Machine Learning for Automotive Radar Signal Processing

Yi Jin, Robert Prophet, Martin Vossiek

Friedrich-Alexander University Erlangen-Nuremberg, Institute of Microwaves and Photonics
Outline

1) Motivation
2) Databases
3) Examples
4) Summary
Motivation

1) FMCW spectrum

\[ R = f \cdot \frac{cT}{2B} \]

Range

- Analytical mapping is well suited!

2) Radar point cloud

Ghost detection: Yes/No?

- Analytical mapping is hard to find!
Databases

- ML methods identify automatically the searched mapping
- Prerequisites: **Suitable method** and **training data**

<table>
<thead>
<tr>
<th>Database</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Radar cube</td>
<td>• Conventional computer vision+classifier</td>
</tr>
<tr>
<td></td>
<td>• Simple CNN</td>
</tr>
<tr>
<td></td>
<td>• Object detection CNN</td>
</tr>
<tr>
<td>2) Radar point cloud</td>
<td>• Conventional classifier (SVM, random forest,…)</td>
</tr>
<tr>
<td></td>
<td>• Semantic segmentation CNN</td>
</tr>
<tr>
<td>3) Radar grid</td>
<td>• Simple CNN</td>
</tr>
<tr>
<td></td>
<td>• Semantic segmentation CNN</td>
</tr>
<tr>
<td></td>
<td>• Object detection CNN</td>
</tr>
</tbody>
</table>
Ex. 1: Ghost Detection Classification

- **Aim**: Distinction between moving real and ghost detections
- **Database**: Radar point cloud
- **Method**: Conventional classifier

Ground Truth

![Radar Point Cloud](image)

- Real Moving Det.
- Ghost Moving Det.
- Test Car
- Stationary Det. (accumulated) = Infrastructure
Ex. 1: Ghost Detection Classification

Point Cloud

Feature Set

Detection Class

Classifier

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$</td>
<td>Range</td>
</tr>
<tr>
<td>$F_2$</td>
<td>Radial velocity</td>
</tr>
<tr>
<td>$F_3$</td>
<td>Azimuth</td>
</tr>
<tr>
<td>$F_4$</td>
<td>Signal to noise ration</td>
</tr>
<tr>
<td>$F_5$</td>
<td>Occupancy grid cell value</td>
</tr>
<tr>
<td>$F_6$</td>
<td>Number of stationary neighbors</td>
</tr>
<tr>
<td>$F_7$</td>
<td>Velocity difference</td>
</tr>
<tr>
<td>$F_8$</td>
<td>Number of moving neighbors</td>
</tr>
</tbody>
</table>
### Ex. 1: Ghost Detection Classification

#### Comparison of Individual Classifiers

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
<th>Memory Requirement</th>
<th>Computation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear SVM</td>
<td>74.4%</td>
<td>4 MB</td>
<td>4.0 ms</td>
</tr>
<tr>
<td>Fine Gaussian SVM</td>
<td>88.2%</td>
<td>9 MB</td>
<td>7.6 ms</td>
</tr>
<tr>
<td>Coarse KNN</td>
<td>79.7%</td>
<td>4 MB</td>
<td>4.3 ms</td>
</tr>
<tr>
<td>Fine KNN</td>
<td>87.9%</td>
<td>4 MB</td>
<td>4.1 ms</td>
</tr>
<tr>
<td>Random Forest</td>
<td>91.2%</td>
<td>21 MB</td>
<td>44.9 ms</td>
</tr>
</tbody>
</table>

#### Calculation

**Actual class**

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true-positive $TP$</td>
<td>false-positive $FP$</td>
</tr>
<tr>
<td>2</td>
<td>false-negative $FN$</td>
<td>true-negative $TN$</td>
</tr>
</tbody>
</table>

**Accuracy**

$$\text{Accuracy} := \frac{TP+TN}{TP+TN+FN+FP}$$

#### Source


Ex. 1: Ghost Detection Classification

Video
Ex. 2: Moving Object Classification I

- **Aim**: Distinction between various moving objects
- **Database**: Radar Cube
- **Method**: Conventional Computer Vision + Classifier

Ground Truth and corresponding Range-Doppler-Matrix (RDM)
Ex. 2: Moving Object Classification I

Radar Cube → ROI Search → Feature Descriptor → Object Class

Ground Truth

RDM (Power)

RDM (Azimuth)

Filtered RDM

Feature Descr.

