ActionSense: A Multimodal Dataset and Recording Framework for Human Activities Using Wearable Sensors in a Kitchen Environment

Joseph DelPreto*, **Chao Liu***, **Yiyue Luo, Michael Foshey, Yunzhu Li,**
Antonio Torralba, Wojciech Matusik, and Daniela Rus
MIT CSAIL, Cambridge, MA 02139
{delpreto, chaoliu, yiyueluo, mfoshey, liyunzhu}@csail.mit.edu
{torralba, wojciech, rus}@csail.mit.edu

Supplementary Materials

*These authors contributed equally to this work

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

A Overview of Dataset Contents

The `ActionSense` dataset contains high-resolution synchronized data streams from a suite of wearable sensors, environment-mounted sensors, and ground truth labels. A summary of the dataset is provided in Figure 1. This dataset label follows the structure proposed by [1, 3], and the remainder of the supplementary materials addresses the additional questions suggested by this framework.

A.1 Dataset Size

The current dataset contains 10 subjects, and is actively growing with a target of containing approximately 25 subjects. It currently spans approximately 778.0 minutes of recorded data, averaging 77.8 ± 16.4 minutes per subject. Approximately 543.5 minutes of that time is occupied by performing kitchen activities (55.6 ± 13.7 minutes per subject), while the remainder is occupied by calibration routines. In the current dataset, data for one of the subjects only contains camera data due to an anomalous technical issue with the wearable streaming, although the data can still be useful for computer vision pipelines. The remainder of the dataset statistics focus on the other 9 subjects. In addition, the last 5 subjects performed the experiments without finger-tracking gloves or tactile sensors; these provide examples of unobstructed hands that can be useful for computer vision pipelines, but future experiments will use all sensors.

The dataset provides synchronized labels as ground truth data, spanning 20 unique activities. Of the time spent performing activities, 64.9% of the data has ground-truth labels entered in real time during the experiment. This leaves approximately 19.5 minutes of unlabeled data per subject, or 0.98 minutes per activity per subject; this generally represents the time spent providing instructions between activities.

Table 1 summarizes the number of instances and durations of labeled activities across 9 subjects. Note that some labels can be further segmented into sub-tasks in the future via manual post-processing or auto-labeling pipelines based on subsets of the sensors; for example, the dataset currently labels slicing a cucumber as a single activity, but individual slices could also be labeled either manually or via sensors such as audio.

Figure 1 presents additional information regarding the dataset size and contents. It includes the typical file sizes expected for the wearable and environment-mounted sensors.

A.2 Data Formats and Organization

A.2.1 Wearable Sensors

Data from all wearable sensors is organized hierarchically into a single `HDF5` file [6]. This is a cross-platform file format that can be used with multiple programming languages including Python, Matlab, Java, and C++. Data is organized according to devices and then streams. Each stream contains at least 3 entries: the data, timestamps for each row of the data as seconds since epoch, and timestamps for each row of data as a date-time string. Streams may also choose to include extra entries, such as timestamps generated internally to the sensor that can be used during post-processing.

Table 2 summarizes the data streams and organization for the wearable sensors and labels. It includes the typical sampling rates achieved, the data size where N is the number of timesteps, and the metadata that is embedded within the `HDF5` file. Note that this file format is also easily modifiable, making it amenable to such tasks as fixing or refining labels and adding new sensors.

A.2.2 Video and Audio Data

Five FLIR GS2-GE-20S4C-C cameras are positioned as shown in Figure 2. The data from every camera contains raw sequential images, a list of frame timestamps, and a generated video. Each image is named as, for example, `frameXXXXXX` in which `XXXXXX` is the image index. The data from the Intel RealSense D415 depth camera contains point cloud data and raw color images. The depth data is streamed using the `PointCloud` message type defined in the ROS `sensor_msgs` package [10]. Data is provided using this structure as well as additional file formats and representations.

ActionSense Dataset Facts

Dataset ActionSense

Motivation

Summary A multimodal dataset and recording framework emphasizing wearable sensors and synchronized ground-truth data to record humans performing kitchen tasks
Example Use Cases Analyze human behavior, train learning pipelines, teach robots
Original Authors Joseph DelPreto, Chao Liu, Yiyue Luo, Michael Foshey, Yunzhu Li, Antonio Torralba, Wojciech Matusik, Daniela Rus
Original Funding Gwangju Institute of Science and Technology (GIST)

Metadata

URL	https://action-sense.csail.mit.edu
Keywords	Wearable sensors, multimodal, activities of daily living, kitchen
Format	.hdf5, .csv, .mp4, .wav, .raw
Ethical review	Approved
License	Creative Commons
First released	2022

Wearable Sensors

Body Tracking	Xsens MTw Awinda
Finger Tracking	Manus Prime II Xsens gloves
Eye Tracking	Pupil Core headset
Muscle Activity	Myo armband
Tactile Data	Custom hand sensors

Environment-Mounted Sensors

RGB Cameras	FLIR cameras
RGBD Cameras	RealSense depth camera
Audio	Omnidirectional and directional microphones

Labeled Activities

Peeling	Cucumbers, potatoes
Slicing	Cucumbers, potatoes, bread
Spreading	Almond butter, jelly
Opening/Closing	Screw-top jar
Pouring	From pitcher into glasses
Cleaning	Plates and pans with sponge and towel
Fetching	Tableware, silverware, and food from cabinets, drawers, and refrigerator
Dishwasher	Loading, unloading

Subjects

Target count	Approximately 25
Current count	10 as of July 2022
Gender	70.0% Male
Hand dominance	90.0% Right
Eye dominance	70.0% Right
Age	27.3 ± 3.7 years

Data Size

Total duration	778.0 min
Activity duration total across subjects, average across activities	16.2 ± 13.1 min
Wearable sensors (non-video): average size per subject	4.4 GB
Eye-tracking video: average size per subject	20.7 GB
5 RGB cameras: average size per subject	1 TB (521,100 frames)
RGBD camera: average size per subject	345 GB (63,540 frames)
Microphones: average size per subject	6 GB

Figure 1: A dataset informational card for `ActionSense`, constructed based on [1, 3].

Table 1: Counts and Durations of Labeled Activities

Activity Label	Total Count	Total Duration [min]
Get/replace items from refrigerator/cabinets/drawers	73	62.37
Clear cutting board	84	26.95
Peel a cucumber	27	25.12
Slice a cucumber	27	22.06
Peel a potato	27	26.84
Slice a potato	27	14.21
Slice bread	27	10.65
Spread almond butter on a bread slice	27	10.15
Spread jelly on a bread slice	27	9.24
Open/close a jar of almond butter	27	6.19
Pour water from a pitcher into a glass	45	5.17
Clean a plate with a sponge	27	5.54
Clean a plate with a towel	27	5.57
Clean a pan with a sponge	27	4.85
Clean a pan with a towel	27	4.82
Get items from cabinets: 3 each large/small plates, bowls, mugs, glasses, sets of utensils	10	19.66
Set table: 3 each large/small plates, bowls, mugs, glasses, sets of utensils	11	18.53
Stack on table: 3 each large/small plates, bowls	9	5.98
Load dishwasher: 3 each large/small plates, bowls, mugs, glasses, sets of utensils	9	18.81
Unload dishwasher: 3 each large/small plates, bowls, mugs, glasses, sets of utensils	9	21.54
Total	**574**	**324.26**

Two microphones are positioned as shown in Figure 2. The overhead microphone is omnidirectional, while the sink-level microphone has a cardioid pickup pattern. Audio data is streamed at 48 kHz as chunks of 2,048 samples, with 2 bytes per sample. The dataset includes audio data from each microphone as an uncompressed `WAV` file. Timestamps for each chunk of audio data is included in an `HDF5` file, which also contains metadata about the data format and interpretation.

B Dataset Publishing and Usage

The dataset, code, and instructions will be made available for researchers using long-term hosting and accessible licenses. The hosting is approached in two tiers: an easily updateable MIT-based storage, and archival third-party services.

A front-end website is being created to describe the `ActionSense` project, stream data visualizations, explore or download the data and code, access instructions for using the data or recording framework, and download desired subsets. It is hosted on MIT CSAIL servers in storage space that is intended for long-term persistent websites and maintained by the infrastructure team. The website acts as a portal to point to all relevant visualizations, data, code, and instructions.

Regarding the dataset itself, CSAIL-based storage offers sufficient space to hold all collected data, which can be up to 2 TB per subject including video data, and is thus the main storage repository. This space is maintained by the CSAIL infrastructure team, and is intended for long-term storage and data publishing. It is also easily accessed and updated by the research team, allowing new data to be added as it is collected. This storage thus allows the dataset to be both dynamic and persistent.

In addition, third-party archival services will be explored once the dataset nears its final size. While storage limits may preclude including raw frame image data, they should be sufficient to include all wearable data, split or compressed audio data, and split or compressed video data.

Code will be made available via GitHub [4]. In addition, third-party services for archival code repositories will be explored. The code includes `ReadMe` files describing the code structure, installation, and usage.

The data will be available for use under a Creative Commons license, such as a CC BY-NC-SA 4.0 license [2]. Code will be available under an open-source license, such as the MIT License [9].

