3.3

Radar-Based Human-Robot Interfaces

Hans Cappelle[1], Ali Gorji Daronkolaei[1], Ing Jyh Tsang[1,2], Björn Debaillie[1] and Ilja Ocket[1]

[1]IMEC, Belgium
[2]University of Antwerp, Belgium

Abstract

In this article, two implementations of a radar-based human-robot interface are presented. These implementations represent two classes of inference approaches that are investigated in the radar group at imec. The first class exploits traditional machine learning classification techniques. The second class uses spiking neural networks. The machine learning classification system presented in this article supports nine gestures and achieves a gesture classification accuracy of 93%. This compares to an accuracy of 98% for our spiking neural network system operating on four gestures. Based on public data sets, the accuracy of the spiking neural network approach exceeds the published state of the art. Misclassification is however significant, which is still precluding safety critical interactions when using a single radar sensor. As proof-of-concept, a discrete control of a robot will be demonstrated by means of radar-based gesture recognition using five gestures. We present the main concepts of this demonstrator. For pre-validation, we use emulation of the gesture recall statistics and timing characteristics to model the radar part.

Keywords: gesture recognition, 60-GHz radar, machine learning, random forest classification, neuromorphic computing, spiking neural networks, micro-Doppler, human-robot interaction, discrete robot control.

3.3.1 Introduction and Background

In tomorrow's factories, production robots and cobots will need to interact more closely with humans in different types of settings, ranging from advanced assembly lines to the use of exoskeletons to enhance worker capabilities. To ensure safety and active control of those robots, advanced sensors will need to be integrated both on the robots as in the fixed factory infrastructure. These sensors must be reliable and fast while being able to operate in harsh conditions. Often, vision-only approaches will be found to be vulnerable to failure in low visibility conditions.

Millimeter wave radar has the advantage of operating under visually difficult conditions such as darkness, smoke, and dust. Moreover, radar enables to measure the surrounding including the speed of approaches and receding targets. Therefore, radar is excellently suited for collision avoidance. No wonder that this technology is intensively applied in automotive. Radar also enables to use the temporal velocity changes (so-called micro-Doppler patterns) to identify/classify the target. As such, road users can be identified [1], or different hand gestures can be distinguish as demonstrated by Google in their Soli project [2].

Many different approaches can be explored to create a radar-based robot interface. To enhance the intuitive interaction between the operator and the machine, a contact/touchless interface via hand gestures is preferred. These hand gestures could, for example be used to select from a menu (= discrete gestures), or the hand movements could be tracked at real-time to operate the robot (continuous control). Although real-time interaction can be perceived very natural , it comes with significant technical challenges and security risks. Therefore, in this work, we opt for detecting the hand discrete gestures, and we construct a vocabulary allowing the operator to control discrete robot actions. Our envisioned proof-of-concept will control a robot arm taking pictures of an object from different positions and angles. This will enable 3D object modelling.

In this article, we consider two hand gesture recognition implementations using the same 60 GHz radar platform (TI IWR6843 [3]), as well as their suitability for the proof-of-concept demonstrator for interfacing with a robot arm. In our first implementation, the hand gesture type is identified before augmenting it based on the hand location.

This implementation relies on traditional classification techniques based on engineered features [4].

The second implementation does not rely on segmenting the space in quadrants, but only aims to recognize gestures independent on the hand location. This implementation uses a spiking neural network organized as a liquid state machine (LSM) in combination with a trained output classifier layer [5]. For pre-validation of the demonstrator the radar part is modelled by means of gesture command recall statistics and timing behavior.

In the following sections, we start with describing both interface implementations separately. Then we compare both implementations in terms of performance and implementation complexity, as well as how they can be integrated in the proof-of-concept use case.

3.3.2 Gesture Recognition Using a Machine Learning Approach

3.3.2.1 Concept and Experimental Setup

Although the radar system [3] offers only a moderate angular resolution (3x4 MIMO with singe patch antennas), the angular dimensions (both azimuth and elevation) can be exploited to determine the hand position. This provides an extra degree of freedom to design the control interface. The implemented concept is depicted in Figure 3.3.1, showing the training process (in red) and the inference process (in blue).

