Multimodal Movement Activity Recognition Using a Robot's Proprioceptive Sensors

Robin Schmucker[1(✉)], Chenghui Zhou[2], and Manuela Veloso[2]

[1] Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
robin.schmucker@online.de
[2] Carnegie Mellon University, Pittsburgh, PA 15213, USA
{chenghuz,mmv}@cs.cmu.edu

Abstract. By recognizing patterns in streams of sensor readings, a robot can gain insight into the activities that are performed by its physical body. Research in Human Activity Recognition (HAR) has been thriving in recent years mainly because of the widespread use of wearable sensors such as smartphones and activity trackers. By introducing HAR approaches to the robotics domain, this work aims at creating agents that are capable of detecting their own body's activities. An activity recognition pipeline is proposed that allows a robot to classify its actions by analyzing heterogeneous, asynchronous data streams provided by its inbuilt sensors. The approach is evaluated in two experiments featuring the service robot Pepper. In the first experiment, a set of base movements is recognized by analyzing data from various proprioceptive sensors. The findings indicate that a multimodal activity recognition approach can achieve more accurate classifications than single-sensor approaches. In the second experiment, a person interferes with the forward movement of the robot by pulling its base backward. This happens in a way that is not detected by Pepper's inbuilt systems. The approach can detect the unexpected behavior and could be used to extend Pepper's inbuilt capabilities. Through its generality, this work can be used to recognize activities of other robots with comparable sensing capabilities.

Keywords: Learning from sensory data · Activity recognition · Behavior verification

1 Introduction

While the planning layer captures a robot's intended activity execution, it makes no statement about its actual state and activity. Assume a robot wants to move a certain distance forward. During its movement, it might collide with an obstacle and fall over. A robot that recognizes the unexpected behavior can try to recover or call a human operator for help. In another scenario, a robot might be pushed by a human. If the robot recognizes what is happening to its body, it can respond

D. Holz et al. (Eds.): RoboCup 2018, LNAI 11374, pp. 299–310, 2019.
https://doi.org/10.1007/978-3-030-27544-0_25

with a warning when it is being moved in a way that is overly demanding on its mechanics. In case of remote control, the robot might even reject a user command to prevent damage.

Sensor-based Human Activity Recognition (HAR) uses wearable sensors such as accelerometers and gyroscopes to capture human activity and finds application in areas including mobile computing [17], ambient-assisted living [5] and health care [2]. HAR can detect activities of the human activities such as walking, running, riding escalator, eating, opening door, and lifting object [11] by detecting patterns in streams of sensor data. While sensor-based HAR needs to attach and calibrate sensors for each individual user, robots feature a variety of inbuilt sensors that give insight into their physical states.

By combining HAR approaches with a robot's rich sensor data, this work aims at creating an agent that recognizes its own body's activity. A recognition pipeline is proposed that enables the robot to detect its own activities by analyzing asynchronous, heterogeneous streams of sensor data. A Long Short Term Memory (LSTM) [8] based neural network is used for activity recognition. The approach is evaluated in two experiments featuring the service robot Pepper. In the first experiment, a set of 7 movement activities is recognized. Here, the robot detects if it moves forward, backward, left or right, rotates clockwise or counterclockwise or stands still (see Fig. 3). The pipeline combines information from heterogeneous, propriocepetive sensors to achieve accurate classifications. The multimodal sensor data comprises joint states, electrical current, orientation, angular velocity and acceleration data. In the second experiment a human interferes with the forward movement of the robot by pulling its base backward in a way that is not detected by Pepper's inbuilt capabilities. Through its generality, the in this work presented approach can be used to recognize activities of other robots with comparable sensing capabilities.

2 Related Work

This work uses a robot's inbuilt sensors to allow it to detect the actions performed by its own body. This is achieved by recognizing activity patterns in streams of sensor data. In robotics, related work can be found in Collision Detection (CD) and Execution Monitoring (EM).

The area of CD uses a robot's sensors to handle intentional or accidental contact of its body with its physical environment [7]. One of the central motivations is to enable robots to share a common work space with humans by preventing injuries caused by forceful impacts as well as preventing damage to the robot's body.