Table 2: HDF5 File Organization for Wearable Sensor Data and Ground-Truth Labels

Type	Device > Stream	Data Size	Sampling Rate [Hz]	Notes	Metadata Included
Labels	**experiment-activities**			Start/Stop entries for each activity,	Activity list; target coordinates
	activities	N x 4	Asynchronous	with rating and notes	Data headings; description
	experiment-calibration				Target coordinates; list of poses; list of calibration routines;
	body	N x 6	Asynchronous	Known body poses at known locations	Data headings; description
	tactile_gloves	N x 5	Asynchronous	Pressing on a scale; holding objects in known poses	Data headings; description
	third_party	N x 7	Asynchronous	Ex. Pupil Labs and Manus gloves routines	Data headings; description
	experiment-notes				
	notes	N x 1	Asynchronous	Notes from the experimenter	Description
Eye Tracking	**eye-tracking-gaze**				
	confidence	N x 1	115		Range; description; PupilCapture key
	eye_center_3d	N x 3	115		Units; data headings; description; PupilCapture key
	normal_3d	N x 3	115		Units; data headings; description; PupilCapture key
	point_3d	N x 3	115	Position in world coordinates	Units; data headings; description; PupilCapture key
	position	N x 2	115	Normalized gaze in image space	Units; data headings; description; PupilCapture key
	timestamp	N x 1	115	Pupil Capture internal timestamp	Units; data headings; description; PupilCapture key
	eye-tracking-pupil				
	circle3d_center	N x 3	115		Units; PupilCapture key
	circle3d_normal	N x 3	115		Units; PupilCapture key
	circle3d_radius	N x 1	115		Units; PupilCapture key
	confidence	N x 1	115		Range; description; PupilCapture key
	diameter	N x 1	115		Units; Description; PupilCapture key
	diameter3d	N x 1	115		Units; Description; PupilCapture key
	polar_phi	N x 1	115		Units; Description; PupilCapture key
	polar_theta	N x 1	115		Units; Description; PupilCapture key
	position	N x 2	115		Units; Description; Origin; Data headings; PupilCapture key
	projected_sphere_angle	N x 1	115		Units; Description; PupilCapture key
	projected_sphere_axes	N x 2	115		Units; Description; Origin; Data headings; PupilCapture key
	projected_sphere_center	N x 2	115		Units; Description; Origin; Data headings; PupilCapture key
	sphere_center	N x 3	115		Units; Description; Data headings; PupilCapture key
	sphere_radius	N x 1	115		Units; Description; PupilCapture key
	timestamp	N x 1	115	Pupil Capture internal timestamp	Description; PupilCapture key
	eye-tracking-time				
	pupilCore_time_s	N x 1	145	Pupil Capture internal timestamp	Description
	eye-tracking-video-eye			Timestamps for frames from the eye camera	
	frame_timestamp	N x 1	115		Description
	eye-tracking-video-world			Timestamps for frames from the world camera	
	frame_timestamp	N x 1	30		Description
	eye-tracking-video-worldGaze			Timestamps for frames from the world camera	
	frame_timestamp	N x 1	30	with the gaze estimate overlaid	Description
Forearm EMG, IMU, and gestures	**myo-left**			Muscle and motion information for the forearm	Arm; orientation; device information; sync time
	acceleration_g	N x 3	45		Coordinate frame; data headings; units
	angular_velocity_deg_s	N x 3	45		Coordinate frame; data headings; units
	battery	N x 1	Asynchronous		Units
	emg	N x 8	160		Data headings; range; units
	gesture	N x 1	Asynchronous		Description
	orientation_quaternion	N x 4	45		Coordinate frame; data headings; format; units
	rssi	N x 1	Asynchronous		Units
	synced	N x 1	Asynchronous		Description
	myo-right	[same as for myo-left]			
Tactile force and calibration data	**tactile-calibration-scale**			Scale used during tactile calibration routines	
	accuracy_plusMinus_g	N x 1	5	Estimated accuracy of weight readings	Description
	raw_data	N x 1	5	Raw byte array received	Description
	weight_g	N x 1	5	Weight reading	Description
	tactile-glove-left			ADC readings from custom tactile sensors	
	tactile-data	N x 32 x 32	6 [improveable to ~15]		Description; range; mapping from matrix to hand locations
	tactile-glove-right	[same as for tactile-glove-left]			
Xsens body tracking and Manus finger tracking	**xsens-CoM**			Body center of mass	
	acceleration_cm_ss	N x 3	60		Coordinate frame; data headings; units
	position_cm	N x 3	60		Coordinate frame; data headings; units
	velocity_cm_s	N x 3	60		Coordinate frame; data headings; units
	xsens-ergonomic-joints			6 virtual joints indicating posture	
	rotation_xyz_deg	N x 6 x 3	60		Coordinate frame; data headings; units; joint map; matrix
	rotation_zxy_deg	N x 6 x 3	60		ordering; segment mapping
	xsens-foot-contacts			Heel/toe of each foot	
	foot_contact_points	N x 4	60		Description; data headings
	xsens-joints				
	child	N x 28	60	Body joints	Data headings; format; segment mapping
	parent	N x 28	60	Body joints	Data headings; format; segment mapping
	rotation_xyz_deg	N x 60 x 3	60	Body and finger joints	Coordinate frame; data headings; units; joint map; matrix
	rotation_zxy_deg	N x 60 x 3	60	Body and finger joints	ordering; segment mapping
	xsens-segments				
	acceleration_cm_ss	N x 23 x 3	60	Body segments	Coordinate frame; data headings; units; matrix ordering
	angular_acceleration_deg_	N x 23 x 3	60	Body segments	Coordinate frame; data headings; units; matrix ordering
	angular_velocity_deg_s	N x 23 x 3	60	Body segments	Coordinate frame; data headings; units; matrix ordering
	orientation_euler_deg	N x 63 x 3	60	Body and finger segments	Coordinate frame; data headings; units; matrix ordering
	orientation_quaternion	N x 63 x 4	60	Body and finger segments	Coordinate frame; data headings; normalization; matrix ordering
	position_cm	N x 63 x 3	60	Body and finger segments	Coordinate frame; data headings; units; matrix ordering
	velocity_cm_s	N x 23 x 3	60	Body segments	Coordinate frame; data headings; units; matrix ordering
	xsens-sensors			Raw IMU readings	
	free_acceleration_cm_ss	N x 17 x 3	60		Data headings; matrix ordering; units
	magnetic_field	N x 17 x 3	60		Data headings; matrix ordering; units
	orientation_euler_deg	N x 17 x 3	60		Data headings; matrix ordering; units
	orientation_quaternion	N x 17 x 4	60		Data headings; matrix ordering; normalization
	xsens-time			Map internal Xsens time to original streamed time	
	stream_receive_time_s	N x 1	60	Description	

B.1 Intended Uses and Ethical Considerations

The dataset and code are made available for research purposes. Anticipated use cases include extracting insights about how humans perform common tasks, analyzing how various sensing modalities relate to each other, analyzing how various sensing modalities relate to specific tasks, and training learning pipelines that can help teach robots to assist or autonomously perform tasks of daily living.

While subjects consented to having their video and audio included in a public dataset, no attempts should be made to actively identify the subjects included in the dataset. The data should also not be modified or augmented in a way that further exposes the subjects' identities.

When using the dataset, societal and ethical implications should be carefully considered. These include safety, privacy, bias, and long-term impact on society. If using the data to train robot assistants, immediate safety of any nearby subjects should be carefully considered. In addition, if the new pipelines use similar personally identifiable sensors as `ActionSense`, then the privacy of any new subjects should be preserved as highly as possible and clearly described to the subjects; this includes how the new learning pipelines store and process any video or audio data.

In general, `ActionSense` is intended to be a tool for developing the next generation of wearable sensing and robot assistants for the betterment of society. Endeavors using its data or framework should consider the long-term implications of the application. For example, robot assistants have the potential to improve quality of life and mitigate unsafe working conditions, but they can also result in job displacement that could negatively impact people especially in the short term. How a new robot assistant balances these aspects should be carefully considered before embarking on a novel learning pipeline. In addition, `ActionSense` and subsequent expansions or reproductions may contain biased data along dimensions such as subject backgrounds, experience, demographics, and hand or eye dominance. This could lead to unanticipated consequences for learning pipelines based on the data. Information is provided about the subject pool along with the dataset, and this should be taken into account when scoping a new project based on the provided data.

Such considerations and uses will be presented with the dataset and its license. The authors declare that they bear all responsibility in case of any violation of rights during the collection of the data or other work, and will take appropriate action when needed, e.g., by removing data with such issues.

C Experimental Protocol

C.1 Human Subjects Considerations

The experiments used to collect data carefully considered and mitigated safety and privacy risks for the subjects as much as possible. The protocol was approved by MIT's branch of the Institutional Review Board (IRB), namely the Committee on the Use of Humans as Experimental Subjects (COUHES). Experiments were conducted in a mock kitchen environment within a lab at the Computer Science and Artificial Intelligence Lab (CSAIL) of MIT. Each experiment lasted approximately 2-2.5 hours including donning sensors, calibration, activities, and surveys. Each subject was involved in a consenting session before the experiment in which the experimental goals and risks were described, and a consent form and media release form were signed. If a subject works for one of the study personnel, and the subject is not also on the study personnel list, then the consenting session is run by a third party via the MIT Center for Clinical and Translational Research to help avoid undue pressure to participate in the experiments.

Safety risks include those associated with tasks as well as sensors. Tasks involve sharp objects such as knives or peelers, and breakable objects such as mugs or glasses. In addition, some tasks involve lifting objects overhead into cabinets. These pose risks such as cuts, discomfort, and joint fatigue. To mitigate these risks, sharp tools are covered when not in use, the sensorized gloves protect most of the hands, and the weight of lifted objects is kept low. An additional concern is that common tasks may be more awkward or tiring than expected when wearing the suite of sensors. Subjects are allowed to pause or stop the experiments at any time without penalty.

Due to the public nature of the dataset, privacy is a primary concern for subjects. The goal of publishing a public dataset including personally identifiable video and audio was clearly described to each participant. They were made aware that the videos would not be blurred or otherwise altered from their original state to obscure their face. They were allowed to wear a mask if desired, either for health reasons or to help obscure their identity. A dedicated media release form was signed to

acknowledge their willingness for their video and audio to be published in the dataset, as well as for various other optional uses such as publications and media. In addition to video and audio, wearable sensor data and survey results are also recorded. Wearable sensor data is associated with the video and audio, and thus may be identified as associated with the particular subject. Survey data such as experience levels may also be reported for each specific subject. Survey data such as demographics are only reported in aggregations. In all cases, for all types of data, the subjects' names are kept confidential to the study personnel and not published.

Subjects were recruited by email, posters, and word of mouth primarily from CSAIL and more generally the MIT campus. This may introduce bias into the dataset regarding subject backgrounds and their experience with sensors, robots, and kitchen tasks. This will be noted on the dataset website so that researchers can be aware of the limitations. Survey results such as self-reported experience levels help to characterize these dimensions.

C.2 Donning Sensors

After written consent is obtained, the experimenter helps the subject don and set up each of the wearable sensors. The order is chosen to be as streamlined as possible and to avoid interference between the sensors. In brief, the order is as follows: Xsens IMU sensors on the feet, legs, torso, head, and upper arms, followed by the Myo armbands, the Xsens IMU sensors on the forearms and pelvis, the eye tracking headset, and finally the gloves. More information is provided below.

First, 11 body measurements requested by the Xsens body-tracking system are taken using a flexible tape measure and a laser distance measurer. These comprise the subject's height, shoe length, shoulder height, shoulder width, elbow span, wrist span, arm span, hip height, hip width, knee height, and ankle height.

Next, the Xsens jacket is worn and the IMU sensors are placed. The foot sensors are mounted using a custom elastic strap that wraps around the shoe and has strategic velcro strips to hold the sensor and to accommodate a range of foot sizes. The head sensor is placed within a cloth headband. Two shoulder sensors and a sternum sensor are attached to the jacket using velcro. Sensors are placed on the shins, thighs, upper arms, lower arms, and lower back using velcro straps.

Before placing the forearm sensors at the wrists, the Myo armbands are donned to avoid being obstructed by the Xsens sensors. Each Myo is oriented so that a pre-labeled EMG channel faces upwards when the subject holds their arm at their sides with their forearm horizontal; this ensures that the EMG channels will have a known relative orientation around the forearm that is consistent across subjects.

While wrapping a velcro strap around the waist to hold the Xsens IMU sensor on the lower back, an overhead USB extension cable is draped between wraps so that the same strap holds the cable. This cable is for a USB hub that will connect to the eye tracker and to the tactile sensors. Wireless options are also being developed for the tactile sensors, and a small computer may be worn in the future to make the eye tracking wireless. The overhead USB cable and cables going from the hub to the tactile sensors are coiled stretchable cables to maximize freedom of motion for the subject.

The Manus gloves are donned last, after the eye tracking headset, so that the subject retains maximum dexterity to adjust the sensors before for as long as possible. The Xsens IMU sensors for the hands are fastened to the Manus gloves using provided mounts.

Throughout this process, care is taken to ensure that the subject is comfortable and safe. The tightness of each strap is checked with the subject, and sensors are adjusted for freedom of motion and comfort. The subject is requested to place velcro straps on themselves whenever possible, especially on the legs and waist. The experimenter wears latex gloves while assisting the subject.

C.3 Calibration Procedures and Instructions

`ActionSense` includes calibration data for each sensor to allow the information to be as useful as possible for a range of applications and learning pipelines. These include third-party calibrations defined by sensor-specific software as well as custom calibration procedures. Together, they provide information in both relative and global reference frames. All calibrations are performed after donning the sensors and before starting activities. In addition, selected procedures are performed at the end of the experiment to provide information about stability and drift.

C.3.1 Wearable Sensors

The experimental protocol includes calibration routines for the Xsens body-tracking system, the Manus finger-tracking gloves, the Pupil Labs eye tracker, the Myo EMG armbands, and the custom tactile sensors. These are described below in the order that they are performed, including a summary of instructions provided to the subjects.

Body tracking: The Xsens system is calibrated via the MVN Analyze software. The person first stands in *N pose* with their arms relaxed at their sides, walks around for a few seconds, then once again stands in *N pose*. All cameras and microphones are recording during this procedure, but no wearable sensor data is recorded since the Xsens is not yet streaming data.

After the Xsens calibration is completed, streaming and recording data from all sensors is started so that all subsequent calibration routines are captured. Labels are provided for each calibration period.

Eye tracking: The cameras of the eye-tracking headset are first adjusted to ensure the most appropriate field of views. The subject is asked to pretend to chop vegetables, and the first-person world camera is adjusted to capture the entire cutting board, the forearms, and as much of the table as possible. The eye-facing camera is adjusted to look up at the eye such that the pupil is visible when the subject is looking down at the table or around the kitchen environment. The eye-tracking system is then calibrated via the Pupil Capture software by Pupil Labs. A single target is displayed on an external monitor, and the subject is asked to fixate on the target while moving their head in slow spirals. This procedure is performed at both the beginning and end of the experiment.

Finger-tracking: The Manus gloves are calibrated using the Manus Dashboard procedures. IMUs are first calibrated by moving both hands in a figure-eight pattern and then holding them flat and steady. Next, the software requests a series of hand poses to calibrate each glove: a fist, curling the thumb over the palm while extending the remaining four fingers, and a pistol shape with the thumb and index finger extended. The subject performs these routines in front of the depth camera to obtain additional ground truth data.

