**Robotics: Science and Systems 2022**
**New York City, NY, USA, June 27-July 1, 2022**

# **CALI**: **C**oarse-to-Fine **ALI**gnments Based Unsupervised Domain Adaptation of Traversability Prediction for Deployable Autonomous Navigation

Zheng Chen
Luddy School of Informatics, Computing, and Engineering
Indiana University - Bloomington
Indiana 47408
Email: zc11@iu.edu

Durgakant Pushp
Luddy School of Informatics, Computing, and Engineering
Indiana University - Bloomington
Indiana 47408
Email: dpushp@iu.edu

Lantao Liu
Luddy School of Informatics, Computing, and Engineering
Indiana University - Bloomington
Indiana 47408
Email: lantao@iu.edu

*Abstract*—**Traversability prediction is a fundamental perception capability for autonomous navigation. The diversity of data in different domains imposes significant gaps to the prediction performance of the perception model. In this work, we make efforts to reduce the gaps by proposing a novel coarse-to-fine *unsupervised domain adaptation* (UDA) model – CALI. Our aim is to transfer the perception model with high data efficiency, eliminate the prohibitively expensive data labeling, and improve the generalization capability during the adaptation from easy-to-obtain *source domains* to various challenging *target domains*. We prove that a combination of a coarse alignment and a fine alignment can be beneficial to each other and further design a first-coarse-then-fine alignment process. This proposed work bridges theoretical analyses and algorithm designs, leading to an efficient UDA model with easy and stable training. We show the advantages of our proposed model over multiple baselines in several challenging domain adaptation setups. To further validate the effectiveness of our model, we then combine our perception model with a visual planner to build a navigation system and show the high reliability of our model in complex natural environments where no labeled data is available. The robot navigation demonstration can be seen in this video: https://www.youtube.com/watch?v=Nqsegaq_x-o.**

## I. Introduction

We consider the deployment of autonomous robots in the real-world unstructured field environments, where the environments can be extremely complex involving random obstacles (e.g., big rocks, tree stumps, man-made objects), cross-domain terrains (e.g., combinations of gravel, sand, wet, uneven surfaces), as well as dense vegetation (tall and low grasses, shrubs, trees). Whenever a robot is deployed in such an environment, it needs to understand which area of the captured scene is navigable. A typical solution to this problem is the visual traversability prediction that can be achieved by learning the *scene semantic segmentation*.

Visual traversability prediction has been tackled by using deep neural networks where the models are typically trained offline with well-labeled datasets in limited scenarios. However, there is a gap between the data used to train the model and the real world. It is usually challenging for existing datasets to well approximate the true distributions of

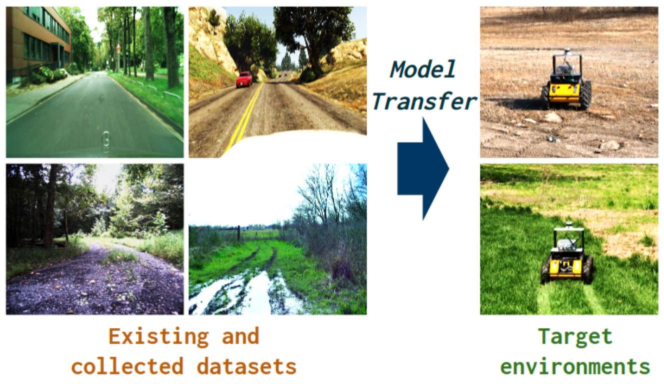


Fig. 1: Transferring models from the available domain to the target domain. The existing available data might be from either a simulator or collecting data in certain environments, at a certain time, and with certain sensors. In contrast, the target deployment might have significantly varying environments, time, and sensors.

the unseen *target* environments where the robot is deployed. Even incrementally collecting and adding new training data on the fly cannot guarantee the target environments to be well *in-distribution* included. In addition, annotating labels for dense predictions, e.g., semantic segmentation, is prohibitively expensive. Therefore, developing a generalization-aware deep model is crucial for robotic systems considering the demands of the practical deployment of deep perception models and the costs/limits of collecting new data in many robotic applications, e.g., autonomous driving, search and rescue, and environmental monitoring.

To tackle this challenge, a broadly studied framework is *transfer learning* [24] which aims to transfer models between two domains – *source domain* and *target domain* – that have related but different data distributions. The prediction on target domain can be considered as a strong generalization since testing data (in target domain) might fall out of the independently and identically distributed (i.i.d.) assumption and follow a very different distribution than the training data (in source domain). The "transfer" process has significant meaning to our model development since we can view the available public datasets [29, 8, 35, 17] as the source domain and treat the data in the to-be-deployed environments as the target domain. In this case, we have access to images and corresponding labels in source domain and images in target domain, but no access to labels in target domain. Transferring models, in this set-up, is called *Unsupervised Domain Adaptation* (UDA) [36, 40].