EM (also known as Fault Detection and Diagnosis) observes sensor readings to detect and classify faults and their causes [1,9,15]. Examples are the detection of mechanical jams and the loss of hydraulic fluid. Conventional EM approaches analyze a robot's activities and determine a set of features that indicates correct execution. These features are then monitored to detect anomalies by either comparing them with the expected system behavior or by subjecting them to pattern recognition methods.

While the areas of CD and EM are interested in fault avoidance and fault detection/recovery respectively, this work aims at allowing a robot to recognize the activities executed by its own body. This goal and the used methods in this work are closely related to the field of HAR which uses wearable sensors to gain insight into human activities.

As part of Human Computer Interaction, HAR creates devices that can recognize their user's physical activities. Wearable sensors are used to capture data about a user's body activity which is then subjected to pattern recognition methods. One example is fall detection in ambient-assisted living. Here, a person is equipped with a wearable device that detects falls and calls help if required [19]. Multimodal HAR approaches use data from multiple sensors to capture activities in greater detail. The use of data from a variety of sensors can achieve a higher classification accuracy than unimodal approaches [12–14,16]. Lara [11] and Cornacchia [6] provide comprehensive surveys about HAR with wearable sensors. Conventional sensor-based HAR approaches use sliding window based techniques combined with manual feature engineering. While these approaches achieve satisfying results on simple activities such as lying, standing and walking, it is difficult to recognize more complex activities. This limitation mainly lies in the manually engineered features that are restricted by human domain knowledge [4]. Recent advances in HAR utilize deep learning techniques because of their automatic feature generation and selection. Deep learning approaches such as LSTMs and Convolutional Neural Networks can come up with task specific non-linear features and provide more accurate classification [18].

3 Robot Activity Recognition

Inspired by similar approaches in the field of HAR [18], this section formulates the task of activity recognition in the context of robotics. Sensors act as a connection between the physical world and the computer and allow to observe a robot's physical state. During task execution, data streams generated by a robot's sensors can be analyzed to gain insight into the performed activities. Assume a robot is executing a sequence of activities belonging to a predefined set A:

$$A = \{a_p\}_{p=1}^{n} \tag{1}$$

where n marks the number of activity types. The robot's sensor readings are observed over time. The observed sequence s contains m consecutive readings r_i, $i \in \{1, \ldots, m\}$, that capture the state of the robot during a period of time at equal intervals. The size of m depends on the sampling rate and the observation duration. For example, the recorded sequences used in Sect. 5.2 each capture 10 readings per second over roughly 5 min. The number of readings that are actually used for a prediction at a given time is dependent on the used model and activity types. Each of the m readings features l attributes.

$$s = (r_1, \ldots, r_m), \quad r_i \in \mathbb{R}^l \tag{2}$$

The goal is to learn a model $\mathcal{M}$ that generates a sequence of predictions $\hat{A}$ about the performed activity at the time of each given reading r_i

$$\mathcal{M}(s) = \hat{A} = (\hat{a}_{r_1}, \ldots, \hat{a}_{r_m}), \quad \hat{a}_{r_i} \in A \tag{3}$$

where the actual performed activity sequence A^* is:

$$A^* = (a^*_{r_1}, \ldots, a^*_{r_m}), \quad a^*_{r_i} \in A \tag{4}$$

A suitable model $\mathcal{M}$ minimizes the discrepancy between predicted sequence $\hat{A}$ and ground truth sequence A^*. Here, A^* can, for example, be determined by a human observer or, as in our later experiments, by logging the commands given by the robot's controller. While this formulation assumes that the readings are sampled synchronously at the same rate, a real robot's sensors usually generate readings asynchronously and at different, sometimes even varying, rates. The following section responds to this by introducing a recognition pipeline that can generate a steady data stream by combining and synchronizing readings from multiple, asynchronous sensors.

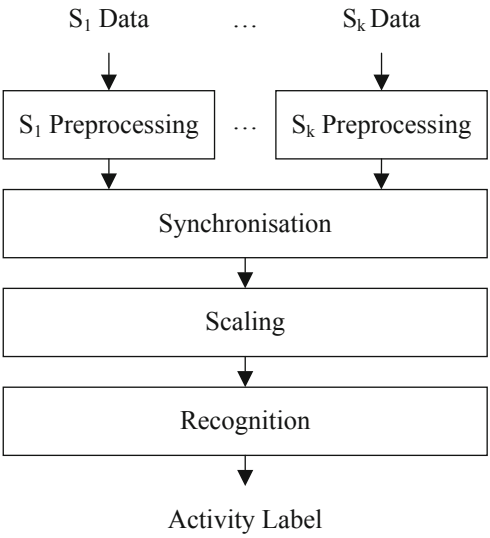

Fig. 1. The proposed activity recognition pipeline. It classifies the current activity performed by the robot by analyzing data streams from multiple sensors.

