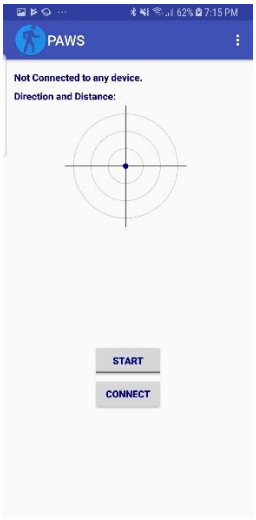
# PAWS Smartphone Application Manual

## I. Setup

Before running the application, copy the “SEUS” folder into your phone’s “Download” folder. This folder contains the machine learning models used in the application.

## II. Using the Application

After launching the application, you will arrive at the screen shown below.



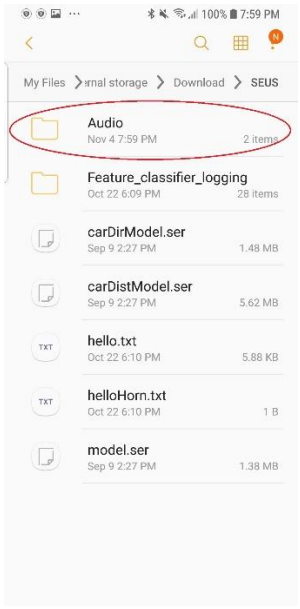

To connect the smartphone application to a PAWS headset via Bluetooth, press the “Connect” button and select the appropriate connection (generally “Nordic_UART_0”). The headset provides car direction estimates to the smartphone application.

To begin running the application, press the “Start” button. If a car is detected, a quadrant in the polar graph will light up. If the headset is connected to the smartphone application, this quadrant will correspond to the relative direction of the car estimated by the headset. If there is no headset, then this will just be a random quadrant. Press the same button again to stop the system.

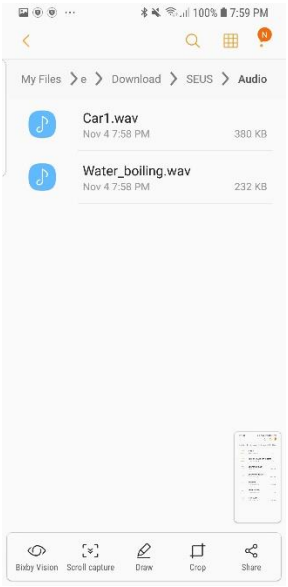
## III. Training a New Detection Model

You may want to train your own model for car detection. To do so you must:

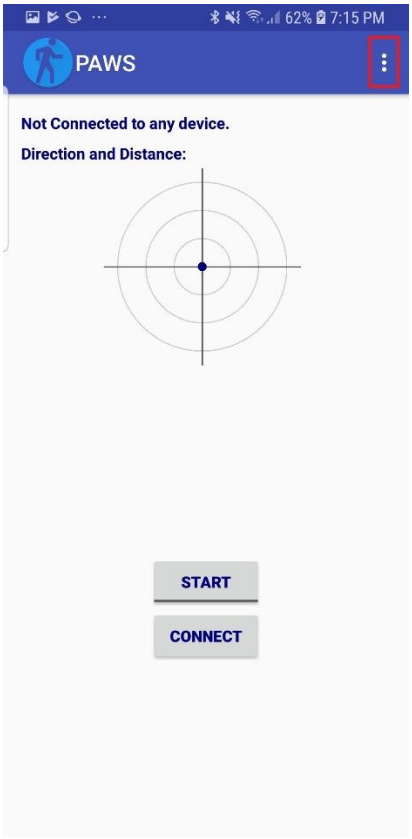
1. Create an "Audio" folder inside the "SEUS" folder you placed in your phone's "Download" folder from part I, as shown below.



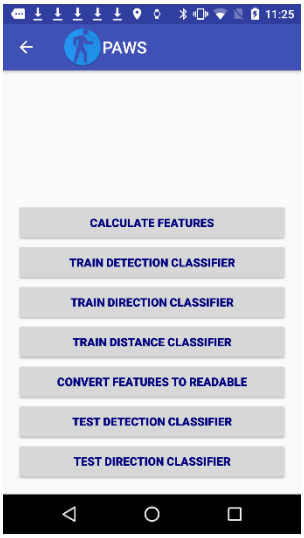

2. Record audio clips of cars and non-car street sounds using a free recording application such as "Easy Voice Recorder". Make sure to set the sampling frequency to 48 kHz, as the PAWS application uses 48 kHz, and save the files inside the "Audio" folder that you just created. Car sounds should have "car" somewhere in the name, and non-car sounds should not contain "car" anywhere in the name. An example is shown below.