Tactile sensors: Custom calibration procedures are performed to provide information about converting tactile readings to physical units and for hand pose training data. First, the subject presses with a flat hand on a Dymo M25 Digital Scale. Weight readings are streamed and recorded from the scale via USB to provide ground truth. The subject is requested to press down three times with each hand. This is then repeated with a 3D-printed textured object placed on the scale, to isolate parts of the tactile sensor over the palm. The subject is then requested to press down twice with each finger sequentially to isolate readings from each finger. After these weight calibrations, the subject is asked to hold five objects with each hand in turn: a mug, a pan, a plate, a knife, and jar. Each one is held for at least 5 s in front of the depth camera. These procedures for calibrating the tactile sensors are performed at both the beginning and end of the experiment. In addition to these calibration routines, a mapping is also provided in the dataset's metadata to indicate which entries of the 32×32 matrix correspond to which location on the hand.

Global positions and Myo poses: Finally, the subject is asked to stand at marked locations in specified poses. This provides information about the global absolute locations. It also provides known poses for the Myo armband that can serve as reference poses for converting its quaternion forearm pose estimates from the built-in arbitrary frame into the world frame. The subject first stands at the origin with their heel on a marked location, facing along the $+x$ direction, with their arms pointing downwards in *N pose*; the origin is marked with white tape towards the lower left of Figure 2, with the long tape indicating the $+x$ direction. The Xsens software is set to move the character to the origin while the subject is in this location. Next, the subject stands at a marked location at coordinates $(100\,\mathrm{cm}, 150\,\mathrm{cm})$ facing along the $-x$ direction, with their arms outstretched in *T pose*; this location is marked with white tape between the table and cabinets in Figure 2. These poses are performed at both the beginning and end of the experiment.

The above calibration procedures are labeled in the dataset similarly to an activity label. They aim to provide researchers with useful information about how to interpret the sensor readings and how to gauge their accuracy.

While most experiments successfully consist of a single continuous recording, occasionally there are technical issues that necessitate stopping and restarting the recording. In particular, the Xsens system

sometimes needs to be recalibrated. In such cases, the two poses described above for calibrating the global location and Myo poses are performed at the beginning of the new recording session. All other procedures are not repeated since the prior calibration data still applies.

C.3.2 Cameras

Every FLIR GS2-GE-20S4C-C camera is calibrated using an 8×6 $60\,\mathrm{cm}$ checkerboard to derive its intrinsic parameters. The Intel RealSense Dynamic Calibration Tool [7] is available for depth camera calibration if necessary. Camera calibration was performed before dataset collection began, and does not need to be repeated for each experiment.

C.4 Activity Ordering and Instructions

After calibration, the main sequence of activities commences. The experimental protocol was design to keep the total duration and effort low for each subject while also spanning a wide variety of tasks. A few considerations used to select the order of activities are described below:

- Generally starting with well-defined short tasks to allow subjects to acclimate to the kitchen and the sensors, then including higher-level planning tasks that require knowledge of the kitchen organization at the end;
- Alternating between tasks at the table and tasks such as fetching or returning items to provide breaks between standing still and moving;
- Performing tasks in logical subgroups, such as peeling then cutting the same vegetable or stacking plates then loading the dishwasher;
- Streamlining which items are needed at which times, such that the kitchen area remains clean and without obstructions;
- Allowing all intermediary tasks, such as cleaning a cutting board or fetching items from the refrigerator, to be labeled as dedicated activities.

In addition to the activity order, the level of detail provided in the instructions was also carefully considered. They were designed to balance consistency and variability to generate informative training data for future learning pipelines. This includes the overall approach to completing the task goal as well as the motion paths or techniques used. It also includes temporal considerations; for example, tasks with multiple iterations such as peeling three cucumbers could be completed in rapid succession or with pauses in between as desired by the subject, and with or without waiting for explicit cues from the experimenter. In all cases, the experimenters observed the subjects and labeled each iteration in real time using the annotation GUI; timestamped notes were submitted if there was a labeling error so it could be fixed in post-processing.

Given the above considerations, the order of activities and their associated instructions are summarized below. Every activity listed, including intermediaries such as fetching or returning items, is labeled in real time by the experimenter.

Fetch items: The subject begins by placing the cutting board on the table, getting three cucumbers from the refrigerator, and getting a chef's knife and peeler. Each group of items is instructed sequentially after the previous group is fetched, and their locations are specified, to avoid confusing or overwhelming the subjects during the initial task.

Peel cucumbers: The subject is instructed to peel the three cucumbers using the peeler at their own pace. They should peel the entire cucumber, but it does not need to be perfectly clean to be considered completely peeled. The approach to the task such as general technique, length of each peeling motion, or grasp poses are at the subject's discretion.

Clear the cutting board: Peelings are scraped from the cutting board into a provided pot. Subjects can choose to use a knife if desired, and can move the pot to the table or carry the board to the pot as desired. This task may be done after each cucumber or after all cucumbers, depending on the subject's preference and how cluttered the cutting board becomes.

Slice cucumbers: Subjects are instructed to slice the three peeled cucumbers using the chef's knife at their own pace and using their preferred technique. Slicing cross-sectionally to create rings is preferred. The subject determines the slice thickness, the cutting technique, and the timing.

Clear the cutting board: The slices are scraped into the aforementioned pot, using similarly flexible instructions regarding timing and technique.

Fetch items: The subject is instructed to fetch three potatoes from the refrigerator, with the target drawer specified.

Peel potatoes: The three potatoes are peeled using the peeler similarly to the cucumbers.

Clear the cutting board: The potato peels are cleared into the pot at the subject's discretion.

Slice potatoes: The three peeled potatoes are sliced using the chef's knife and similar instructions as for the cucumbers.

Clear the cutting board: The potato slices are cleared into the pot at the subject's discretion.

Fetch items: Three sandwich rolls of bread are fetched from the refrigerator, from a bag in a specified location. A bread knife is fetched from a specified drawer.

Slice bread: Each roll is sliced into rings using the bread knife. The timing, technique, and thickness are determined by the subject.

Clear the cutting board: The bread slices are cleared into a specified pan, either after each roll or after all rolls.

Fetch items: Three slices of pre-sliced bread, a jar of almond butter, and a jar of jelly are fetched from the refrigerator. A dinner knife is fetched from a specified drawer.

Spread almond butter: The dinner knife is used to spread almond butter onto three slices of bread. Subjects are instructed to fully cover the slice, but the amount of almond butter used and the technique is at their discretion. For example, some subjects may place the slice on the table while others may hold the slice in their hand.

Spread jelly: Jelly is spread on top of the almond butter on each of the three bread slices using the dinner knife. As with the almond butter, the timing and technique are flexible but the entire slice should be covered.

Clear the cutting board: The prepared slices are cleared onto the pan used previously.

Open and close a jar: The subject is asked to open and close the jar of almond butter three times. This is a relatively well-structured task, but still has variations such as the force used and whether the jar is placed on the table between iterations.

Clear the table: The subject is asked to return the jars to the refrigerator, and to place all remaining items on the table in the sink.

Fetch items: Five glasses are fetched from a specified cabinet, and a pitcher of water is retrieved from a specified location next to the refrigerator.

Pour water: The subject is asked to pour water from the pitcher into all five glasses. They are asked to hold the active glass in a hand rather than leaving it free-standing on the table. The glasses are requested to be relatively full to a comfortable amount. The timing and exact amounts are at the subject's discretion.

Clear the table: The glasses and pitcher are returned to their original location.

Fetch items: A pan, plate, sponge, and towel are retrieved from specified locations.

Clean a plate with a sponge: The subject is asked to pretend that there is some dirt on a plate, and to clean it with a sponge. The motion pattern and applied force are at the subject's discretion. The timing is largely at the subject's discretion, but is gently prompted to be about 5-10 seconds by the experimenter. This task is repeated three times. The subject holds the plate in their hand.

Clean a plate with a towel: The previous task is repeated using similar instructions, except that a towel is used instead of a sponge. Note that this may induce different motion patterns and techniques.

Clean a pan with a sponge: This is similar to cleaning the plate, but a pan is used instead. Note that this may induce different motion patterns, techniques, and tactile forces than when using a plate. The pan is held by its handle.

Clean a pan with a towel: The previous task is repeated, but using a towel instead of a sponge.

Clear the table: The pan, plate, sponge, and towel are returned to their original locations.

Fetch items: To prepare for setting the table, three of each required item are fetched and placed on the table. Three large plates, small plates, and bowls are retrieved from a specified cabinet. Three mugs and glasses are retrieved from a second specified cabinet. Three knives, spoons, and forks are retrieved from a specified drawer.

Set the table: The subject is asked to set three place settings on the table. The desired location of each setting is described. Each setting is specified to have one of each item. The relative placement of each item within a setting, such as whether each utensil is placed on the left or right of the plate, is at the subject's discretion. The overall strategy, such as how to maneuver the items and use staging areas if needed given the cluttered table, is also determined by the subject.

Stack tableware: To prepare for loading the dishwasher, the subject is asked to make stacks of large plates, small plates, and bowls on the table.

Load the dishwasher: The subject is asked to load all items on the table into the dishwasher. This comprises three each of large plates, small plates, bowls, mugs, glasses, knives, forks, and spoons. No other instructions are provided by default, so the subject can use their preferred techniques including how to use staging areas, carrying items one at a time or in groups, and the dishwasher arrangement. If subjects are unfamiliar with loading a dishwasher, then guidance is provided such as placing large plates on the bottom rack and mugs or glasses on the top rack.

Unload the dishwasher: All items are taken out of the dishwasher and returned to their original locations. Guidance can be provided if the subjects do not remember where an item belongs.

These activities create a logical flow of kitchen tasks that aims to keep the subject engaged. The structured set of activities also allows subjects to perform a wide range of tasks involving many tools and objects, while keeping the kitchen clean and not requiring intervention from the experimenter.

C.5 Surveys

After each experiment, the subject is asked to complete a questionnaire. It includes the NASA TLX workload assessment [5] as well as custom questions about how obtrusive each sensor was during the activities. It also asks the subject to rate their level of expertise with various kitchen tasks, and to rate how desirable various roles would be for an autonomous or collaborative robot assistant. The full texts of the surveys are included as Appendix H.

D Infrastructure Overview

D.1 Kitchen Environment and Items

The layout of the kitchen and some important measurements are shown in Figure 2. The kitchen environment is designed to be as similar to a real kitchen as possible. It is equipped with a fridge, a kitchen island, some cabinets, a sink, a dishwasher, a stove, and some other appliances. It also has white tape markings on the floor that indicate the global coordinate system and provide known locations for the subject to stand during the relevant calibration procedures. This environment enables a variety of kitchen-related activities, ranging from object manipulation tasks including cutting and peeling to complex action sequences such as taking food from the fridge and setting the table.

Throughout these activities, subjects interact with a wide variety of items. These are shown in Figure 3. All items are placed in pre-determined locations in the kitchen before each experiment, to streamline the activity sequence and help standardize data across subjects.