4 Activity Recognition Pipeline

A general approach for robot activity recognition is proposed. The architecture features a 4-step pipeline (shown in Fig. 1) that recognizes a robot's activity by

analyzing heterogeneous, asynchronous streams of sensor data during runtime. The readings from the individual sensors (Sect. 4.1) are preprocessed separately (Sect. 4.2) and then fused to a combined synchronous data stream (Sect. 4.3). Afterwards, the combined readings are scaled (Sect. 4.4) and subsequently passed to a recognition module (Sect. 4.5) which classifies the activity that is currently performed by the robot. The used scaler and model are intended to be trained offline with data from annotated activity sequences. In the following, the individual pipeline steps are discussed in detail.

4.1 Sensor Data

Sensors can capture the state of a robot's body over time. Heterogeneous sensors, such as accelerometers and gyroscopes, provide data streams that can be used to recognize the performed activities. Common variables include acceleration, torque, electrical current, voltage, orientation, joint states and temperature. The individual variables vary in significance based on the class of activity that is to be predicted. For example, acceleration and torque capture information about the forces that act on a robot's body at a given time and are suitable for the detection of motion activity. Meanwhile, temperature can be seen as an indicator for long term engine activity by being dependent on the amount of heat that is generated over time.

The pipeline assumes that a robot features k sensors S_j, $j \in \{1, \ldots, k\}$. Each sensor S_j samples signal p_j with sampling rate f_j over time. A reading of sensor S_j at given time t provides a d_j dimensional vector:

$$S_j(t) = (v_1, \ldots, v_{d_j}) \in \mathbb{R}^{d_j} \tag{5}$$

4.2 Preprocessing

The recognition pipeline receives one stream of sensor data from each of the k sensors. It can be favorable to perform sensor specific transformations before learning a model. This can reduce the number of required training samples by adding expert knowledge to the model. Each sensor S_j is associated with a preprocessing function Φ_j that is implemented in a separate module. Thereby, the sensor readings are transformed to d'_j dimensional feature vectors.

$$\Phi_j(S_j(t)) = (v'_1, \ldots, v'_{d'_j}) \in \mathbb{R}^{d'_j} \tag{6}$$

For the evaluation, joint angles and electric current are scaled to unit space based on the respective sensor specifications. A filter is applied to the raw acceleration data generated by the inertial measurement unit (IMU) to separate low frequency gravitational acceleration from high frequency activity acceleration [3].

4.3 Synchronization

In the general case, a robot samples its individual sensors at different rates and provides asynchronous data streams. The recognition module assumes all sensors

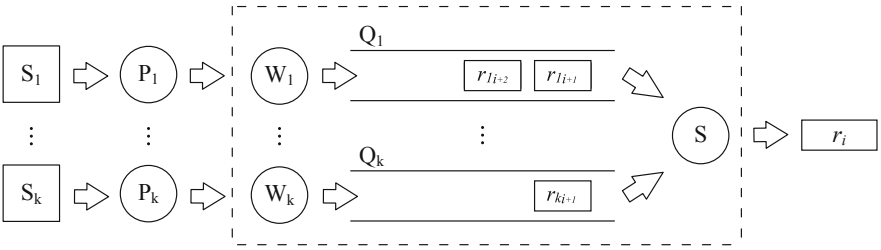

Fig. 2. Synchronization module.

to be sampled synchronously at a predefined rate f, which makes it necessary to synchronize the individual streams of preprocessed sensor data.