Domain Alignment (DA) [10, 12, 13, 32, 33] and Class Alignment (CA) [31] are two conventional ways to tackle the UDA problem. DA treats the deep features as a whole. It works well for image-level tasks such as image classification, but has issues with pixel-level tasks such as semantic segmentation, as the alignment of whole distributions ignores the class features and might misalign class distributions, even the whole features from the source domain and target domain are already well-aligned. CA is proposed to solve this issue for dense predictions with multiple classes.

It is natural and necessary to use CA to tackle the UDA of semantic segmentation as we need to consider aligning class features. However, CA can be problematic and might fail to outperform the DA for segmentation, and in a worse case, might have unacceptable *negative transfer*, which means the performance with adaptation is even degraded than that without adaptation. Intuitively, we need to consider more alignments in CA than in DA. Thus the searching space might be more complicated, and training might be more unstable and harder to converge to an expected minima, leading to larger prediction errors.

To solve the issue of CA, we investigate the relationship of the upper bounds of the prediction error on target domain between DA and CA and provide a theoretical analysis of the upper bounds of target prediction error in UDA setup, and bridge the theoretical analysis and algorithm design for UDA of traversability prediction.

In summary, our contributions include

- We prove that with proper assumptions, the upper bound of CA is upper bounded by the upper bound of DA. This indicates that constraining the training of CA using DA can be beneficial. We then propose a novel concept of *pseudo-trilateral game structure* (PTGS) for integrating DA and CA.
- We propose an efficient coarse-to-fine alignments based UDA model, named CALI, for traversability prediction. The new proposal includes a trilateral network structure, novel training losses, and an alternative training process. Our model design is well supported by theoretical analysis. It is also easy and stable to train and converge.
- We show significant advantages of our proposed model compared to several baselines in multiple challenging public datasets and one self-collected dataset. We combine the proposed segmentation model and a visual planner to build a visual navigation system. The results show high safety and effectiveness of our model.

## II. Related Work

**Semantic Segmentation:** Semantic segmentation aims to predict a unique human-defined semantic class for each pixel in the given images. With the prosperity of deep neural networks, the performance of semantic segmentation has been boosted significantly, especially by the advent of FCN [20] that first proposes to use deep convolutional neural nets to predict segmentation. Following works try to improve the FCN performance by multiple proposals, e.g., using different sizes of kernels or dilation rates to aggregate multi-scale features [6, 7, 38]; building image pyramids to create multi-resolution inputs [41]; applying probabilistic graph to smooth the prediction [19]; compensating features in deeper level by an encoder-decoder structure [30], and employing attention mechanism to capture the long-range dependencies among pixels in a more compact and efficient way [28]. We can also see how excellent the current semantic segmentation SOTA performance is from very recently released work [42, 37]. However, all of those methods belong to fully-supervised learning and the performance might catastrophically be degraded when a domain shift exists between the training data and the data when deploying. Considering the possible domain shift and developing *adaptation-aware* models is extremely practical and urgent.

**Unsupervised Domain Adaptation:** The main approaches to tackle UDA include adversarial training (a.k.a., *distribution alignment*) [10, 12, 13, 32, 31, 33, 21, 34] and self-training [43, 39, 23, 16]. Although self-training is becoming another dominant method for segmentation UDA in terms of the empirical results, it still lacks a sound theoretical foundation. In this paper, we only focus on the alignment-based methods that not only keep close to the UDA state-of-the-art (SOTA) performance but also are well supported by sound theoretical analyses [1, 3, 2].

The alignment-based methods adapt models via aligning the distributions from the source domain and target domain in an adversarial training process, i.e., making the deep features of source images and target images indistinguishable to a discriminator net. Typical approaches to UDA include Domain Alignment (DA) [10, 12, 13, 32, 33], which aligns the two domains using global features (aligning the feature tensor from source or target *as a whole*) and Class Alignment (CA) [31, 21, 34], which only considers aligning features of each class from source and target, no matter whether the domain distributions are aligned or not. In [31], the authors are inspired by the theoretical analysis of [2] and propose a discrepancy-based model for aligning class features. There is a clear relation between the theory guidance [2] and the design of network, loss, and training methods. There are some recent works [21, 34] similar to the proposed work in spirit and show improved results compared to [31], but it is still unclear to relate the proposed algorithms with theory and to understand why the structure/loss/training is designed as the presented way.