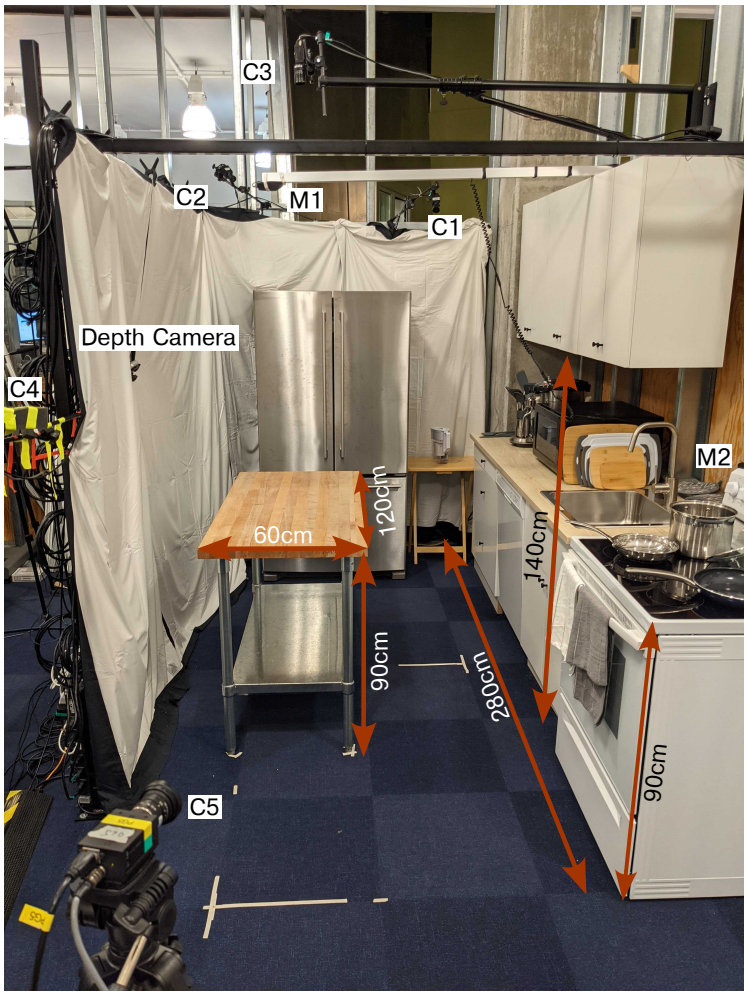


Figure 2: The mock kitchen includes five RGB cameras (C1 through C5), two microphones (M1 and M2), and a depth camera. The white tape on the floor indicates standing locations and orientations for global calibrations. Each carpet square measures 50 cm×50 cm.

D.2 Computing Resources

All wearable sensors and the experiment control GUI were interfaced to a Dell XPS 15 9510 laptop. It has 64 GB of RAM, a 2.5 GHz 8-core processor, an NVIDIA GeForce RTX 3050 Ti Laptop GPU, and integrated Intel UHD Graphics. The laptop concurrently ran the Python code for streaming, recording, visualizing, and labeling, the Xsens software in recording mode, the Pupil Labs software in recording mode, the Manus software, and the Myo software.

The five FLIR GS2-GE-20S4C-C cameras were connected via Ethernet to a desktop computer with 64 GB of RAM, a 3.6 GHz 8-core processor, and an NVIDIA Quadra K1200 GPU. Due to bandwidth limitations, each Ethernet switch can handle two cameras; three switches are thus used, connected to the computer via a three-port Ethernet network interface card. The Intel RealSense camera communicated with a laptop with 16 GB of RAM, a 2.6 GHz 12-core processor, and an NVIDIA GeForce RTX 2060 Laptop GPU.

All computers used to record data have their system times synchronized via Network Time Protocol (NTP) before beginning experiments.

Figure 3: Various items are manipulated by the subjects throughout the selected tasks. These include the (A) pot for clearing vegetable peels and slices, (B) pan for clearing bread, (C) sponge and (D) towel for cleaning plates and pans, (E) peeler, (F) chef's knife, (G) bread knife, (H) cutting board, (I) dinner knife, (J) fork, (K) spoon, (L) large plate, small plate, and bowl, (M) glass, (N) mug, (O) pitcher of water, (P) jar of almond butter, (Q) jar of jelly, (R) sandwich roll, (S) potato, (T) cucumber, and (U) pan to be cleaned.

E Software Overview

Scripts are provided in multiple languages for parsing the data in the dataset and extracting target streams. This aims to facilitate research that leverages the data to develop analysis and learning pipelines. These will be available on the `ActionSense` website alongside the dataset.

In addition, all software for streaming, recording, visualization, labeling, and processing data is provided along with the dataset for researchers to use the data or to recreate the setup. Instructions are included for installing and configuring any dependencies or third-party software, and pre-configured Python environments are available for download. The following sections summarize the framework.

E.1 Wearable Sensor Streaming and Recording

E.1.1 Class Hierarchy and Functionality

The Python code for interfacing with the wearable sensors features a class hierarchy to streamline adding new sensors. It provides functionality for streaming, periodic saving to disk, periodic visualization, and post-processing. Figure 4 summarizes the overall structure of the code.

The code is based around the abstract `SensorStreamer` class, which provides methods for streaming data. Each streamer can have one or more `devices`, and each device can have one or more data `streams`. Data streams can have arbitrary sampling rates, data types, and dimensions. Each subclass specifies the expected streams and their data types. For example, the `MyoStreamer` class may have a left-arm device and a right-arm device connected. Each one has streams for EMG, acceleration, angular velocity, gesture predictions, etc. Its EMG data uses 200 Hz, IMU data uses 50 Hz, and gesture prediction data is asynchronous.

Each `stream` has a few channels that are created automatically: the data itself, a timestamp as seconds since epoch, and the timestamp formatted as a string. Subclasses can also add extra channels if desired. Timestamps for streaming data are automatically logged via the Python `time` module when data arrives. Some devices such as the Xsens also have their own timestamps; these are treated as a separate data stream, timestamped with the Python system time, and can be used in post-processing if desired for refined alignment.

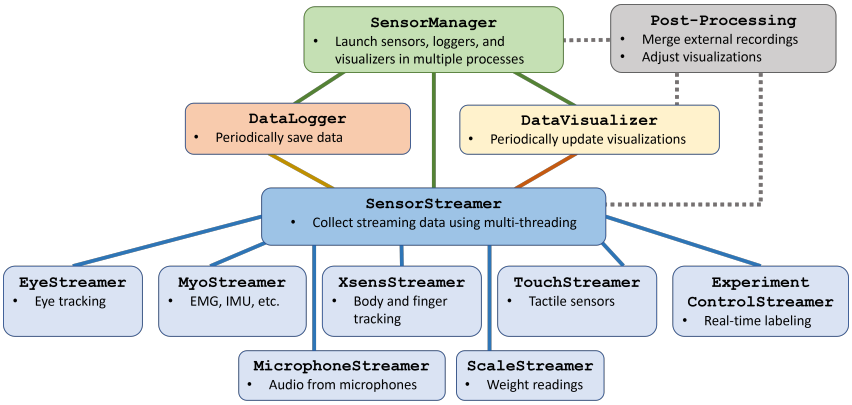


Figure 4: The provided software includes a class hierarchy that enables streaming, saving, visualizing, and post-processing sensor data. It also aims to streamline expansion to additional sensors, and leverages multi-threading and multi-processing to manage resource utilization. All software is made available alongside the dataset to facilitate recreating the recording setup.

The collection of streamers provided by the current software is described below. This can be extended as desired for alternate sensors, and new additions will automatically leverage the streaming and recording infrastructure.

- `MyoStreamer` can connect to one or more Myo devices. Each device will stream EMG, acceleration, angular velocity, quaternion orientation estimates, predicted gestures, battery levels, and RSSI levels.
- `XsensStreamer` streams data from the Xsens body tracking suit as well as the two Manus finger-tracking gloves.
- `EyeStreamer` streams gaze and video data from the Pupil Labs eye tracker. It also overlays the gaze estimate onto the world video, and automatically starts and stops the built-in recording functionality of the Pupil Capture software.
- `TouchStreamer` streams pressure data from one or more of the custom tactile sensors.
- `MicrophoneStreamer`: While the current microphones are not wearable, the same recording framework is used to save their audio data. It streams raw chunked data and timestamps each received chunk.
- `ScaleStreamer` streams weight readings from a Dymo M25 Digital Postal Scale during tactile sensor calibration.
- `ExperimentController` creates a GUI that allows the experimenter label activities, label calibrations, or enter notes at any time. Each note will be timestamped in the same way as any other data, allowing notes to be synchronized with sensor data. The labels and notes are treated as sensor streams. This GUI will be described further below.

A few classes are also provided to manage collections of these streamers. They use multi-threading to coordinate asynchronous operations and non-blocking operations, and they optionally leverage multi-processing to take advantage of multiple cores.

The `SensorManager` class is a helpful wrapper for coordinating multiple streamers and data loggers. It connects and launches all streamers, and creates and configures all desired data loggers. It does so using multiprocessing, so that multiple cores can be leveraged. Streamers can request to be run on the main process if needed, such as for user input applications.

`DataLogger` provides functionality to save data that is streamed from `SensorStreamer` objects. It can write data to `HDF5` and/or `CSV` files. Video data will be excluded from these files, and instead written to video files. Data can be saved incrementally throughout the recording process, and/or at the end of an experiment. Data can optionally be cleared from memory after it is incrementally saved, thus reducing RAM usage. Data from all streamers given to a `DataLogger` object will be

organized into a single hierarchical `HDF5` file that also contains metadata. N-dimensional data will be written to `HDF5` files directly, and will be unwrapped into individual columns for `CSV` files.

`DataVisualizer` periodically updates visualizations of streaming sensor data. Abstract visualization classes are defined, so that streamers can simply specify which type of visualization is best suited to its data. These include scrolling line plots, heat maps, body skeletons, and videos. The `DataVisualizer` can also create a composite visualization, which combines visualizations from multiple streamers into a single window. In all cases, multi-threading allows the visualization overhead to not interfere with the main data streaming.

E.2 Sensor Synchronization and Extensibility

E.2.1 Synchronizing Data Across Sensors

A critical component of streaming from multiple sensors is to ensure that their data can be temporally aligned despite disparate and possibly variable sampling rates. `ActionSense` synchronizes streams by recording a wall-clock timestamp for every sample from every sensor during acquisition. This aims to provide a flexible method of synchronization, such that each sensor interface can operate independently. This facilitates extensibility, and simplifies both online and offline processing. For each sensor, a vector of timestamps is stored in the dataset alongside the vector of data matrices acquired at each timestep. Offline analysis pipelines can then use these time vectors to extract data from each sensor during a desired time window such as a specific activity instance. The time vectors can also be used to interpolate or resample data, which can be helpful for unifying the number of samples from each sensor during an extracted segment.

Two concerns that may arise with this approach are (1) how closely the timestamps are aligned with when the measurements were actually taken, and (2) synchronizing wall clocks for each device if applicable. The approach used by `ActionSense` for each of these aspects is discussed in the following paragraphs.

For every sensor stream, the recording framework acquires a timestamp when Python receives a sample using the system clock via the `time.time()` method of the `time` module. This provides a unified timing strategy across sensors that uses a single clock (the system clock of the computer). It is also applicable regardless of whether a sensor provides an on-board clock, which streamlines extensibility to additional sensor hardware. However, there will be a delay between when the sensor physically measures a reading and when the Python code ingests that reading. For most purposes, this delay is acceptable; for example, it may represent the small delay incurred by a Serial or Bluetooth protocol employed by the sensor. Note that acquisition for each sensor uses a dedicated thread and process, so delays from one sensor do not unduly impact acquisition from other sensors.

Some sensors also provide their own clock and their own strategy for timestamping samples. In such cases, these timestamps are also recorded in the dataset alongside the system timestamps described above. This provides additional information for increasing accuracy and for assessing communication delays. Post-processing scripts can validate their synchronization with the system clock, and replace the system timestamps with the more accurate device timestamps.

For cases with multiple clocks in the recording setup, `ActionSense` aims to synchronize them as closely as possible before recording begins. For example, interfaces for sensors with their own clock include setup code for synchronizing the sensor clock with the system clock if this is supported by the sensor's API. In addition, the current experimental setup uses multiple computers: one for managing all wearable sensors and ground-truth labeling, and one for managing the external cameras and audio. One computer is configured to use the other computer as its time server, and the experimental protocol includes refreshing this synchronization before each experiment. This ensures that the system clocks are aligned as closely as possible.

Synchronization considerations for specific sensors are discussed below:

- *Xsens Body Tracking:* The Xsens system provides per-sample timestamps in its third-party recording format. `ActionSense` includes post-processing scripts for extracting data from these recordings, and uses the device timestamps for enhanced accuracy. The original streamed timestamps are also provided in the dataset if needed. Note that the Xsens system is based on the system clock, so multiple clocks do not need to synchronized in this case.

