APPENDIX

A.1 DATASET EXAMPLES

The supplementary video provides examples of our dataset and also qualitative results of our representation learning model.

A.2 DATA COLLECTION DETAILS

We describe the details of our hardware setup and the alignment method used to synchronize the recordings between our camera, gaze tracker, and movement sensors.

A.2.1 ALIGNMENT AND SYNCHRONIZATION OF DEVICES

There are three different devices in our setup, 1) Tobii Pro eye-tracking glasses to record gaze, 2) BNO055 IMU sensors to record movements, and 3) GoPro camera attached to the forehead to capture ego-centric videos. These different devices record data independently. Therefore, it is necessary to synchronize all recordings.

Gaze Tracker and GoPro Alignment. Tobii Pro2 eye-tracking captures a video and the center of the gaze in the camera frame. Due to the low quality of the video captured by the eye-tracking glasses, we use an additional high-quality GoPro hero 6 camera (with resolution 1920×1080 and 60 frames per second). To synchronize the videos from the gaze tracker and GoPro, we extract SIFT (Lowe, 2004) features and use a brute force algorithm for feature matching and the RANSAC method to find a homography that maps the gaze from Tobii's camera frame to GoPro's camera coordinate. Note that the gaze might be missing for some frames due to the device noise.

IMU Sensors and GoPro Synchronization. The outputs of the IMU sensors are recorded on a Raspberry Pi board. There is also a microphone on the Raspberry Pi that records the audio. We synchronize the IMU and video recordings using two methods, 1) synchronize the audio from the GoPro video and the voice recording on the Raspberry Pi board, and 2) repeat a specific pattern of body movements in front of a mirror, so it can be uniquely identified in both the movements depicted by the IMU sensors and the GoPro camera (which recorded the participant's body pose in the mirror).

A.2.2 MOVEMENT CALCULATIONS

We record the body part movements using BNO055 Inertial Measurement Units (IMUs) in 10 different locations (torso, neck, 2 triceps, 2 forearms, 2 thighs, and 2 legs). The body parts may not appear in ego-centric video frames, therefore, the task of predicting the exact orientation and location of a body part (e.g., arm) using ego-centric videos can be very challenging. We train the model using the simpler task of predicting whether a part has moved or not. This still contains rich information about the action that is happening in the video, for example, walking can be defined as periodic movements of the left and right legs.

To compute the loss function, we need to distinguish between *movement* and *no movement* in the dataset. One way is finding a threshold in the domain of the angles of the part movements, and label all the moves smaller than the threshold as *no movement* and the rest as *movement*. However, this might result in ambiguities for the movements close to the threshold, and the network might over penalize the wrong predictions in the neighborhood close to the threshold. Therefore, we add a third *gray area* label, where the network is not penalized for wrong predictions. We divide the range of movements for each sensor into three equal ranges, the first 33% is labeled as *no movement*, the last 33% is labeled as *movement* and the remaining interval is the *gray area*, for which there is no penalty for misprediction.

A.3 DATASET ANALYSIS

To ensure that the videos in the dataset consist of a wide range of activities, we do not provide any specific instructions to the subjects, and we ask them to perform their daily routine activities. Hence, the dataset includes a variety of different situations including but not limited to driving,

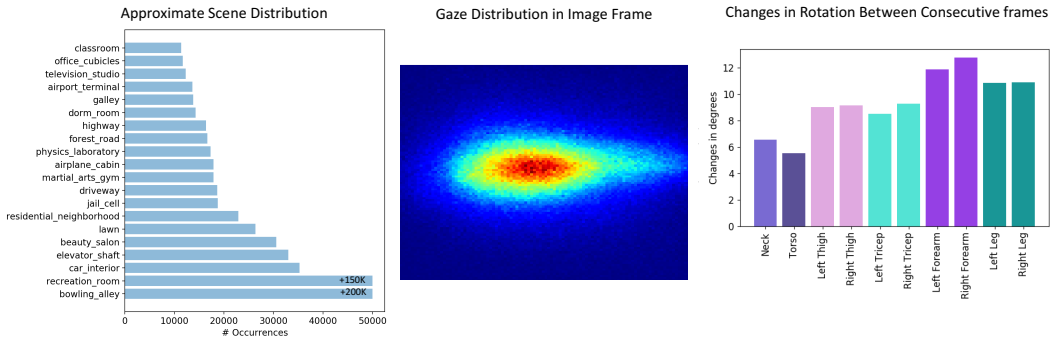

Figure 4: **Dataset Statistics.** Left: The approximate distribution of top 20 scene classes according to a scene classifier trained on the Places (Zhou et al., 2017) dataset. We show how often each scene category is predicted as top-1. Middle: The distribution of gaze across the dataset. Right: The average magnitude of the change in the orientation of body parts between consecutive frames.

cycling, playing pool, cooking, cleaning, walking in the streets, and shoveling snow. Purely for dataset analysis purposes, we gather proxy scene labels for each image. We use an off-the-shelf scene classification model trained on the Places dataset (Zhou et al., 2017) and record the top-1 prediction for each frame of our dataset. Many scene categories in our data are not present in Places, so we see a moderate amount of misclassification. However, the classifier confidently (more than 70%) predicts 101 of the 365 classes exist somewhere within our dataset, showing the diversity of our data. Figure 4 Left shows the 20 most frequent predictions. Even though there are some mispredictions among them (such as jail cell which is frequently mistaken for dark rooms), we observe that our data is fairly diverse.