- HOG
- SURF
- GLOH
- ...
## Ex. 2: Moving Object Classification I

Comparison of Individual Classifiers for 2 Classes (Pedestrian and Other)

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy ((B_1 = 0.4 \text{ GHz}))</th>
<th>Accuracy ((B_2 = 1.6 \text{ GHz}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDM filter</td>
<td>No RDM filter</td>
<td>RDM filter</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>69.1%</td>
<td>86.4%</td>
</tr>
<tr>
<td>Quadratic SVM</td>
<td>69.0%</td>
<td>87.9%</td>
</tr>
<tr>
<td>Cubic SVM</td>
<td>67.7%</td>
<td>87.4%</td>
</tr>
<tr>
<td>Medium Gaussian SVM</td>
<td>69.5%</td>
<td>87.8%</td>
</tr>
<tr>
<td>Coarse Gaussian SVM</td>
<td>67.3%</td>
<td>85.8%</td>
</tr>
<tr>
<td>Quadratic SVM</td>
<td>66.8%</td>
<td>82.3%</td>
</tr>
<tr>
<td>Cubic SVM</td>
<td>67.0%</td>
<td>81.2%</td>
</tr>
<tr>
<td>Medium Gaussian SVM</td>
<td>68.5%</td>
<td>81.3%</td>
</tr>
<tr>
<td>Coarse Gaussian SVM</td>
<td>66.9%</td>
<td>80.4%</td>
</tr>
</tbody>
</table>

### Source: R. Prophet et al., "Image-Based Pedestrian Classification for 79 GHz Automotive Radar", 2018.
Ex. 2: Moving Object Classification I

Video
Ex. 3: Moving Object Classification II

- **Database**: Radar Point Cloud
- **Method**: Semantic Segm. CNN

### Network Structure

- **MSG 1**
  - $N_{\text{sample}} = 1024$
  - $r_1 = 1m$, $N_{\text{neigh}} = 8$
  - $r_2 = 3m$, $N_{\text{neigh}} = 32$
- **MSG 2**
  - $N_{\text{sample}} = 512$
  - $r_1 = 2m$, $N_{\text{neigh}} = 8$
  - $r_2 = 4m$, $N_{\text{neigh}} = 32$
- **MSG 3**
  - $N_{\text{sample}} = 256$
  - $r_1 = 3m$, $N_{\text{neigh}} = 16$
  - $r_2 = 6m$, $N_{\text{neigh}} = 32$
- **FP 1**
  - Kernel sizes: 256, 256
- **FP 2**
  - Kernel sizes: 128, 128
- **FP 3**
  - Kernel sizes: 128, 128, 128

### Operations
- **ID Conv.**
  - Kernel size: 256
- **Dropout**
  - Keep probability: 50%
- **ID Conv.**
  - Kernel size: 128
- **Dropout**
  - Keep probability: 50%
- **ID Conv.**
  - Kernel size: 6

### Softmax
Ex. 3: Moving Object Classification II

Input Features

\[ F_0 = (x, y, v_r, \sigma) \]
\[ F_1 = (x, y, v_r) \]
\[ F_2 = (x, y, \sigma) \]
\[ F_3 = (x, y) \]

\[ F_0 \text{ Average} = 0.7425 \]
\[ F_1 \text{ Average} = 0.7303 \]
\[ F_2 \text{ Average} = 0.6492 \]
\[ F_3 \text{ Average} = 0.5939 \]

**F1 Scores Object-Wise**

**Source:**

Precision \( P := \frac{TP}{TP+FP} \), Recall \( R := \frac{TP}{TP+FN} \), \( F1 := 2 \frac{PR}{P+R} \)
Ex. 4: Moving Object Classification III

- **Database**: Radar Cube & Point Cloud
- **Method**: Simple CNN

Algorithm Structure:

1. **Pre-processing**
   - Radar cube (low-level)
   - Radar targets (target-level)
   - Odometry

2. **Map radar targets to cube and crop surrounding block**

3. **Ego-motion compensation**

4. **RTCnet**
   - I. Conv 6x3x3x3, MP 6x2x2x1, Conv 25x3x3x3, MP 25x2x2x1
   - II. Conv 16x1x1x7, MP 16x1x1x3, Conv 32x1x1x7, MP 32x1x1x3
   - III. FC 128, FC 128

5. **Post-processing**
   - Ensemble classification
   - Object clustering
   - Classified object list
## Ex. 4: Moving Object Classification III

