On the other hand, the thumb consists of distal, proximal phalange, and metacarpal bones with the following joints: Interphalangeal (IP), MCP, and Trapeziometacarpal (TM) joints. The above finger joints can be moved in complex forms to create different gestures/poses. The MCP joints have two degrees of motion of flex/extensions and abductions/adductions as depicted in Fig. 3c, the two angles are denoted by $\phi_{mcp,f/e}$ and $\phi_{mcp,a/a}$ respectively. The PIP and DIP angles only have one degree of motion (flex/extensions), and these angles are denoted by ϕ_{dip} and ϕ_{pip} respectively. Thus, each of the four fingers have four degrees of freedom (DoF). The thumb has a slightly different anatomy as its MCP and TM joints can both flex/extend and abduction/adduction, and its IP (interphalangeal) joint can only flex or extend with a single DoF. Thus, the thumb has five DoFs including $\phi_{mcp,f/e}$, $\phi_{mcp,a/a}$, $\phi_{tm,f/e}$, $\phi_{tm,a/a}$, and ϕ_{ip}, forming a 21-dimension ($\mathbb{R}^{21}$) space of finger joint angle with the other four fingers.



(a) Anatomical Structure (b) Kinematic Model (c) Flex/Extensions and Abduction/Adduction

Fig. 3. Anatomical and kinematic structure of finger joints

Wrist Motion: In addition to the finger skeletal model above, we also track the wrist joint angles. Several bones (e.g., distal ends of the Radius, Ulna and the carpal bones) form the wrist joint (depicted in Fig. 4a), which can connect the forearm to the hand palm, allowing a flexible range of motion [89]. Fig. 4b depicts 3 categories (3 DoFs) of the wrist joint motion: i) pronation/supination, $\phi_{wrist,p/s}$; ii) flexion/extension, $\phi_{wrist,f/e}$; and iii) radial/ulnar deviation, $\phi_{wrist,r/u}$. Note that unlike finger joints, each DoF of the wrist involves both positive and negative rotations from the neutral position [8]. As discussed earlier, fingers and thumb have 21 DoFs whereas the wrist has additional 3 DoFs, thus totaling to 24. *ssLOTR* aims to track these 24 degrees of freedom (24 DoF).



(a) Wrist Joint [2] (b) Range of Motion of Wrist[8]

Fig. 4. Anatomy and range of motion of the wrist

Hand Motion Constraints: The various joint angles responsible for finger articulation exhibit a high degree of correlation and interdependence [16, 66]. Intrafinger constraints are defined as constraints between different joints of one finger. Such intrafinger constraints are noted below:

$$\phi_{ip} = \frac{1}{2}\phi_{mcp,f/e}, \quad \phi_{dip} = \frac{2}{3}\phi_{pip}, \quad \phi_{mcp,f/e} = k\phi_{pip}, \qquad 0 \leq k \leq \frac{1}{2} \tag{1}$$

Equation 1 suggests that in order to bend the DIP joint, the PIP joint must also bend under normal finger motion (assuming no external force is applied on the fingers). Similarly, the range of motion for PIP is very much limited by the MCP joint (Equation 1). Besides, there exists range of motion constraints as depicted below:

$$-15^\circ \leq \phi_{mcp,aa} \leq 15^\circ,\ 0^\circ \leq \phi_{dip} \leq 90^\circ,\ 0^\circ \leq \phi_{pip} \leq 110^\circ \tag{2}$$

In addition, there are also complex interdependencies between joints of different fingers, which cannot be directly modeled by equations, but we believe that our ML models will automatically learn those constraints.

4 PLATFORM DESIGN

In this section, we outline the design and implementation details of a new wireless wearable sensor device based on embedded IMUs and a SoC micro-controller with integrated WiFi and BLE modules.

4.1 Hardware Architecture

Our goal is to design an unobtrusive, comfortable, and portable sensor device to fit the finger while reliably collecting IMU sensor data. Compact size, lightweight, low-latency, and wireless streaming are our main criteria to build the sensor device so as to facilitate comfortable wearing under a natural setting. Towards this end, we selected the ICM20948 IMU (InvenSense, USA) [44] as the main sensing unit and the Tinypico (Unexpected Maker, USA) (Fig. 5b) [78] as the micro-controller with integrated BLE and WiFi modules. The ICM20948 provides 9 DoFs (triple-axis MEMS accelerometer, triple-axis MEMS gyroscope, and triple-axis magnetometer) and supports on-chip 16-bit Analog to Digital Converters (ADC), and an inter-integrated circuit (I^2C) interface [114] for communication with the micro-controller. The Tinypico, a breakout of ESP32, is based on Espressif [120] SoC wireless micro-controller with WiFi and BLE. The chip ESP32-Pico-D4 (ESPRESSIF SYSTEMS, CHINA) [28], embedded in Tinypico, has a 32-bit dual low-power Xtensa 32-bit LX6 microprocessor [138] operating at 240 MHz and a Bluetooth BLE 4.2 and WiFi 802.11 b/g/n complaint 2.4 GHz transceiver and supports I^2C bus interface. The Tinypico including its external antenna has a package size of 18mm × 32mm.

We designed different IMUs breakouts for fingers and wrist using two-layer PCB technology so that we could integrate different electronic components together to minimize the size of the device. As shown in Fig. 5e and Fig. 5c, PCBs of the finger sensor module and wrist sensor module were designed using Autodesk EAGLE (Autodesk, CA, USA) [6]. They are two-layer boards that were fabricated with JLCPCB Shenzhen, Guangdong, China [109]. The two-layer PCB enables the assembly of electronic components on both sides of the board for minimizing the size of the device. Both layers are built using material of flame retardant standard (FR4) [31] and appearance quality of Institute of Printed Circuits (IPC) Class 2 standard [81]. For the finger sensor module as shown in Fig. 5a, we used Fused Deposition Modeling (FDM) technology [32] to build 3D printed housing designed by SolidWorks 3D CAD software (Dassault Systems, Vélizy-Villacoublay, France) [112], and we chose PLA and TPU as material of finger sensor modules for high standard of wearability, flexibility and comfortableness. Each finger sensor module is firmly attached inside ring-shaped housing on the proximal phalanx bone of each finger and the wrist sensor module (Fig. 5) is mounted inside watch-shaped housing made of PLA. The TPU material was used for the ring because of elasticity and flexibility. On the other hand, the PLA material was used for casings on

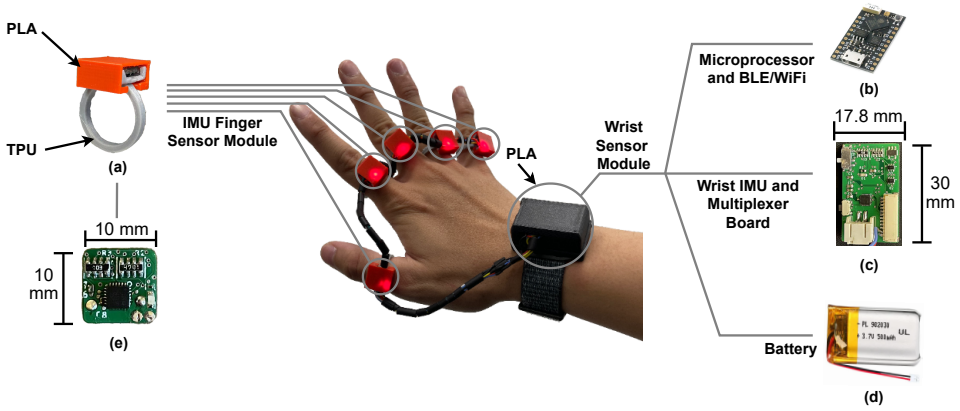


Fig. 5. Design of a portable wireless sensing device in *ssLOTR*: (a) IMU finger sensor module (b) Tinypico integrated with microprocessor and BLE/WiFi module (c) Wrist IMU and multiplexer board (d) Battery (e) Finger sensor board

both fingers and the wrist because of rigidity and sturdiness. While we create several different sizes of the ring to accommodate different users, the elasticity of the TPU material helps in snug fitting and comfortable wearing.

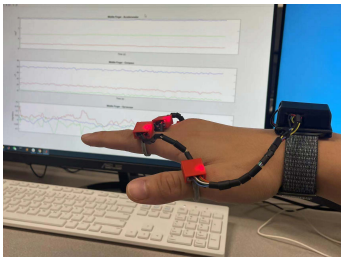
Depicted in Fig. 5e, each finger sensor module embeds an ICM20948 IMU. Each finger sensor module has I^2C bus for communication, using 4 channels (SCL, SDA, VIN, GND) [114] to connect with the wrist sensor module. In addition to Tinypico (Fig. 5b), the wrist sensor module includes a PCB integrating an ICM2cr0948 IMU and a TCA9548A multiplexer [42] (Fig. 5c). The multiplexer, an 8 channel I^2C switch by TEXAS Instruments helps Tinypico in uniquely addressing sensors on different fingers for communication. We also embed a side tactile switch to turn on/off the device. Overall, the size of breakout for finger sensor module and wrist sensor module are 10 mm × 10 mm × 1.6 mm (W × L × H) and 17.78 mm × 30.48 mm × 1.6 mm (W × L × H) respectively. The dimensions of the finger sensors are comparable to Tapstrap2 and the dimensions of the wrist sensor are comparable to a smartwatch. The device is designed to be worn on both right and left hands with equal comfort by simply rotating the device by 180 degrees about an axis perpendicular to the plane of the PCB boards.

To power our device, we use a 3.7V, 500mAh LiPo battery (Fig. 5d). Our device has a power consumption of 198mA while the device is actively streaming sensor data for finger motion tracking. However, the device is put under sleep for further power savings when it is not being actively used for an interactive application, thus prolonging the battery life. The total weight of the device is 34.9g. The average weight of each finger sensor is 2.5g, including the TPU and PLA housing (1.3g). The total weight of wrist sensor module is 22.4g, including the PLA housing (7.3g) and the LiPo battery (9.7g). Since the SoC, wireless communication, and battery components are housed on the wrist which offers more space with a smartwatch form factor, this makes the sensors on fingers to be lightweight, thus allowing dexterous hand motions. The hardware specifications are summarized in Table 2.

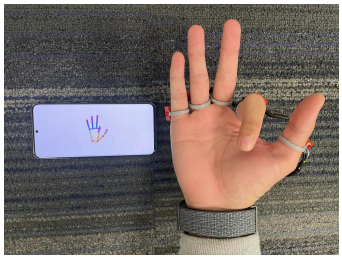
4.2 Software Framework

The software side of the sensor device includes three main components: (i) Collecting the data from IMU sensors on different fingers and wrist at the microcontroller; (ii) Packaging them into a packet; (iii) Sending the packets over BLE connection to a smartphone. The software is implemented on the ESP32 microcontroller using C++ and appropriate libraries for IMU, BLE streaming, and sensor addressing via multiplexers [27, 41, 55]. The Tinypico first reads data from six IMUs over I^2C and then streams the data to a mobile device over BLE using Universal Asynchronous Receiver/Transmitter (UART) profile [55] with a sampling rate at 100 Hz. The Tinypico has only two I^2C bus interfaces, which precludes it from communicating with six IMU sensors. To solve the addressability of different IMUs, we integrate TCA9548A into the wrist sensor module. It serves as an extension module to

communicate with six IMUs including five IMUs on fingers and one IMU on wrist. Therefore, we can control individual channels SCn/SDn according to different pins using Sparkfun Qwicc Mux software [27], thus solving the addressability issue. This allows us to detect and process 9 DoFs data from six IMUs breakout efficiently, with a low-latency of 9.7 ms. Fig. 6a depicts the IMU data streamed in real-time and plotted using Matplotlib [80] on a desktop computer. Fig. 6b shows real-time streaming to a smartphone device that implements the ML modules from Section 5 for 3D finger motion tracking (rendered using OpenGL ES [34]). More details on data processing are discussed in Sections 5 and 6.