The result of the radar inference is then used to control a robot arm (in green).Specifically:

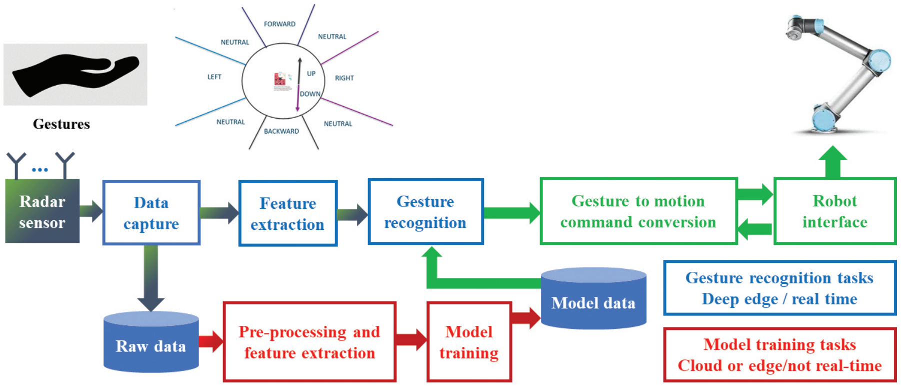

Figure 3.3.1 Human-robot interaction concept using 60 GHz radar.

- For training, indicated in red, raw data is collected. Next radar signal pre-processing is applied, and features are extracted. These features are then used to train the model, and the machine learning model data is saved. These tasks do not require real-time processing and can be done on the edge or in the cloud.
- For gesture recognitions, indicated in blue, features are extracted, and the model is used to recognize the gesture by means of a classifier that uses the machine learning model data.
- The recognized gestures are communicated to the robot part, indicated in green. Gestures are transformed into robot commands which are sent to the robot interface.

The proof-of-concept setup consists of an upwards facing radar mounted in front of the robot operator. The center of detection is approximately 50 cm above the radar sensor. The operated can perform the following hand gestures:

- Palm/hand wave. If a waving hand is detected, then the zone of this gesture is also detected (left/right/front/back/up/down) relative to the detection center. These zones are between 20 and 30 cm away from the center. For example: doing a palm wave at 70 cm (50 cm + 20 cm) above the radar sensor is detected as a "palm-wave/up".
- Pinch. This corresponds to pinching the thumb and index finger.
- Thumbs down. This corresponds to a "thumbs down" gesture with the thumb facing to the radar sensor.
- Tick. This corresponds to making a "V" check movement in the air.

During the measurement campaign, also other gestures were recorded. These were used to model unknown gestures for testing the robustness of the classifier.

3.3.2.2 Inference Pipeline, Training Algorithm

Radar systems exploit electromagnetic waves to detect and locate objects in their environment.

A radar system comprises a transmitter, receiver, and signal processing modules. The implementation uses an FMCW radar [3].

Figure 3.3.2 illustrates an FMCW radar with one transmitter and one receiver, depicting the linear sawtooth and digital signal processing to recover range and Doppler information after the ADC convertor.

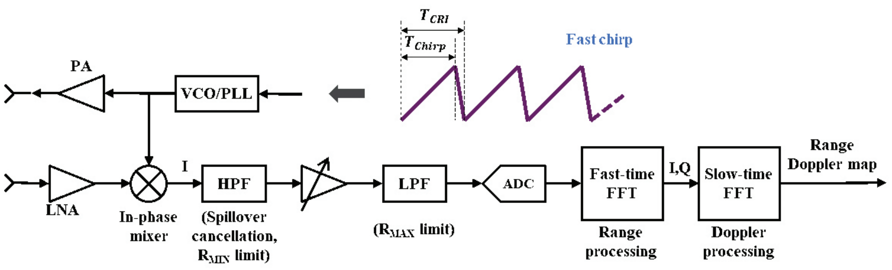

Figure 3.3.2 Simplified radar block diagram.

Transmit Part

The transmit signal is a sinusoid whose frequency is swept linearly from a start frequency to a stop frequency, forming a chirp with a duration of T_{Chirp}. These chirps are repeated with a chirp repetition interval T_{CRI}. A voltage-controlled oscillator (VCO) is used to steer a phased locked loop (PLL) which produces the output signal, which is amplified with a power amplifier (PA) and sent to the transmit antenna.

