Domain Reconstruction for UWB Car Key Localization Using Generative Adversarial Networks

Aleksei Kuvshinov,[1] Daniel Knobloch,[2] Daniel Külzer,[2] Elen Vardanyan,[3] Stephan Günnemann[1, 4]

[1]Technical University of Munich, Department of Informatics
[2]BMW Group, Hufelandstraße 1, 80937 Munich, Germany
[3]Technical University of Munich, Department of Mathematics
[4]Technical University of Munich, Munich Data Science Institute
kuvshino@in.tum.de, {daniel.kd.knobloch, daniel.kuelzer}@bmw.de, elen.vardanyan@tum.de, guennemann@in.tum.de

Abstract

We consider the car key localization task using ultra-wideband (UWB) signal measurements. Given labeled data for a certain car, we train a deep classifier to make the prediction about the new points. However, due to the differences in car models and possible environmental effects that might alter the signal propagation, data collection requires considerable effort for each car. In particular, we consider a situation where the data for the new car is collected only in one environment, so we have to utilize the measurements in other environments from a different car. We propose a framework based on generative adversarial networks (GANs) to generate missing parts of the data and train the classifier on it, mitigating the necessity to collect the real data. We show that the model trained on the synthetic data performs better than the baseline trained on the collected measurements only. Furthermore, our model closes the gap to the level of performance achieved when we would have the information about the new car in multiple environments by $35\,\%$.

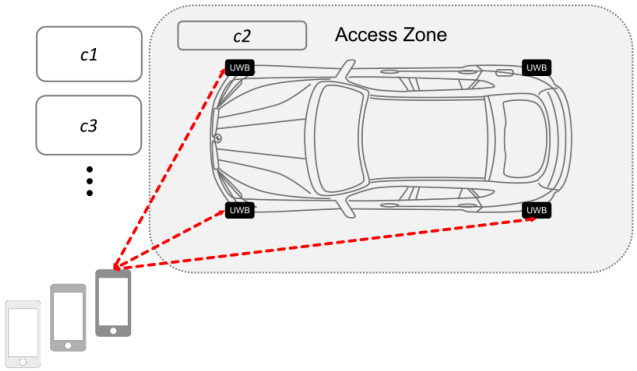


Figure 1: Illustration of the UWB car access use case with an exemplary number of UWB antennas ($N = 4$) and exemplary clusters (*c1*, *c2*, *c3*).

Key Localization Task

Following the trend of mobile payment solutions, a new and emerging use case is formed by using a smart device (e.g., cell phone) as a car key. Starting from recently introduced near-field communication (NFC)-based car access, the next step constitutes hands-free or passive entry to the car via the personal smart device. In other words, this means that the vehicle owner can keep his smart device in his pocket, and the car unlocks automatically upon approach. The 'Digital Car Key' is standardized (Car Connectivity Consortium (CCC) 2021), and its commercial launch was recently announced (BMW 2021). Its underlying technology relies on two different wireless technologies, Bluetooth and ultra-wideband (UWB). Moreover, a back end is required for owner pairing (key setup) or friend sharing (key lending), while the car access functionality works independently of an available internet connection.

Upon approaching the vehicle, a Bluetooth connection between the smart device and car is established. Security protocols are exchanged, and the communication partners agree upon UWB connection parameters. Next, said UWB connection is established to allow secure ranging of the smart device (see Figure 1). For that, the vehicle is equipped with N UWB antennas or anchors. The UWB technology enables precise localization thanks to its pulse duration of $2\,\mathrm{ns}$. Moreover, the available time of flight (ToF) data prevents relay attacks, fulfilling highest security requirements. With the help of the UWB car key (CCC Digital Key) localization, the vehicle locks or unlocks and targeted lights might welcome the vehicle owner at a few meters distance.

However, the UWB-based key localization task requires complex and costly vehicle calibration due to its frequency range of $3\,\mathrm{GHz}$ to $10\,\mathrm{GHz}$ and resulting reflections or shielding for most materials. The digital car key has to work in many environments (e.g., free-field or underground parking) that feature individual and different propagation characteristics. For solid system performance, all possible smart device locations in and outside the vehicle have to be measured in a total of 4 to 5 environments. On top of that, each new car model requires a new measurement campaign as a vehicle's form factor, interior, and materials usually differ. Even model variants might require special treatment due to glass roofs or engine types.