## III. Background and Preliminary Materials

We consider segmentation tasks where the input space is $\mathcal{X} \subset \mathbb{R}^{H\times W\times 3}$, representing the input RGB images, and the label space is $\mathcal{Y} \subset \{0,1\}^{H\times W\times K}$, representing the ground-truth $K$-class segmentation images, where the label for a single pixel at $(h, w)$ is denoted by a one-hot vector $y^{(h,w)} \in \mathbb{R}^K$ whose elements are by-default 0-valued except at location $(h, w)$ which is labeled as 1. Domain adaptation has two domain distributions over $\mathcal{X} \times \mathcal{Y}$, named source domain $\mathcal{D}_S$ and target domain $\mathcal{D}_T$. In the setting of UDA for segmentation, we have access to $m_s$ *i.i.d.* samples with labels $\mathcal{U}_S = \{\mathbf{x}_{si}, \mathbf{y}_{si}\}_{i=1}^{m_s}$ from $\mathcal{D}_S$ and $m_t$ *i.i.d.* samples without labels $\mathcal{U}_T = \{\mathbf{x}_{tj}\}_{j=1}^{m_t}$ from $\mathcal{D}_T$.

In the UDA problem, we need to reduce the prediction error on the target domain. A *hypothesis* is a function $h : \mathcal{X} \to \mathcal{Y}$. We denote the space of $h$ as $\mathcal{H}$. With the loss function $l(\cdot, \cdot)$, the expected error of $h$ on $\mathcal{D}_S$ is defined as

$$\epsilon_S(h) := \mathbb{E}_{(x,y)\sim\mathcal{D}_S} l(h(x), y). \tag{1}$$

Similarly, we can define the expected error of $h$ on $\mathcal{D}_T$ as

$$\epsilon_T(h) := \mathbb{E}_{(x,y)\sim\mathcal{D}_T} l(h(x), y). \tag{2}$$

Two important upper bounds related to the source and target error are given in [2]. Basically,

**Theorem 1** For a hypothesis $h$,

$$\epsilon_T(h) \le \epsilon_S(h) + d_1(\mathcal{D}_S, \mathcal{D}_T) + \lambda, \tag{3}$$

where $d_1(\cdot, \cdot)$ is the $L^1$ divergence for two distributions, and the constant term $\lambda$ does not depend on any $h$. However, it is claimed in [2] that the bound with $L^1$ divergence cannot be accurately estimated from finite samples, and using $L^1$ divergence can unnecessarily inflate the bound. Another divergence measure is thus introduced to replace the $L^1$ divergence with a new bound derived.

**Definition 1** Given two domain distributions $\mathcal{D}_S$ and $\mathcal{D}_T$ over $\mathcal{X}$, and a hypothesis space $\mathcal{H}$ that has finite VC dimension, the $\mathcal{H}$-divergence between $\mathcal{D}_S$ and $\mathcal{D}_T$ is defined as

$$\begin{aligned} d_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h\in\mathcal{H}} |\mathrm{P}_{x\sim\mathcal{D}_S}[h(x) = 1] - \\ \mathrm{P}_{x\sim\mathcal{D}_T}[h(x) = 1]\,|, \end{aligned} \tag{4}$$

where $\mathrm{P}_{x\sim\mathcal{D}_S}[h(x) = 1]$ represents the probability of $x$ belonging to $\mathcal{D}_S$. Same to $\mathrm{P}_{x\sim\mathcal{D}_T}[h(x) = 1]$.

The $\mathcal{H}$-divergence resolves the issues in the $L^1$ divergence. If we replace $d_1(\mathcal{D}_S, \mathcal{D}_T)$ in Eq. (3) with $d_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T)$, then a new upper bound for $\epsilon_T(h)$, named as $\mathbb{UB}_1$, can be written as

$$\begin{aligned} &\epsilon_T(h) \le \mathbb{UB}_1, \\ &\mathbb{UB}_1 = \epsilon_S(h) + d_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda. \end{aligned} \tag{5}$$

An approach to compute the empirical $\mathcal{H}$-divergence is also proposed in [2].

**Lemma 1** For a symmetric hypothesis class $\mathcal{H}$ (one where for every $h \in \mathcal{H}$, the inverse hypothesis $1 - h$ is also in $\mathcal{H}$) and two sample sets $\mathcal{U}_S = \{x_i, i = 1, \cdots, m_s, x_i \sim \mathcal{D}_S\}$ and $\mathcal{U}_T = \{x_j, j = 1, \cdots, m_t, x_j \sim \mathcal{D}_T\}$.

$$\begin{aligned} \hat{d}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2\Bigg(1 - \min_{\eta\in\mathcal{H}} \Bigg[\frac{1}{m_s}\sum_{i=1}^{m_s} \mathbb{I}[\eta(x_i) = 0] + \\ \frac{1}{m_t}\sum_{j=1}^{m_t} \mathbb{I}[\eta(x_j) = 1]\Bigg]\Bigg), \end{aligned} \tag{6}$$

where $\mathbb{I}[a]$ is an indicator function which is 1 if $a$ is true, and 0 otherwise.