### $F1$ Scores Object-Wise

<table>
<thead>
<tr>
<th>Method</th>
<th>Ped.</th>
<th>Cyclist</th>
<th>Car</th>
<th>Average</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prophet</td>
<td>0.48</td>
<td>0.50</td>
<td>0.23</td>
<td>0.40</td>
<td>Cube</td>
</tr>
<tr>
<td>Schumann</td>
<td>0.54</td>
<td><strong>0.60</strong></td>
<td>0.31</td>
<td>0.48</td>
<td>Point Cloud</td>
</tr>
<tr>
<td>RTCnet</td>
<td><strong>0.61</strong></td>
<td>0.59</td>
<td><strong>0.47</strong></td>
<td><strong>0.56</strong></td>
<td>Both</td>
</tr>
</tbody>
</table>

Precision $P := \frac{TP}{TP+FP}$, Recall $R := \frac{TP}{TP+FN}$, $F1 := 2 \frac{PR}{P+R}$

### Challenging Cases for Clustering

- **Schumann**
- **RTCnet**

**Source:** A. Palffy et al., "CNN Based Road User Detection Using the 3D Radar Cube", 2020.
Ex. 5: Infrastructure Classification

- **Aim:** Distinction between stationary objects
- **Database:** Radar grid
- **Method:** Semantic Segm. CNN

![Diagram of infrastructure classification process]

1. **Radar Grid**
2. **CNN**
3. **Segmented Grid**

**Ground Truth**
- Background
- Passable Surface
- Car
- Test Car
- Barrier

![2D Occupancy Grid](Image)
Ex. 5: Infrastructure Classification

- Point Cloud
- 3D RCS Grid
- 3D Occupancy Grid
Ex. 5: Infrastructure Classification

### Network Structure

![Network Structure Diagram]

### Results

<table>
<thead>
<tr>
<th>Network</th>
<th>Dataset</th>
<th>Recall</th>
<th>Intersection over Union (IoU)</th>
<th>Boundary Contour Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SegNet</td>
<td>Real-World</td>
<td>90.4%</td>
<td>77.3%</td>
<td>81.0%</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
<td>95.6%</td>
<td>83.8%</td>
<td>96.2%</td>
</tr>
<tr>
<td>FCN-8s</td>
<td>Real-World</td>
<td>90.1%</td>
<td>65.3%</td>
<td>57.7%</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
<td>95.9%</td>
<td>68.9%</td>
<td>87.8%</td>
</tr>
<tr>
<td>U-Net</td>
<td>Real-World</td>
<td>46.2%</td>
<td>31.1%</td>
<td>32.3%</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
<td>59.8%</td>
<td>35.2%</td>
<td>45.4%</td>
</tr>
</tbody>
</table>

Recall $R := \frac{TP}{TP+FN}$

IoU $:= \frac{TP}{TP+FN+FP}$

Ex. 5: Infrastructure Classification

Video
Ex. 6: Road Course Estimation

- **Aim:** Prediction of the road course
- **Database:** Radar grid
- **Method:** Simple CNN

![Radar Point Cloud with Road Course](image)

- **Ground Truth:**
- **Radar Grid**
- **CNN**
- **Road Course Coefficients**
- **Test Car**
- **Stationary Det.** (accumulated)
- **Road Course**
- **Infrastructure**
Ex. 6: Road Course Estimation

- Road course as **clothoid**: 
  \[ y(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0 \]

- Ground truth positions from DGPS
Ex. 6: Road Course Estimation

<table>
<thead>
<tr>
<th>CNN</th>
<th>ARCSGM</th>
<th>SRCSSGM</th>
<th>OGM</th>
<th>SGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>70.96</td>
<td>99.38</td>
<td>62.70</td>
<td>74.16</td>
</tr>
<tr>
<td>DenseNet-201</td>
<td>68.63</td>
<td>103.10</td>
<td>58.37</td>
<td>62.99</td>
</tr>
<tr>
<td>NasNet</td>
<td>64.47</td>
<td>97.80</td>
<td>60.32</td>
<td>68.04</td>
</tr>
<tr>
<td>ShuffleNet</td>
<td>114.11</td>
<td>150.33</td>
<td>109.64</td>
<td>111.91</td>
</tr>
</tbody>
</table>

Results: RMSE in cm at +-75 m depending on CNN and Gridmap

Comparisons between Ground Truth (red) and Prediction (green)

Ex. 6: Road Course Estimation

Video
Summary

• Machine Learning is very helpful
  ➢ Almost every problem can be solved!
  ➢ But not always useful (range calculation, ...)

• Which **database**? Which basic **conditions** (computation time, memory, ...)?
  ➢ Choose an appropriate **method**!

• Need for **training data**!
  ➢ Currently only a few and small public datasets