Data The localization decision is based on the anchors' received signal strength and ToF measurements. Considering vehicle A parked in environment 1, our input data to the

classifier is $x_{\mathrm{A1}} \in \mathbb{R}^{2N}$. Our classifier performs the 'simple' decision of the key being outside or inside the vehicle. For that, a total of 4000–6000 measurements for both cases (forming a balanced data set) and per environment are used. Additionally, the measurements are labeled by one of the 17 regions in or around the car (e.g., inside rear, inside front) where they were taken. We refer to these location labels as clusters (see Figure 1 for examples showing clusters *c1*, *c2*, and *c3*).

Challenges The main challenge from an automotive manufacturer's perspective is the measurement complexity. Each model requires measurements in multiple environments, constituting a lengthy, costly, and error-prone task. The novel idea of this work is to use artificial intelligence (AI) to replace part of the measurements. Hence, regarding the emerging AI application, the challenge for the deep learning framework is to transform available measurement data to a completely missing domain.

For the rest of the paper, we assume to have one car used for extensive data collection (the default one) in multiple environments and one new car (the target one) where only the measurements from a single environment are available to train the model. The task is to train a classifier for the target car data under the described conditions.

Contributions We propose a framework based on generative adversarial networks (GANs) that allows for the reconstruction of a completely missing domain by learning a transition mapping from the available data sets. This enables the UWB key localization application rollout to an original equipment manufacturer (OEM)'s model range at significantly reduced cost. Using the generated data, we train a classifier for a new car model to decide whether the smart device is inside or outside the vehicle. Our experiments show a superior performance of the proposed framework compared to the baseline trained on the available data only.

Missing Domain Reconstruction

For the following sections, we denote the default and target cars as A and B, as well as the default and target environment as environments 1 and 2. The corresponding domains are called A1, A2, B1, and the missing one B2. Note that we have multiple target environments and train the described models for each of them separately.

To be able to deploy the classifier and have good performance in the real-life scenario, we have to train it using the knowledge from as much available data as possible. The simplest approach would be to train the classifier on the available data from the target car, that is, its default environment B1. However, as we show in the experiments, utilizing the data A1 and A2 from the default car significantly improves upon that baseline.

The idea is to reconstruct B2 and use the new data to train the classifier. Note that we have the data either from the same car but the wrong environment (B1) or the same environment but the different car (A2). Therefore, our ultimate goal is to construct a mapping that, given the input points from A2 and B1, generates data points in B2. That means the task is to learn a transfer function that simulates measurements in a different environment and the same car (B1 $\rightarrow$ B2 transfer, G_{12} in Figure 2). In addition, we learn a mapping that generates the data points in the same environment but from a car with a different geometry (A2 $\rightarrow$ B2 transfer, G_{AB} in Figure 2).

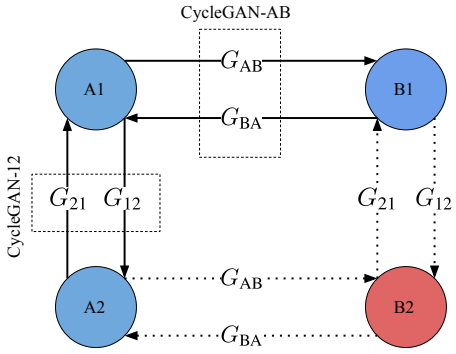


Figure 2: Four domains (depicted as circles) and four generators (arrows). The CycleGAN-12 unit (with the generators G_{12} and G_{21} and discriminators D_{12} and D_{21}) operates between the environments while the CycleGAN-AB unit (G_{AB}, G_{BA}, D_{AB} and D_{BA}) maps data between the cars. Dotted arrows indicate the difference in the loss function when the generators are trained for the B1–B2 and A2–B2 pairs (see Section Missing Domain Reconstruction for details).

CycleGAN Framework Unlike standard generative models (e.g., standard GAN proposed by Goodfellow et al.), during the training phase, we do not have any data samples from B2, the data distribution that we have to approximate. Instead, we have a sample from the source domain that we have to convert into a sample from a new domain. Thus, the generation procedure has to be conditioned on that input. Still, since the target domain B2 is unavailable during training, it is impossible to train a conditional generator directly on B1 and B2.

For that purpose, we adapted the CycleGAN model developed to solve the image-to-image translation task by Zhu et al. (2017). One CycleGAN unit consists of two GANs (two generator-discriminator pairs, for the forward and backward direction) coupled with each other by the so-called cycle and identity losses. Having two dimensions in our set of domains, cars, and environments, we learn two CycleGAN units for each of the mappings described above. The full setup is shown in Figure 2. Once trained on an available pair of domains, we use the generators G_{12} and G_{AB} to correspondingly reconstruct domain B2 from B1 and A2. We show the hyperparameters used to train CycleGANs in Section Experiments.