- *Pupil Labs Eye Tracking:* The Pupil Labs software also has its own clock, and can stream timestamps along with data samples. The API includes methods for setting the time of this clock by sending a desired timestamp via the network sockets. `ActionSense` uses these methods during initialization to synchronize the Pupil Core clock with the system clock, including a routine for estimating the communication delay when sending the new timestamp so it can compensate for this and achieve higher accuracy.
- *Cameras:* The environment-mounted FLIR cameras provide timestamps for every recorded frame. These are the timestamps included in the dataset.
- *Myo Armbands, Tactile Sensors, and Calibration Scale:* These sensors do not provide on-board timestamps for streamed data, so the system timestamp acquired via Python is the one included in the dataset. For most purposes, the delay incurred by the communication protocol should be acceptable. The Myo armbands transmit data via Bluetooth, the tactile sensors transmit via Serial, and the scale communicates via a USB protocol.
- *Microphones:* Audio data is buffered via the `pyaudio` acquisition library and then provided to the main Python code in chunks. A system-clock timestamp is associated with each chunk of samples in the dataset. The current configuration uses an audio sampling rate of 48 kHz and a chunk size of 2,048 samples, so timestamps are recorded at approximately 23.4 Hz. Post-processing can assign each sample a timestamp if desired using interpolation; enforcing a constant sampling rate within each chunk should be a sufficient assumption for more most applications.

E.2.2 Extensibility: Modifying the Sensor Suite

An important design principle of the recording framework is to ensure streamlined extensibility to additional sensor hardware. The class hierarchy allows parent classes to encapsulate functionality such as writing data to disk, logging, visualizing, and organizing data hierarchically across sensors. Each sensor interface also operates independently in its own thread, including the decoupled synchronization scheme described above.

All together, these aspects allow new sensors to be added to the system by merely creating a new subclass of the `SensorStreamer` class. The new subclass provides methods for setting up the sensor connection, acquiring samples at a desired rate, and for associating samples with timestamps from the system clock or an on-board clock. The rest of the code infrastructure then handles spawning threads to run these methods at appropriate times, and manages the acquired data. Adding, removing, or replacing a sensor thus does not need to impact the rest of the pipeline or any other sensor.

The code repository includes a template class for adding a new sensor. This has already been used by other researchers to implement interfaces for insole pressure sensors and a smart watch, thus demonstrating the framework's extensibility; these new classes are included in the repository. In addition, as `ActionSense` continue to grow, the RealSense depth camera may be joined by additional depth cameras or replaced by different hardware such as a Kinect depth camera; such adjustments will not affect the rest of the pipeline, and will only involve creating new interface code to acquire and timestamp frames of data for the new sensors.

E.3 Post-Processing and Analysis

In addition to software for handling real-time sensor streaming, code is provided for post-processing, parsing, and analyzing data from the dataset.

Post-processing code includes scripts for merging data recorded via sensor-specific software with streamed data. In particular, the Xsens and Pupil Capture programs can record data on their own at higher and more consistent rates than they can stream. The Xsens software can also re-process the data after an experiment to improve its pose estimates. The Python recording framework activates these third-party recordings during the experiment, and then merges the data with streamed data so that the dataset contains data at as high of a resolution and rate as possible.

In addition, example scripts are provided to demonstrate how to segment and parse data according to labeled activities. They include parsing the streams of label data, extracting data from multiple

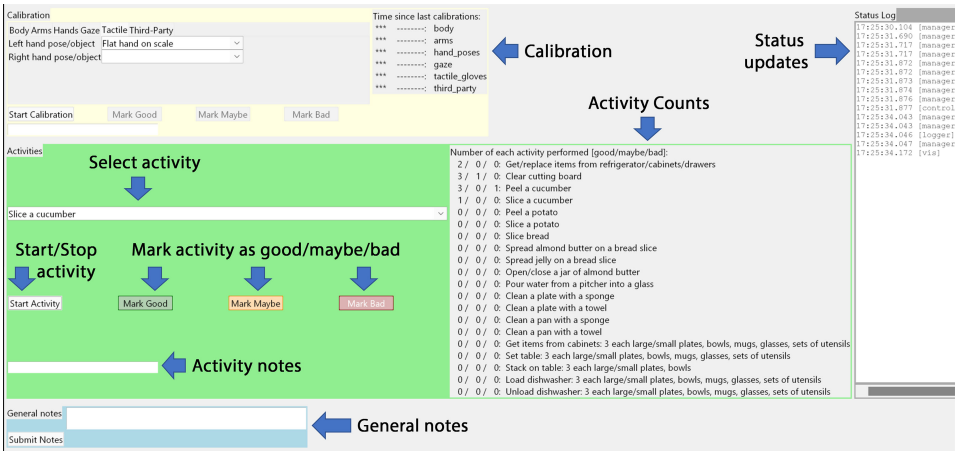


Figure 5: A GUI allows experimenters to label, annotate, and monitor activities in real time. Labels and notes are treated as data streams that are timestamped alongside sensor data.

sensors that correspond to a desired activity, and resampling sensor data if desired to unify the number of samples from each sensor in the extracted segments.

All code used to perform the analyses in Section F and Section G is also included in the provided repository. The activity classification demonstration includes extracting training segments, generating feature vectors from multiple sensor streams, training neural networks, and analysis of results. The cross-modal analysis demonstration includes data extraction, signal processing, and correlational analysis. These examples span multiple programming languages.

E.4 Real-Time Labeling and Status Monitoring

As introduced above and shown in Figure 5, the software framework includes a GUI for investigators to use during experiments. It provides status updates to help monitor the experiment including sensor streaming rates, elapsed time since calibrations were performed, and the number of times each activity has been performed.

Most importantly, it allows the investigator to label and annotate activities in real time. Lists of activities and calibrations are pre-populated, and the experimenter selects the one that will be performed next. After providing instructions to the subject, they watch the subject to determine an appropriate label start time and simply click a button to record the beginning of the activity. They similarly click a button to mark the end of the label once the activity finishes. Afterwards, the experimenter can rate the recorded label as good or bad and enter any notes if desired; this accommodates unexpected events that may have occurred during the activity, and can indicate labels that should be discarded during analysis. The notes can also be used to indicate corrections that should be made offline, such as having selected the wrong activity label or needing to adjust the start or end times. Depending on the subject's preference, they may wait for the experimenter's cue before proceeding with each iteration of an activity or they may proceed at their own pace. In all cases, the experimenter watches the subject and clicks the start and stop buttons appropriately. The experimenter may also request the subject to pause if extra time is needed to select the upcoming activity label or enter notes.

This generates a stream of labeling information that is timestamped and stored alongside the data. In particular, two timestamped entries are created for each label: one denoting its start and one denoting its end. The end timestamp is recorded when the stop button is pressed, and does not include time taken to rate the activity or enter notes. Each one uses the system clock to acquire a timestamp when the button was pressed, thus aligning with the timestamps recorded for the sensor streams. Storing only the start and end times essentially denotes rising and falling edges of labeled activities that can be used to easily segment sensor streams offline regardless of the sensor's sampling rate. They could also be used to create square waves of ground-truth indicators at arbitrary sampling rates if desired.

This real-time labeling approach greatly reduces post-processing effort. If needed though, its minimal storage representation of only two entries per activity allows labels to be adjusted afterwards if needed.

Label texts, start times, and end times can be directly edited in the HDF5 file either interactively or via a script. Example scripts are provided to facilitate this process. The composite videos provided with the data include a bottom banner that displays the system timestamp of each frame and any active activity label; a researcher can thus use these videos to check when activities were being performed and copy the new label timestamps if needed.

E.5 Cameras

All five FLIR GS2-GE-20S4C-C cameras are recorded using a customized ROS package. All data is first stored in the ROS bag file format, and then processed to generate images and timestamp data. The Intel RealSense camera is recorded using an Intel RealSense driver [8]. All data is first stored in the ROS bag file format and then processed to generate depth information. This recording is done on a dedicated computer, but the times of all computers are synchronized via NTP so the vision data is aligned with the wearable data and activity labels.

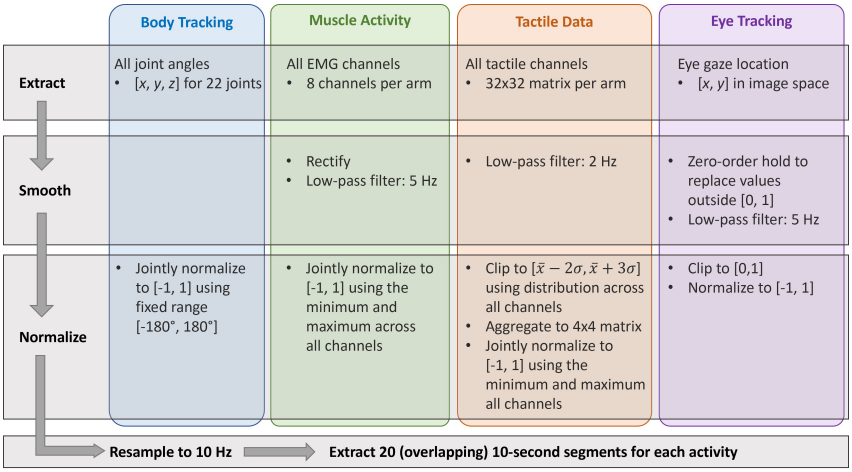


Figure 6: Data streams from 4 wearable sensing modalities are processed and used to generate features for activity classification.

F Application: Activity Classification and Modality Ablation Studies

Since a primary goal of the `ActionSense` dataset is to provide rich descriptions of human behavior, it is important to explore how the inclusion of multiple modalities can enhance extracted insights. As discussed in Section 3.1, envisioned applications of the `ActionSense` dataset include comparing modalities to inform future wearable deployments, cross-modal analyses, and activity segmentation.

To begin exploring these aspects and demonstrate the applicability of `ActionSense` to these use cases, a preliminary machine learning pipeline was developed and trained to classify the 20 labeled activities using a neural network. Ablation studies were then performed to compare results when only subsets of the sensing modalities are used. Results suggest that each modality can provide valuable information about performance, and that using multiple modalities enhances the potential insights and improves classification performance. The following sections discuss the methodology and results in more detail.

F.1 Data Processing and Feature Generation

Data from each of the wearable sensing modalities included in `ActionSense` were processed and used to create feature matrices. These include the EMG muscle activity from the Myo armbands, tactile sensor data, joint angles estimated by the Xsens system, and gaze location estimated by the Pupil Labs eye tracking system. Details about how each modality is processed is included below and summarized in Figure 6. While future pipelines can use more data streams for richer information, the current pipeline focuses on representing each main modality for an initial exploration.

Forearm Muscle Activity: The 8 channels of muscle activity recorded from each forearm are processed to highlight general muscle activation levels. Each channel is rectified by taking the absolute value, and then a low-pass filter with cutoff frequency 5 Hz is applied. All 8 channels from an armband are then jointly normalized and shifted to the range $[-1, 1]$ using the minimum and maximum values across all channels. This preserves relative magnitude comparisons across locations on the forearm. This process results in 8 channels of normalized data from each of the 2 arms.

Tactile Data: Each tactile sensor provides a 32 x 32 matrix of data for each timestep for a total of 1,024 channels from each hand. The values represent raw analog-to-digital-converter (ADC) readings. Each channel is first low-pass filtered with a cutoff frequency of 2 Hz. Since some of the conductive thread intersections may occasionally short together or disconnect, values are clipped to avoid outliers unduly impacting results. The distribution of readings across all channels is considered, and values are clipped to the range $[\bar{x} - 2\sigma, \bar{x} + 3\sigma]$ where x and σ are the mean and standard deviation, respectively, across all channels and all timesteps. Note that across the whole experiment, the mean will likely be close to the baseline value recorded from the sensors when no object is being held; more values are thus included above the mean than below the mean, to capture more of the values when force is applied to the sensors. Each matrix of data is then sub-sampled to a coarser

tactile matrix to reduce spatially smooth the readings and reduce the number of features. A 4 x 4 matrix is created for each timestep, with each entry representing the mean of the 64 original matrix entries that fall within that quadrant. Data from each hand is then normalized to the range $[-1, 1]$ using the minimum and maximum values across all aggregated channels and all timesteps from that sensor. Jointly normalizing all channels preserves relative magnitude comparisons across locations on the hand. This process yields 16 channels of normalized data for each of the 2 hands.

Body Joints: The Xsens system uses the wearable IMUs to estimate 3D joint angles for 22 body joints. The current pipeline considers the x, y, and z angles of each joint. It normalizes each one to the range $[-1, 1]$ using fixed bounds of $[-180°, 180°]$. This process yields 66 channels of normalized data.