The synchronization module fuses the separate streams to one combined data stream. First, an initial start time t_0 is being determined. Subsequently, for each $t_i = t_0 + i * T$ ($T = 1/f$, $i \in \mathbb{N}$), one combined measurement is being interpolated. The module observes all data streams in parallel. For each sensor S_j, one reading r_{j_m} (the m-th reading of S_j) is kept in a buffer together with the timestamp of its creation t_{j_m}. When a new reading $r_{j_{m+1}}$ arrives at time $t_{j_{m+1}}$, the condition $t_{j_m} \leq t_i < t_{j_{m+1}}$ is checked. If the condition is not met, the synchronization module updates its buffer with $r_{j_{m+1}}$ and continues listening to the stream until another reading matches the condition. If the condition is met, the buffer is updated likewise and a linear interpolation between r_{j_m} and $r_{j_{m+1}}$ is performed to determine

$$r_{j_i} = \frac{r_{j_m} * (t_{j_{m+1}} - t_i) + r_{j_{m+1}} * (t_i - t_{j_m})}{t_{j_{m+1}} - t_{j_m}} \tag{7}$$

where r_{j_i} is the representative feature vector for sensor S_j at time t_i which will be used in the recognition process. Subsequently, the module buffers r_{j_i} in a queue and continues to determine $r_{j_{(i+1)}}$. After one vector r_{j_i} for each Sensor S_j has been determined for time t_i, the synchronization module dequeues the vectors, concatenates them and passes the combined reading to the next pipeline step.

A scheme of the synchronization module is shown in Fig. 2. For each sensor S_j, $j \in \{1, \ldots, k\}$, a worker process W_j analyzes the stream of data published by preprocessing node P_j. For time t_i, worker W_j interpolates a representative feature vector as described above and puts it into queue Q_j. Subsequently it continues to interpolate an entry for t_{i+1}. After one feature vector for each sensor has been determined, synchronizer S dequeues the individual vectors and fuses them to one combined reading r_i. This reading is then given to the scaling module for further processing.

4.4 Scaling

The scaling module subtracts the mean from the individual features contained in the synchronized readings and scales them to zero mean unit variance. This

pipeline step reduces the numerical difficulties in the training process and prevents the features in a greater numerical range to have a negative impact on the model. In the experiments, the StandardScaler implementation of the scikit-learn library was used and trained with data from multiple prerecorded activity sequences.

4.5 Recognition

The recognition module receives a stream of synchronized and scaled sensor readings from the previous pipeline step. The stream matches the requirements for the activity recognition formulation described in Sect. 3. Each reading captures the physical state of the robot at a given point in time. Depending on the activities that are intended to be recognized, a suitable model is selected by the programmer. The chosen model analyzes the multimodal sensor data and outputs an activity label that describes the activity the robot is currently performing. The model is trained offline with annotated activity sequences.

In the evaluation, an LSTM based neural network receives a description of the robot's state as an input matrix containing multiple consecutive sensor readings. This matrix is prepared by a small buffer that proceeds the network. The network was trained on 5 readings containing 50 features each. This input goes through two LSTM layers consisting of 32 neurons each. Afterwards, a softmax layer associates each sequence with one of 7 classes (see Sect. 5.2). Each LSTM layer is followed by a batch normalization layer and is regularized by l_1 and l_2 regularizers each with coefficient 0.05. The categorical cross-entropy function is used to calculate the loss and Adam [10] is the used optimizer. The network is trained over 20 epochs with a batch size of 100.

5 Evaluation

An implementation of the activity recognition pipeline is evaluated in two experiments featuring the service robot Pepper. In the first experiment, the pipeline is used to recognize a set of 7 movement activities. The classification accuracy achieved when using single and multimodal sensor data is analyzed. In the second experiment, a human interferes with Pepper's forward movement. It is shown that the pipeline responds to the interference and could be used to verify activity execution.

5.1 Pipeline Implementation

The pipeline was realized with the Robotic Operating System (ROS). The modules are implemented as individual ROS nodes which communicate over ROS topics. The ROS community provides a NAOqi/ROS API to communicate with Pepper's NAOqi operating system. The API offers joint state (50 Hz) and IMU (10 Hz) readings via designated topics. An additional wrapper node was implemented which samples electrical current at 10 Hz and publishes the data to a topic. Each reading is associated with the time of its creation.