New losses To summarize, we have eight networks to train: a pair of a generator and discriminator for each of the four directions (A$\rightarrow$ B, B$\rightarrow$ A, 1$\rightarrow$ 2, 2$\rightarrow$ 1, see Figure 2). Note that while B2 is not available, we still have to ensure that the model is aware of its presence. Otherwise, assume one would train CycleGAN-12 consisting of G_{12} and G_{21}

separately from CycleGAN-AB. This is done by using A1 as the source domain and A2 as the target domain. Then the network would learn the mapping on the data from car A and be applicable to its specific geometry only. Instead, we are aiming at learning a transfer function applicable to the target car B as well.

To make it possible, in addition to the standard CycleGAN losses (marked by '–' below), we introduce the new ones (marked by '*'). The full list of the loss terms we use for training CycleGANs is shown below.

- Adversarial and identity losses, CycleGAN-12.
  - [LG1.1] G_{12} loss: $(D_{12}(G_{12}(x_{\mathrm{A1}}))-1)^2$
  - [LD1.1] D_{12} loss: $(D_{12}(x_{\mathrm{A2}})-1)^2+D_{12}(G_{12}(x_{\mathrm{A1}}))^2$
  - [LG1.2] G_{21} loss: $(D_{21}(G_{21}(x_{\mathrm{A2}}))-1)^2$
  - [LD1.2] D_{21} loss: $(D_{21}(x_{\mathrm{A1}})-1)^2+D_{21}(G_{21}(x_{\mathrm{A2}}))^2$
  - [LI1] $\|G_{12}(x_{\mathrm{A2}})-x_{\mathrm{A2}}\|_1+\|G_{21}(x_{\mathrm{A1}})-x_{\mathrm{A1}}\|_1$
- The adversarial and identity losses associated with CycleGAN-AB ([LG2.x], [LD2.x] and [LI2]) are of the same form as the losses above, but with the second domain being B1 instead of A2 (the first one is A1).
- Cycle losses associated with CycleGAN-12.
  - [LC1.1] A1 $\to$ A2 $\to$ A1: $\|x_{\mathrm{A1}}-G_{21}(G_{12}(x_{\mathrm{A1}}))\|_1$
  - [LC1.2] A2 $\to$ A1 $\to$ A2: $\|x_{\mathrm{A2}}-G_{12}(G_{21}(x_{\mathrm{A2}}))\|_1$
  - * [LC1.3] B1 $\to$ B2 $\to$ B1: $\|x_{\mathrm{B1}}-G_{21}(G_{12}(x_{\mathrm{B1}}))\|_1$
  - * no B2 $\to$ B1 $\to$ B2 cycle since x_{B2} is missing.
- Cycle losses associated with CycleGAN-AB.
  - [LC2.1] A1 $\to$ B1 $\to$ A1: $\|x_{\mathrm{A1}}-G_{\mathrm{BA}}(G_{\mathrm{AB}}(x_{\mathrm{A1}}))\|_1$
  - [LC2.2] B1 $\to$ A1 $\to$ B1: $\|x_{\mathrm{B1}}-G_{\mathrm{AB}}(G_{\mathrm{BA}}(x_{\mathrm{B1}}))\|_1$
  - * [LC2.3] A2 $\to$ B2 $\to$ A2: $\|x_{\mathrm{A2}}-G_{\mathrm{BA}}(G_{\mathrm{AB}}(x_{\mathrm{A2}}))\|_1$
  - * no B2 $\to$ A2 $\to$ B2 cycle since x_{B2} is missing.
- Cycle losses associated with both CycleGANs.
  - * [LC3.1] A1 $\to$ B1 $\to$ B2 $\to$ A2 $\to$ A1 cycle: $\|x_{\mathrm{A1}}-G_{21}(G_{\mathrm{BA}}(G_{12}(G_{\mathrm{AB}}(x_{\mathrm{A1}}))))\|_1$
  - * [LC3.2] A1 $\to$ A2 $\to$ $B2$ $\to$ B1 $\to$ A1 cycle: $\|x_{\mathrm{A1}}-G_{\mathrm{BA}}(G_{21}(G_{\mathrm{AB}}(G_{12}(x_{\mathrm{A1}}))))\|_1$
  - * [LC3.3] A1 $\to$ B1 $\to$ B2 vs. A1 $\to$ A2 $\to$ B2: $\|G_{12}(G_{\mathrm{AB}}(x_{\mathrm{A1}}))-G_{\mathrm{AB}}(G_{12}(x_{\mathrm{A1}}))\|_1$

Note that almost all new losses are of the same form as the cycle loss from the standard CycleGAN and compare a given ground truth sample to its reconstruction. However, the reconstruction is now generated using the synthetic points from B2. This causes CycleGANs to learn meaningful mappings not only between the 'simple' domains A1, A2, and B1 but for B1 $\rightleftarrows$ B2 and A2 $\rightleftarrows$ B2 as well.