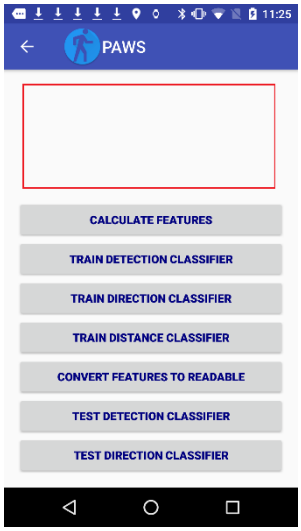
3. Inside the application, tap the triple-dot icon in the upper right corner and select “Train Classifier”




4. On the new screen shown below, first tap the “Calculate Features” button to extract the features from the audio files.




5. Wait until some text shows up in the display telling you how many files were processed and number of features extracted. The text will show up in the area marked in red below.




6. Next, press the “Train Detection Classifier” button to train the models and wait for the display to update, which will signify that the training is finished.
7. Restart the application to use the new models.

## IV. Viewing and Logging Features from PAWS Headset

Sometimes it is helpful to view and log the features computed from the headset. There are two ways to do this. You can either log and view features, in real time, computed from the headset using the *SEUS Features* application, or you can just log features by using the *PAWS* application.

a. Logging and Viewing Features using the *SEUS Features* Application:

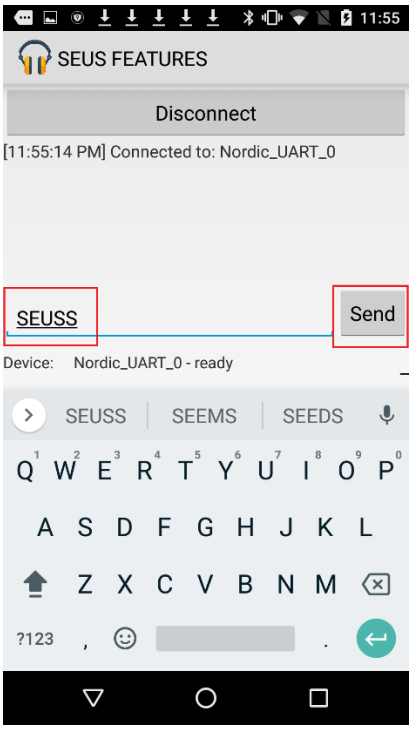
Install the *SEUS Features* application (Android Studio project files located: https://github.com/Columbia-ICSL/PAWS-Smartphone/tree/master/PAWS_Features/SEUS_Features).

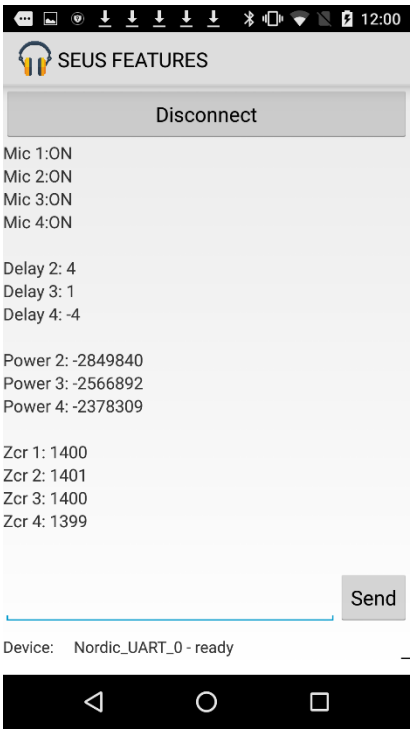
Before starting the application, make sure to give *SEUS Features* the "Location" permission to access Bluetooth and the "Storage" permission to log outputs.

Additionally, you must create a folder named "HEADDATA" in the "/mnt/sdcard/" folder on your phone.