The recognition module uses an LSTM based neural network (Sect. 4.5). The network is realized with Keras and trained with annotated activity sequence data. TensorFlow serves as Keras backend. During the data collection process, a training script lets Pepper perform an activity sequence and publishes annotation information whenever it sends a command to the robot. The recognition pipeline runs partially up to the synchronization module (Sect. 4.3) and publishes combined readings containing joint state (17 features), electrical current (20 features) and IMU (13 features) data. Two logging nodes store annotation information and synchronized sensor data in a SQLite database. For the training of the model, the readings are annotated corresponding to their timestamps.

5.2 Recognizing Movement Activities

The activity recognition approach is evaluated on a set of 7 movement activities executed by the Pepper robot. An activity sequence (shown in Fig. 3) is executed by the robot. Sensor readings are captured, annotated and used to train and evaluate scaling and recognition module. The pipeline uses joint state, electrical current and IMU data for its classifications.

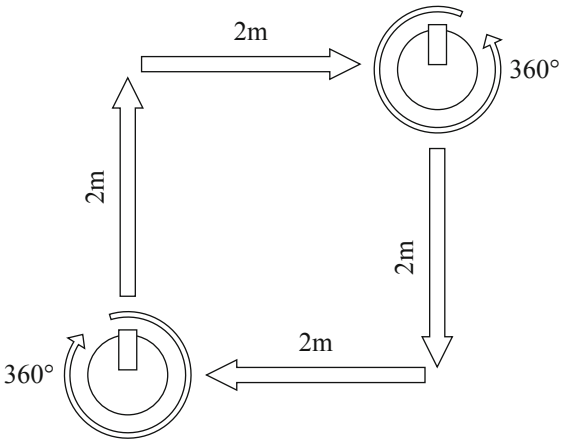

Fig. 3. The recorded activity sequence.

Recorded Data: A control script lets Pepper perform the movement sequence shown in Fig. 3. The robot performs a full clockwise rotation, moves forward, moves right, performs a full counterclockwise rotation, moves backward and finally moves left to its initial position. Between the individual movements Pepper stands still for 2.5 s. The control allows the robot to perform 5 repetitions of the activity sequence and sends corresponding commands and annotation information. The synchronization module interpolates combined sensor readings

at 10 Hz containing joint state, electrical current and IMU data. During the experiments, 10 recordings containing combined sensor readings and annotation information are collected, each capturing little above 5 min of Pepper's movement activity. For evaluation, a 10-fold cross-validation is performed. Each of the 10 folds consists of 2450 annotated samples from one individual recording (350 samples per class picked at random from the respective recording).

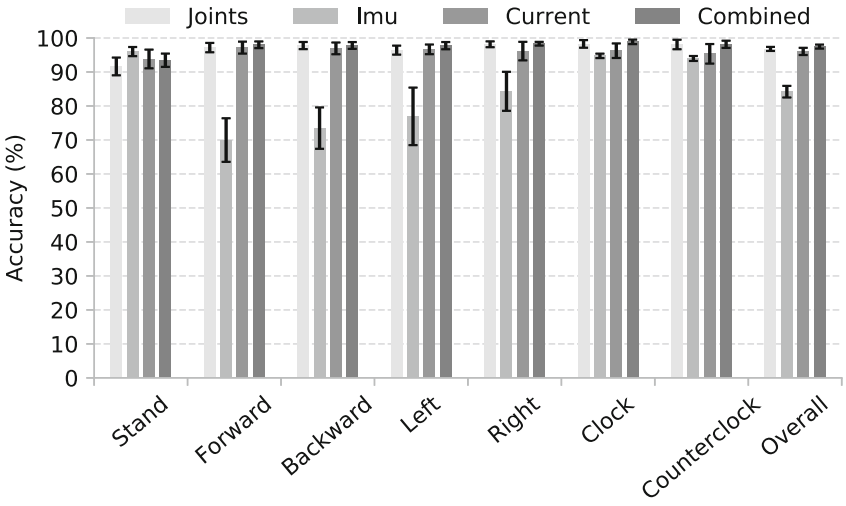

Fig. 4. The classification accuracy and standard deviations achieved on Pepper's base movements when using different sensor data for the recognition process.