Classification After the training of CycleGANs is completed, we create new data $x_{\mathrm{B2}}=G_{12}(x_{\mathrm{B1}})$ from B1 samples as well as $x_{\mathrm{B2}}=G_{\mathrm{AB}}(x_{\mathrm{A2}})$ from A2 samples. In both cases, we take the corresponding label and assign the same one to the new sample in B2. Once we have the synthetic data substituting the real points in B2, we train the classifier on the union of B1 and synthetic B2. This way, it incorporates more knowledge about the geometry of car B, and we use as much information as possible about the target environment as well.

Experiments

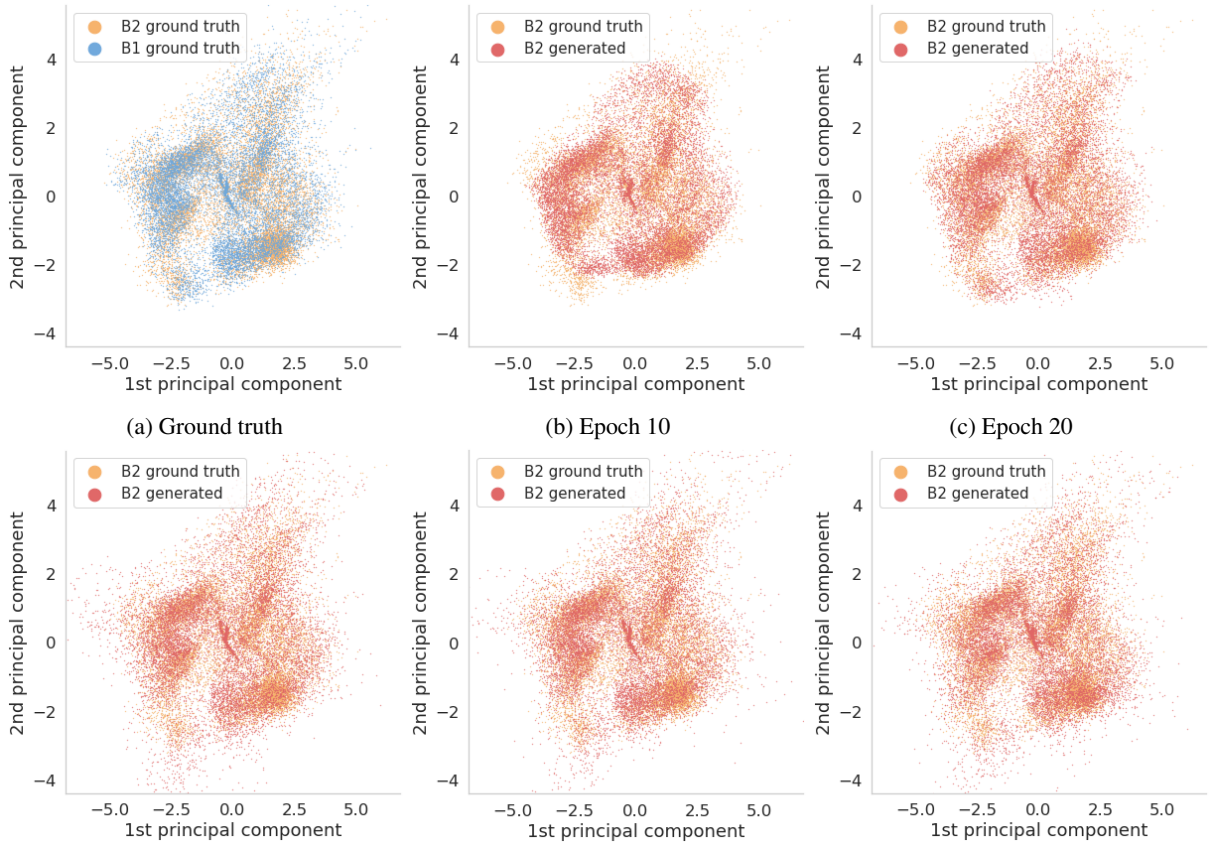
We refer to the classifier trained as described above as GEN. In addition to it, we train two baseline models. The first is called SRC and trained on a union of B1 and A2, the ground truth (source) data available without training CycleGANs. To have an upper bound that one achieves having the full information about the target car, we train a model on B1 and the true B2 (target data, collected for testing purposes) and call it TGT.