Receive Part

A reflected signal is picked up by the receive antenna and amplified with a low noise amplifier (LNA). It is mixed with the transmitted signal producing a beat signal that has a frequency that depends on the range (delay) of the reflected signal. A high pass filter (HPF) is used to removed unwanted signal components at low frequencyies, determining the minimum range and cancelling spillover from the transmit signal. Next the signal is amplified by a second amplifier and passed through a low pass filter (LPF) to limit the maximum range. Next, the signal is digitized with an analog to digital convertor (ADC) and a range Doppler map is produced by doing a fast-time fast Fourier transform (FFT) to recover range and slow time FFT to produce Doppler information. Note that by doing so only moving objects can be observed.

In this implementation, all data after the ADC is processed on a laptop, using a data capture board [6]. Angular information can be obtained by combining multiple transmitters with multiple receivers. Figure 3.3.3 shows the signal processing to obtain angular and micro-Doppler information, used to generate feature data. To generate a point cloud a constant false alarm rate (CFAR) detector is used to identify targets, and the MUSIC algorithm is used

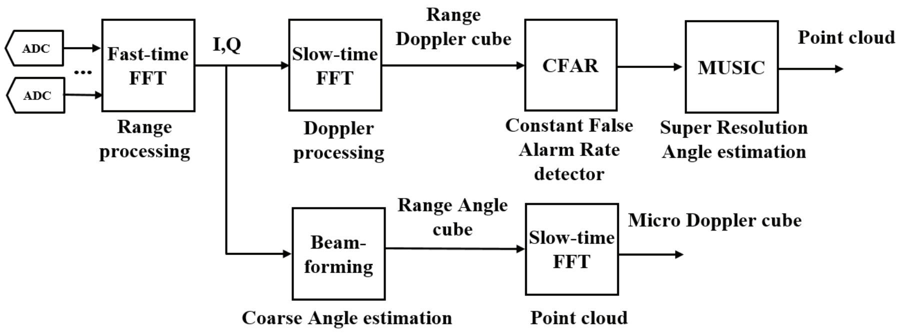

Figure 3.3.3 Signal processing pipeline to provide input to the feature generator.

to annotate identified targets with angular information. To generate micro-Doppler data, beamforming is applied to do a coarse angle estimation, and a slow time FFT to generate a micro-Doppler cube.

To train and evaluate a random forest classifier, feature data is extracted from these signal processing blocks, as illustrated in Figure 3.3.4. We extract ten features, subdivided in four classes [4]:

1. MD: micro-Doppler features:
   - RAW: a sub-sampled micro-Doppler cube
   - ENV: a curve fit of the micro-Doppler envelopes
2. RD_ROI: range-Doppler region of interest features. This corresponds to a denoised and subsampled version of the range Doppler information.
3. POINT: point cloud features. This tracks the average, mode and standard deviation of range (RNG), elevation (ELV), azimuth (AZM) and Doppler (DOP) over several radar frames (of 90 milliseconds each).

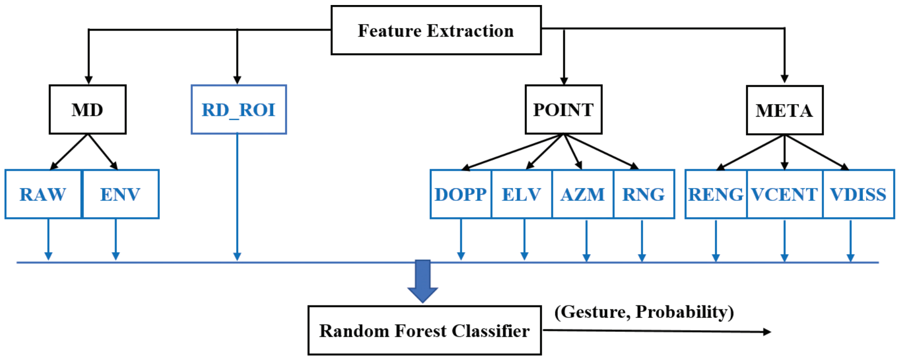

Figure 3.3.4 Feature extraction for the random forest classifier.

4. META. From the range Doppler cube, the following meta parameters are derived:
    - RENG: range instantaneous energy
    - VCENT: centroid velocity
    - VDISS: dispersion of velocity

3.3.2.3 Data Recording and Results

The machine learning approach was based on a supervised framework. First, the setup collected a dataset of gestures for the learning phase. Table 3.3.1 gives the radar parameter used for these measurements [3].

The TI DCA1000 [6] data capture board was used to obtain raw data samples. While the maximum unambiguous range is 11.3 meter, the maximum range was restricted to 1.5 m since larger heights are not relevant for the upward facing radar.