The second upper bound is based on a new hypothesis called symmetric difference hypothesis.

**Definition 2** For a hypothesis space $\mathcal{H}$, the symmetric difference hypothesis space $\mathcal{H}\Delta\mathcal{H}$ is the set of hypotheses

$$g \in \mathcal{H}\Delta\mathcal{H} \Leftrightarrow g(x) = h(x) \oplus h'(x) \quad \text{for some } h, h' \in \mathcal{H}, \tag{7}$$

where $\oplus$ denotes an XOR operation. Then we can define the $\mathcal{H}\Delta\mathcal{H}$-distance as

$$\begin{aligned} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h,h'\in\mathcal{H}} |\mathrm{P}_{x\sim\mathcal{D}_S}[h(x) \ne h'(x)] - \\ \mathrm{P}_{x\sim\mathcal{D}_T}[h(x) \ne h'(x)]\,|. \end{aligned} \tag{8}$$

Similar to Eq. (5), if we replace $d_1(\mathcal{D}_S, \mathcal{D}_T)$ with $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T)$, the second upper bound for $\epsilon_T(h)$, named as $\mathbb{UB}_2$, can be expressed as

$$\begin{aligned} &\epsilon_T(h) \le \mathbb{UB}_2, \\ &\mathbb{UB}_2 = \epsilon_S(h) + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda, \end{aligned} \tag{9}$$

where $\lambda$ is the same term as in Eq. (3).

A standard way to achieve the alignment for deep models is to use the adversarial training method, which is also used in Generative Adversarial Networks (GANs). Therefore we explain the key concepts of adversarial training using the example of GANs.

GAN is proposed to learn the distribution $p_r$ of a set of given data $\{\mathbf{x}\}$ in an adversarial manner. The architecture consists of two networks - a generator $G$, and a discriminator $D$. The $G$ is responsible for generating fake data (with distribution $p_g$) from random noises $\mathbf{z} \sim p_{\mathbf{z}}$ to fool the discriminator $D$ that is instead to accurately distinguish between the fake data and the given data. Optimization of a GAN involves a mini-maximization over a joint loss for $G$ and $D$.

$$\begin{aligned} &\min_G \max_D V(G, D) \\ &V(G, D) = \mathbb{E}_{\mathbf{x}\sim p_r} \log[D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log[1 - D(G(z))]. \end{aligned} \tag{10}$$

where we use 1 as the real label and 0 as the fake label. Training with Eq. (10) is a bilateral game where the distribution $p_g$ is aligned with the distribution $p_r$.

The two bounds (Eq. (5) and Eq. (9)) for the target domain error are separately given in [2]. It has been independently demonstrated that domain alignment corresponds to optimizing over $\mathbb{UB}_1$ [10], where optimization over the upper bound $\mathbb{UB}_1$ (Eq. (5) with the divergence Eq. (6)) is proved as equivalent to Eq. (10) with a supervised learning in the source domain, and that class alignment corresponds to optimizing over $\mathbb{UB}_2$ [31], where the $d_{\mathcal{H}\Delta\mathcal{H}}$ is approximated by the discrepancy between two different classifiers.

Training DA is straightforward since we can easily define binary labels for each domain, e.g., we can use 1 as the source domain label and 0 as the target domain label. Adversarial training over the domain labels can achieve domain alignment. For CA, however, it is difficult to implement as we do not have target labels, hence the target class features are completely

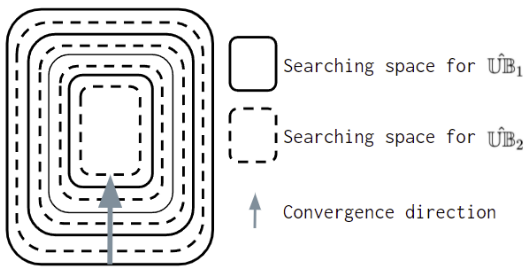

Searching space for $\hat{\mathbb{UB}}_1$
Searching space for $\hat{\mathbb{UB}}_2$
Convergence direction

Fig. 2: An ideal iterative training process by integrating DA and CA.

unknown to us, thus leading naively using adversarial training over each class impossible. The existing way well supported by theory to perform CA [31] is to indirectly align class features by devising two different classifier hypotheses. The two classifiers have to be well trained on the source domain and are able to classify different classes in the source domain with different decision boundaries. Then considering the shift between source and target domain, the trained two classifiers might have disagreements on target domain classes. Note since the two classifiers are already well trained on the source domain, the agreements of the two classifiers represent those features in the target domain that are close to the source domain, while in contrast, the features where disagreements happen indicate that there is a large shift between source and target. We use the disagreements to approximate the distance between source and target. If we are able to minimize the disagreements of the two classifiers, then features of each class between source and target will be enforced to be well aligned.