Eye Tracking: While the eye-tracking system provides both first-person video and an estimated gaze location, the current pipeline only considers the gaze location. This was chosen to simplify the processing pipeline and to focus on non-video wearable data which may create a fairer comparison to the other modalities. Note that this may introduce inconsistencies across subjects since the head may be held in different positions and thus the same gaze position may focus on different objects, but it still provides information about how gaze changes throughout a task. Towards this end, the current system considers the x and y gaze estimates generated by the Pupil Labs system. These are provided as normalized values corresponding to a normalized image location. They may be outside the range $[0, 1]$ though if the gaze is outside the camera's field of view or if the pupil detection is uncertain. These can lead to erratic jumps in the gaze estimates that do not correspond to physical eye motions. The current pipeline seeks to smooth these artifacts to create more robust gaze trajectories. For each of the x and y channels, it removes any samples that are outside the range $[0, 1]$ and then uses a zero-order hold interpolation to replace them. Finally, it applies a low-pass filter with cutoff frequency 5 Hz to each channel. Values are then clipped to the range $[0, 1]$, and then shifted and scaled to the range $[-1, 1]$. This process yields 2 channels of normalized data.

All together, this produces 116 channels of data with each one normalized to the range $[-1, 1]$. Each one is then resampled to a common time vector that uses a 10 Hz sampling rate, using linear interpolations when needed. This allows the same number of samples to be extracted from each modality within a given time window, which will simplify feature generation and avoid network biases based on the amount of data from each sensor.

All processed channels are concatenated to form a 116-element feature vector for each timestep.

F.2 Segmentation

To prepare a corpus of training examples, segments are extracted from each of the 20 activities described in Section 3.2. Within each experiment, the stream of label timestamps is used to determine time windows that represent each instance of each activity. The pipeline then evenly distributes 20 10-second windows across each activity to use as training example windows. These windows may overlap depending on the total time spent performing the activity. In addition, 20 10-second windows are selected that are distributed across the experiment during times when no labeled task was being performed; these are associated with a baseline label. All together, this process yields 420 labeled time windows per experiment.

The processed data streams from each sensing modality described above are then segmented according to these windows. The feature vectors associated with all timesteps within a window are concatenated to create a feature matrix. Since all data streams were resampled to 10 Hz, this produces a labeled 100 x 116 feature matrix for each example. The current pipeline considers data from 5 subjects, creating a total of 2,100 examples. As `ActionSense` continues to grow, additional subjects can be incorporated to augment the corpus and improve performance.

F.3 Network Architecture and Training

A model and training procedure were designed to perform activity classification. These were implemented in Python 3.9 using version 2.5 of TensorFlow and Keras.

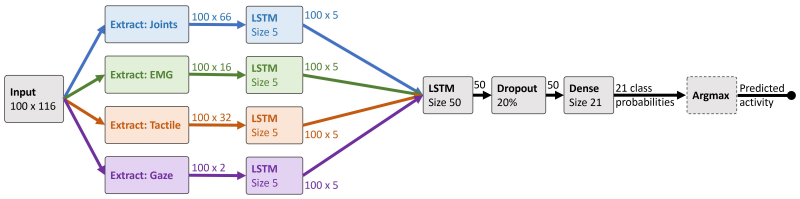


Figure 7: The preliminary network architecture is based on long short-term memory (LSTM) recurrent neural networks. It features parallel pathways for each modality, whose outputs are concatenated to ultimately predict probabilities for the 20 labeled activities and the baseline label.

F.3.1 Neural Network Model

The model is based on long short-term memory (LSTM) recurrent neural networks. Since LSTMs have feedback connections to process sequences of data, they are well-suited to the task of classifying the segments of wearable data sequences. Future pipelines could explore alternative structures, such as convolutional approaches.

The network is summarized in Figure 7. The first portion consists of parallel pathways that each process a single sensor modality. Each one consists of a single LSTM layer that outputs a sequence matrix of size 100 x 5. These outputs are concatenated and passed to an LSTM layer that outputs a vector of size 50. This is followed by a 20% dropout layer, and a dense output layer with softmax activations. The output has 21 classes: the 20 activities and a baseline class representing that no activity is being performed.

This architecture was designed to facilitate ablating various subsets of the modalities by simply removing the corresponding parallel pathway, and to be relatively lightweight while also having sufficient capacity to discover useful characteristics in the time-series feature matrices. Having separate modality pathways that each output the same size also unifies the amount of data that the main LSTM receives from each modality; this aims to help avoid over-reliance on a single modality based on a bias in the number features. These pathways also create embedded networks that can extract insights from each individual modality, which can be useful to probe in the future to investigate the benefit of each modality. The dropout layer aims to reduce overfitting during training. Alternative structures can be explored in the future, but the current pipeline is sufficient to demonstrate applicability of the `ActionSense` data to activity classification and to explore the impact of using multiple modalities.

F.3.2 Training Methodology

A 5-fold leave-one-subject-out cross validation strategy was employed for training and evaluating the model. All examples from a subject are used as the test set, such that the network will be tested on data from an experimental session that did not influence the training at all. Each subject is iteratively treated as the holdout, so 5 different networks are trained. Using each experiment as a test set instead of using randomized k-fold cross validation helps avoid data leakage between training and testing sets, since data within a session is likely correlated along such aspects as user behavior or sensor properties. The selected procedure aims for a more robust evaluation by simulating performance that would be expected on a new subject without network retraining.

Each test set has all 420 examples from the holdout experiment. Examples from the remaining 4 experiments are then split into training and validation sets, with the validation set having the same size as the test set. This corresponds to the training set having 1,260 examples. The random split into training and validation sets is implemented to maintain the original proportions of labels. Each set thus has an equal distribution of examples across the 21 classes.

Each network was trained for 200 epochs using a batch size of 32, the Adam optimizer, and a categorical cross-entropy loss. Accuracy on the training, validation, and test sets was computed after every epoch. The median of the test set accuracies over the last 50 epochs is used in the following sections to represent a network's accuracy; this aims to smooth any spurious jumps in accuracy as the

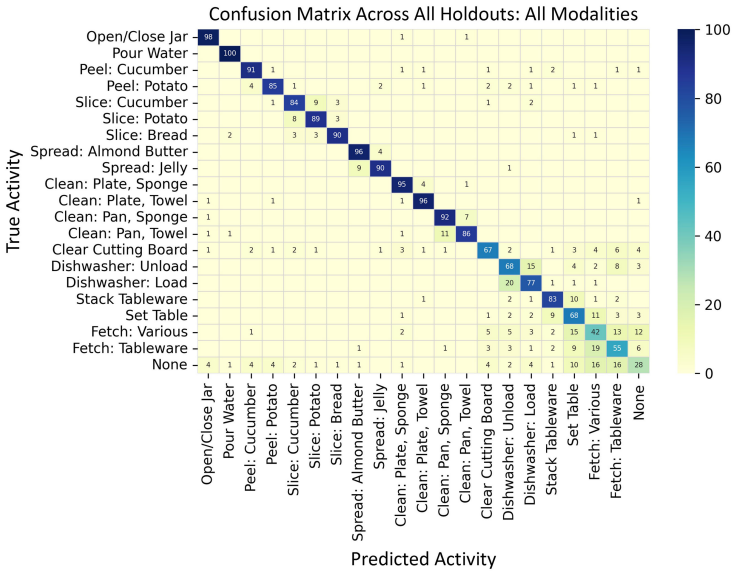


Figure 8: Using features extracted from all 4 sensing modalities, the network successfully classified examples spanning 21 activity classes.

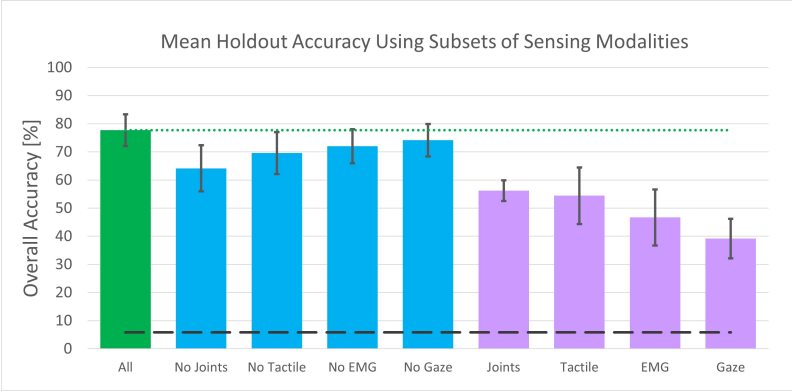


Figure 9: Ablation studies train the network on various subsets of the sensing modalities. The first bar (green) uses all sensors, and the dotted green line extends its mean holdout accuracy for comparison reference. The next 4 bars (blue) only use a single sensing modality, while the last bars (purple) each remove a single modality. Error bars indicate one standard deviation on each side of the mean across holdout experiments. The dashed black line represents the chance level.

network trains, although future investigations can employ early-stopping criteria instead to choose the most promising network.

F.4 Results and Discussion

Using all sensor pathways, the trained network achieved a mean accuracy and standard deviation of $77.7\% \pm 5.6\%$ with a median of 79.9% across the 5 holdout experiments. Figure 8 probes this further by illustrating a confusion matrix among the 20 activities and the baseline class. Since the distribution of examples was balanced across all classes, the chance level for this task is approximately 4.8%. This was verified experimentally by repeating the cross-validation training and evaluation process with randomized labels, where the distribution of labels was kept balanced across the classes; these networks averaged $5.8\% \pm 0.7\%$. All together, the results indicate that the network successfully learned to distinguish activities based on the selected features within 10-second example windows. This demonstrates the applicability of the `ActionSense` dataset to activity detection, and suggests that it can provide useful insights about how people perform each task.

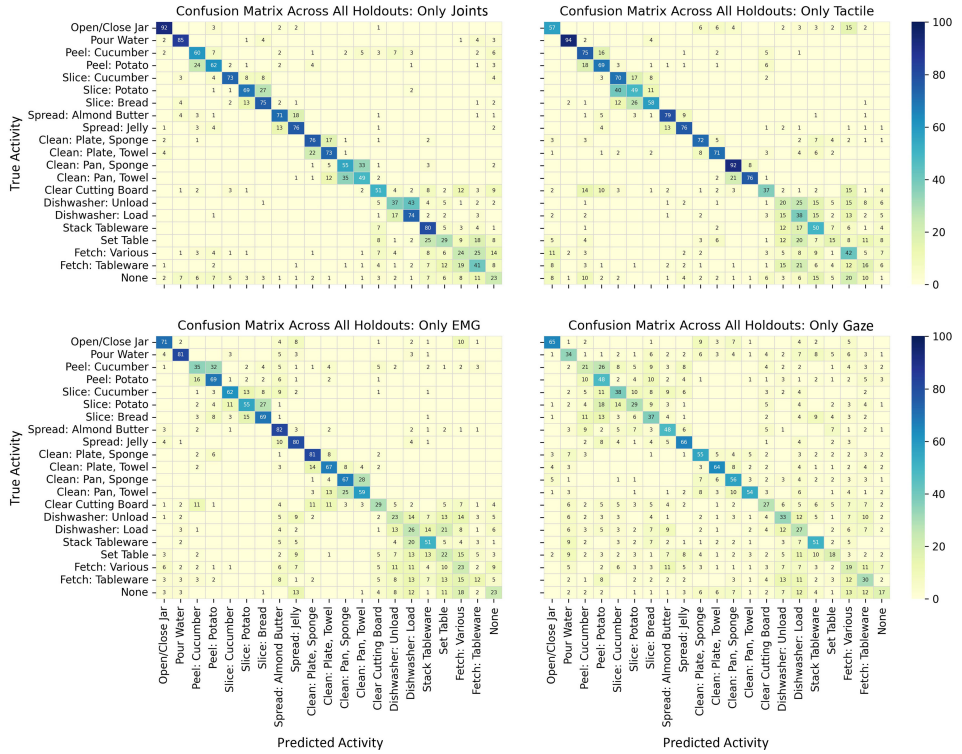


Figure 10: Using only a single modality, the network can still learn to distinguish various activity classes albeit with lower performance than using multiple modalities. Each cell indicates the percent of examples from the holdout set that were classified as the column activity, averaged across all 5 folds of the cross validation. Each row thus summarizes classifications of 100 examples.

To investigate this further, ablation studies were performed that remove sensing modalities or only use a single sensing modality. Figure 9 illustrates the results. Using any single modality achieves classification performance that significantly outperforms the chance level, demonstrating that each of the selected sensors provides valuable information about the task and behavior. Performance could likely be improved even further in the future by using a more sophisticated network architecture.