To train the machine leaning model, 22 different test subjects with two types of gesture were recorded.

- In 6 zones (left, right, up, down, backwards, forward) measurements were done for a palm wave gesture under different conditions (normal speed, fast speed, left arm, right arm).
- Six gestures were done in the central position: **pinch**, thumb-up, **thumb-down**, cross, **tick**, palm tilt with different speeds and hand. After analysis it was decided to retain only the pinch, thumb-down and tick gestures.

Table 3.3.1 Chirp/Frame (a) and scene (b) radar parameters.

Start Frequency (GHz)	60.2
Slope (MHz/us)	60.0
Samples per chirp	280
Chirps per frame	128
Sampling rate (Msps)	5.47
Sweep bandwidth (GHz)	2.70
Frame period (msec)	90
Transmit antennas	3

(a)

Range resolution (cm)	4.46
Maximum unambiguous range (m)	11.3
Maximum radial velocity (m/s)	1.75
Radial velocity resolution (m/s)	0.0273
Azimuth resolution (Degrees)	14.5

(b)

Table 3.3.2 Recall and precision statistics of the machine learning based detector.

Percentage	Tick	Pinch	Thumb down	Left	Right	Up	Down	Forward	Backward	Unknown
Recall	87.3	74.4	78.7	96.1	93.6	94.8	96.8	93.6	89.4	83.2
Precision	87.3	84.2	76.6	80.3	81.0	88.5	86.0	83.0	90.8	89.7

Labeling the data took most effort, and unsupported or poorly executed gestures were labeled as unknown. For the palm-wave, some gestures were relabeled to another type if the test subject made the gesture in the wrong zone.

For training the model a 5-fold cross validation was used, doing 5 runs using 80% users for training and 20% for testing the model. To assess the performance, we look at detector statistics, timing, and real time inference performance.

Machine Learning Detector Statistics

Achieved detection accuracy is 86.1% with 13.8% misdetections. The detection rate is significantly impacted by using unknow data for input stimuli. If this data is not included, then detection performance increases to 92.8% with 7.2% misdetections.

Table 3.3.2 shows the achieved recall and precision statistics of the detected gestures. Recall shows the probability that a gesture is detected correctly, while precision indicates the percentage that a reported gesture is correct.

Machine Learning Real Time Inference

We use an Intel Core i7-8750H @ 2.20GHz based laptop to run all signal processing after ADC, feature extraction and classification in python on an Ubuntu 16.04 operating system. Critical parts are optimized in C or C++. While 12 cores are available, we use no explicit multi-threading. Real-time performance is achieved, and processing delay is less than 120 milliseconds.

3.3.3 Gesture Recognition Using a Spiking Neural Network

For the second implementation, a spiking neural network (SNN) approach was used for the radar-based hand gesture recognition (HGR). For this implementation, the same FMCW millimeter-wave radar was used. After

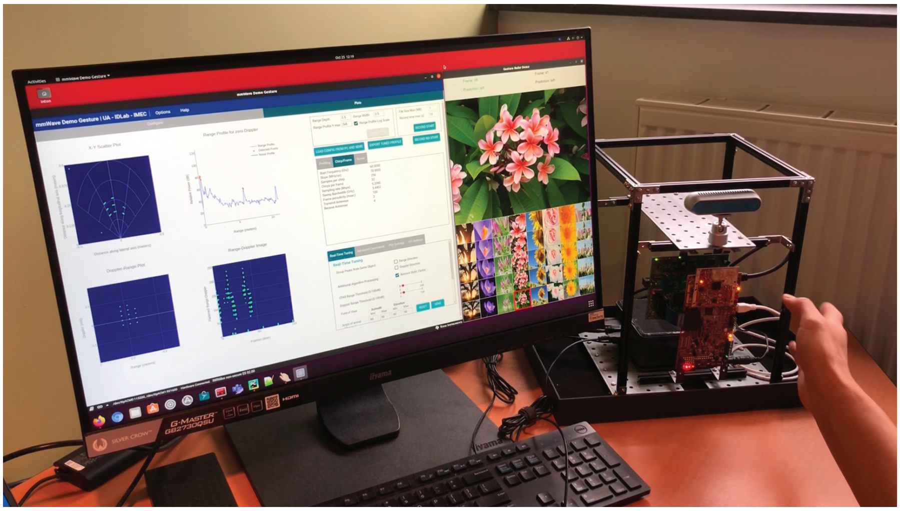

Figure 3.3.5 SNN-based gesture demonstrator.

pre-processing the range-Doppler radar signal, we use a signal-to-spike conversion scheme that encodes radar Doppler maps into spike trains. The spike trains are fed into a spiking recurrent neural network, a liquid state machine (LSM). The readout spike signal from the SNN is then used as input for a logistic regression which is used as a classifier in a supervised learning machine learning framework.