(a) Wireless streaming of sensor data to desktop



(b) Wireless streaming of sensor data to mobile device which performs 3D finger motion tracking

Fig. 6. Interaction of *ssLOTR*'s sensing device with smartphones and desktops

Table 2. Summary of *ssLOTR*'s hardware design

Component	Purpose and Description	Specs and References
Sensor Chip	Collecting data from accelerometer, gyroscope, and magnetometer for finger motion tracking	ICM 20948 [44]
Microcontroller (Mic)	Assembling sensor data from individual fingers, packaging them into a BLE packet and streaming to a smartphone or desktop	TinyPico [78]
Hardware Communication Protocol	For transferring sensor data between sensor chips and microcontroller	I2C [114]
Multiplexer	Creates unique addresses for different sensor chips for communication with microcontroller	TCA9548A [42]
Wireless Streaming Protocol	Streaming of sensor data to a smartphone or desktop	BLE 4.2 [11]
Ring Material	Flexible material used for 3d printing the ring that supports the sensor	TPU [125]
Sensor and Wrist Case	A sturdy material to hold sensors, microcontroller, battery, and multiplexer	PLA [98]
Battery	Powering the sensor and microcontroller	3.7V, 500mAh LiPo battery [68]
PCB Technology	Two-layer printing for double sided PCB that minimizes hardware size	Provided via JLCPCB [109]
Dimensions (Finger Module)	The overall dimensions of finger sensor module	10mm x 10mm x 1.6 mm
Dimensions (Wrist Module)	The overall dimensions of the wrist sensor module	17.8mm x 30mm x 1.6 mm
Weight (Finger Module)	The overall weight of the finger sensor module	2.5g
Weight (Wrist Module)	The overall weight of the wrist sensor module	22.4g

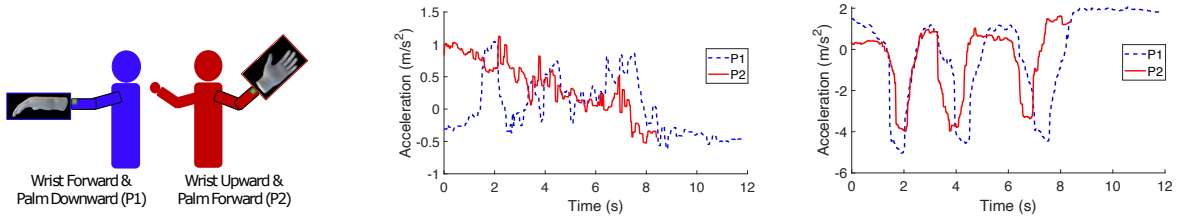
5 TECHNICAL MODULES

In this section, we describe the key signal processing and machine learning modules in self-supervised tracking of 3D finger motion.

5.1 Sensor Data Transformation

Identical finger motion can generate different sensor data depending on the position and orientation of the wrist because of differences in coordinate frames as well as the influence of gravity [142]. Therefore, we first transform the sensor data to a consistent frame of reference. This helps reduce variation in the distribution of the input sensor data thus facilitating efficient learning of the ML models in *ssLOTR*. We elaborate on the details below.

Wrist Coordinate Frame: By computing the orientation of the sensor, the accelerometer measurements can be converted from a local frame of reference (sensor's x, y, and z axes) to a global frame of reference (east, north, and up) [142]. As the hand moves continuously in 3D space, the orientation of fingers, as well as the projection of gravity on the local frame of reference of the sensor, will change. Therefore, a similar finger motion pattern such as curling all fingers from an open palm into a fist will generate different sensor data depending upon the position and orientation of the wrist. Fig. 7a shows an example where the fingers were moved in *curling motion*



(a) (b) (c)

Fig. 7. (a) Curling of fingers with two different wrist orientations (b) They generate different sensor data (c) After alignment with WCF, the sensor input is similar across two wrist orientations. This increases the efficiency of learning by containing variation in input.

with two different wrist positions/orientations – P1 and P2 as noted. As expected, the accelerometer data (of the x-axis of index finger) for the two cases as depicted in Fig. 7b look different.

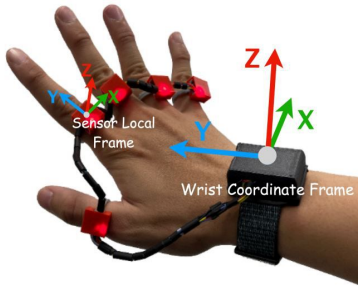


Fig. 8. Wrist Coordinate Frame and Sensor Local Frame

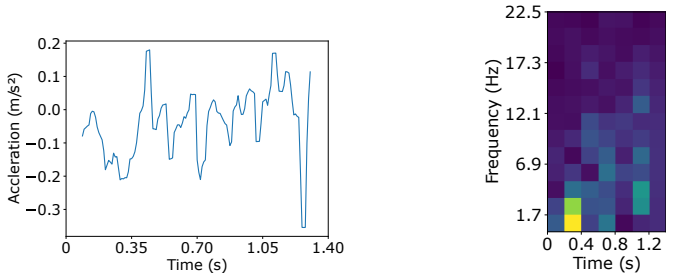
While it might be possible to train the ML models with such a variation in input across wrist configurations, that will entail complex models and larger training data. Therefore, we first transform the finger sensor data to a consistent frame of reference such that similar finger motion patterns will generate similar sensor data irrespective of the state of the wrist. Transforming the sensor data to the global frame of reference does not solve this problem, since the direction of motion will still be dependent on the state of the wrist. Therefore, we define the *Wrist Coordinate Frame* (WCF) as depicted in Fig. 8 and consider the wrist as the reference for measuring the

finger sensor data. The data are transformed into the global frame and then into the WCF based on the following equation.

$$\begin{bmatrix} X_{wcf} & Y_{wcf} & Z_{wcf} \end{bmatrix} = \begin{bmatrix} X_l & Y_l & Z_l \end{bmatrix} R_{finger} R_{wrist}^T , \tag{3}$$

where, $[X_l\ Y_l\ Z_l]$ denotes the acceleration of the finger in the local frame of reference after subtraction of gravity. R_{wrist} and R_{finger} denote the orientations of the wrist and the fingers respectively. Finally, $[X_{wcf}\ Y_{wcf}\ Z_{wcf}]$ denotes the acceleration of the finger in the WCF. Fig. 7c shows the index finger accelerometer data (of the x-axis) for the two configurations after gravity compensation and transformation to WCF, which are now similar. Such consistency will reduce the variation in the input distribution thus decreasing the overhead in training [45].

5.2 Preprocessing and Feature Extraction



(a) Time domain data (b) Spectrogram

Fig. 9. Example IMU data in time and frequency domains.

Therefore, we preprocess the data to extract frequency domain features that can exploit sparsity in human motion for efficient learning [61]. We also perform a low pass filter (cutoff frequency of 25 Hz) to eliminate high-frequency noise. We perform short time fourier transforms (STFT) to create 2D spectrograms from the sensor input. The sensor input includes 3-axis accelerometer data that have been subtracted with gravity and converted to the WCF as discussed in Section 5.1. In addition to the accelerometer input, we also include the direction vector of the y-axis ($[v_x\ v_y\ v_z]$) of each IMU with respect to WCF:

$$\begin{bmatrix} v_x & v_y & v_z \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} R_{finger} R_{wrist}^T , \tag{4}$$

Under normal wearing conditions shown in Fig. 8, the y-axis is roughly aligned with the direction between the MCP and PIP joints. However, a careful alignment or special calibration is not needed. We note that the direction vectors are noisy given that a small motion can produce a large change in the finger poses. Change in wrist angles, as well as the changes in finger angles, can together influence the direction vectors in complicated ways. Nevertheless, the ML models presented next are trained to be robust to such noises as well as combine information across time for higher accuracy. While x-axis and z-axis direction vectors can also be similarly included as inputs to the model, the information thus captured will have some redundancy with the y-axis. Considering the tradeoff between a higher number of parameters with more inputs to the model and the accuracy, we did not notice a significant difference in performance by including x and z axes direction vectors, therefore, we only include the y-axis direction vector as an input to the ML model.

We compute the 2D spectrograms from such accelerometer and direction vectors in WCF at a sampling rate of 100 Hz. We use a Fast Fourier Transform (FFT) window size of 50, with a 50% overlap between successive FFT computations. A Hamming window is applied before computing FFT to reduce spectral leakage and counteract the assumptions made by FFT that the data are infinite [29]. Fig. 9 shows an example of an input accelerometer

data converted into a 2D spectrogram. Since we have data from 6 sensors (5 fingers and 1 wrist), each with 6 axes (acceleration and direction vectors), we will have a total of 36 spectrograms as input to the ML models.

5.3 Self-Supervised Learning for Capturing Efficient Representations from IMU Data

Fig. 10 depicts the high-level architecture of the self-supervised learning framework used in *ssLOTR*. The raw sensor input (2D spectrograms) x^i is first transformed into two variants (x_1^i and x_2^i) based on data augmentation techniques. The transformations add perturbations to the data while still retaining the overall pattern. x_1^i and x_2^i are then fed as input to an encoder neural network that extracts representations h_1^i and h_2^i as shown in the *self-supervised stage* of the figure. Finally, the representations are projected into a latent space before comparing them using a *contrastive loss function* that attempts to maximize the similarity between y_1^i and y_2^i while minimizing the similarity between y^k and y^j, where $k \neq j$. Since such a network tries to enforce similarity among representations even though the inputs have been perturbed by data augmentation techniques, it is known to learn efficient representations. Learning such representations can be done in an entirely self-supervised manner without any labeled training data.

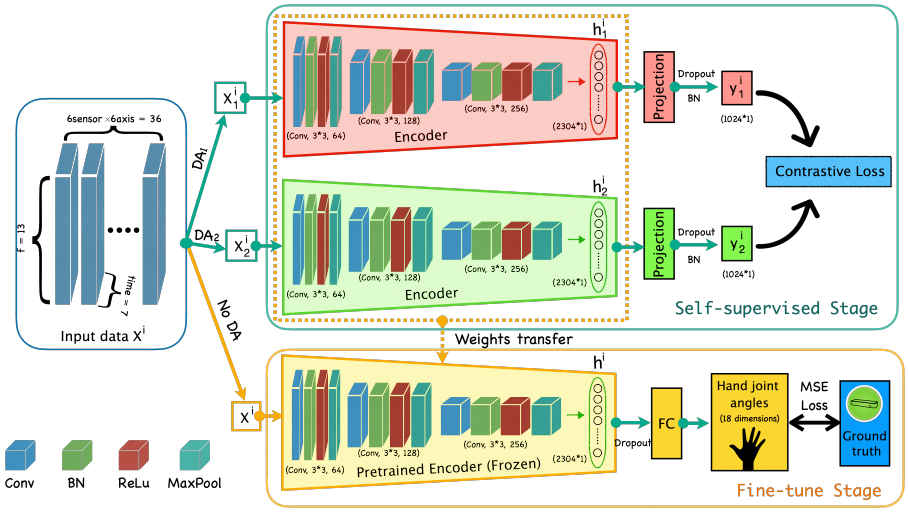


Fig. 10. Architecture for self-supervised and fine-tuning stages (DA = Data Augmentation)

Finally, using the representations thus learned instead of original raw inputs will dramatically decrease the amount of labeled training data needed for supervised learning [7, 14, 20]. We now elaborate on various components of the architecture.