## IV. Methodology

In this work we further investigate the relation between the $\mathbb{UB}_1$ and $\mathbb{UB}_2$ and prove that $\mathbb{UB}_1$ turns out to be an upper bound of $\mathbb{UB}_2$, meaning DA can be a necessary constraint to CA. This is also consistent to our intuition: DA aligns features globally in a coarse way while CA aligns features locally in a finer way. Constraining CA with DA is actually a coarse-to-fine process, which makes the alignment process efficient and stable. By carefully studying the internal structure of existing DA and CA work, we propose a novel concept, *pseudo-trilateral game structure*, for efficiently integrating DA and CA. We follow our theoretical analysis and proposed PTGS to guide the development of CALI, including designs of model structure, losses and training process.

Notations used in this paper is explained as follows. we denote the segmentation model $h$ as $h^{\theta,\phi}(x) = C^{\theta}(G^{\phi}(x))$ which consists of a feature extractor $G^{\phi}$ parameterized by $\phi$ and a classifier $C^{\theta}$ parameterized by $\theta$, and $x$ is a sample from $\mathcal{U}_S$ or $\mathcal{U}_T$. If multiple classifiers are used, we will denote the $j^{th}$ classifier as $C_j$. We denote the discriminator as $D^{\psi}$ parameterized by $\psi$.

### A. Bounds Relation

We start by examining the relationship between the DA and the CA from the perspective of target error bound. We propose to use this relation to improve the segmentation performance of class alignment, which is desired for dense prediction tasks. We provide the following theorem:

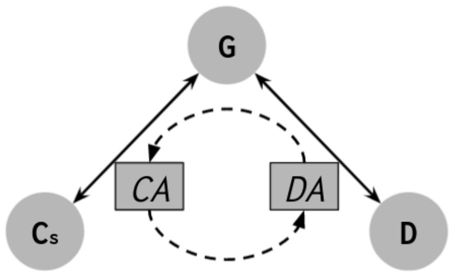


Fig. 3: Pseudo-trilateral game structure (PTGS). Three players are in the game, a feature extractor $G$, a domain discriminator $D$ and a family of classifiers $C_s$. The game between $G$ and $Cs$ is the CA while the game between $G$ and $D$ is the DA. The DA and CA are connected by sharing the same feature extractor $G$. Both $D$ and $C_s$ are trying to adjust the $G$ such that the features between source and target generated from $G$ could be well aligned globally and locally.

**Theorem 2** If we assume there is a hypothesis space $\mathcal{H}$ for segmentation model $h^{\theta,\phi}$ and a hypothesis space $\mathcal{H}_D$ for domain classifiers $D^{\psi}$, and $\mathcal{H}\Delta\mathcal{H} \subset \mathcal{H}_D$, then we have

$$\begin{aligned} &\epsilon_T(h) \leq \hat{\mathbb{UB}}_2 \leq \hat{\mathbb{UB}}_1, \\ &\hat{\mathbb{UB}}_1 = \epsilon_S(h) + \frac{1}{2} d_{\mathcal{H}_D}(\mathcal{D}_S, \mathcal{D}_T) + \lambda, \\ &\hat{\mathbb{UB}}_2 = \epsilon_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda. \end{aligned} \tag{11}$$

The proof of this theorem is provided in Appendix. VI-A.

Essentially, we limit the hypothesis space $\mathcal{H}$ and $\mathcal{H}_D$ in Eq. (11) into the space of deep neural networks. Directly optimizing over $\hat{\mathbb{UB}}_2$ might be hard to converge since $\hat{\mathbb{UB}}_2$ is a tighter upper bound for the prediction error on target domain. The bounds relation in Eq. (11) shows that the $\hat{\mathbb{UB}}_1$ is an upper bound of $\hat{\mathbb{UB}}_2$. This provides us a clue to improve the training process of class alignment, i.e., *the domain alignment can be a global constraint and narrow down the searching space for the class alignment*. This also implies that integrating the domain alignment and class alignment might boost the training efficiency as well as the prediction performance of UDA. An ideal training process is illustrated in Fig. 2 where the searching space of $\hat{\mathbb{UB}}_2$ (CA) is constantly bounded by that of $\hat{\mathbb{UB}}_1$ (DA), ensuring the whole training process converge stably. This inspires us to design a new model, and we are explaining next in details about our model structures, losses and training process.