3.3.3.1 Concept and Experimental Setup

The proof-of-concept setup of the second implementation is shown in Figure 3.3.5. This implementation differs from the previous one in two ways. Firstly, the hand is now placed at a more or less fixed distance to the radar. No attempt is made to identify where the hand is positioned in space in front of the sensor. Secondly, the demonstration focuses on accurately identifying the gesture the person is making. The gesture vocabulary is “Swipe left”, “Swipe right”, “Zoom out” and “Zoom in”, allowing the user of the demonstrator, e.g., to navigate through a series of pictures and zoom in/out on each of them.

3.3.3.2 Inference Pipeline, Training Algorithm

The radar system collects the range-Doppler frames, representing the velocity and distance of the reflected object (i.e., hand). These frames were mapped as

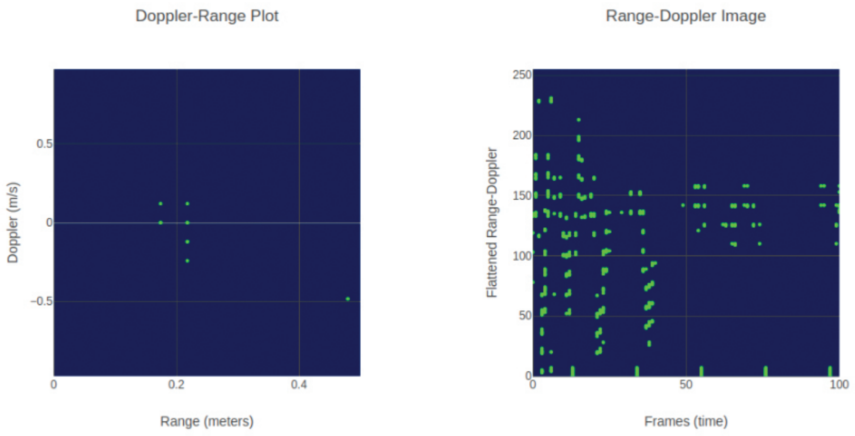

Figure 3.3.6 Range vs Doppler (left), flattened range-Doppler vs frames (right).

16×16 images (Figure 3.3.6 left), where a sequence of frames represents the complete gesture. Each gesture has different time durations, thus different amounts of frames. Each frame was unrolled and vertically stacked for the data representation, creating a frame versus flattened range-Doppler matrix (Figure 3.3.6 right).

The flattened range-Doppler can be seen as a unique pixel location of the 16×16 range-Doppler images. The frame vs. pixel representation captures the information in time, which is ideal for the signal-to-spike neural encoding to produce the spike train input for the LSM network.

The LSM is a type of reservoir computer capable of universal function approximation [7]. The basic formulation of LSM maps an input function $u(\cdot)$ onto a filter, or liquid neurons, L^M while the output $x^M(t) = \left(L^M u\right)(t)$ is fed to a second component, a readout map f^M, which is task-specific and generates the output $y(t) = f^M\left(x^M(t)\right)$.

The readout maps in our context will be a classifier that receives a state as input. Different classifiers can be used for this second component, such as logistic regression, random forest, or support vector machine. For simplicity and ease of in hardware implementation, we focus on the logistic regression in these experiments.

Figure 3.3.7 shows the LSM and how it has been used to build an end-to-end system for gesture recognition.

The top part of Figure 3.3.7 depicts the timing and how each gesture is sampled in the LSM. T_{s0} and T_{s1} are the boundaries of the time interval reserved for a gesture, wherein the spike train of a gesture can have a variable stimulus length duration.

After the end of the stimulus, a readout delay t_r determines the readout window interval, during which the state of the liquid is measured and stored or passed to the classifier, depending on whether it is used in a real-time online or offline learning and inference system.

When mapping to the LSM, each sample had a different stimulus length. As a result, the readout window varies according to the sample frame duration.