Data Augmentation: Towards learning self-supervised representations as described above, we employ the following four data augmentation techniques to perturb the input. This helps avoid overfitting as well as teaches the algorithm to look for stable features that measure similarity. The architecture in Fig. 10 needs two data augmentation techniques at a given instant of time. We use various combinations of two techniques from the four techniques presented below. The results of these combinations are summarized in Section 6. *(i) Time Masking:* We mask parts of the input spectrogram so as to help the ML model in capturing the spectro-temporal dependencies in the sensor data. Fig. 11b depicts an example where the masked column is highlighted in red. **Benefits:** Such a strategy is popular in language modeling such as BERT [23] where words are masked in a sentence to force the language model to predict these words, thus facilitating learning of efficient representations of sentences. Inspired by the success of BERT, we apply similar masking techniques to the sensor data. In *time masking*, a few columns

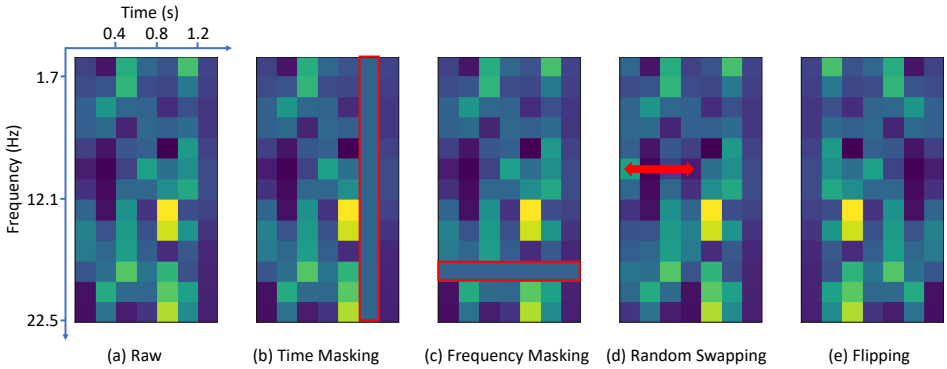


Fig. 11. Data augmentation techniques: (a) Raw Input (b) Time Masking (c) Frequency Masking (d) Random Swapping (e) Flipping

(around 14% $\approx \frac{1}{7}$ based on the empirical results in Fig. 22) are randomly masked in the input spectrogram. *(ii) Frequency Masking:* Similar to the temporal masking above, we perform masking across frequencies (rows) of the input spectrogram. Fig. 11c depicts an example where the masked row is highlighted in red. **Benefits:** This is the frequency domain analogue for temporal masking with similar benefits in capturing frequency domain representations. Moreover, masking can prevent models overfitting as it masks some portions of input data, which makes models more robust. *(iii) Random Swapping:* We swap columns randomly in the spectrogram corresponding to the input data. Fig. 11d depicts an example where the swapped columns are shown in red. **Benefits:** The idea of random swapping helps the model to develop robustness in the face of outliers, ultimately facilitating in capturing efficient representations from sensor data. (iv) Horizontal Flip: We flip the sensor data across time so as to reverse the time series. Fig. 11e is a flipped version of the spectrogram in Fig. 11a. **Benefits:** Flipping is inspired by image flipping from computer vision which is an effective strategy for increasing the diversity of the dataset [111]. Likewise, flipping in *ssLOTR* generates a mirrored version of the sensor data in the opposite direction at a negligible computation cost, while increasing the diversity of the sensor dataset. This will help the learning framework identify the key latent features of similarity across the augmented versions. In a nut shell, the sensor data augmentations improve the quality of the feature representations in the self-supervised framework and also prevent the model from overfitting.

Representation Learning: The encoder architecture is depicted in Fig. 10. The encoder maps a sequence of data augmented IMU input to compact representations that can support efficient learning with minimal training data. The dimensions of our input (i.e., x^i) is $36 \times 13 \times 7$ which includes 36 IMU spectrograms (over 1.4s of data) from 6 sensors at a sampling rate of $100Hz$, which are computed as discussed earlier. The dimensions of the output representations (i.e., h^i) is 2304×1. The input first passes through the encoder network that consists of a series of convolutional layers with the input downsized at each layer with maxpool operation. Without any labeled training data, the encoder attempts to learn representations that are invariant of different distortions of the same input. The learned representations are later used for 3D finger motion tracking using a small scale labeled training data. Batch normalization is used at each layer for accelerating convergence by controlling variation in the input distribution at each layer [45].

Contrastive Loss Function: The encoded representations h are passed through a projection head as shown in Fig. 10, resulting in an output $y = p(h)$ where p represents the action of the projection head. We apply the contrastive loss function on y that maximizes the similarity between two differently augmented copies of the same input. The contrastive loss function is applied on y whereas we use representations h in the later phases for prediction of 3D finger motion. The reasons for such a design choice are as follows. (i) Since the contrastive loss function's main goal is to maximize the similarity between differently augmented versions of the same input, it might lose some information during the process. (ii) On the other hand, the encoded representation h is one level

before the projection head, and it offers the best trade-off between capturing high-level robust representations without losing much information. Our evaluation results validate that such a design choice results in considerably better performance than applying the contrastive loss function directly on h.

The mathematical form of the contrastive loss function that enforces similarity between differently augmented samples of the same input x^i is given by:

$$\ell^i = -\log \frac{\exp(sim(y_1^i, y_2^i)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{i \neq j} \exp(sim(y^i, y^j)/\tau)}, \quad sim(u, v) = \frac{u^T v}{||u|| \cdot ||v||} \tag{5}$$

Here, (y_1^i, y_2^i) represents the output of the projection head from Fig. 10 that acts on differently augmented versions x_1^i, x_2^i of the same input x^i. Given a batch of N input examples $\{x^i, \forall\, i \in [1, N]\}$, we have $2N$ similarity examples of the form $\{(x_1^i, x_2^i), \forall\, i \in [1, N]\}$. For each similarity pair of the form (x_1^i, x_2^i), we have $2(N-1)$ dissimilar pairs of the form $\{(x^i, x^j), i \neq j\}$. $1_{i \neq j}$ indicates dissimilar pairs when $i \neq j$, and τ denotes a temperature parameter that controls the penalty strength on dissimilar samples [133]. In our experiments, we set τ to 0.2 to encourage the model to have tolerance for similar samples within a batch. Both similar and dissimilar pairs are considered in evaluating the contrastive loss function in Equation 5 thus training the network to maximize the similarity between similar pairs as well as minimize the similarity between dissimilar pairs. The similarity metric $sim(u, v)$ is also indicated in Equation 5.

5.4 Downstream Prediction of 3D Finger Joints from Self-Supervised Representations

The representations h learned above based on the architecture in Fig. 10 are used for estimating 3D finger joint angles. This is indicated as the *fine-tune stage* in Fig. 10. The input IMU data is first passed through the encoder which extracts representations h. For predicting the finger joint angles using h, we follow a widely used evaluation protocol [56] which can be used as a proxy indicator for the efficiency of self-supervised learning. Specifically, a simple linear model with two fully-connected layers takes the representations h and predicts joint angles. The weights of the linear model are trained on top of the encoder network (encoder's weights are frozen after *self-supervised stage* in Fig. 10) in a supervised fashion requiring only 15% of labeled training data as fully supervised learning for achieving similar performance.

As discussed in Section 3, finger and thumb motion has 21 DoFs whereas the wrist has additional 3 DoFs, thus totaling to 24. Among these, the DIP angle (ϕ_{dip}) and the IP angle (ϕ_{ip}) can be predicted directly using constraints from Equations 1 , whereas one of the wrist angles ($\phi_{wrist,p/s}$) can be computed directly from the orientation of the wrist relative to the body [72, 107, 108]. Therefore, the actual output of the network has 18 DoFs as indicated in Fig. 10 of the fine-tune stage. Towards tracking these 18 DoFs, we observe that the sensors placed on the proximal phalanx bone (Fig. 3a), are fairly sensitive to the motion of various finger joints. By fusing the sensor data with constraints, both implicit (Equations from Section 3) and explicit (learned by ML models), *ssLOTR* is able to track all finger joints consistently as evaluated in Section 6.

Loss Functions and Optimization: In all equations below, $\hat{\phi}$ denotes the prediction by the ML model, whereas ϕ denotes the training labels from a depth camera (leap sensor [59]).

$$loss_{mcp,f/e} = \sum_{i=1}^{i=4} (\hat{\phi}_{i,mcp,f/e} - \phi_{i,mcp,f/e})^2, \quad loss_{pip} = \sum_{i=1}^{i=4} (\hat{\phi}_{i,pip} - \phi_{i,pip})^2, \quad loss_{mcp,a/a} = \sum_{i=1}^{i=4} (\hat{\phi}_{i,mcp,aa} - \phi_{i,mcp,aa})^2 \tag{6}$$

The above equations capture the Mean Squared Error (MSE) loss in the prediction of joint angles of MCP (flex/extensions and adduction/abduction) joints and PIP joints of the four fingers.

$$loss_{thumb} = (\hat{\phi}_{th,mcp,aa} - \phi_{th,mcp,aa})^2 + (\hat{\phi}_{th,mcp,f/e} - \phi_{th,mcp,f/e})^2 + (\hat{\phi}_{th,tm,aa} - \phi_{th,tm,aa})^2 + (\hat{\phi}_{th,tm,f/e} - \phi_{th,mcp,f/e})^2 \tag{7}$$

The above equation captures the MSE loss in the MCP and TM joints of the thumb.

$$loss_{wrist} = (\hat{\phi}_{wrist,f/e} - \phi_{wrist,f/e})^2 + (\hat{\phi}_{wrist,r/u} - \phi_{wrist,r/u})^2 \tag{8}$$

The above equation captures the MSE loss in the flex/extension and radial/ulnar deviation angles of the wrist. The pronation/supination is directly computed from the orientation of the wrist sensor [72, 107, 108].

$$loss_{smoothness} = ||(\nabla\hat{\phi}_t - \nabla\hat{\phi_{t-1}})||_2^2 \tag{9}$$

The above equation enforces constant velocity smoothness constraint in the predicted joint angles where ϕ_t above is a representative vector of all joint angles across all fingers at a time step t.

The overall loss function is given by the below equation.

$$loss = loss_{mcp,f/e} + loss_{mcp,aa} + loss_{pip} + loss_{thumb} + loss_{wrist} + loss_{smoothness} \tag{10}$$

Note that the loss function does not include ϕ_{dip}, or ϕ_{ip} because we compute them directly from anatomical constraints: Equations 1.

Finger motion range constraints: As described in Section 3, each finger joint has a certain range of motion for both flex/extensions and abduction/adductions. In order to apply these constraints, we first normalize the predicted output of a joint angle by dividing it by the range constraint (for example, by 110° for ϕ_{pip}). We then apply the bounded Rectified Linear Unit (ReLU) activation (bReLU) function [64] to the last activation layer in our network. The bReLU adds an upper bound to constrain its final output. The bReLU outputs are multiplied again with their range constraints such that the unit of the output is in degrees. The bReLU, in conjunction with other loss functions based on temporal constraints (Equation 9) facilitates predicting anatomically feasible as well as temporally smooth tracking results.

6 PERFORMANCE EVALUATION

6.1 User Study

6.1.1 Data Collection Methodology. Our study has been approved by the IRB committee. We conduct a study with 12 users (8 males, 4 females). The users are aged between 20-33 and weigh between 47-96kgs. The users wear the sensor device as shown in Fig. 5 with the sensor snugly fit on the fingers. The users were then instructed to perform random finger motions and wrist motions that include flexing or extending of fingers/wrist as well as abduction or adduction thus incorporating all range of possible hand poses across all fingers. This ensures good convergence of the ML models as well as generalizability to arbitrary finger motions. There are no discrete classes of gestures. The motion patterns are entirely arbitrary thus making the data collection easier.

6.1.2 Labels for Training and Testing. The collected data includes 9-axis IMU from the fingers, wrist as well as the fingers' 3D coordinates and joint angles captured by the leap sensor [59]. While the IMU sensors provide motion data for 3D pose tracking, the leap sensor data serves as ground truth for validation as well as provides labels for training *ssLOTR*'s ML models. These labels include joint angles for each finger. Since *ssLOTR* performs continuous finger tracking instead of identifying discrete gestures, we simply employ the MSE (instead of cross-entropy) loss between predicted joint angles (from *ssLOTR*) and ground truth (from leap sensor) for training and testing.

6.1.3 Training Data. Each user participates in 5 separate sessions with each session lasting for 2 minutes, with sufficient rest between sessions. The sensor is removed and remounted across each session so as to make the ML models robust to natural variations in sensor mounting positions and orientations. The wrist position and orientation were varied across sessions as shown in Fig. 12. One of the configurations includes mobility of the wrist, both translation, and rotation. Using the above training data, we mainly evaluate *ssLOTR* against the following two baselines.