For our experiments, we use data collected for two cars in a different number of environments: 8 and 14. We treat the car with less available environments as the default car A and use the underground parking environment as the default environment (available for both cars). The remaining seven environments of car A are the target environments A2. For each of these, we train the CycleGAN models as described in Section Missing Domain Reconstruction separately. Note that for the remaining six environments of car B, we do not generate any synthetic B2 data. Therefore, we separately evaluate models' performance on the environments available for both cars and the completely new environments from car B. The former explains how well the generated data approximates the corresponding true car B data from the common environments. Whereas the latter indicates whether the models SRC and GEN are able to generalize to completely new environments, which is a highly desirable property when deployed.

Additionally, since we have the information about the device's location, we want to train CycleGANs tailored to the specific clusters. It is helpful because the transition mappings should not be the same for, e.g., inside and outside regions. To achieve that, we first train the CycleGANs on the data from all clusters. Then, starting from the latest model checkpoint, we continue training a separate set of generators and discriminators on each cluster individually.

Setup: CycleGAN Training The architecture of the generators and discriminators is based correspondingly on the ResNet and PatchGAN architectures adapted by Zhu et al. (2017) (see the ResNet paper by He et al. (2016) and Isola et al. (2017) for PatchGAN). However, to enable its usage for the vector data, we add the first dense linear layer to all networks with the output of shape $3\times64\times64$. Additionally, for the generators, we add the corresponding final dense linear layer with the vector-valued output. For the main part of the generators, we use two downsampling blocks consisting of a convolutional, batch normalizing, and activation layer, followed by six ResNet and two upsampling blocks (with a similar structure as the downsampling blocks). Each generator has $580\,000$ parameters. For more information, see the CycleGAN paper and PyTorch implementation. For the discriminators, we use PatchGAN with three convolutional blocks from the CycleGAN implementation. Each discriminator has $335\,000$ parameters. In all eight networks, we have 16 filters in the last convolutional layer.

Each of the two CycleGANs is trained with their adversarial losses ([LDx.x] and [LGx.x] above) as described by Zhu et al. (2017), where we use the MSE GAN loss ('lsgan' option, introduced by Mao et al. (2017)). Furthermore, in addi-



(a) Ground truth (b) Epoch 10 (c) Epoch 20

(d) Epoch 40 (e) Epoch 200 (f) Epoch 450

Figure 3: Two-dimensional principal component analysis (PCA)-visualization of the ground truth B2 data in comparison with B1 (showing the difference between the environments) and generated B2 at different epochs during the training. The different between B1- and B2-measurements in Figure 3a is reduced by the generated samples after the CycleGAN training is completed, see Figure 3f.

tion to the cycle losses that improve the quality of the generated samples, we use the identity losses from the CycleGAN framework. The idea was initially proposed by Taigman, Polyak, and Wolf (2017) and applied by Zhu et al. (2017) on some of the use cases to stabilize the training and further improve the results. As we observe a similar effect in our case, the training is more stable, and we keep the identity loss for a certain number of epochs.

The weights are distributed as follows: discriminator and generator losses ([LGx.x] and [LDx.x]) – 10, cycle losses associated with only one CycleGAN ([LC1.x] and [LC2.x]) – 10, cycle losses coupling both CycleGANs ([LC3.x]) – 1, identity losses ([LIx]) – 5 for the first 200 epochs then 0. We use a starting learning rate of 0.0002 and keep it for 50 epochs. Afterward, it linearly decays for 200 epochs until it reaches 1% of the initial value and stays at that level for another 200 epochs. Overall, the training lasts 450 epochs.

Figure 3 shows the two-dimensional projection of the standardized true and generated data throughout the learning process. We use the PCA applied on a pair of datasets and take the environment with five people inside the car in the underground parking as the target one. In all six images, the first two principal components of the target environment from car B (B2 domain) are shown in orange. In Figure 3a, we compare these points to the default environment from car B (B1, depicted in blue). The overall coverage of the two principal components shows the similarity between both environments. Note that we use the points from B1 as the input for the generator, and it motivates the usage of the identity loss in our case. It prevents the generator from outputting points that are too different from B1 that already provides a good starting point for an approximation of true B2. Nevertheless, some gaps between these distributions are visible, and we are able to improve by training the CycleGANs. Figures 3b–3f show the true and generated B2 (red points) at a certain epoch during the training of CycleGANs. We see that the difference between both datasets becomes smaller during the training, and the approximation of the true B2 domain improves.