The conversion from spike at pixel position i, of frame n to spike $s_i(t)$ at time t, is a direct map from n to t, i.e., if frame n has a spike at pixel i, then $s_i(t)$ has a spike at $t = n$. An alternative way to map the LSM was to normalize the frames to a predefined fixed stimulus length, whereas all the samples have the same readout window duration.

For every pixel position i, we convert the spike in frame n to a relative time regarding a fixed stimulus length S_l . Thus, the spike train sequence is given by:

$$s_i(t) = \frac{f_n * S_l}{f_l}$$

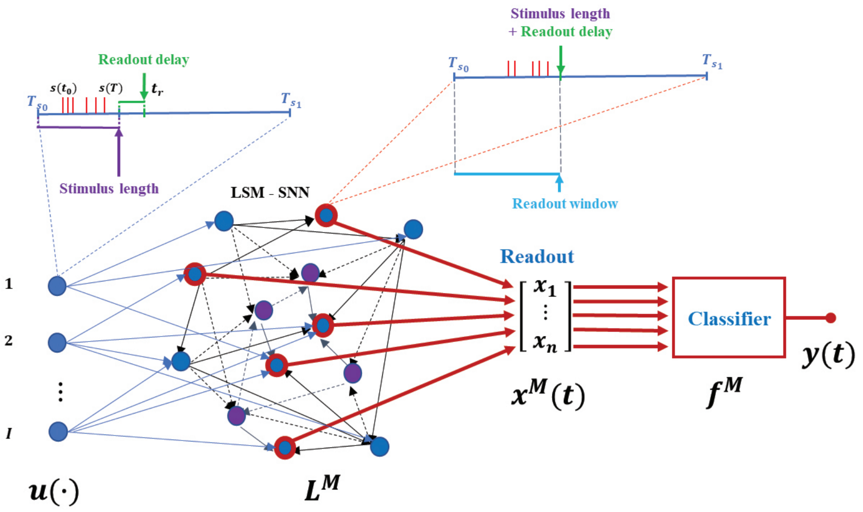

Figure 3.3.7 LSM network with trainable output layer.

Where:

S_l= predefined fixed stimulus length;

f_n= frame number,n that contains a spike;

f_l= length of the particular sample in number of frames.

In constructing the LSM, we focus on achieving the most compact and simpler to implement network without sacrificing accuracy. Each pixel i will produce a spike train as an input to the LSM, and each input is randomly connected to C_{inp} excitatory neurons.

All excitatory neurons are used for readout. For the neuron unit, we used a leaky integrate-and-fire neuron model with exponential postsynaptic currents with the associated synaptic model, based on [8].

3.3.3.3 Data Recording and Results

The SNN approach was based on a supervised framework. First, we collect a dataset of gestures for the learning phase.

Table 3.3.3 details the radar parameter used for the radar [3] in this demo setup. Notice that while it was configured for 32 chirps per frame, we rescaled to a 16 x 16 range-Doppler image to compose the frame versus pixel representation, reducing to a total of 256-pixel channels as input to the LSM.

The range depth, width and resolution were configured to around 0.5 m, thus only the reflected signal directly in front of the radar receivers were captured for the range-Doppler frames.

Table 3.3.3 Chirp/Frame (a) and scene (b) radar parameters.

Start Frequency (GHz)	60.0
Slope (MHz/us)	70.0
Samples per chirp	256
Chirps per frame	32.0
Sampling rate (Msps)	5.21
Sweep bandwidth (GHz)	3.44
Frame period (msec)	100
Transmit antennas	2

(a)

Range resolution (cm)	4.36
Maximum unambiguous range (m)	8.93
Maximum radial velocity (m/s)	0.974
Radial velocity resolution (m/s)	0.122
Azimuth resolution (Degrees)	14.5

(b)

Ideally, the more diverse and generic the learning dataset, the better generalization can be achieved by the machine learning framework.

Conversely, a personalized dataset can be used to tune the system for a specific user. In this case, we have collected four gestures from a single person. Collecting data from multiple persons will be done in the next phase.

Each gesture was collected in 30 separate sessions, wherein each session, a gesture was repeated 15 times.

The learning set contained then 450 samples for each of the four gestures. Figure 3.3.8 shows the confusion matrix of a 90%-10% learning (1620 samples) and test (180 samples) split of the dataset.

The LSM consisted of 600 neurons and a normalized stimulus length $S_l = 20$.