The results also indicate that using combinations of modalities improves performance, with using all sensors providing the best performance. Each case of using a single modality yields a lower distribution of accuracies across the 5 holdout experiments than using all modalities, with $p < 0.01$ using a one-tailed paired t-test. In addition, using a single modality yields lower accuracies than using 3 modalities, with $p < 0.1$ in each of the 16 pairwise comparisons (and with $p < 0.05$ in all pairs except comparing the joints-only case to the no-joints case). Finally, each case of removing a single modality yields lower accuracies than using all modalities with $p < 0.1$.

For more fine-grained analysis of the results, Figure 10 and Figure 11 present confusion matrices when a single modality is used or when a single modality is removed, respectively. These can be compared to the confusion matrix when using all modalities shown in Figure 8. In general, shorter tasks with more structure are classified more reliably than longer planning tasks; for example, opening/closing a jar, pouring water, peeling, slicing, and cleaning are detected better than setting the table or loading/unloading the dishwasher. This is likely due in part to the amount of variation between subjects, and to how repetitive motions are within the task; increasing the dataset size may help the network learn more varied tasks better. When fewer sensing modalities are used, there is generally increased confusion among tasks that are similar to each other; these include the 2 variations of peeling (cucumbers or potatoes), the 3 variations of slicing (cucumbers, potatoes, or bread), and the 4 variations of cleaning (wiping a pan or a plate with a sponge or towel). There is also generally more confusion about the longer higher-level tasks that involve moving around the kitchen, such as fetching items or using the dishwasher. This indicates that using combinations of modalities can help

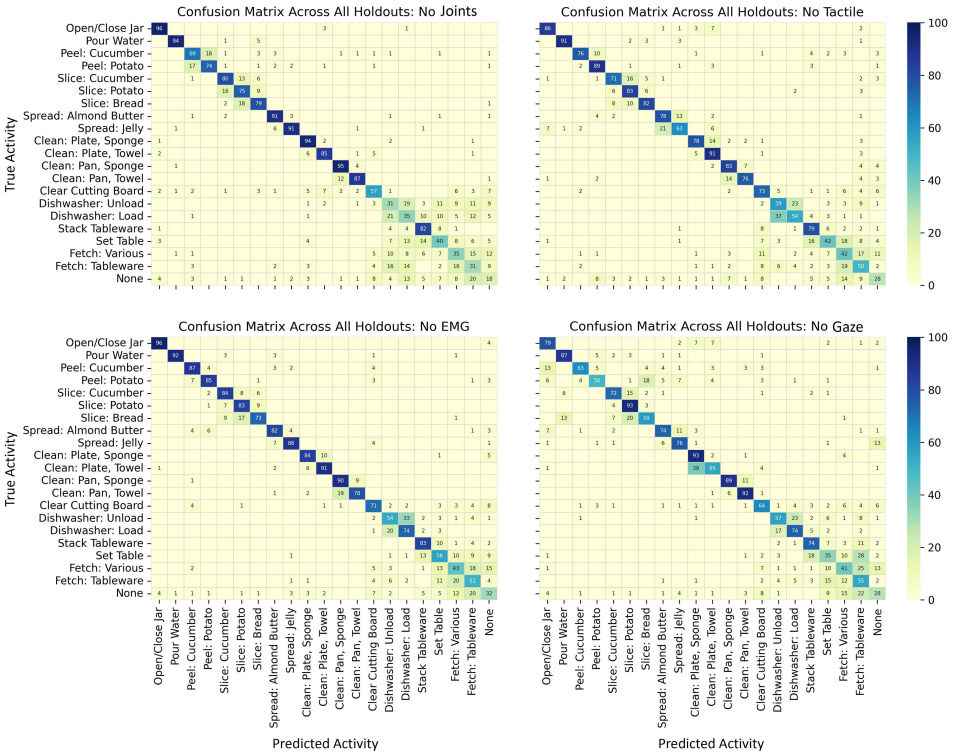


Figure 11: Ablating any single modality generally yields lower performance than using all modalities, but the network can still distinguish activities. Each cell indicates the percent of examples from the holdout set that were classified as the column activity, averaged across all 5 folds of the cross validation. Each row thus summarizes classifications of 100 examples.

to resolve ambiguities between otherwise similar tasks, and can help to extract patterns in tasks that involve sequences of varied behaviors.

Overall, the presented pipeline represents a preliminary machine learning application aimed at demonstrating the applicability of `ActionSense` data. Its initial results lend support to the multimodal approach pursued by `ActionSense`. They suggest that multiple modalities can improve learning pipelines and facilitate behavioral or sensor insights. They also suggest that the selected `ActionSense` sensors in particular can provide complementary information about task performance. Future studies can explore these aspects in more detail, including different feature extraction procedures and different network architectures.

G Application: Cross-Modal Analysis

A valuable anticipated use case for the `ActionSense` dataset is to perform cross-modal analysis between various subsets of the rich sensing suite. This can help extract insights about how sensing streams relate to each other, and about which sensors may be most informative for specific tasks. This in turn could help inform a more streamlined collection of sensors for future activities and guide the development of future smart textiles. In addition, the varied suite of sensors could facilitate training learning pipelines that predict one subset of sensor data from another; for example, predicting hand pose from muscle activity or audio from motion and force data. Finally, cross-modal analysis could yield insights about using subsets of the sensors to automatically label data and create fine-grained action segmentations. All together, such endeavors take a step towards a deeper understanding of human behavior during common tasks, a deeper understanding of wearable sensing, and groundwork for more effective robot assistants.

To demonstrate the feasibility of such analysis using the `ActionSense` data, a preliminary correlational cross-modal investigation was performed between EMG data and tactile sensing data. For certain tasks such as slicing, it may be expected that each downward stroke induces increased force on the hand holding the knife and uses increased forearm muscle activity to stiffen the wrist and grasp the knife. This is thus a reasonable test case for seeing whether muscle activation can provide information about hand pressure and vice versa. Since slicing is composed of many small repetitive motions, it is also a good case for exploring the possibility of automatic labeling.

G.1 Data Processing

The EMG and tactile data streams are first pre-processed to estimate overall activation levels. The absolute value of EMG data across all 8 forearm channels are summed together in each timestep to indicate overall forearm activation; this provides an estimate of wrist stiffness, which is induced by activating the antagonistic muscle pairs, and grasp strength. Similarly, all entries of each 32×32 tactile sensing matrix are summed together to indicate overall force applied to the hand.

The streams are then smoothed to focus on low-frequency signals on time scales comparable to slicing motions. The EMG signal is filtered by a 5^{th} order Butterworth filter with cutoff frequency 0.5 Hz. The tactile stream is smoothed by a moving mean filter with a 1 second trailing window. After these filters, each stream is normalized such that its values are between $[0, 1]$.

While the EMG stream is sampled at approximately 160 Hz, the tactile data is sampled at approximately 6 Hz by the current infrastructure. To facilitate analysis, the EMG stream is resampled to the timestamps recorded by the tactile stream.

Finally, the ground-truth labels and associated timestamps are used to segment the EMG and tactile streams to focus on the desired activities. Each one is cut to start when the first instance of the desired task commences, and end when the last instance of the desired task concludes. It may thus include intermediary activities such as clearing the cutting board if subjects chose to do so in between task iterations, but this is sufficient for the current preliminary exploration.

G.2 Results and Discussion

Figure 12 and Figure 13 illustrate results for slicing cucumbers and potatoes, respectively. Each subplot presents data from a different subject, and thus spans a different duration depending on how long the subject took to perform the task.

It can be seen that the muscle activity and hand pressure readings generally have similar overall trends and peak locations. The correlation coefficient averaged 0.66 ± 0.12 for slicing cucumbers, and 0.61 ± 0.05 for slicing potatoes. While more evaluation will be required to investigate this further, these correlations and qualitative results suggest that information from one modality contains useful information about the other modality, opening the possibility for using a sensor to predict data other than its directly intended stream. As expected though, the correlation is not perfect since the modalities are fundamentally measuring different aspects of the person's activity; this therefore confirms that there is also valuable information to be gleaned by having both modalities present. Depending on the task, this can help inform which modality to select or how best to combine the streams in a learning pipeline.

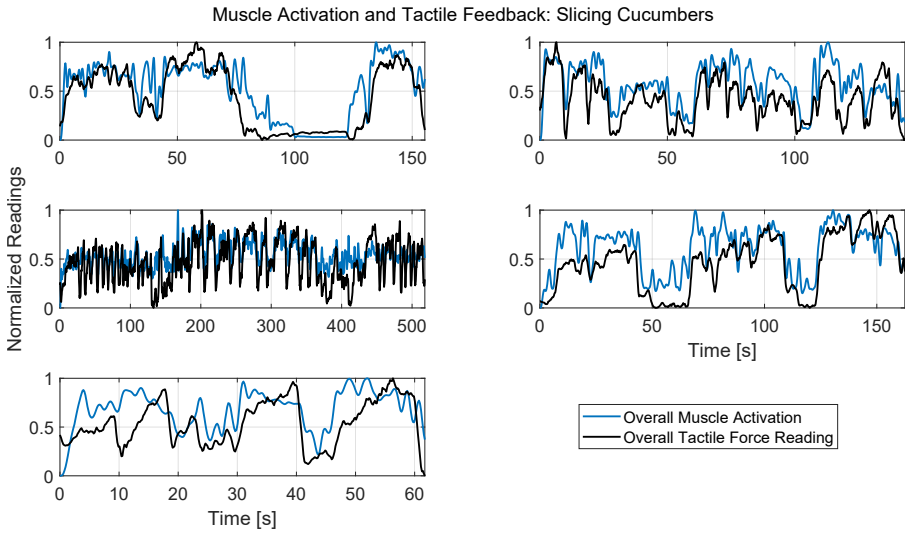


Figure 12: Normalized EMG and tactile data are compared, to demonstrate preliminary cross-modal analysis facilitated by the `ActionSense` data. Each subplot represents data from a different subject. Three cucumbers were completely sliced during the duration of each subplot.

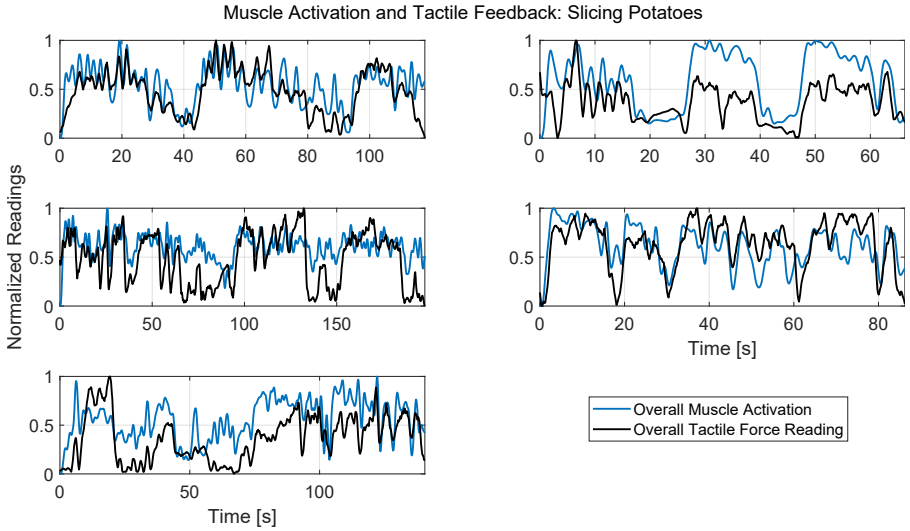


Figure 13: Normalized EMG and tactile data are compared, to demonstrate preliminary cross-modal analysis facilitated by the `ActionSense` data. Each subplot represents data from a different subject. Three potatoes were completely sliced during the duration of each subplot.

In addition to comparing the two modalities to each other, the signals also suggest valuable information about the task. Each subplot spans three iterations of the activity, i.e., slicing three cucumbers or three potatoes. For many of the subjects, these three segments can be clearly identified by inspection of the overall activation trends; for other subjects, these may be clouded by intermediary tasks such as clearing the cutting board. Furthermore, the peaks within each iteration may represent individual slicing actions. It can be seen that the number of slices was variable between subjects, but that there was typically a consistent cadence to the activity. The potato slices are often more salient than cucumber slices, which may be related to the vegetable hardness, the required technique, or the amount of practice the subjects had wearing the sensors since potatoes were always sliced after cucumbers. The preliminary exploration thus suggests that the data may facilitate automatic labeling and fine-grained action segmentation. Future investigations can explore this more rigorously, such as by adjusting the filtering to highlight the desired peaks, by augmenting the analysis with additional data streams such as audio to hear when the knife contacts the table, or by comparing with manually annotated video data.