Setup: Classifier Training Since the main focus of this work lies in the domain recovery task, we use a simple dense classifier that already yields reasonable performance to save

	GEN		*TGT*		*SRC*	
	mean	*std*	*mean*	*std*	*mean*	*std*
all	0.913	0.001	0.954	0.001	0.891	0.001
known	0.933	0.001	0.963	0.002	0.915	0.001
new	0.887	0.002	0.943	0.002	0.870	0.002
in	0.882	0.001	0.961	0.007	0.855	0.001
out	0.942	0.001	0.947	0.006	0.934	0.001
c1	0.860	0.008	0.853	0.036	0.876	0.009
c2	0.997	0.001	0.994	0.002	0.992	0.002
c3	0.995	0.001	0.993	0.002	0.994	0.001
c4	0.964	0.003	0.966	0.003	0.961	0.003
c5	0.986	0.001	0.983	0.002	0.985	0.001
c6	0.998	0.001	0.998	0.002	0.999	0.001
c7	0.804	0.012	0.853	0.020	0.718	0.010
c8	0.812	0.003	0.899	0.013	0.748	0.007
c9	0.996	0.001	0.997	0.002	0.997	0.002
c10	0.986	0.001	0.997	0.001	0.987	0.002
c11	0.993	0.002	0.998	0.001	0.990	0.002
c12	0.997	0.001	0.993	0.003	0.998	0.001
c13	0.996	0.001	0.992	0.003	0.991	0.002
c14	0.852	0.007	0.782	0.021	0.900	0.005
c15	0.741	0.005	0.973	0.006	0.686	0.004
c16	0.642	0.008	0.888	0.023	0.531	0.005
c17	0.983	0.002	0.951	0.011	0.974	0.003

Table 1: Performance of the classifiers GEN, SRC and TGT on the different parts of the test data: *all* – complete test set, *known* – common environments used for training CycleGANs, *new* – new environments, unseen during CycleGAN training, *in* – points inside the car, *out* – points outside the car, *cx* – points from the cluster *x*.

the training time in that stage. To solve the classification task, we use a dense network with ReLU activations containing two hidden layers. It has 16 input neurons ($N = 8$ anchors), 15 neurons in the first hidden layer, and 5 neurons in the second one. We train it for 70 epochs with a learning rate of 0.001 using Adam optimizer by Kingma and Ba (2015). We use Adam's PyTorch implementation with the standard parameters.

After the CycleGANs are trained, the points from car B (both B1 and B2) are split into $70\,\%$ train and $30\,\%$ test data. Model TGT is then trained on the train set, and all models are evaluated on the test set without the default environment B1 since GEN and SRC used it during training. Thus, the test set consists of $30\,\%$ of the data from the B2 environments. We do ten runs with random data splits to ensure the statistical significance of the results. In Table 1, we show the mean correct classification rate as well as its standard deviation over the ten runs of the three competitors on the test set.

Results In Table 1, we report the performance of the three models described above. Each column contains the correct classification rate of the corresponding model on a specific part of the test data. We report the performance on the full test set (row *all*), on the environments used to train CycleGANs and the completely new environments from the target car (*known* and *new*), on the points inside and outside of the car (*in* and *out*) as well as on each of the clusters individually (*cx*, all environments).

Classifier GEN trained on the generated B2 data outperforms SRC baseline by over two percentage points on the whole data set and both environment groups. Having $91.3\,\%$ correctly predicted points against $95.4\,\%$ for TGT (with the absolute knowledge about car B), we close the gap between the performance of the TGT and SRC models. Note that initially, only the default environment from that car was available for training GEN out of 14 used to train TGT. Furthermore, for *known* environments, the correct classification rate of GEN almost reaches TGT with $93.3\,\%$ against $96.3\,\%$ correspondingly. It illustrates that the synthetic B2 points approximate the ground truth well enough to reach that high level of performance on the reconstructed environments. On the other hand, while the test error is larger on *new* environments for all models, GEN is still outperforming SRC, showing better capability to generalize to the unseen environments.