Table 3.3.4 summarizes recall and precision statistics of the SNN based detector, for the confusion matrix in Figure 3.3.8.

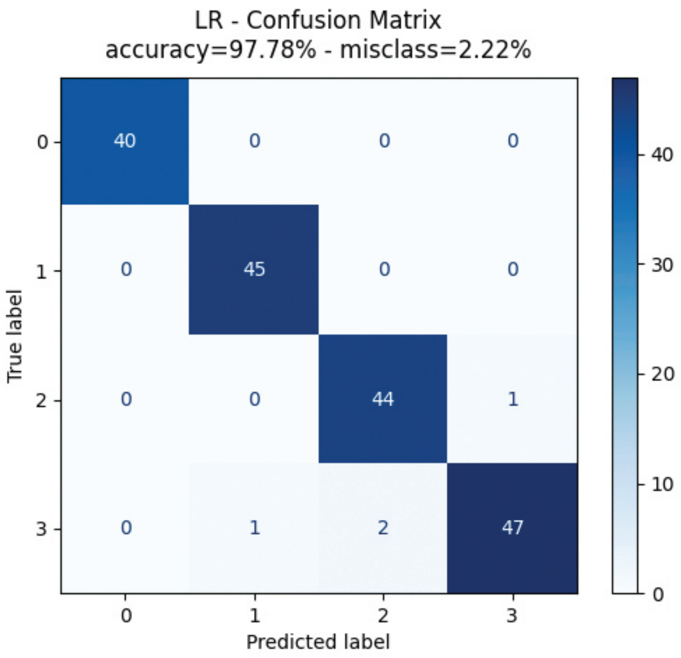

Figure 3.3.8 Confusion matrix for a 90%-10% learning (1620 samples) and test (180 samples) split of the dataset.

Table 3.3.4 Recall and precision statistics of the SNN based detector.

Percentage	Zoom out (label 0)	Zoom in (label 1)	Swipe left (label 2)	Swipe right (label 3)
Recall	100	100	97.8	94.0
Precision	100	97.8	95.7	97.9

3.3.3.4 Discussion

The setup was based on an Intel NUC Core i7-10710U (12MB Cache, 1.10GHz) with 32 GB DDR4 RAM. A complete capture, classification, and image cursor movement took between 0.5 to 1 sec for inference. The recognition rates reflect the confusion matrix shown in Figure 3.3.8, depending on the hand's relative position to the radar. The learning phase is relatively fast as the SNN was designed to be compact and efficient, considering the possibility of being deployed on an embedded system. The learning process was performed when launching the program, and uses a few minutes. The collection of the learning dataset was the most time-consuming element.

Dataset personalization shows that the system can be tuned specifically to the user operating the robot or other device to be controlled. Generalization to many users or a generic user base depends on the learning dataset. Moreover, extending the dataset to recognize more gestures can be easily done. In both cases, other classification schemes, such as support vector machine (SVM) or random forest, can be applied and integrated straightforwardly into the system. The spiking neural network algorithms were also validated on public data sets [5], achieving a gesture detection accuracy of 98%, which is better than the published state of the art. Still there is a misclassification chance of 2%. This restricts gesture input to non-safety-critical applications.

3.3.4 Proof of Concept Demonstration

We envisioned a proof-of concept demonstrator with five gestures to control the position of a robot arm:

- The robot arm positions a camera at discrete locations around an object. The robot arm can be moved to the left or to the right of the object, following a pre-defined trajectory. Also, the camera can be tilted up and down.
- A picture can be made of a 3D object and stored for post processing.

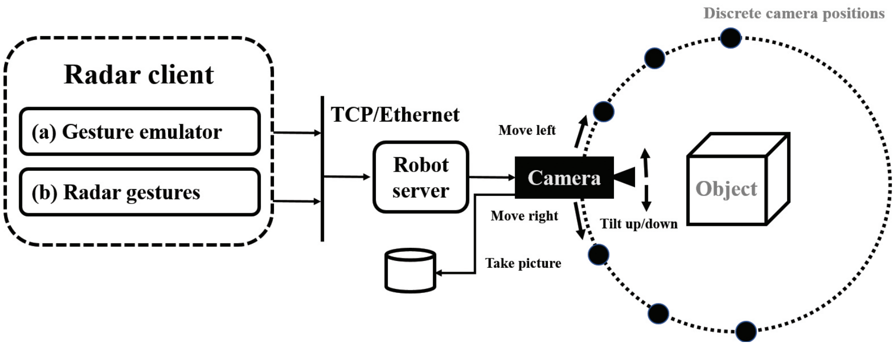

Figure 3.3.9 Proof of concept radar/robot interface block diagram.