### B. CALI Structure

The existing DA or CA works usually involve a bilateral game. In CA, the game is between a feature extractor and a family of classifiers. The two players are optimized over the discrepancy of the two classifiers (note here the two players are the two classifiers vs. the feature extractor) in an opposite manner. In DA, the game happens between a segmentation net and a domain discriminator. The two players are optimized over the domain discrimination in an opposite way. It has been empirically showed [33, 32] that DA performs well if the

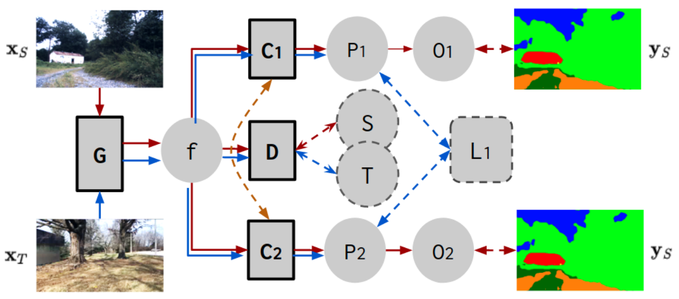


Fig. 4: CALI network structure. $\mathbf{x}_S$ and $\mathbf{x}_T$ represent the source and target images, respectively. $\mathbf{y}_S$ is the label of the source image. See Section. IV-B for more details.

domain alignment happens to the prediction probability (after `Softmax()`). However, according to the identified relation in Eq. (11), the two upper bounds $\hat{\mathbb{UB}}_1$ and $\hat{\mathbb{UB}}_2$ need to use the same feature, hence we connect the domain alignment and class alignment using a shared feature extractor and propose a novel concept called PTGS (see Fig. 3) to illustrate an interesting structure to integrate DA and CA. Both $C_s$ and $D$ have game with $G$, but there is no game between $C_s$ and $D$, hence we call this game as *pseudo-trilateral game*. Furthermore, as defined in Eq. (8), $h$ and $h^{'}$ are two different hypotheses, thus we have to ensure the classifiers in $C_s$ are different during the training.

Following the concept of PTGS, we design the structure of our CALI model as shown in Fig. 4. Four networks are involved, a shared feature extractor $G$, a domain discriminator $D$ and two classifiers $C_1$ and $C_2$. $f$ represents the shared features; $P_1/O_1$ and $P_2/O_2$ are the probability/class predictions for $C_1$ and $C_2$, respectively; $S/T$ represent the source domain label (1) and target domain label (0); and $L_1$ represents the $L_1$ distance measure between two probability distributions. The one-way solid arrows indicate the forward propagation of the data flow while the two-way dashed arrows indicate losses are generated. The red arrows represent the source-related data while the blue ones represent the target-related data. The orange two-way dashed line indicates the structural regularization loss between the $C_1$ and $C_2$.

Note that the supervision signal from source domain ($\mathbf{y}_S$ in Fig. 4) is used to train $C_1$ and $C_2$ because both classifiers are expected to generate correct decision boundaries on source domain. No label from target domain is used during the training process.

### C. CALI Losses

We denote raw images from source or target domain as $\mathbf{x}$, and the label from source domain as $\mathbf{y}$. We use semantic labels in source domain to train all of the nets, but the domain discriminator, in a supervised way, see the solid red one-way arrow in Fig. 4. We need to minimize the supervised segmentation loss since Eq. (11) and other related Eqs suggest that the source prediction error is also part of the upper bound of target error. The supervised segmentation loss for training CALI is defined as

$$\begin{aligned}\mathcal{L}_{seg}(G, C_1, C_2) &= \frac{1}{2}\Bigg(\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_S}\left[-\mathbf{y}\log(C_1(G(\mathbf{x})))\right] + \\ &\qquad \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_S}\left[-\mathbf{y}\log(C_2(G(\mathbf{x})))\right]\Bigg) \\ &= -\frac{1}{2}\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_S}[\mathbf{y}_S\log\left((C_1(G(\mathbf{x}))\odot(C_2(G(\mathbf{x}))))\right)],\end{aligned} \tag{12}$$

where $\odot$ represents the element-wise multiplication between two tensors.