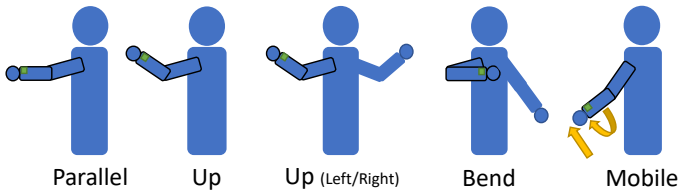


Fig. 12. Wrist orientations/positions

■ *SL*-100%: This is fully supervised learning which needs 360 seconds (more details explained in Fig. 13c) of training data from a given user. In contrast, with self-supervised learning, *ssLOTR* only requires 15% (54 seconds) of labeled training data and 5760 seconds of unlabeled training data obtained from different users. Given that the unlabeled data can be continuously generated without supervision, it is known to be substantially less overhead than labeled training data that needs supervision [15, 71].

■ *SL*-15%: Fully supervised learning that needs the same amount of labelled training data as *ssLOTR* (15%).

Justification for the above choice of models: *ssLOTR* vs *SL*-100% vs *SL*-15% provides a quantitative measure of effectiveness of self-supervised learning techniques. For example, if the performance of *ssLOTR* is close to *SL*-100%, this indicates that with only 15% labelled training data, *ssLOTR* can achieve as much performance as fully supervised learning, thereby dramatically cutting down the training overhead. Also, the comparison between *ssLOTR* and *SL*-15% provides a quantitative metric for the gain due to self-supervision in *ssLOTR*.

For *ssLOTR*, we develop three versions of models based on the dependence of the training data with the user:

■ *User Independent model (UI):* Here, the training data are derived entirely from a different user than the one on which testing is performed.

■ *User Dependent model (UD):* The training data are derived from the same user on whom the testing is done.

■ *User Adaptation (UA):* Here, 90% of the labeled training data are derived from a different user, and only 10% are derived from the user on whom inferences are performed.

Justification for the above choice of models: The goal here is to understand the effectiveness of domain adaptation strategies in *ssLOTR* in decreasing the user dependent training overhead. For example, *User Adaptation* only uses 10% of user-specific training data in comparison to *User Dependent model:*. If the performance of the two models are comparable, this validates the effectiveness of domain adaptation strategies in *ssLOTR* to dramatically cut down user-specific training data by exploiting a pre-trained model from another user. Similarly, given that *User Independent* model does not need user-specific training data, its comparison with the *User Dependent* models helps understand the loss in performance due to lack of user-specific training data. The *User Adapt* model is expected to offer a sweetspot in the tradeoff between training overhead and accuracy. Table 3 depicts the specific models used in the evaluation figures. More details are provided during the description of these figures.

Table 3. Summary of models used across evaluation figures.

	***ssLOTR* (UA)** Default version of *ssLOTR*	*ssLOTR* (UD)	*ssLOTR* (UI)	*SL* − 100% (UA)	*SL* − 15% (UA)
Fig. 13a, 13b	✓			✓	
Fig. 14, 15a, 20a	✓			✓	✓
Fig. 15b	✓	✓	✓		
Fig. 16, 17, 20b, 20c, 21, 22	✓				

UA: User Adaptation **UD**: User Dependent **UI**: User Independent

6.1.4 Testing Data. For the model with user adaptation, we use user adaptation training data from one session, and we evaluate the joint angle prediction accuracy over test cases from other sessions where: (i) The sensor has

been removed and remounted on the user's wrist and fingers. (ii) The wrist position is completely different from the one used to train the models. Similarly, for the user dependent model, we use 3 sessions of data as training, and the rest as test data in a 3:2 randomized cross validation across sessions. The wrist position, orientations, and sensor positions are completely different across training and testing datasets.

6.2 Implementation

ssLOTR is implemented on a combination of desktop and smartphone devices. The ML models are implemented with Pytorch [97] library and the training is performed on a desktop with Intel i7-6700K CPU, 16GB RAM memory, and an NVIDIA GTX 1070 GPU. We use the Adam optimizer [54] with a learning rate of 1e-3, β_1 of 0.9 and β_2 of 0.999. To avoid over-fitting issues that may happen in the training process, we apply the L2 regularization[10] on each convolutional layer with a parameter of 0.01 and also add dropouts[132] with a parameter of 0.05 following each ReLU activation. Once a model is generated from training, the inference is done entirely on a smartphone device using TensorFlowLite [36] on Samsung S20 and OnePlus 9 Pro smartphones.

6.3 Performance Results

We compare the performance of *ssLOTR* against the two baselines *SL*-100% and *SL*-15% as discussed before. If not stated otherwise, the reported results are under the following conditions: (i) The errors reported are for flex/extension angles for both fingers and wrist as they are prone to more errors with a high range of motion. The errors for abduction/adduction (fingers) and ulnar/radial deviation (wrist) are discussed separately (Fig. 16a). (iii) The model with *User Adaptation* is used. *User Dependent* and *User Independent* results are discussed separately. (iv) The error bars denote the 10^{th} percentile and the 90^{th} percentile errors.

6.3.1 Qualitative Results. Fig. 2 depicts the qualitative results of finger motion tracking. A demo video has been published with this paper. The figure compares the tracking by *ssLOTR* with reference to the real hand and ground truth (Leap). Evidently, *ssLOTR* is able to capture a wide range of finger motions with decent accuracy. We believe these results are promising in the context of applications in augmented and virtual reality.

6.3.2 Overall Performance of ssLOTR. Fig. 13a depicts the comparison between *ssLOTR* and fully supervised learning. Evidently, with only 15% of labeled training data as fully supervised learning, the performance of *ssLOTR* is close to fully supervised learning. *ssLOTR* can also improve the median and 90^{th} percentile error 2.67 and 1.89 times over *SL-15%*, a version of supervised learning that uses the same amount of labeled training data as *ssLOTR*. While Fig. 13a reports joint angle errors, Fig. 13b reports the joint position errors. Specifically, the 10^{th}, 50^{th} and 90^{th} percentile of *ssLOTR* in degrees are 1.57°, 9.07° and 28.58°. And joint position errors of those are 1.14*mm*, 6.55*mm* and 20.64*mm*.

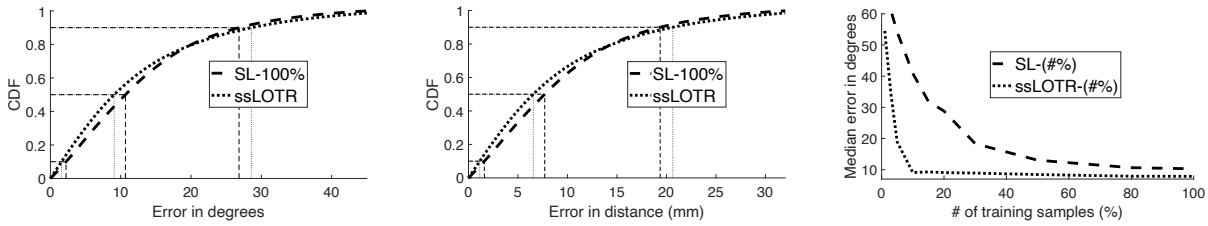


Fig. 13. Overall accuracy and training overhead. (a) Overall Accuracy *ssLOTR* vs Fully Supervised Learning (b) Joint position errors of Fig. 13a measured in millimeters. (c) Training Overhead. *ssLOTR* converges quickly with only a small amount of labeled training data

6.3.3 Accuracy vs Training Overhead. Fig. 13c depicts the accuracy as a function of the size of labeled training data. *ssLOTR* is able to quickly learn with a small amount of labeled data and achieve convergence. The proposed self-supervised learning framework captures better representations from sensor data, thus leading to fast and efficient learning. Because of this, with more labeled training data, *ssLOTR* can even outperform supervised learning. Given that unlabeled data can be constantly acquired without the overhead of labeling with a high precision ground truth, we believe the overhead of training with unlabeled data in *ssLOTR* is negligible. Another observation based on the figure is that the accuracy variation for $SL - 100\%$ saturates completely beyond 360s. Based on this observation, we note that $SL - 100\%$ needs 360 seconds of training data.

6.3.4 Robustness to Wrist Position and Orientation. Fig. 14a evaluates the robustness to changes in wrist positions and orientation. *ssLOTR* incorporates appropriate WCF transformations (Section 5.1). Therefore, the predictions by the ML model are independent of changes to wrist configurations. Accordingly, the accuracy is consistent across changes to wrist positions/orientations. Therefore, we believe this can improve the usability in applications like augmented reality and sports analytics where the wrist configuration can change continuously.

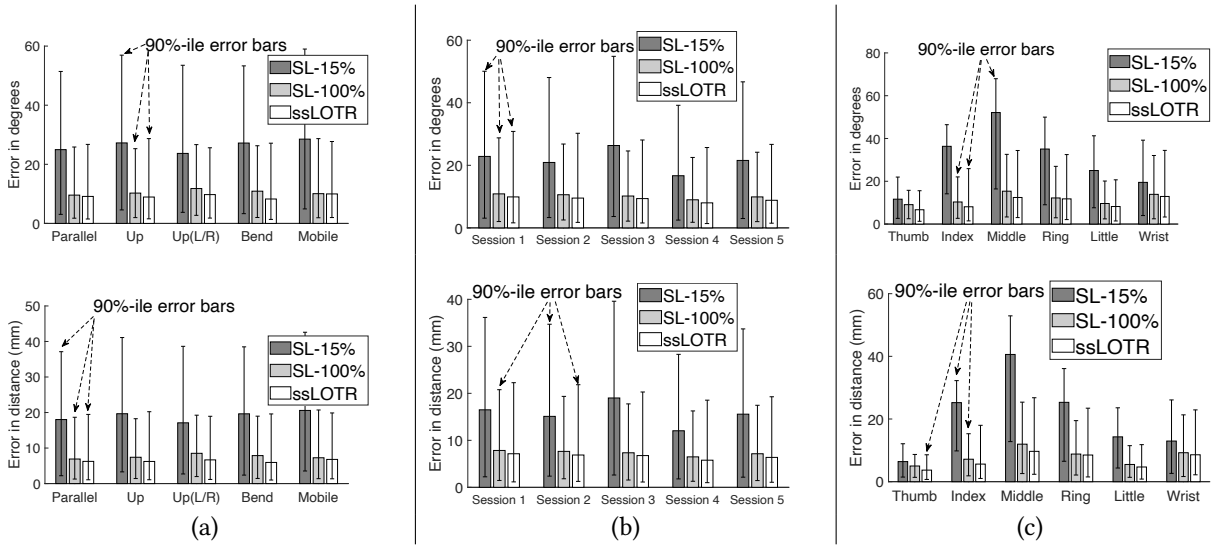


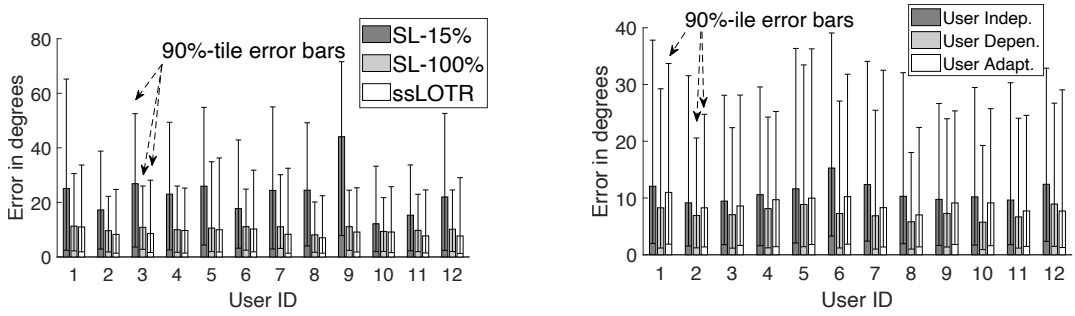
Fig. 14. Robustness of *ssLOTR* to various settings where top figures report error in degrees and bottom figures report error in millimeter (a) Wrist position and orientation (defined in Fig. 12) (b) Different sessions with changes to IMU sensor position and orientation (c) Different fingers and wrist

6.3.5 Robustness to Mobility. Fig. 14a also indicates the tracking accuracy when the wrist is mobile. The mobility includes both translational and rotational motion of the wrist with the fingers being simultaneously in active motion. Regardless of the mobility, the orientation tracking and WCF transformations are accurate, and the sensors on the fingers are still being able to capture the dominant component of finger motions. Therefore, the accuracy under mobility of the wrist is comparable to the accuracy under static conditions.