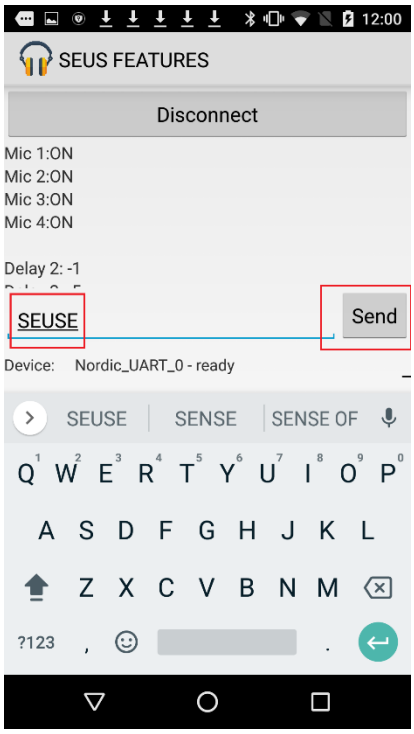
Once these folders are created, turn on *SEUS Features* and the headset and connect.

Once connected, send the command: *SEUSS* to the headset and you will begin to see a live update of the features computed from the headset (the three relative powers, four zero delays, and three relative delay values). You will also see the status of each microphone (“ON” or “OFF”), which will notify if you need to change the batteries that power the microphone.

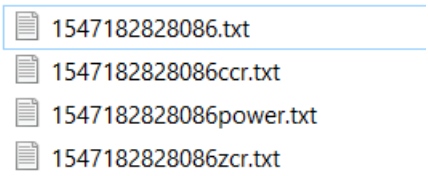


To stop the transmission, either hit the disconnect button or send the command: *SEUSE* to the headset.




All windows of features will be stored in the "/mnt/sdcard/HEADDATA" folder, that you created before running the application, as .txt files in .csv format using the UNIX time of when you first connected to the headset as the file name. Each time you connect to the headset and collect data, four files are generated.




- [UNIX TIME].txt: Raw byte stream received from headset
- [UNIX TIME]ccr.txt: Relative delays between microphones computed from the headset; each row is one window and each window will have three relative delays (three columns)
- [UNIX TIME]power.txt: Relative microphone powers computed from the headset; each row is one window and each window will have three relative powers (three columns)
- [UNIX TIME]zcr.txt: Zero crossing rate of each microphone computed from the headset; each row is one window and each window will have four values, corresponding to each of the four microphones (four columns)

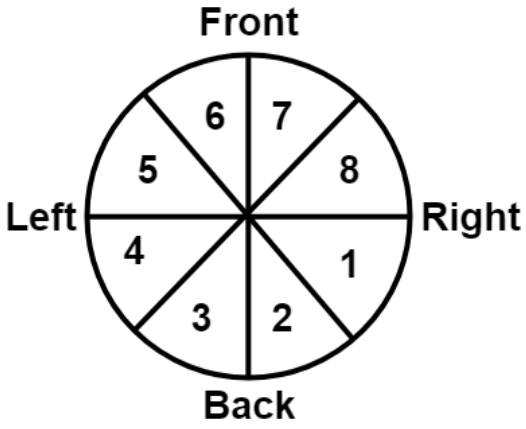
b. Logging Features using the *PAWS* Application:

A second way to log the output from the headset is using the *PAWS* application. Unlike using the *SEUS Features* application, the *PAWS* application will not only log output from the headset, but it will also log output of the detection classifier, output of the direction classifier, output of the distance classifier, as well as the vector of features computed on the smartphone used for detection.

To enable logging, make sure that the "LOGGING_FLAG" is set to "1"; set it to "0" to disable logging. The "LOGGING_FLAG" can be found in the file "app/src/main/java/bashima/cs/unc/seus/constant/Constant.java". By default, this flag is set high, so you will not have to modify the project files if you have a fresh download of the project.

Connect to the headset and start running the application. Once you disconnect or stop the application, a new folder named "Feature_classifier_logging" will be created in the "SEUS" folder. The .csv file containing the features and classifier outputs will be stored in this folder with the UNIX time as the file name. Each row contains one window of information (e.g. one window of features and classifier outputs) with the following structure:

- Columns 1 – 4: Zero crossing rate of each microphone computed from the headset; These values will be "-Inf" if not connected to headset
- Columns 5 – 7: Relative microphone powers computed from the headset; These values will be "-Inf" if not connected to headset
- Columns 8 – 10: Relative delays between microphones computed from the headset; These values will be "-Inf" if not connected to headset
- Column 11: Detection classifier output; This value will be "1" if car is detected and "0" otherwise.
- Column 12: Direction classifier output; Value mapped to direction is shown below