To perform domain alignment, we need to define the joint loss function for $G$ and $D$

$$\mathcal{V}_1(G, D) = -\left(\mathcal{CE}_S(\mathbf{x}) + \mathcal{CE}_T(\mathbf{x})\right), \tag{13}$$

where no segmentation labels but domain labels are used, and we use the standard cross-entropy to compute the domain classification loss for both source ($\mathcal{CE}_S(\mathbf{x})$) and target data ($\mathcal{CE}_T(\mathbf{x})$). We have

$$\begin{aligned}\mathcal{CE}_S(\mathbf{x}) &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_S}[\mathcal{CE}([1,0]^T, [D(G(\mathbf{x})), 1-D(G(\mathbf{x}))]^T)] \\ &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_S}[-\log(D(G(\mathbf{x})))].\end{aligned} \tag{14}$$

and

$$\begin{aligned}\mathcal{CE}_T(\mathbf{x}) &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_T}[\mathcal{CE}([0,1]^T, [D(G(\mathbf{x})), 1-D(G(\mathbf{x}))]^T)] \\ &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_T}[-\log(1-D(G(\mathbf{x})))],\end{aligned} \tag{15}$$

Note we include $G$ in Eq. (14) since both the source data and target data are passed through the feature extractor. This is different than standard GAN, where the real data is directly fed to $D$, without passing through the generator.

To perform class alignment, we need to define the joint loss function for $G$, $C_1$, and $C_2$

$$\mathcal{V}_2(G, C_1, C_2) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_T}\left[d(C_1(G(\mathbf{x})), C_2(G(\mathbf{x})))\right], \tag{16}$$

where $d(\cdot,\cdot)$ is the distance measure between two distributions from the two classifiers. In this paper, we use the same $L_1$ distance in [31] as the measure, thus $d(p,q) = \frac{1}{K}|p-q|_1$, where $p$ and $q$ are two distributions and $K$ is the number of label classes.

To prevent $C_1$ and $C_2$ from converging to the same network throughout the training, we use the cosine similarity as a weight regularization to maximize the difference of the weights from $C_1$ and $C_2$, i.e.,

$$\mathcal{WR}(C_1, C_2) = \frac{\mathbf{w}_1\cdot\mathbf{w}_2}{\|\mathbf{w}_1\|\,\|\mathbf{w}_2\|}, \tag{17}$$

where $\mathbf{w}_1$ and $\mathbf{w}_2$ are the weight vectors of $C_1$ and $C_2$, respectively.

### D. CALI Training

We integrate the training processes of domain alignment and class alignment to systematically train our CALI model. To be consistent with Eq. (11), we adopt an iterative mechanism that alternates between domain alignment and class alignment. Our training process is pseudo-coded in Algorithm 1.

**Algorithm 1:** CALI Training Process

```
1  Input: Source dataset $\mathcal{U}_s$; Target dataset $\mathcal{U}_t$; Initial model
     $G, C_1, C_2$ and $D$; Maximum iterations $M$; Iteration
     interval $I$.
2  Output: Updated model parameters $\phi_G, \theta_{C_1}, \theta_{C_2}$ and $\psi_D$
3  Initialization: is_domain=True; is_class=False;
4  for $m \leftarrow 1$ to $M$ do
5      if $m\%I == 0$ and $m \neq 0$ then
6          is_domain = not is_domain;
7          is_class = not is_class;
       // Eq. (12)
8      $\min_{\phi_G, \theta_{C_1}, \theta_{C_2}} \mathcal{L}_{seg}(G, C_1, C_2)$;
       // Eq. (17)
9      $\min_{\theta_{C_1}, \theta_{C_2}} \mathcal{WR}(C_1, C_2)$;
10     if is_domain then
           // Eq. (13)
11         $\max_{\psi_D} \min_{\theta_G} \mathcal{V}_1(G, D)$;
12     if is_class then
           // Eq. 16
13         $\max_{\theta_{C_1}, \theta_{C_2}} \min_{\phi_G} \mathcal{V}_2(G, C_1, C_2)$;
14 Return $\phi_G$, $\theta_{C_1}$, $\theta_{C_2}$ and $\psi_D$;
```

Note the adversarial training order of $\mathcal{V}_1$ in Algorithm 1 is $\max_{\psi_D} \min_{\phi_G}$, instead of the $\min_{\phi_G} \max_{\psi_D}$, meaning in each training iteration we first train the feature extractor and then the discriminator. The reason for this order is because we empirically find that the feature from $G$ is relatively easy for $D$ to discriminate, hence if we train $D$ first, then the $D$ might become an accurate discriminator in the early stage of training and there will be no adversarial signals for training $G$, thus making the whole training fail. The same order applies to training of the pair of $G$ and $Cs$ with $\mathcal{V}_2$.