For the most individual clusters with a significant difference between the error of TGT and SRC, we improve upon the error of SRC. For the outside regions close to the windows (cluster *c7* – roof and *c8* – windows), the classification task is more challenging, resulting in worse performance for all competitors (GEN, TGT, and SRC) than the error on the full dataset. Another region with performance below average is in and around the open trunk (clusters *c14* and *c16*). In addition to the same problem as for *c7* and *c8*, we observe a higher error inside the open trunk (cluster *c16*) for SRC compared to TGT. This phenomenon arises because several data points inside the open trunk were recognized as outside due to car A's influence on the classifier.

Related Work

Note that our task falls into the broad category of transfer learning problems solved by the machine learning methods that utilize the knowledge from a known domain to achieve better performance in a new one. Here we mention the most relevant approaches that are related to our use case and the chosen approach.

Conditional GANs There is a family of approaches that use conditional GANs (Goodfellow et al. 2014; Zhao, Mathieu, and LeCun 2017; Mirza and Osindero 2014; Mao et al. 2017) to solve the image-to-image translation problem (e.g., recoloring or changing the style of a picture). Some of these approaches require pairs of input-output samples (Isola et al. 2017; Karacan et al. 2016; Sangkloy et al. 2017) and other work with unpaired data sets as in our use case (Aytar et al. 2017; Bousmalis et al. 2017; Liu and Tuzel 2016; Shrivastava et al. 2017; Taigman, Polyak, and Wolf 2017; Zhu et al. 2017). CyCADA, presented by Hoffman et al. (2018), builds upon the idea of the cycle consistent sample generation of CycleGAN but uses it in a context of a downstream task. While currently, we keep the training of the models for the domain recovery and classification task separated, using CyCADA or other approaches based on conditional GANs that are aware of the downstream problem instead of CycleGAN is a promising direction for further improvement.

Multi-Domain Setting While the approaches above operate in a situation with one source and one target domain, there is a line of work that deals with more than two domains and the (bi-directional) transitions across them. StarGAN (Choi et al. 2018), StarGANv2 (Choi et al. 2020), and RadialGAN (Yoon, Jordon, and Schaar 2018) are the recent approaches that learn a single generator that, given input from one of the domains, is capable of producing images from a different one. A typical use case is transitioning between facial expressions of the same person. Finally, CollaGAN (Lee et al. 2019) was proposed specifically for missing image data imputation and trains a generator that can estimate the data in any domain utilizing the other clean data set.

While these models work in a similar scenario, two major differences do not allow us to use them directly in our use case, as shown in Figure 2. First, in our case, one of the domains is completely missing, and its samples cannot be used for training. Note that the approaches mentioned above require data from all relevant domains for their training. For example, if a certain facial expression is not present in the training data set, it cannot be learned and afterward generated by the approaches like StarGAN. Second, the domains in our case have a certain relation to each other (either the same environment or the same car). This structure of the data set is different from a collection of subsets with a changing semantic feature. It is exactly this structure that allows us the training of car-to-car and environment-to-environment translation mappings as described in Section Missing Domain Reconstruction.

UWB Key Localization The proposed framework serves as a means to reduce cost in the rollout of using a smart device as a car key. Besides apparent and described benefits, the UWB key localization application poses several challenges (Knobloch 2017). In short, the major problem is that the signal might be blocked, and a UWB antenna only receives multiple reflections, i.e., non-line of sight (NLOS) propagation paths or no signal at all. Classifying a received signal as line of sight (LOS) or NLOS propagation would considerably improve UWB localization accuracy. However, the task is challenging and computationally complex, and a first approach is described in (Jiang et al. 2020). While our feature set has proven robust in practice, more complex features for multi-path environments, and without direct NLOS determination, are listed by Park et al. (2021). For a broader overview of UWB-based (indoor) localization, we refer the reader to the work by Shi and Ming (2016).

Conclusion

In this work, we present the UWB car key localization problem needed to be solved in order to enable a reliable way of using a smart device as a car key. We discuss the challenges related to a large number of application scenarios and the need for extensive, costly data collection.

To overcome these issues, we propose an AI-based solution to replace most of the data required to train the model for a new car with the generated samples. The novel approach allows training a simple classifier that consistently outperforms the baseline trained on the available measurements only. Furthermore, it nearly reaches the performance of the model that had the full information about the environments from the test set of the new car. In summary, with the proposed approach, we contribute to the future deployment of the UWB car key technology and its efficient performance.

Future Work Finally, we mention two directions of possible improvements.