Figure 4 Middle shows the distribution of the gaze in the images. As expected, the focus is mostly in the center of the image. In Figure 4 Right, we show the change in the orientation of the body parts between two consecutive frames. We observe more movements in the limbs compared to the torso and neck. We additionally notice more right arm movement than the left which is likely caused by more of our participants being right-handed.

A.4 Architecture & Training Details

In this section, we describe the details of our network architectures as well as the hyperparameters and optimization methods that we used, for reproducibility purposes. Also, the code and data will be made publicly available for further research.

A.4.1 Backbone Network

We train the feature extractor network by using a sequence of images of length $k = 5$, which are $\frac{1}{6}th$ of a second apart, as input. We use the ResNet18 (He et al., 2016) convolution layers as the feature extraction backbone. To preserve spatial information, which is essential for gaze prediction, we use the $512 \times 7 \times 7$ features before average pooling. We then add a 1×1 convolutional layer on top to reduce the feature size to $64 \times 7 \times 7$. The flattened feature is then input to a 3 layer LSTM with hidden size 512, which encodes the input video into a hidden feature vector. Next, the embedded video feature vector is decoded using a 3 layer LSTM, to predict the binary movement vector and gaze. For $\mathcal{L}_{visual}$, we use the feature size 128 (obtained by a fully connected layer on top of the ResNet18 features) and a memory bank of size 16384. We choose $\delta = 1$ in Equation 1, $\alpha = 0.09, \beta = 0.01, \gamma = 0.9$ in Equation 3, and $\tau = 0.07$ in Equation 2. We use a dropout of 0.5 and weight decay of 0.1. For data augmentation, we only use color jitter and random flip. We flip the entire sequence of images, swap the part movements for right and left arms and legs, and calculate the updated gaze in the flipped images.

A.4.2 Target Task Networks

Shared Implementation Details. During the target task training, the weights for the backbone are frozen. For all of our experiments, we use the Adam optimizer (Kingma & Ba, 2015) and images

are reshaped to 224×224. The size of the hidden layer in our LSTM in all temporal experiments is 512. We use leaky-ReLU non-linearities between all network layers except for LSTMs.

Self-supervised Baseline Details. When training the MoCo encoder network, we use the SGD optimizer with 0.03 learning rate for the MoCo baseline since it performs best in this setting. For data augmentation, we use random cropping, horizontal flipping, gray scaling, and color jittering with the same parameters used in that work. We train the baseline with batch size 256 on 8 GPUs for 200 epochs with the same training regime as the original work.

Scene Classification. We use a decoder network of a single 1×1 convolution layer that reduces the feature size from $512 \times 7 \times 7$ to $64 \times 7 \times 7$, followed by two fully connected layers, that convert these features to a vector of size 512 and then 397. We use the cross-entropy loss for training, and evaluate using mean per class top-1 accuracy.

Action Recognition. For this task, we train a single 1×1 convolution layer that reduces the feature size from $512 \times 7 \times 7$ to $64 \times 7 \times 7$, followed by an LSTM to embed the video in one hidden vector of size 512, and two fully connected layers that convert the features to a vector of size 200 and then to the number of actions. As before, we use the cross-entropy loss as the objective and evaluate with mean per class top-1 accuracy. Since some of the action classes in this dataset appear in a limited set of videos, following one of the EPIC challenge finalists (Damen et al., 2019), we choose 9 verbs that result in a state transition, namely, *take*, *put*, *open*, *close*, *wash*, *cut*, *mix*, *pour* and *peel* and ignore verbs that do not cause a state transition (e.g., check).

Dynamic Prediction. We create a binary rectangular mask using the object bounding box. We use this mask image as the input to a two-layer convolutional network. As the result, we obtain a feature of size $64 \times 7 \times 7$. These two convolutional layers are trained for both our method and the baseline. We concatenate the mask feature with the feature vector obtained from the image and add two fully convolutional layers on top, to obtain the class labels. Again, we optimize the network using the cross-entropy loss and use mean per class top-1 accuracy as the evaluation metric.

Depth Estimation. The ResNet backbone is connected to a Feature Pyramid Network (FPN) (Lin et al., 2017). We use Pixel Shuffle layers (Shi et al., 2016) for up-scaling the lower level features. The learned ResNet backbone is frozen; the 5 up-convolution layers are the only layers trained for the target task. We use the Huber loss as the objective.

Walkable Surface Estimation. The architecture of this network is the same as the depth estimation network. The ResNet backbone is frozen and five up-convolution layers are the only layers that are trained for the target task. We use the binary cross-entropy loss as the objective, where the goal is segmenting walkable and non-walkable pixels. For evaluation, we use the standard Intersection over Union (IOU) metric for segmentation tasks.

A.5 Result of Full Supervision

As a point of reference, we also provide the results using a fully supervised backbone that is trained using ImageNet. Neither our method nor the purely visual baselines use any supervision for representation learning. Therefore, a direct comparison is not fair. The results are shown in Table 5. The corresponding self-supervised results are shown in Table 1.

Datasets		SUN397 Xiao et al. (2010)	Epic Kitchen Damen et al. (2018)	VIND Mottaghi et al. (2016a)	NYUv2 Nathan Silberman & Fergus (2012)	
Method	Training Objective	(a) Scene (Top-1 ↑)	(b) Action (Top-1 ↑)	(c) Dynamics (Top-1 ↑)	(d) Walkable (IOU ↑)	(e) Depth (RMSElog ↓)
MoCo (He et al., 2020)	InfoNCE	35.18	30.28	16.37	63.95	0.135
Supervised	Classification	47.27	32.09	20.01	65.80	0.132

Table 5: **Results of pre-training with ImageNet.** We provide the results of full supervision for pre-training using ImageNet and also the MoCo model trained on ImageNet data.