6.3.6 Robustness to Sensor Position. Fig. 14b depicts the variation in accuracy across different sessions. The sensors were removed and remounted across sessions thereby helping validate any effects of changes in sensor position or orientation with respect to the human body. Evidently, the accuracy is stable across various sessions. This is because the sensors are fit snugly to the hands, and any minor variation in positions/orientation across sessions is typically much smaller than the hardware noise floor, thus having a negligible impact on the accuracy.

6.3.7 Accuracy vs Fingers and Wrist. Fig. 14c depicts the accuracy as a function of fingers and wrist. The training data in *ssLOTR* incorporates the entire range of motion across all fingers and wrist. Therefore, *ssLOTR* is able to track all fingers with a stable accuracy. The robustness also extends towards the tracking of wrist angles. Across all cases, *ssLOTR* performs comparably to fully supervised learning with a clear gain due to self-supervision. The middle finger has a slightly higher error because it can be more influenced by the motion of other fingers causing minute vibrations in the middle finger. Nevertheless, we believe the overall tracking results are promising.

6.3.8 Accuracy vs Users. Fig. 15a and Fig. 15b depict the variation in accuracy across users. Our inspection of the data suggests that the variation happens due to the following reason. Some users perform faster and complex finger motion in comparison to other users. Nevertheless, the user with the worst performance in *ssLOTR* (User Adapt. in Fig. 15b) only has 2.13° higher error than the average case. Therefore, we believe *ssLOTR* provides an accuracy that is consistent across users with diversity in gender, body masses, sizes, etc. The contrastive learning framework together with data augmentation techniques introduce deliberate perturbations during the process of self-supervised learning. We believe this also helps develop inherent robustness to diversity across users. *ssLOTR*, with only limited labeled training data, achieves a performance that is close to the fully supervised learning (Fig. 15a) and the user dependent model (Fig. 15b).



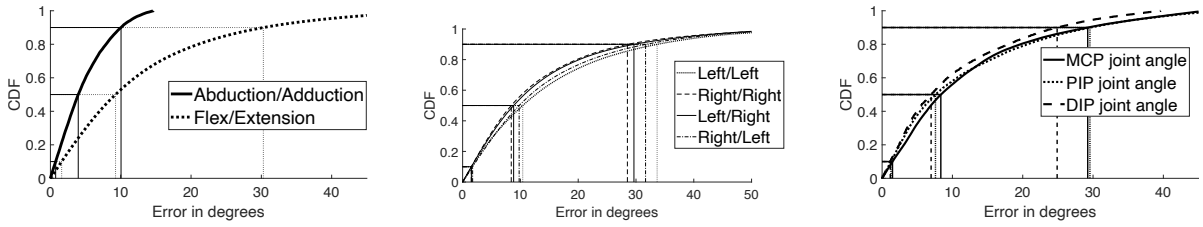
(a) (b)

Fig. 15. Accuracy across users (a) *ssLOTR* vs. supervised learning (b) Training process: User dependent vs. independent vs. adaptation

6.3.9 User Dependent vs. User Adaptation. Fig. 15b depicts the accuracy of the three versions of self-supervised learning – *User Dependent*, *User Independent*, and *User Adaptation (*ssLOTR*)*. While the performance of the *User Independent* model is still reasonable, with only a small number of user-specific training data, *User Adaptation model (*ssLOTR*)* can achieve accuracy close to the *User Dependent* model. Overall, *User Adaptation model (*ssLOTR*)* uses only a small fraction (≈ 10%) of user-specific training data in comparison to the *User Dependent* model, but achieves close to 88.6% of the performance of the *User Dependent* model.

6.3.10 Flex/Extension and Abduction/Adduction. Fig. 16a shows the CDF of errors for Flex/Extension angles in comparison with Abduction/Adduction. The figure shows the feasibility of *ssLOTR* in capturing the Abduction/Adduction angles which also includes radial/ulnar deviation angles for the wrist. Because of the smaller range of motion, the Abduction/Adduction angles have a smaller error in comparison to Flex/Extension angles. Overall, *ssLOTR* can capture all dimensions of finger motion reliably.

6.3.11 Transfer between Left and Right hand. Fig. 16b shows the accuracy when the model trained on the left and right hands are used for inferences across each other. When the sensor device (Fig. 5) is removed from the left hand and worn on the right hand, all sensors are rotated by 180 degrees about their z-axes (axis conventions in Fig. 8). Therefore, if the model is trained on the left hand and the inference is done on the right hand, the signs of x and y-axes of the data from the right hand have to be inverted. Evidently, the model trained on one hand is



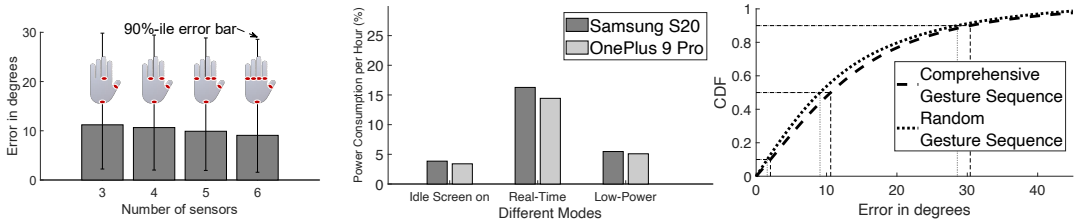
(a) (b) (c)

Fig. 16. (a) Accuracy for Flex/Extension in comparison to Abduction/Adduction (b) Accuracy for right and left hands (c) Accuracy over joints

applicable for performing inferences on the other hand, which offers scope for further decreasing the training overhead and making the model more consistent by aggregating data from both hands.

6.3.12 Accuracy vs Finger Joints. Fig. 16c depicts the accuracy as a function of various finger joints - MCP ($\phi_{mcp,fe}$), PIP (ϕ_{pip}), and DIP (ϕ_{dip}). *ssLOTR* can track all joint angles with decent accuracy. While the sensors placed on the fingers are sensitive to the motion of all finger joints, the motion of various joints has a high degree of interdependence because of the constraints (Section 3), which can help reduce the search space for the correct angle. By combining these opportunities, *ssLOTR*'s ML models can reliably compute various joint angles.

6.3.13 Accuracy vs. Number of Sensors. Fig. 17a depicts the accuracy as a function of the number of sensors. The specific set of sensors used for each case is also indicated in the figure, with the bottom-most red dot denoting the wrist sensor. Evidently, there is a graceful degradation in accuracy with fewer sensors. Because of the high



(a) (b) (c)

Fig. 17. (a) Accuracy over number of sensors used (b) Power consumption analysis on different modes (c) Comparison between our prior user study (Random Gesture Seq.) and the new user study (Comprehensive Gesture Seq.)

degree of interaction between fingers, the motion of one finger will cause other fingers to move, thus enabling capturing 3D joint angle information even without placing sensors on all fingers. For applications that do not need a higher precision, the footprint of the sensor can be minimized further to enhance the comfort levels.

6.3.14 Accuracy vs. Gesture Sequences. We conduct a new user study by having 12 users perform a sequence of finger motion gestures that are known to account for all possible hand poses within the constraints of anatomical feasibility (based on literature [67]). Specifically, the 28 finger pose gestures in Fig. 18 are known to be base states of human hand poses, and majority of possible hand poses are known to be one of these *base states* or transitioning between these poses [127] based on anatomical feasibility constraints. In our new user study, the users are instructed to pass through all of these *base states* (multiple times in no specific order), thus ensuring that all possible sequences of hand poses are accounted for. The accuracy is depicted in Fig. 17c. The median error with the new user study (*Comprehensive Gesture Sequence*) is only 1.4° higher (tail errors are similar) and comparable to the prior user study (*Random Gesture Sequence*) where the users were instructed to perform random finger

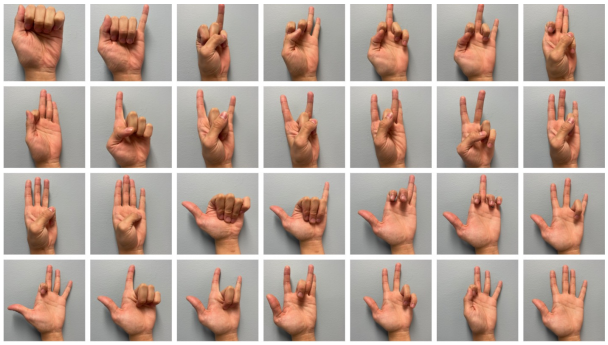


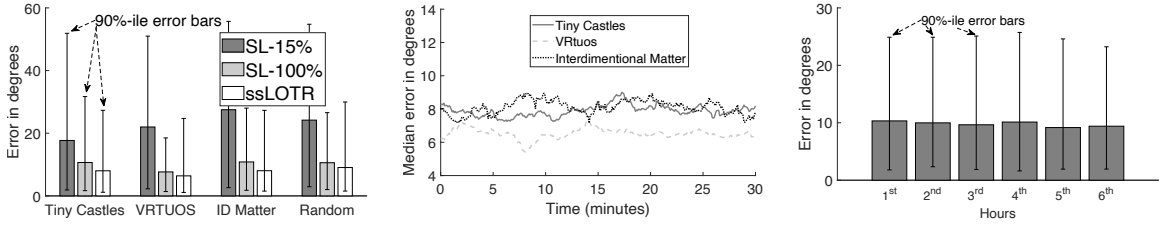
Fig. 18. Anatomically feasible base states of human hand poses.

motion while covering a full range of motion as possible for all fingers. This suggests that *ssLOTR* can accurately track anatomically feasible human hand poses.

Fig. 19. Users play VR games with Oculus Quest while wearing *ssLOTR*'s sensor

6.3.15 Longer Session AR/VR Experiments. Towards evaluating the efficacy of *ssLOTR* in a realistic setting, we perform some VR experiments with 12 users. Fig. 19 depicts a user wearing an Oculus Quest 2 [84] while being engaged in a VR game. The fingers are simultaneously being tracked by *ssLOTR*'s sensors as well as the leap sensor. Under this setting, we have 12 users who play the following three games: (i) Tiny Castles [126] (ii) VRTUOS [91] (iii) Interdimentional matter [90]. Each game lasted for about 30 minutes. Fig. 20a depicts the results. *ssLOTR* provides consistent accuracy across multiple games. When compared to the accuracy from prior user studies, the accuracy for some games might be better. This is because the finger motions in prior user studies cover the entire range of motion for all fingers to stress test the system under more challenging conditions. Fig. 20b shows the median accuracy as a function of time for these VR games. Evidently, errors do not accumulate with time, and we do not observe drift in errors because *ssLOTR* exploits opportunistic error compensation strategies from A3 [142].

6.3.16 Longer Session Experiments (Free Living Conditions). We conduct studies under free living conditions with 12 users to study long term effects like potential drifts. Users were instructed to wear the sensor continuously for 6 hours at their apartments. At the end of each hour, we conduct a 5-minute session of finger motion as per our user study protocol described earlier. In between the sessions, the users conducted normal daily life activities



(a) (b) (c)

Fig. 20. (a) VR Games (b) Accuracy as a function of time in three VR Games (c) Long sessions testing for drifts

which included working on their laptops (typing, browsing, etc), eating, drinking, watching movie, etc while they continue to wear the sensor. The results are depicted in Fig. 20c. As expected, the accuracy does not degrade with time. This is because *ssLOTR* does not perform long term integration of sensor data, which is the main source of drift errors [134]. In contrast, *ssLOTR* opportunistically resets drift based on ideas in prior works like A3 [142].