Experimental Results: Multiple neural networks are trained to recognize Pepper's movement activities by analyzing different sensor data. Three single-sensor networks are trained with joint states, IMU and electrical current data respectively. One multimodal model is trained to analyze combined readings. The mean per-class and overall accuracies achieved by the different models and respective standard deviations are visualized in Fig. 4. The combined model achieves an mean overall accuracy of 97.47%, which outperforms the joint states (96.78%), IMU (84.20%) and electrical current (96.05%) model. While the combined model outperforms the others in terms of overall accuracy, there are differences in the individual class accuracies. The IMU model achieves a lower overall accuracy than the other models, but achieves the highest accuracy for the standing activity. Also, the IMU model achieves good results for the rotations while achieving worse results on the directional movements. The joint and current model perform similarly except when recognizing the rotating and right moving robot.

5.3 Detecting Human Interference

In this experiment, a human interferes with Pepper's forward movement by pulling its base backward in a way that is not detected by the robot's inbuilt systems. The output of the recognition pipeline is analyzed. It is shown that the approach can detect unexpected behavior that defers from the commands given by the robot's controller and could be used to extend Pepper's inbuilt capabilities.

Recorded Data: The Pepper robot is controlled by a simple script. It first stands still for 5 s, then moves 3 m forward and concludes the sequence by standing still for another 5 s. The control sends the corresponding commands to the robot and publishes annotation information in parallel. The synchronization module of the pipeline publishes combined sensor readings which contain joint state, IMU and electrical current information at 10 Hz. Sensor and annotation information are collected and stored in a database for later analysis. After the robot has executed about half of its forward movement, the experimenter grabs the base of the robot in a way that is not detected by its inbuilt systems and pulls it backward.

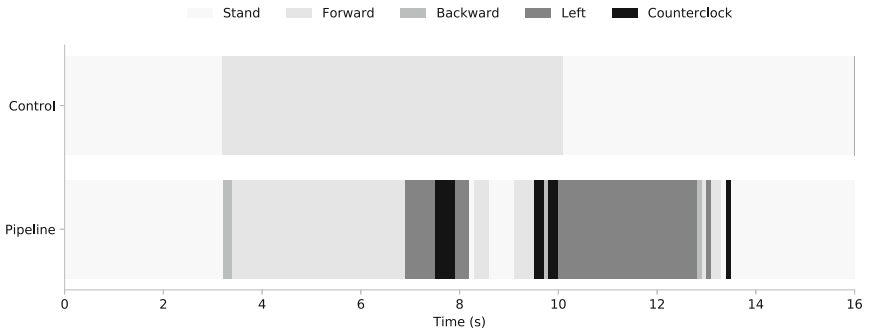

Fig. 5. The robot's activity over time as perceived by the robot's controller and recognition pipeline.

Experimental Results: For this experiment, pipeline scaler and model are trained with the 10 recordings of the previous experiment (Sect. 5.2). Figure 5 compares the robot activity over time as perceived by control layer and recognition pipeline. The pipeline recognizes the standing activity correctly. At the transition between standing and forward movement, the model makes two wrong predictions (200 ms) before recognizing the forward movement correctly. After about 7 s, the experimenter starts pulling Pepper's base backward. While the control does not respond to the interference, the pipeline recognizes the change in the robot's movement and classifies it first as backward, then left and then

backward again. Because the experimenter pulls the robot unevenly, the recognized state then swings for a while between several states which are followed by a long backward movement. After about 10 s, the control sends a standing command and assumes Pepper came to a halt while in reality it is still being pulled by the experimenter. After about 13 s, the interference stops.

To verify correct activity execution, the output of the recognition pipeline can be compared to the robot's commands. If control and pipeline do not agree on the same activity for a certain amount of time (e.g. 0.5 s), an observation system can detect the unexpected behavior of the robot. This can then be used to allow the agent to communicate its problem to a remote supervisor.

6 Conclusion

This work introduced a activity recognition pipeline inspired by HAR methods to the robotics domain. The approach analyzes multiple streams of asynchronous sensor data to recognize the type of action a robot is performing and by doing so allows it to detect its own activities in the physical world. The robot could use this capability to verify its own activity execution or to narrate its actions to a remote person.

The recognition pipeline was evaluated in two experiments. In the first experiment, a set of 7 base movements executed by the service robot Pepper was recognized. It was shown how a model can achieve higher classification accuracy by analyzing combined data from heterogeneous sensors. In the second experiment, a person interfered with the forward movement of the robot by pulling its base backward. While the robot's inbuilt capabilities were not able to detect the external interference, the pipeline successfully recognized the unexpected state. This suggests that the approach could be used to extend Pepper's inbuilt capabilities with an additional verification system.