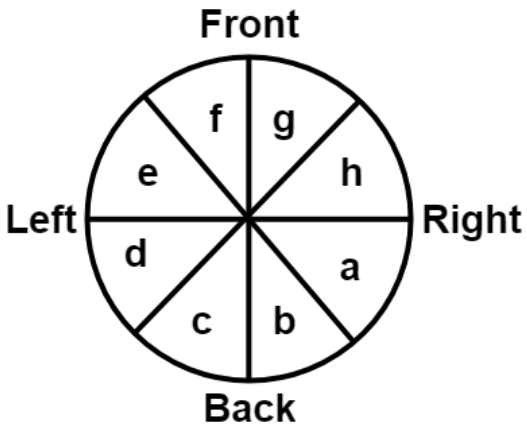

- Column 13: Distance classifier output;
  - "1" if car is close (between 0 – 30 m)
  - "2" if car is between (30 – 60 m)
  - "3" if car is 60 meters or greater away.
- Columns 14 – end (33 by default): Detection feature vector computed on smartphone

## V. Training a New Direction Classification Model

It may be useful to train or retrain the direction classifier in certain cases (e.g. after changing the configuration of the microphones).

To do this, you need labeled feature data, which you can obtain using either the *SEUS Features* or the *PAWS* application, as explained in Section III.

Generating labeled data using *SEUS Features*:
Once you have logged your data using the *SEUS Features* application, copy the contents in the file named "[UNIX TIME]ccr.txt" into a file named "carWeka.csv" and place it inside the "SEUS" folder. The new .csv file should contain three columns (each column corresponding to a relative delay feature), and in the fourth column label each window. The label that should be applied for each direction is shown below.

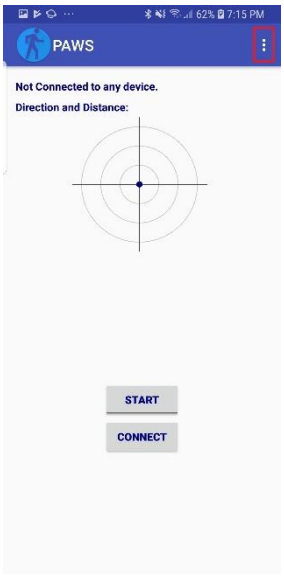


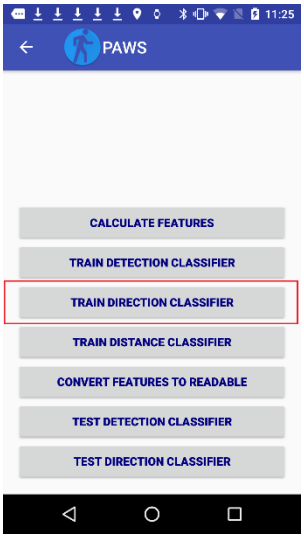

An example of the "carWeka.csv" file is shown below.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 11 | -5 | 13 | f |
| 2 | 9 | -10 | 12 | f |
| 3 | 6 | -12 | 12 | f |
| 4 | 7 | -11 | 12 | f |
| 5 | -2 | -13 | 12 | f |
| 6 | -2 | -15 | 10 | f |
| 7 | -2 | -15 | 9 | f |
| 8 | -3 | -15 | 9 | f |
| 9 | -5 | -17 | 7 | f |
| 10 | -8 | -18 | 7 | f |
| 11 | 11 | -4 | 13 | f |
| 12 | 13 | -4 | 13 | f |
| 13 | 11 | -2 | 13 | f |
| 14 | 9 | -8 | 13 | f |
| 15 | 8 | -8 | 12 | f |
| 16 | 6 | -10 | 11 | f |

Generating labeled data using *PAWS*:
Once you have logged your data using the *PAWS* application, copy columns 8 – 10 in the file named "[UNIX TIME].csv" into a file named "carWeka.csv" and place it inside the "SEUS" folder. The rest of the steps are the same as the process of labeling data using *SEUS Features*.

Training the model:
Once the "carWeka.csv" file is generated and placed in the "SEUS" folder, open the *PAWS* application and tap the triple-dot icon in the upper right corner and select "Train Classifier". In the new menu, select the "Train Direction Classifier" button and wait until the application is finished training. When the application finishes training, you will receive a message that says that the direction classifier was successfully built. To use the newly trained model, restart the *PAWS* application.