6.3.17 Power Consumption and Latency. The power consumption of the sensor device itself was discussed in Section 4. Here, we analyze the power consumption of the ML model of *ssLOTR* as implemented on smartphones. For profiling the energy of the TensorflowLite model, we use Batterystats and Battery Historian [5] tools. We compare the difference in power between two states: (i) The device is idle with the screen on. (ii) The device is making inferences using the TensorflowLite model. The idle display-screen on discharge rate 3.63% per hour while the discharge rates for various modes are shown in Fig. 17b. The power consumption is very low. Since the architecture in *ssLOTR* from Fig. 10 processes data in chunks of 1.4 seconds, it will incur a delay of at least 1.4 seconds if we process the data only once in 1.4 seconds. Towards making it real-time, we make a modification where at any given instant of time, previous 1.4 seconds segment of data is input to the network to obtain instantaneous real-time results. This provides real-time tracking at the expense of power. Depicted in Fig. 17b this entails continuous/redundant processing thus increasing the discharge rate to 15.35%. The low-power mode trades off real-time performance (1.4 seconds delay) for power savings. Depending on the requirements of real-time latency or energy efficiency, a user can choose between the two modes. The latency of execution of the ML model on Samsung S20 and Oneplus 9 Pro are 6.8 ms and 6.3 ms respectively.

6.3.18 Contrastive Loss vs MSE. Fig. 21a shows the variation in accuracy as a function of the loss function used for pretraining *ssLOTR* with unlabeled data. As expected, the contrastive loss function provides better accuracy than the conventional MSE loss function. While the MSE loss function considers only a pair of similar samples at a time and maximizes their similarity, the contrastive loss function tries to simultaneously maximize the similarity between similar pairs and minimize the similarity between dissimilar pairs as outlined in Section 5.3 and Equation 5. Therefore, the contrastive loss function results in better efficiency in capturing the representations.

6.3.19 Motivation for the Projection Head. In Section 5.3, in reference to the *ssLOTR*'s architecture in Fig. 10, we discussed the possibility of applying the contrastive loss function directly to the representation h, or to its projection $y = p(h)$. In this section, we experimentally validate the two choices. Fig. 21b compares the two design choices. Evidently, applying the contrastive loss to y is better by 48.7% in the median case in comparison to h. While projections y might help enforce similarity between differently augmented versions of the same input, there might be some loss of information. On the other hand, the representations h which are one level before y offer a good tradeoff in capturing high-level features without losing much of the information.

6.3.20 Variation across Data Augmentation Techniques. ssLOTR's architecture in Fig. 10 needs two differently augmented copies of the same input. We discussed four techniques for performing this data augmentation in Section 5.3. We now evaluate all six combinations of selecting two such data augmentation techniques for

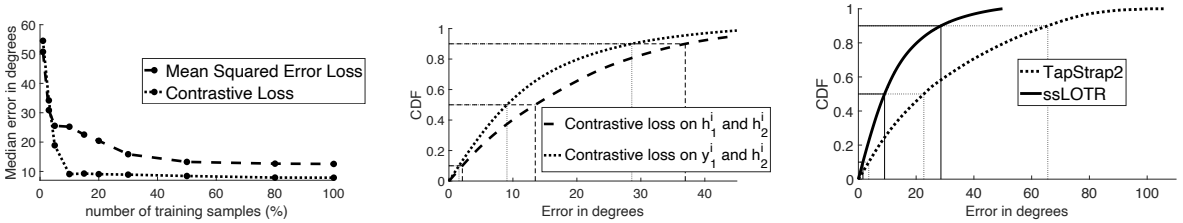
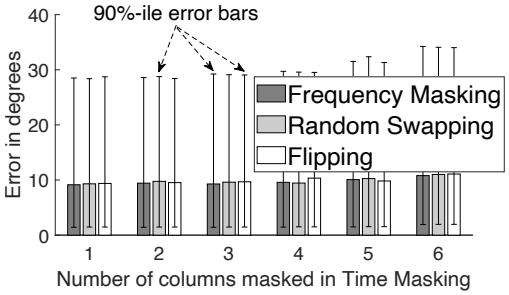


Fig. 21. (a) Contrastive loss results in efficient learning than MSE loss (b) Contrastive loss on different representations (c) *ssLOTR* vs TapStrap2



TM	FM	RS	F	10%-ile error	50%-ile error	90%-ile error
✓	✓			1.43	9.14	28.50
✓		✓		1.47	9.31	28.37
✓			✓	1.47	9.38	28.32
	✓	✓		1.44	8.91	27.49
	✓		✓	1.52	9.43	28.68
		✓	✓	1.42	9.37	28.48

TM: Time Masking. **FM**: Frequecy Masking. **RS**: Random Swapping. **F**: Flipping.

Fig. 22. (a) Figure shows empirical results of time masking with different number of masked columns with one of the other three data augmentations (b) Table describes comparison between combinations of data augmentation techniques

augmenting the input to pass it into the architecture in Fig. 10. Our results in Table of Fig. 22 indicate that all of the combinations provide consistent performance without any significant difference. Moreover, we study the effect of number of masked columns in time masking technique. Our results (Left-sided figure in Fig. 22) suggests that randomly masking one column yields better accuracy.

6.3.21 Comparison with TapStrap2. Tapstrap2 only has accelerometers. The gyroscope and magnetometer sensors are not embedded. We develop self-supervised learning algorithms with such data accessed from the Tapstrap2, based on the architecture in Fig. 10. Fig. 21c shows the performance of *ssLOTR* in comparison to TapStrap2. With the same amount of labeled and unlabeled training data, the performance of *ssLOTR* in contrast to TapStrap2 is substantially higher. TapStrap2 also has a longer tail due to convergence issues. Accelerometer contains less information in comparison to a 9-axis IMU sensor. Moreover, the lack of gyroscope and magnetometer makes it difficult to estimate orientation and perform gravity compensation and coordinate alignment techniques discussed in Section 5.1, thereby causing issues of stability and convergence of the ML models with TapStrap2 data.

7 DISCUSSION AND FUTURE WORK

Usefulness of form factor: Our primary requirement includes the ability to track 24 DoF motion of fingers while being comfortable for wearing that allows performing natural finger motion activities without any hindrance from the sensor. While tracking 24 DoF finger joints has been systematically validated in Section 6, we discuss the usability aspects of the form factor in comparison to other design choices as follows. (i) Our sensors are worn like connected rings on fingers and a smartwatch on the wrist. Based on the free-living study conducted in Section 6 (Fig. 20c), the users were able to comfortably perform daily life activities which included working on their laptops (typing, browsing, etc), eating, drinking, watching movie, etc while they continue to wear the sensor. Therefore we believe that the platform is comfortable to wear, but we note that there is scope for improvement as

noted in further comments. (ii) Prior research on glove prototypes suggest that it is difficult to perform daily activities with precise finger motion such as typing, knitting, opening a lock, etc, while wearing gloves [103]. In contrast, *ssLOTR*'s sensor device allows free motion of fingers. Wristbands and armbands might be other alternatives. Wristband based solutions like FingerTrak [40] can use infrared cameras for 24 DoF finger motion tracking, but as pointed out by the authors themselves, the system is not robust to background temperatures (sun, heater, etc) as well as changes in sensor position due to wrist motion. EMG armbands on the otherhand need calibration and warming of the skin to be in proper contact with the electrodes which can even take up to 5 minutes during each instance of wearing, leading to usability issues [87, 128, 136]. (iii) Nevertheless, we note that the form factor in *ssLOTR* is still a work in progress and there is scope for improvement. One of the goals of the paper is to explore the limits and bounds of the accuracy as a function of the number of sensors. The experimental evidence in Section 6, Fig 17a suggests a graceful degradation in accuracy with decreasing number of sensors. Therefore, we believe the form factor of the prototype can further be reduced with a graceful degradation in accuracy. Building upon this promise, we will also consider prototyping ring based form factors , as a part of our future work. Depending on the application and accuracy requirements, perhaps the rings can only be worn on a few fingers instead of all fingers.

Target population and potential use cases: While *ssLOTR* shows the feasibility of generic finger motion tracking with the ability to be extended to any application, we list a few potential use cases below.

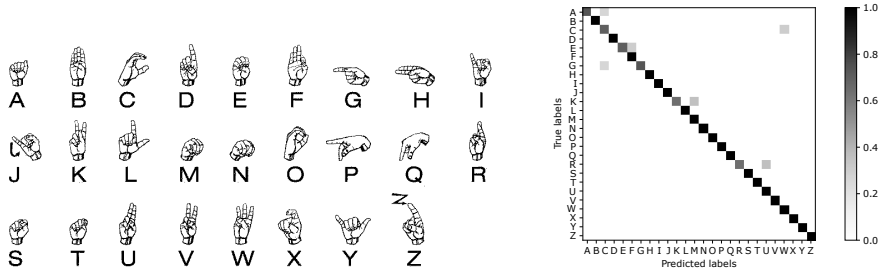
■ *Virtual and Augmented Reality*: Experiments in Fig. 20a validate the applicability of *ssLOTR* in a AR/VR setting. Building on this promise, we plan to leverage the finger motion tracking from *ssLOTR* to create more interactive AR/VR applications such as cooking, artwork training, sports coaching, etc. This can enable hands-on learning experience where the tutorials are often packaged in the form of engaging game-like activities. We believe the generic nature of 3D finger motion tracking enabled by *ssLOTR* can be extended towards such applications.

■ *Accessibility*: Accessibility applications [116] such as Sign Language Recognition and Translation can bridge the communication gap between deaf people and hearing people. Techniques based on cameras are limited by the need for good lighting, and resolution. Furthermore, they lack portability and ubiquity because the sensing is limited to the range of the camera. Therefore, we believe wearable based solutions like *ssLOTR* can be used in the context of Sign Language Recognition to offer a ubiquitous solution. We conduct a simple study to classify alphabets (Fig. 23a) in American Sign Language (ASL) using *ssLOTR*'s finger motion tracking for classification. Depicted in Fig. 23b, *ssLOTR* is able to accurately classify the alphabets. We plan to extend *ssLOTR* in the future towards recognition over a larger vocabulary. *ssLOTR* could also be useful for people with vision impairments, where they are required to draw or perform some gestures for user interface applications [48].

■ *User Interfaces (Typing)*: Similar to TapStrap 2, we believe typing is another potential use case with the following benefits over conventional keyboard: (i) Typing can be done anywhere, without a keyboard, thus offering a more ubiquitous way of interacting with IoT devies, particularly the ones with smaller form factor (ii) Tap based typing can be much faster (almost twice) than conventional typing [124].

■ *Robotic Teleoperation*: Complex and unstructured robotic operation, especially in an unregulated environment may require human intelligence in addition to mechanical sturdiness and robustness of a robot. This might include applications ranging from controlling a home assistant robotic avatars or a robotic avatar in a dangerous industrial setting [25] in tasks including grasping and manipulating objects in complex ways. Towards this end, we believe 24 DoF finger motion tracking in *ssLOTR* can provide a ubiquitous solution for robotic avatar control, which is particularly useful if the control is desired from anywhere, anytime.

Scope for improving the appearance and mechanical design: Our goal with the paper is to show the feasibility of designing a lightweight wireless sensing device for 3D finger motion tracking. However, we believe there is more work to be done before the sensor can be turned into a finished product. The mechanical design and appearance can be optimized. For example, stereolithograph apparatus (SLA) technology [117] can be used to enhance the resolution of 3D printing. However, the liquid resins used in SLA can be sensitive to UV exposure



(a) (b)

Fig. 23. (a) American Sign Language Alphabet (b) Confusion matrix for American Sign Language characters classification

and much more expensive than PLA and TPU material used in *ssLOTR* [30]. Moreover, the SLA printing material made of resin is highly toxic with safety issues for operation in a lab setting. The tradeoffs in resolution, material cost, and manufacting cost must be carefully considered.