In future work, we want to recognize more complex activities that go beyond simple base movements. In particular, the application of plan detection is of interest to us. Here, we want to analyze how the recognized activities fit into a meaningful context. This work used proprioceptive sensors because they are more homogenous than external sensors and are closely relate to conventional HAR methods. In further research, readings from external sensors such as rgb cameras and depth sensors can be incorporated into the pipeline to enrich the activity information. Another question is if a model that was trained on one robot can provide accurate predictions when deployed on another robot of the same type.

References

1. Alwi, H., Edwards, C., Tan, C.P.: Fault Detection and Fault-Tolerant Control Using Sliding Modes. Springer, Heidelberg (2011). https://doi.org/10.1007/978-0-85729-650-4
2. Avci, A., Bosch, S., Marin-Perianu, M., Marin-Perianu, R., Havinga, P.: Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: a survey. In: 2010 23rd International Conference on Architecture of Computing Systems (ARCS), pp. 1–10. VDE (2010)
3. Bayat, A., Pomplun, M., Tran, D.A.: A study on human activity recognition using accelerometer data from smartphones. Procedia Comput. Sci. **34**, 450–457 (2014)
4. Bengio, Y.: Deep learning of representations: looking forward. In: Dediu, A.-H., Martín-Vide, C., Mitkov, R., Truthe, B. (eds.) SLSP 2013. LNCS (LNAI), vol. 7978, pp. 1–37. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39593-2_1
5. Chernbumroong, S., Cang, S., Atkins, A., Yu, H.: Elderly activities recognition and classification for applications in assisted living. Expert Syst. Appl. **40**(5), 1662–1674 (2013)
6. Cornacchia, M., Ozcan, K., Zheng, Y., Velipasalar, S.: A survey on activity detection and classification using wearable sensors. IEEE Sens. J. **17**(2), 386–403 (2017)
7. Haddadin, S., De Luca, A., Albu-Schäffer, A.: Robot collisions: a survey on detection, isolation, and identification. IEEE Trans. Robot. **33**(6), 1292–1312 (2017)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
9. Khalastchi, E., Kalech, M.: On fault detection and diagnosis in robotic systems. ACM Comput. Surv. (CSUR) **51**(1), 9 (2018)
10. Kinga, D., Adam, J.B.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
11. Lara, O.D., Labrador, M.A.: A survey on human activity recognition using wearable sensors. IEEE Commun. Surv. Tutor. **15**(3), 1192–1209 (2013)
12. Liu, K., Chen, C., Jafari, R., Kehtarnavaz, N.: Fusion of inertial and depth sensor data for robust hand gesture recognition. IEEE Sens. J. **14**(6), 1898–1903 (2014)
13. Ofli, F., Chaudhry, R., Kurillo, G., Vidal, R., Bajcsy, R.: Berkeley MHAD: a comprehensive multimodal human action database. In: 2013 IEEE Workshop on Applications of Computer Vision (WACV), pp. 53–60. IEEE (2013)
14. Ordóñez, F.J., Roggen, D.: Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors **16**(1), 115 (2016)
15. Pettersson, O.: Execution monitoring in robotics: a survey. Robot. Auton. Syst. **53**(2), 73–88 (2005)
16. Radu, V., Lane, N.D., Bhattacharya, S., Mascolo, C., Marina, M.K., Kawsar, F.: Towards multimodal deep learning for activity recognition on mobile devices. In: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct, pp. 185–188. ACM (2016)
17. Shoaib, M., Bosch, S., Incel, O.D., Scholten, H., Havinga, P.J.: A survey of online activity recognition using mobile phones. Sensors **15**(1), 2059–2085 (2015)
18. Wang, J., Chen, Y., Hao, S., Peng, X., Hu, L.: Deep learning for sensor-based activity recognition: a survey. arXiv preprint arXiv:1707.03502 (2017)
19. Zhang, T., Wang, J., Liu, P., Hou, J.: Fall detection by embedding an accelerometer in cellphone and using KFD algorithm. Int. J. Comput. Sci. Netw. Secur. **6**(10), 277–284 (2006)