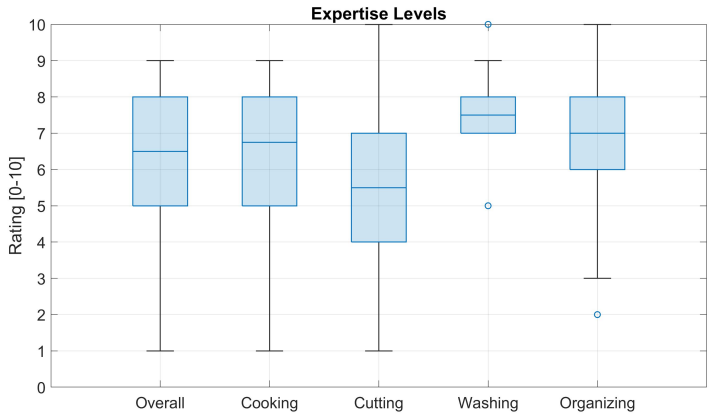


Figure 14: Subjects self-reported their expertise levels for a variety of kitchen tasks related to the activities explored by the dataset. In particular, they assess cooking, cutting/chopping, washing/cleaning plates or countertops, and organizing/tidying. Each rating scale is from 0 to 10, inclusive, with 0 indicating novice and 10 indicating expert.

H Survey Results

Results of the survey using the current dataset start to summarize how sensors impact task performance, characterize the subject pool, and suggest directions for future research.

Subjects: The subject pool recruited so far comprises 10 subjects and is 70% male, 90% right-hand dominant, 70% right-eye dominant, and 27.3 ± 3.7 years old. Figure 14 indicates the self-reported levels of expertise regarding various kitchen tasks. Note that the subject pool continues to grow, and that the dataset aims to reach 25 subjects over the next few months.

Sensors: Figure 15 summarizes how obtrusive the subjects found each sensor. The body tracking was highly variable depending upon the strap tightness, while the eye tracker and the muscle sensor did not generally interfere with the task. The gloves and the tactile sensors could be improved and streamlined in the future to reduce their impact on task performance.

Workload: Across all subjects, the raw workload estimate using the NASA TLX averaged 3.40 ± 1.62 with lower numbers indicating lower workload on a scale from 0 to 10, inclusive. This indicates that the workload was relatively low despite the mild sensor obtrusiveness.

Robot roles: Figure 16 indicates that there is significant variability among desired roles for robot assistants. This spans both autonomous and collaborative cases, although there are a few tasks such as loading or unloading the dishwasher where an autonomous assistant would be greatly valued over a collaborative robot. Some considerations provided by the subjects include safety especially regarding a robot wielding sharp objects, how much time the task would take to do manually, and a desire for the robot to anticipate what task or tool will be needed next.

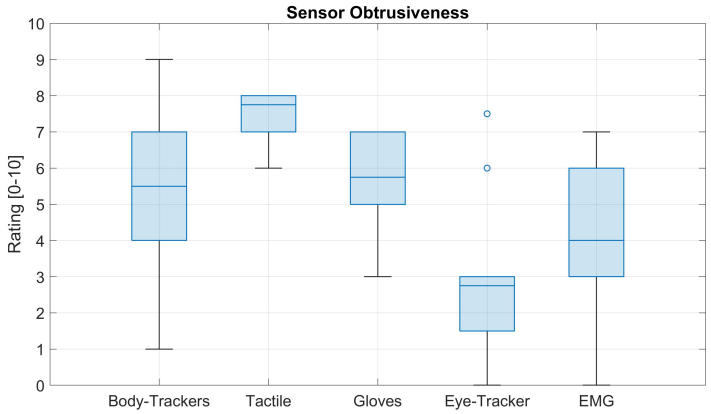


Figure 15: The survey included ratings of how obtrusive the subjects found each sensor to be during the activities. The included sensors are the Xsens body trackers, the tactile sensors comprising the resistive material and the wires, the gloves alone, the eye-tracking headset, and the Myo armbands. Each rating scale is from 0 to 10, inclusive, with 0 indicating the sensor was not noticed and 10 indicating it was very obtrusive.

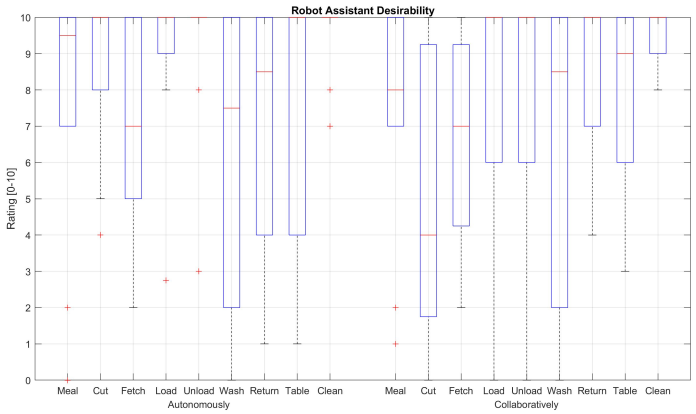
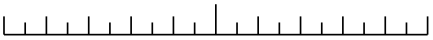


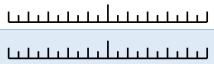
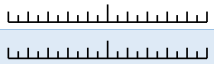
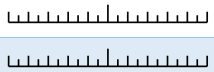
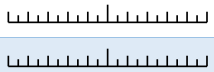
Figure 16: Subjects rated how desirable various robot assistant roles would be, including whether the robot would perform the role autonomously or collaboratively. The specified roles are preparing an entire meal, cutting vegetables/fruit or bread, fetching ingredients, loading the dishwasher, unloading the dishwasher, manually washing/drying plates and whatnot, putting away plates or utensils and whatnot, setting/clearing the table, and general cleaning. Each rating scale is from 0 to 10, inclusive, with 0 indicating no desirability and 10 indicating a definite desire.

I Full Text of Post-Experiment Surveys

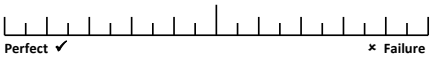
*For each sensor, please **circle the vertical line** that best describes your experience performing tasks while wearing it.*

	Didn't notice it | Neutral | Very obtrusive	Reason or general comments
Body trackers		
Tactile sensors on hands (the black pads and associated wires)		
Gloves alone		
Eye-tracking headset		
Muscle-sensing armbands		
Any other general comments?		

*How likely would you be to want a robot assistant for the following tasks? Please **circle the most appropriate vertical lines**.*

	The robot does it on its own		The robot assists you		Reason or general comments
	Not at all |	Definitely |	Not at all |	Definitely |	
Prepare an entire meal					
Cut (vegetables/fruit, bread, etc.)					
Fetch ingredients					
Load the dishwasher					
Unload the dishwasher					
Manually wash/dry plates etc.					
Put away plates, utensils, etc.					
Set/clear the table					
General cleaning					
Any other tasks that you would want a robot to do on its own or while working with you?					

Considering the tasks as a whole while wearing the sensors, please circle the vertical line that best matches your experience.

Mental Demand: How mentally demanding was the task?		
Physical Demand: How physically demanding was the task?		
Temporal Demand: How hurried or rushed was the pace of the task?		
Performance: How successful were you in accomplishing what you were asked to do?		
Effort: How hard did you have to work to accomplish your level of performance?		
Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you?		
Any other general comments?		

For each pair, circle the Scale Title that represents the more important contributor to workload for the specific task(s) you performed in this experiment.

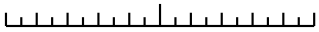
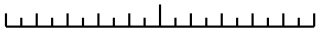
Effort or Performance	Temporal Demand or Frustration	Temporal Demand or Effort
Physical Demand or Frustration	Performance or Frustration	Physical Demand or Temporal Demand
Physical Demand or Performance	Temporal Demand or Mental Demand	Frustration or Effort
Performance or Mental Demand	Performance or Temporal Demand	Mental Demand or Effort
Mental Demand or Physical Demand	Effort or Physical Demand	Frustration or Mental Demand

Any additional feedback, comments, or suggestions:

To be completed by the research team: Subject ID: _____ Session ID: _____ Date: _____

Study Participant Information

*How would you characterize your kitchen expertise? Please **circle the most appropriate vertical lines.***

	Novice | Expert |	Comments
Overall		
Cooking		
Cutting/chopping		
Washing / cleaning plates, countertops, etc.		
Organizing / tidying		

Handedness: ☐ Right ☐ Left

Eye dominance: ☐ Right ☐ Left

Gender: ☐ Female ☐ Male

Age: _____

What is your race?
☐ American Indian/Alaska Native
☐ Asian
☐ Native Hawaiian or Other Pacific Islander
☐ Black or African American
☐ White
☐ More Than One Race
☐ Do not wish to report

Are you Hispanic or Latino? ☐ Yes ☐ No ☐ Do not wish to report

To be completed by the research team: Subject ID: _____ Session ID: _____ Date: _____

References

[1] Jack Bandy and Nicholas Vincent. Nutrition label template for dataset documentation, 2021. URL https://www.overleaf.com/latex/templates/nutrition-label-template-for-dataset-documentation/gxzpbfmncyfp.

[2] Creative Commons. Attribution Non-Commercial Share-Alike license, 2022. URL https://creativecommons.org/licenses/by-nc-sa/4.0.

[3] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna M. Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *CoRR*, abs/1803.09010, 2018. URL http://arxiv.org/abs/1803.09010.

[4] GitHub. GitHub, 2022. URL https://github.com.

[5] Sandra G Hart and Lowell E Staveland. Development of NASA-TLX (task load index): Results of empirical and theoretical research. In *Advances in Psychology*, volume 52, pages 139–183. Elsevier, 1988. doi: 10.1016/S0166-4115(08)62386-9. URL https://doi.org/10.1016/S0166-4115(08)62386-9.

[6] HDF Group. The HDF5 library and file format, 2022. URL https://www.hdfgroup.org/solutions/hdf5.

[7] Intel RealSense. Intel RealSense D400 series dynamic calibration tool, 2021. URL https://www.intel.com/content/www/us/en/download/645988/intel-realsense-d400-series-dynamic-calibration-tool.html.

[8] Intel RealSense. ROS wrapper for Intel RealSense devices, 2022. URL https://github.com/IntelRealSense/realsense-ros.

[9] Open Source Initiative. The MIT License, 2022. URL https://opensource.org/licenses/MIT.

[10] ROS. Documentation for sensor_msgs, 2022. URL https://wiki.ros.org/sensor_msgs.

NeurIPS Conference Paper Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] They describe the dataset which is actively growing, the multimodal recording framework, and the envisioned use cases.
   (b) Did you describe the limitations of your work? [Yes] In particular, note that Section 4 and Section 5 describe the limitations of each modality as it is included in the presented framework. Section 7 and others also discuss future extensions that can leverage the data or improve the recording framework.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Relevant considerations are introduced in Section 3.5. They are discussed further in the supplementary materials, particularly in the section on intended uses that is recommended to be in the supplementary document. They will also be discussed in the dataset metadata such that it is visible to anyone downloading the data.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplementary material or as a URL)? [Yes] The URL to the dataset, code, and instructions is provided to access data and reproduce the setup.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the supplementary materials section on the preliminary classification pipeline.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Preliminary classification results in the supplementary materials include error bars based on cross validation.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The supplementary materials include information about the computers currently used for recording, processing, and preliminary analyses.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [Yes] Section 6 and supplementary materials.
   (c) Did you include any new assets either in the supplementary material or as a URL? [Yes] The URL to the dataset is provided in the abstract and Section 6.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Section 3 summarizes the human subjects considerations and approvals. The supplementary materials include more details.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Section 3 briefly describes privacy concerns related to publishing audio and video. Supplementary materials contain more details.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] See the supplementary materials.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [Yes] Section 3 summarizes the human subjects considerations and approvals. The supplementary materials include more details.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes] Section 3 summarizes the compensation and duration. The supplementary materials include more experimental details.