Scope for decreasing the number of IMU sensors: Our results in Fig. 17a show a graceful degradation in accuracy with the number of sensors. Because of the high degree of interaction between fingers, the motion of one finger will cause other fingers to move, thus enabling capturing 3D joint angle information even without placing sensors on all fingers. This provides opportunities to further minimize the size of the sensing device by only using a few sensors. We will consider alternative ways to package the sensing device with fewer sensors.

Human Body Pose Detection: *ssLOTR* shows the feasibility of sensing 3D finger motion with limited training data using self-supervised learning techniques. Motivated by the promising results, we plan to explore full-body motion tracking using a similar framework. In particular, we are interested in understanding the tradeoffs between the number of sensors placed on the body, the amount of training data, and the achievable accuracy.

8 CONCLUSION

This paper shows the feasibility of self-supervised 3D finger motion tracking using small-scale training data. A novel wireless sensing device was designed by exploiting advances in PCB design, 3D printed packaging, and SoC microcontrollers with integrated WiFi/BLE for efficient sensing and comfortable wearing that enables dexterous motion of fingers. Extensive user study with 12 users shows a median tracking error of 9.07° (or 6.55 mm) with robustness to user diversity, body masses, sensor wearing positions, etc. While the results are promising, we believe this opens ample opportunities for future research in several areas including augmented and virtual reality, self-supervised learning for body pose tracking with non-obtrusive, and sparse sensors and limited training data.

ACKNOWLEDGMENTS

This research was partially supported by NSF grants: CAREER-2046972, and CNS-1909479.

REFERENCES

[1] 5DT Data Glove Ultra - 5DT 2004. 5DT Data Glove Ultra. https://5dt.com/5dt-data-glove-ultra/.
[2] ACRO. 2017. Anatomy of the Wrist. https://www.acropt.com/blog/2017/5/28/anatomy-of-the-wrist-mfkkp.
[3] Rocco Agostino et al. 2003. Impairment of individual finger movements in Parkinson's disease. *Movement disorders* 18, 5 (2003), 560–565.
[4] Mohamed Aktham Ahmed et al. 2018. A review on systems-based sensory gloves for sign language recognition state of the art between 2007 and 2017. *Sensors* 18, 7 (2018), 2208.
[5] Android Developer 2021. Profile battery usage with Batterystats and Battery Historian. https://developer.android.com/topic/performance/power/setup-battery-historian.
[6] Auto Desk Eagle 2021. ADE. https://en.wikipedia.org/wiki/EAGLE_(program).

[7] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477* (2020).

[8] Neil M Bajaj, Adam J Spiers, and Aaron M Dollar. 2015. State of the art in prosthetic wrists: Commercial and research devices. In *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*. IEEE, 331–338.

[9] Benjamin Levin 2019. The Tap Strap 2 turns your hand into a keyboard and mouse. https://www.cnn.com/2019/12/23/cnn-underscored/tap-strap-2-wearable-keyboard-mouse-review/index.html.

[10] Mario Bertero, Christine De Mol, and Giovanni Alberto Viano. 1980. The stability of inverse problems. In *Inverse scattering problems in optics*. Springer, 161–214.

[11] BLE v4.2 2021. Core Specification 4.2. https://www.bluetooth.com/specifications/specs/core-specification-4-2/.

[12] Arij Bouazizi, Julian Wiederer, Ulrich Kressel, and Vasileios Belagiannis. 2021. Self-Supervised 3D Human Pose Estimation with Multiple-View Geometry. *arXiv preprint arXiv:2108.07777* (2021).

[13] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. 2018. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *ECCV*.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[15] Yujin Chen, Zhigang Tu, Liuhao Ge, Dejun Zhang, Ruizhi Chen, and Junsong Yuan. 2019. So-handnet: Self-organizing network for 3d hand pose estimation with semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6961–6970.

[16] Fai Chen Chen et al. 2013. Constraint study for a hand exoskeleton: human hand kinematics and dynamics. *Journal of Robotics* 2013 (2013).

[17] James Connolly et al. 2017. IMU sensor-based electronic goniometric glove for clinical finger movement analysis. *IEEE Sensors Journal* (2017).

[18] Francesca Cordella et al. 2012. Patient performance evaluation using Kinect and Monte Carlo-based finger tracking. In *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE, 1967–1972.

[19] CyberGlove Systems LLC 2017. CyberGlove. http://www.cyberglovesystems.com/.

[20] Aneesh Dahiya, Adrian Spurr, and Otmar Hilliges. 2021. Exploring self-supervised learning techniques for hand pose estimation. In *NeurIPS 2020 Workshop on Pre-registration in Machine Learning*. PMLR, 255–271.

[21] Dan Harriman 2011. Finger Placement When Shooting a Basketball. https://www.sportsrec.com/476507-finger-placement-when-shooting-a-basketball.html.

[22] Ashwin De Silva et al. 2020. Real-Time Hand Gesture Recognition Using Temporal Muscle Activation Maps of Multi-Channel sEMG Signals. *arXiv:2002.03159* (2020).

[23] Jacob Devlin et al. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[24] Driveline Baseball 2020. How to throw a knuckle ball. https://www.drivelinebaseball.com/2020/10/how-to-throw-a-knuckleball/.

[25] Guanglong Du, Ping Zhang, Jianhua Mai, and Zeling Li. 2012. Markerless kinect-based hand tracking for robot teleoperation. *International Journal of Advanced Robotic Systems* 9, 2 (2012), 36.

[26] Yu Du et al. 2017. Semi-Supervised Learning for Surface EMG-based Gesture Recognition.. In *IJCAI*.

[27] SparkFun Electronics. 2021. SparkFun Qwiic I2C Mux Arduino Library. https://github.com/sparkfun/SparkFun_I2C_Mux_Arduino_Library.

[28] ESP32 2021. ESP32 Pico d4 Datasheet. https://www.espressif.com/sites/default/files/documentation/esp32-pico-d4_datasheet_en.pdf.

[29] Haytham M. Fayek. 2016. Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between. https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

[30] FormLabs. 2021. FDM vs. SLA: Compare the Two Most Popular Types of 3D Printers. https://formlabs.com/blog/fdm-vs-sla-compare-types-of-3d-printers/.

[31] FR4 2014. FR-4. https://en.wikipedia.org/wiki/FR-4.

[32] Fused filament fabrication 2021. 3D printing. https://en.wikipedia.org/wiki/Fused_filament_fabrication#Fused_deposition_modeling.

[33] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. 2018. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728* (2018).

[34] Dan Ginsburg, Budirijanto Purnomo, Dave Shreiner, and Aaftab Munshi. 2014. *OpenGL ES 3.0 programming guide*. Addison-Wesley Professional.

[35] Oliver Glauser et al. 2019. Interactive hand pose estimation using a stretch-sensing soft glove. *ACM Transactions on Graphics (TOG)* (2019).

[36] Google. 2019. Deploy machine learning models on mobile and IoT devices. "https://www.tensorflow.org/lite".

[37] Mahanth Gowda, Ashutosh Dhekne, Sheng Shen, Romit Roy Choudhury, Lei Yang, Suresh Golwalkar, and Alexander Essanian. 2017. Bringing IoT to sports analytics. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 499–513.

[38] Harish Haresamudram, Apoorva Beedu, Varun Agrawal, Patrick L Grady, Irfan Essa, Judy Hoffman, and Thomas Plötz. 2020. Masked reconstruction based self-supervision for human activity recognition. In *Proceedings of the 2020 International Symposium on Wearable Computers*. 45–49.

[39] Stefan Hesse, H Kuhlmann, J Wilk, C Tomelleri, and Stephen GB Kirker. 2008. A new electromechanical trainer for sensorimotor rehabilitation of paralysed fingers: a case series in chronic and acute stroke patients. *Journal of neuroengineering and rehabilitation* 5, 1 (2008), 21.

[40] Fang Hu et al. 2020. FingerTrak: Continuous 3D Hand Pose Tracking by Deep Learning Hand Silhouettes Captured by Miniature Thermal Cameras on Wrist. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2020).

[41] Adafruit Industries. 2021. Adafruit ICM20X. https://github.com/adafruit/Adafruit_ICM20X.

[42] TEXAS INSTRUMENTS. 2019. "TCA9548A" datasheet [online]. https://www.ti.com/lit/ds/symlink/tca9548a.pdf?ts=1635823287205&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTCA9548A.

[43] Interhaptics 2021. Hand tracking for virtual reality (VR) and mixed reality (MR). https://www.interhaptics.com/products/hand-tracking-for-vr-and-mr.

[44] Invense. 2021. "ICM20948" datasheet [online]. https://3cfeqx1hf82y3xcoull08ihx-wpengine.netdna-ssl.com/wp-content/uploads/2021/10/DS-000189-ICM-20948-v1.5.pdf.

[45] Sergey Ioffe et al. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[46] Umar Iqbal et al. 2018. Hand pose estimation via latent 2.5 d heatmap regression. In *ECCV*.

[47] Jamie Feltham 2020. Cooking Simulator VR Lets You Cut Onions Without The Tears. https://uploadvr.com/cooking-simulator-vr-trailer/.

[48] Shaun K Kane, Jacob O Wobbrock, and Richard E Ladner. 2011. Usable gestures for blind people: understanding preference and performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 413–422.

[49] Hsin-Liu Cindy Kao, Abdelkareem Bedri, and Kent Lyons. 2018. SkinWire: Fabricating a Self-Contained On-Skin PCB for the Hand. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 1–23.

[50] Abdelwahed Khamis, Branislav Kusy, Chun Tung Chou, Mary-Louise McLaws, and Wen Hu. 2020. RFWash: a weakly supervised tracking of hand hygiene technique. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 572–584.

[51] Wolf Kienzle, Eric Whitmire, Chris Rittaler, and Hrvoje Benko. 2021. ElectroRing: Subtle Pinch and Touch Detection with a Ring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[52] David Kim et al. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *ACM UIST*.

[53] Kinect 2021. Microsoft Kinect2.0. https://developer.microsoft.com/en-us/windows/kinect.

[54] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[55] Neil Kolban. 2017. ESP32 BLE for Arduino. https://github.com/nkolban/ESP32_BLE_Arduino.

[56] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. 2019. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1920–1929.

[57] Paul G Kry, Doug L James, and Dinesh K Pai. 2002. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 153–159.

[58] Hyeokhyen Kwon, Catherine Tong, Harish Haresamudram, Yan Gao, Gregory D Abowd, Nicholas D Lane, and Thomas Ploetz. 2020. Imutube: Automatic extraction of virtual on-body accelerometry from video for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–29.

[59] Leap Motion Developer 2012. Leap Motion. https://developer.leapmotion.com/.

[60] Stepan Lemak, Viktor Chertopolokhov, Ivan Uvarov, Anna Kruchinina, Margarita Belousova, Leonid Borodkin, and Maxim Mironenko. 2020. Inertial sensor based solution for finger motion tracking. *Computers* 9, 2 (2020), 40.

[61] Gang Li, Rui Zhang, Matthew Ritchie, and Hugh Griffiths. 2017. Sparsity-based dynamic hand gesture recognition using micro-Doppler signatures. In *2017 IEEE Radar Conference (RadarConf)*. IEEE, 0928–0931.

[62] Hong Li, Wei Yang, Jianxin Wang, Yang Xu, and Liusheng Huang. 2016. WiFinger: talk to your smart devices with finger-grained gesture. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 250–261.

[63] 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.

[64] Shan Sung Liew et al. 2016. Bounded activation functions for training stability of deep neural networks on visual pattern recognition problems. *Neurocomputing* (2016).

[65] Bor-Shing Lin et al. 2018. Design of an inertial-sensor-based data glove for hand function evaluation. *Sensors* (2018).