These interactions are shown in Figure 3.3.9. The gesture-detecting radar system and the robot arm will interface over TCP/IP via an Ethernet link. A radar client collects gestures and transmits them to a robot server, which converts the gestures to robot commands to manipulate the camera location/position. There is a risk that a gesture is misclassified, resulting in a faulty recall. Such errors need to be corrected by the operator. This can be done without an increased safety risk. We still need to decide on the process to take the pictures and to handle the associated risks for sub-optimal captured data.

The radar client can either be emulated (a) or use a radar part (b) as shown in Figure 3.3.9. Either one of these modes is used. In emulation mode, it is sufficient to model the recall statistics of the different gestures together with their latency. This allows pre-validation of the robot part of the use case, without requiring a radar part or classification. For demonstration the emulation part is replaced by a suitable radar and classification. The TCP/IP communication scheme stays identical.

For the radar machine learning approach, sufficient gestures are available to support this use case. A logical choice is to use gestures with the best recall statistics. At least one gesture needs to be added to the spiking neural network if this approach is chosen.

3.3.5 Comparison and Conclusion

Both implementations (the machine learning and the spiking neural network) successfully detect gestures using a single radar. We observe that the SNN implementation achieves a better detection performance of 97.8%. The main reason is that larger training sets are used for a single user. For the machine learning implementation, we observe that the classifier sometimes generates valid gestures for unseen data. Excluding this (for fair comparison), the detection performance improves from 86.1% to 92.8%. Although the obtained detection performances are rather high, the current implementations are not yet suited for safety critical applications. Both approaches require some time to detect a gesture, which may exceed half a second. This enables discrete control of a robot but precludes real time control. We envision a proof-of-concept demonstrator to illustrate the interaction between the radar system and the robot arm. This system will control the position/location of a camara mounted on the robot arm based on hand gestures detected by the radar system. A statistical model of the radar system is being created to allow early evaluation. This model combines gesture recall statistics with latency characteristics. The proof-of-concept demonstrator will be fully developed within the frame of the AI4DI project together with other consortium partners.

Acknowledgements

This work is conducted under the framework of the ECSEL AI4DI "Artificial Intelligence for Digitising Industry" project. The project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826060. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Austria, Czech Republic, Italy, Latvia, Belgium, Lithuania, France, Greece, Finland, Norway.

References

[1] Dimitrievski, M., Shopovska, I., Van Hamme, D., Veelaert, P., & Philips, W. (2020). Weakly supervised deep learning method for vulnerable road user detection in FMCW radar. In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Proceedings. Rhodes, Greece. https://doi.org/10.1109/ITSC45102.2020.9294399

[2] Wang, S., Song, J., Lien, J., Poupyrev, I.., Hilliges, O. (2016). Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology, Tokyo, Japan, 16–19 October 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 851–860.

[3] Texas Instruments, “IWR6843 intelligent mmWave overhead detection sensor (ODS) antenna plug-in module”, https://www.ti.com/tool/IWR6843ISK-ODS.

[4] A. Gorji, A., Khalid, H. U. R., Bourdoux A., and Sahli, H. (2021). “On the Generalization and Reliability of Single Radar-Based Human Activity Recognition,” in IEEE Access, vol. 9, pp. 85334-85349, 2021. https://doi.org/10.1109/ACCESS.2021.3088452

[5] Tsang, I.J., Corradi, F., Sifalakis, M., Van Leekwijck, W., Latré, S. (2021). Radar-Based Hand Gesture Recognition Using Spiking Neural Networks. Electronics 2021, 10, 1405. https://doi.org/10.3390/electronics10121405

[6] Texas Instruments, “DCA1000EVM: Real-time data-capture adapter for radar sensing evaluation module”, https://www.ti.com/tool/DCA1000EVM.

[7] Maass, W., Natschläger, T., Markram, H. (2002). Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. Neural Comput. 2002, 14, 2531–2560.

[8] Tsodyks, T., Uziel, A., Markram, H. (2020). Synchrony Generation in Recurrent Networks with Frequency-Dependent Synapses. J. Neurosci. 2000, 20, RC50