The first one is related to the scalability and training time of the described approach. Currently, we train two CycleGAN units for each environment-cluster combination. Even when it is not a concern now, since CycleGAN training and data generation is done a priori, we consider training a joint framework for multiple environments with fewer generators and discriminators as a possible way to make the framework more efficient.

The second idea for future work goes into the direction of combining both stages: CycleGAN and classifier training. Currently, the information about the labels is used for training CycleGANs only indirectly by training the models on individual clusters (each of them is either an inside or outside region). The idea of using the labels and training all models jointly similar to CyCADA's approach is an intriguing direction for future research.

References

Aytar, Y.; Castrejon, L.; Vondrick, C.; Pirsiavash, H.; and Torralba, A. 2017. Cross-modal Scene Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(10): 2303–2314.

BMW. 2021. BMW Announces BMW Digital Key Plus with Ultra-Wideband Technology Coming to the BMW iX. https://www.press.bmwgroup.com/global/article/detail/T0324128EN/bmw-announces-bmw-digital-key-plus-with-ultra-wideband-technology-coming-to-the-bmw-ix?language=en. Accessed: 2021-09-03.

Bousmalis, K.; Silberman, N.; Dohan, D.; Erhan, D.; and Krishnan, D. 2017. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3722–3731.

Car Connectivity Consortium (CCC). 2021. Digital Key Release 3.0 (CCC-TS-101), Ver. 1.0.0.

Choi, Y.; Choi, M.; Kim, M.; Ha, J.-W.; Kim, S.; and Choo, J. 2018. Stargan: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 8789–8797.

Choi, Y.; Uh, Y.; Yoo, J.; and Ha, J.-W. 2020. Stargan v2: Diverse Image Synthesis for Multiple Domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8188–8197.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 27.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.-Y.; Isola, P.; Saenko, K.; Efros, A.; and Darrell, T. 2018. Cycada: Cycle-Consistent Adversarial Domain Adaptation. In *International Conference on Machine Learning (ICML)*, 1989–1998. PMLR.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1125–1134.

Jiang, C.; Shen, J.; Chen, S.; Chen, Y.; Liu, D.; and Bo, Y. 2020. UWB NLOS/LOS Classification Using Deep Learning Method. *IEEE Communications Letters*, 24(10): 2226–2230.

Karacan, L.; Akata, Z.; Erdem, A.; and Erdem, E. 2016. Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts. arXiv:1612.00215.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations (ICLR), Conference Track Proceedings*.

Knobloch, D. 2017. Practical Challenges of Particle Filter Based UWB Localization in Vehicular Environments. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 1–5.

Lee, D.; Kim, J.; Moon, W.-J.; and Ye, J. C. 2019. CollaGAN: Collaborative GAN for Missing Image Data Imputation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2487–2496.

Liu, M.-Y.; and Tuzel, O. 2016. Coupled Generative Adversarial Networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 29: 469–477.

Mao, X.; Li, Q.; Xie, H.; Lau, R. Y.; Wang, Z.; and Paul Smolley, S. 2017. Least Squares Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2794–2802.

Mirza, M.; and Osindero, S. 2014. Conditional Generative Adversarial Nets. arXiv:1411.1784.

Park, J.; Choi, H.-B.; Ko, Y.-B.; and Lim, K.-W. 2021. Locate UWB Smart Keys: Smart and Faster. In *Proceedings of the ACM International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 169–171.

Sangkloy, P.; Lu, J.; Fang, C.; Yu, F.; and Hays, J. 2017. Scribbler: Controlling Deep Image Synthesis with Sketch and Color. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5400–5409.

Shi, G.; and Ming, Y. 2016. Survey of Indoor Positioning Systems Based on Ultra-Wideband (UWB) Technology. In *Wireless Communications, Networking and Applications*, 1269–1278. Springer.

Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; and Webb, R. 2017. Learning from Simulated and Unsupervised Images Through Adversarial Training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2107–2116.

Taigman, Y.; Polyak, A.; and Wolf, L. 2017. Unsupervised Cross-Domain Image Generation. In *5th International Conference on Learning Representations (ICLR), Conference Track Proceedings*.

Yoon, J.; Jordon, J.; and Schaar, M. 2018. RadialGAN: Leveraging Multiple Datasets to Improve Target-Specific Predictive Models Using Generative Adversarial Networks. In *International Conference on Machine Learning (ICML)*, 5699–5707. PMLR.

Zhao, J. J.; Mathieu, M.; and LeCun, Y. 2017. Energy-Based Generative Adversarial Networks. In *5th International Conference on Learning Representations (ICLR), Conference Track Proceedings*.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2223–2232.