[66] John Lin, Ying Wu, and Thomas S Huang. 2000. Modeling the constraints of human hand motion. In *Proceedings workshop on human motion*. IEEE, 121–126.

[67] John Lin, Ying Wu, and Thomas S Huang. 2000. Modeling the constraints of human hand motion. In *Proceedings workshop on human motion*. IEEE, 121–126.

[68] LiPo 2021. Lithium polymer battery - Wikipedia. https://en.wikipedia.org/wiki/Lithium_polymer_battery.

[69] Dongxin Liu, Tianshi Wang, Shengzhong Liu, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. 2021. Contrastive Self-Supervised Representation Learning for Sensing Signals from the Time-Frequency Perspective. In *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 1–10.
[70] Jiayang Liu et al. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* (2009).
[71] Xialei Liu, Joost Van De Weijer, and Andrew D Bagdanov. 2019. Exploiting unlabeled data in cnns by self-supervised learning to rank. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2019), 1862–1878.
[72] Yang Liu et al. 2019. Real-time Arm Skeleton Tracking and Gesture Inference Tolerant to Missing Wearable Sensors. In *ACM MobiSys*.
[73] Yang Liu, Chengdong Lin, and Zhenjiang Li. 2021. WR-Hand: Wearable Armband Can Track User's Hand. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–27.
[74] Yilin Liu, Shijia Zhang, and Mahanth Gowda. 2021. NeuroPose: 3D Hand Pose Tracking using EMG Wearables. In *Proceedings of the Web Conference 2021*. 1471–1482.
[75] Yilin Liu, Shijia Zhang, and Mahanth Gowda. 2021. When Video meets Inertial Sensors: Zero-shot Domain Adaptation for Finger Motion Analytics with Inertial Sensors. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*. 182–194.
[76] Yongsen Ma, Gang Zhou, Shuangquan Wang, Hongyang Zhao, and Woosub Jung. 2018. Signfi: Sign language recognition using wifi. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 23.
[77] Maurizio Maisto, Claudio Pacchierotti, Francesco Chinello, Gionata Salvietti, Alessandro De Luca, and Domenico Prattichizzo. 2017. Evaluation of wearable haptic systems for the fingers in augmented reality applications. *IEEE transactions on haptics* 10, 4 (2017), 511–522.
[78] Unexpected Maker. 2017. "ICM20948" datasheet [online]. https://www.tinypico.com.
[79] Manus VR 2016. Industry leading VR techology. https://manus-vr.com/.
[80] Matlab. 2021. matlab. https://matplotlib.org/.
[81] Matric Group 2019. IPC Class Definitions for Class 1, 2 and 3 Electronics. https://blog.matric.com/ipc-class-definitions-class-1-2-3-electronics.
[82] Pedro Melgarejo, Xinyu Zhang, Parameswaran Ramanathan, and David Chu. 2014. Leveraging directional antenna capabilities for fine-grained gesture recognition. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 541–551.
[83] Leonardo Meli, Davide Barcelli, Tommaso Lisini Baldi, and Domenico Prattichizzo. 2017. Hand in air tapping: A wearable input technology to type wireless. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 936–941.
[84] Meta 2020. Oculus Quest 2. https://www.oculus.com/quest-2/.
[85] Tomas Mikolov, Kai Chen, Gregory S Corrado, and Jeffrey A Dean. 2015. Computing numeric representations of words in a high-dimensional space. US Patent 9,037,464.
[86] Franziska Mueller et al. 2018. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *IEEE CVPR*.
[87] Myo 2021. Tutorial. https://support.getmyo.com/hc/en-us/articles/203910089-Warm-up-while-wearing-your-Myo-armband.
[88] Rajalakshmi Nandakumar et al. 2016. Fingerio: Using active sonar for fine-grained finger tracking. In *ACM CHI*.
[89] BID Needham. 2021. Anatomy: Hand and Wrist. https://www.bidneedham.org/departments/orthopaedics/hand-program/anatomy-hand-and-wrist.
[90] Oculus Quest 2020. Interdimensional Matter. https://sidequestvr.com/app/506/interdimensional-matter.
[91] Oculus Quest 2020. VRTUOS. https://www.oculus.com/experiences/quest/3827275690649134/.
[92] Oura Ring 2021. Oura Ring: The most accurate sleep and activity tracker. https://ouraring.com/.
[93] Lizhi Pan et al. 2014. Continuous estimation of finger joint angles under different static wrist motions from sEMG signals. *Biomedical Signal Processing and Control* (2014).
[94] Abhinav Parate et al. 2014. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In *ACM MobiSys*.
[95] Farshid Salemi Parizi, Eric Whitmire, and Shwetak Patel. 2019. AuraRing: Precise Electromagnetic Finger Tracking. *ACM IMWUT* (2019).
[96] Keunwoo Park, Sunbum Kim, Youngwoo Yoon, Tae-Kyun Kim, and Geehyuk Lee. 2020. DeepFisheye: Near-surface multi-finger tracking technology using fisheye camera. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 1132–1146.
[97] 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), 8026–8037.
[98] PLA 2021. Polylactic Acid. https://en.wikipedia.org/wiki/Polylactic_acid.

[99] Chen Qu et al. 2019. BERT with history answer embedding for conversational question answering. In *ACM SIGIR Conference on Research and Development in Information Retrieval*.

[100] Fernando Quivira et al. 2018. Translating sEMG signals to continuous hand poses using recurrent neural networks. In *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*. IEEE.

[101] Sumit Raurale et al. 2018. Emg acquisition and hand pose classification for bionic hands from randomly-placed sensors. In *IEEE ICASSP*.

[102] Sai Deepika Regani, Chenshu Wu, Beibei Wang, Min Wu, and KJ Ray Liu. 2021. mmWrite: Passive Handwriting Tracking Using a Single Millimeter Wave Radio. *IEEE Internet of Things Journal* (2021).

[103] Alba Roda-Sales et al. 2020. Effect on manual skills of wearing instrumented gloves during manipulation. *Journal of biomechanics* (2020).

[104] Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. 2019. Multi-task self-supervised learning for human activity detection. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 2 (2019), 1–30.

[105] Panneer Selvam Santhalingam, Yuanqi Du, Riley Wilkerson, Ding Zhang, Parth Pathak, Huzefa Rangwala, Raja Kushalnagar, et al. 2020. Expressive ASL Recognition using Millimeter-wave Wireless Signals. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[106] Jiacheng Shang and Jie Wu. 2017. A robust sign language recognition system with multiple wi-fi devices. In *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*. ACM, 19–24.

[107] Sheng Shen et al. 2016. I am a Smartwatch and I can Track my User's Arm. In *ACM MobiCom*.

[108] Sheng Shen et al. 2018. Closing the Gaps in Inertial Motion Tracking. In *ACM MobiCom*.

[109] Shenzhen JIALICHUANG Electronic Technology Development Co.,Ltd 2021. JLCPCB. https://jlcpcb.com/.

[110] Michael Sherman et al. 2014. User-generated free-form gestures for authentication: Security and memorability. In *ACM MobiSys*.

[111] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of big data* 6, 1 (2019), 1–48.

[112] SolidWorks. 2015. SolidWorks. https://en.wikipedia.org/wiki/SolidWorks.

[113] Ivan Sosin et al. 2018. Continuous gesture recognition from sEMG sensor data with recurrent neural networks and adversarial domain adaptation. In *International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE.

[114] Sparkfun 2021. I2C - learn.sparkfun.com. https://learn.sparkfun.com/tutorials/i2c/all.

[115] Sorawit Stapornchaisit, Yeongdae Kim, Atsushi Takagi, Natsue Yoshimura, and Yasuharu Koike. 2019. Finger Angle estimation from Array EMG system using linear regression model with Independent Component Analysis. *Frontiers in Neurorobotics* 13 (2019).

[116] Thad Starner, Joshua Weaver, and Alex Pentland. 1998. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on pattern analysis and machine intelligence* 20, 12 (1998), 1371–1375.

[117] Stereolithography 2021. Stereolithography. https://en.wikipedia.org/wiki/Stereolithography.

[118] Wei Sun, Franklin Mingzhe Li, Congshu Huang, Zhenyu Lei, Benjamin Steeper, Songyun Tao, Feng Tian, and Cheng Zhang. 2021. ThumbTrak: Recognizing Micro-finger Poses Using a Ring with Proximity Sensing. *arXiv preprint arXiv:2105.14680* (2021).

[119] Evan A Susanto et al. 2015. Efficacy of robot-assisted fingers training in chronic stroke survivors: a pilot randomized-controlled trial. *Journal of neuroengineering and rehabilitation* (2015).

[120] Espressif System. 2021. Espressif Systems LTD. https://www.espressif.com/en/products/socs.

[121] Ryo Takahashi, Masaaki Fukumoto, Changyo Han, Takuya Sasatani, Yoshiaki Narusue, and Yoshihiro Kawahara. 2020. TelemetRing: A Batteryless and Wireless Ring-shaped Keyboard using Passive Inductive Telemetry. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 1161–1168.

[122] TapWithUs 2019. Tap Strap 2: Wearable Keyboard, Mouse, and Air Gesture Control. https://www.tapwithus.com/.

[123] TapWithUs. 2021. Customer Reviews. https://www.tapwithus.com/reviews/.

[124] TapWithUs 2021. How Tap Works: A Detailed Look at the World's Most Advanced Input Device. https://www.tapwithus.com/how-tap-works/.

[125] Thermoplastic polyurethane 2021. TPU. https://en.wikipedia.org/wiki/Thermoplastic_polyurethane.

[126] TinyCo, Inc. 2010. Tiny Castle. https://www.oculus.com/experiences/quest/3647163948685453/.

[127] Carlo Tomasi, Slav Petrov, and Arvind Sastry. 2003. 3D Tracking= Classification+ Interpolation.. In *ICCV*, Vol. 3. 1441.

[128] Timothy Torres. 2015. Myo Gesture Control Armband Review. https://www.pcmag.com/reviews/myo-gesture-control-armband.

[129] Hoang Truong et al. 2018. CapBand: Battery-free Successive Capacitance Sensing Wristband for Hand Gesture Recognition. In *ACM SenSys*.

[130] Jason Tu, Angeline Vidhula Jeyachandra, Deepthi Nagesh, Naresh Prabhu, and Thad Starner. 2021. Typing on Tap: Estimating a Finger-Worn One-Handed Chording Keyboard's Text Entry Rate. In *2021 International Symposium on Wearable Computers*. 156–158.

[131] Yu-Chih Tung and Kang G Shin. 2015. Echotag: Accurate infrastructure-free indoor location tagging with smartphones. In *ACM MobiCom*.

[132] Stefan Wager, Sida Wang, and Percy S Liang. 2013. Dropout training as adaptive regularization. In *Advances in neural information processing systems*. 351–359.

[133] Feng Wang and Huaping Liu. 2021. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2495–2504.
[134] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. 2012. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. 197–210.
[135] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. 2016. Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 851–860.
[136] Jørgen Winkel et al. 1991. Significance of skin temperature changes in surface electromyography. *European journal of applied physiology and occupational physiology* (1991).
[137] Erwin Wu, Ye Yuan, Hui-Shyong Yeo, Aaron Quigley, Hideki Koike, and Kris M Kitani. 2020. Back-Hand-Pose: 3D Hand Pose Estimation for a Wrist-worn Camera via Dorsum Deformation Network. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 1147–1160.
[138] Xtensa lx6 2021. Datasheet. https://mirrobo.ru/wp-content/uploads/2016/11/Cadence_Tensillica_Xtensa_LX6_ds.pdf.
[139] Cheng Zhang et al. 2018. FingerPing: Recognizing Fine-grained Hand Poses using Active Acoustic On-body Sensing. In *ACM CHI*.
[140] Haoyu Zhang, Jianjun Xu, and Ji Wang. 2019. Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243* (2019).
[141] Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful image colorization. In *European conference on computer vision*. Springer, 649–666.
[142] Pengfei Zhou et al. 2014. Use it free: Instantly knowing your phone attitude. In *ACM MobiCom*.