## VI. Training a New Distance Classification Model

It may be useful to train or retrain the distance classifier in certain cases.

To do this, you need labeled feature data, which you can obtain using either the *SEUS Features* or the *PAWS* application, as explained in Section III.

Generating labeled data using *SEUS Features*:
Once you have logged your data using the *SEUS Features* application, copy the following contents in the data files into a file named "carDistWeka.csv":

- Copy "[UNIX TIME]zcr.txt" into columns 1 – 4 in "carDistWeka.csv"
- Copy "[UNIX TIME]power.txt" into columns 5 – 7 in "carDistWeka.csv"
- Copy "[UNIX TIME]ccr.txt" into columns 8 – 10 in "carDistWeka.csv"

Place "carDistWeka.csv" inside the "SEUS" folder. The new .csv file should contain 10 columns, and the 11$^{th}$ column should contain the label of each window. The label that should be applied for each distance is listed below:

- "a" if car is close (between 0 – 30 m)
- "b" if car is between (30 – 60 m)
- "c" if car is 60 meters or greater away.

An example of the "carDistWeka.csv" file is shown below.

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1071 | 1205 | 402 | 828 | 62482 | -96574 | 37214 | 17 | 16 | 17 | c |
| 2 | 1120 | 1217 | 367 | 881 | 75538 | -77028 | -12566 | 15 | 12 | 17 | c |
| 3 | 912 | 1233 | 348 | 775 | 125600 | -46541 | 22102 | 15 | 13 | 18 | c |
| 4 | 829 | 1146 | 426 | 706 | 121865 | -27628 | 73814 | 19 | 13 | 16 | c |
| 5 | 744 | 990 | 412 | 626 | 66863 | -8221 | 40471 | 16 | 13 | 18 | c |
| 6 | 632 | 1013 | 400 | 572 | 78684 | 11126 | 35048 | 18 | 13 | 18 | c |
| 7 | 396 | 908 | 308 | 446 | 121855 | 26387 | 62348 | 17 | 13 | 18 | c |
| 8 | 329 | 841 | 299 | 394 | 99101 | 15114 | 53526 | 18 | 14 | 18 | c |
| 9 | 397 | 832 | 346 | 427 | 56881 | -13907 | 23040 | 19 | 14 | 18 | c |
| 10 | 386 | 734 | 328 | 414 | 83479 | -13527 | 54195 | 19 | 14 | 18 | c |
| 11 | 296 | 625 | 282 | 427 | 55866 | -25752 | 15407 | 19 | 14 | 17 | c |
| 12 | 375 | 683 | 277 | 444 | 45508 | -16916 | 17064 | 18 | 15 | 18 | c |
| 13 | 523 | 882 | 347 | 522 | 192364 | 19368 | 136111 | 16 | 15 | 17 | c |
| 14 | 1088 | 1382 | 693 | 1189 | 343906 | -123716 | 237353 | 15 | 16 | 17 | c |
| 15 | 1356 | 1727 | 816 | 1421 | 882081 | -255672 | 323526 | 11 | 15 | 17 | c |
| 16 | 906 | 1232 | 569 | 806 | 678753 | -171119 | 250359 | 16 | 15 | -3 | c |
| 17 | 590 | 784 | 446 | 486 | 28337 | -56689 | 78347 | 18 | 14 | 17 | c |
| 18 | 677 | 996 | 467 | 745 | 150660 | -13750 | 85593 | 18 | 16 | 16 | c |

Generating labeled data using *PAWS*:
Once you have logged your data using the *PAWS* application, copy columns 1 – 10 in the file named "[UNIX TIME].csv" into a file named "carDistWeka.csv" and place it inside the "SEUS" folder. The rest of the steps are the same as the process of labeling data using *SEUS Features*.

Training the model:
Once the "carDistWeka.csv" file is generated and placed in the "SEUS" folder, open the *PAWS* application and tap the triple-dot icon in the upper right corner and select "Train Classifier". In the new menu, select the "Train Distance Classifier" button and wait until the application is finished training. When the application finishes training, you will receive a message that says that the distance classifier was successfully built. To use the newly trained model, restart the *PAWS* application.