## *E. Visual Planner*

We design a visual receding horizon planner to achieve feasible visual navigation by combining the learned image segmentation. Specifically, first we compute a library of motion primitives [14, 15] $\mathcal{M} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n\}$ where each $\mathbf{p}_* = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ is a single primitive. We use $\mathbf{x}_* = \begin{bmatrix} x & y & \psi \end{bmatrix}^T$ to denote a robot pose. Then we project the motion primitives to the image plane and compute the navigation cost function for each primitive based on the evaluation of collision risk in image space and target progress. Finally, we select the primitive with minimal cost to execute. The trajectory selection problem can be defined as:

$$\mathbf{p}_{optimal} = \underset{\mathbf{p}}{\text{argmin}}\ w_1 \cdot C_c(\mathbf{p}) + w_2 \cdot C_t(\mathbf{p}), \qquad (18)$$

where $C_c(\mathbf{p}) = \sum_j^m c_c^j$ and $C_t(\mathbf{p}) = \sum_j^m c_t^j$ are the collision cost and target cost of one primitive $\mathbf{p}$, and $w_1$, $w_2$ are corresponding weights, respectively.

To efficiently evaluate the collision risk in the learned segmentation images, we first classify the classes in terms of their navigability, e.g., in off-road environments, grass and mulch are classified as navigable while tree and bush are classified as non-navigable. In this case, we are able to extract the boundary between the navigable space and the non-navigable space. We treat the boundary part close to the bottom line of the image as the *obstacle boundary*. We further use the obstacle boundary to generate a Scaled Euclidean Distance Field (SEDF), where the values fall in $[0, 1]$, representing the risk level at the pixel position. Examples of different SEDF with different scale factors can be seen in Fig. 6.

Assume $\mathbf{x}^j$ is the $j^{th}$ pose in one primitive and its image coordinates are $(u^j, v^j)$, then the collision risk for $\mathbf{x}^j$ is

$$c_c^j = E[u^j, v^j], \qquad (19)$$

where $E$ represents the SEDF image.

To evaluate target progress during the navigation progress, we propose to use the distance on $SE(3)$ as the metric. We define three types of frames: world frame $F_w$, primitive pose frame $F_{pj}$, and goal frame $F_g$. The transformation of $F_{pj}$ in $F_w$ is denoted as $\mathbf{T}_{wpj}$ while that of $F_g$ in $F_w$ is $\mathbf{T}_{wg}$. A typical approach to represent the distance is to split a pose into a position and an orientation and define two distances on $\mathbb{R}^3$ and $SO(3)$. Then the two distances can be fused in a weighted manner with two strictly positive scaling factors $a$ and $b$ and with an exponent parameter $p \in [1, \infty]$ [5]:

$$d(\mathbf{T}_{wpj}, \mathbf{T}_{wg}) = \Big[ a \cdot d_{rot}(\mathbf{R}_{wpj}, \mathbf{R}_{wg})^p + b \cdot d_{trans}(\mathbf{t}_{wpj}, \mathbf{t}_{wg})^p \Big]^{1/p}. \qquad (20)$$

We use the Euclidean distance as $d_{trans}(\mathbf{t}_{wpj}, \mathbf{t}_{wg})$, the Riemannian distance over $SO(3)$ as $d_{rot}(\mathbf{R}_{wpj}, \mathbf{R}_{wg})$ and set $p$ as 2. Then the distance (target cost) between two transformation matrices can be defined [25] as:

$$\begin{aligned} c_t^j &= d(\mathbf{T}_{wpj}, \mathbf{T}_{wg}) \\ &= \Big[ a \cdot \left\| \log(\mathbf{R}_{wpj}^{-1} \mathbf{R}_{wg}) \right\|^2 + b \cdot \left\| \mathbf{t}_{wpj} - \mathbf{t}_{wg} \right\|^2 \Big]^{1/2}. \end{aligned} \qquad (21)$$

## V. Experiments

### *A. Datasets*

We evaluate CALI together with several baseline methods on a few challenging domain adaptation scenarios, where several public datasets, e.g., GTA5 [29], Cityscapes [8], RUGD [35], RELLIS [17], as well as a small self-collected dataset, named MESH (see the first column of Fig. 8), are investigated. The GTA5 dataset contains 24,966 synthesized high-resolution images in the urban environments from a video game and pixel-wise semantic annotations of 33 classes. The Cityscapes dataset consists of 5,000 finely annotated images whose label is given for 19 commonly seen categories in urban environments, e.g., road, sidewalk, tree, person, car, etc. The RUGD and RELLIS are two recently released datasets that aim to evaluate segmentation performance in off-road environments. The RUGD and the RELLIS contain 24 and 20 classes with 8,000 and 6,000 images, respectively. RUGD and RELLIS cover various scenes like trails, creeks, parks, villages, and puddle terrains. Our dataset, MESH, includes

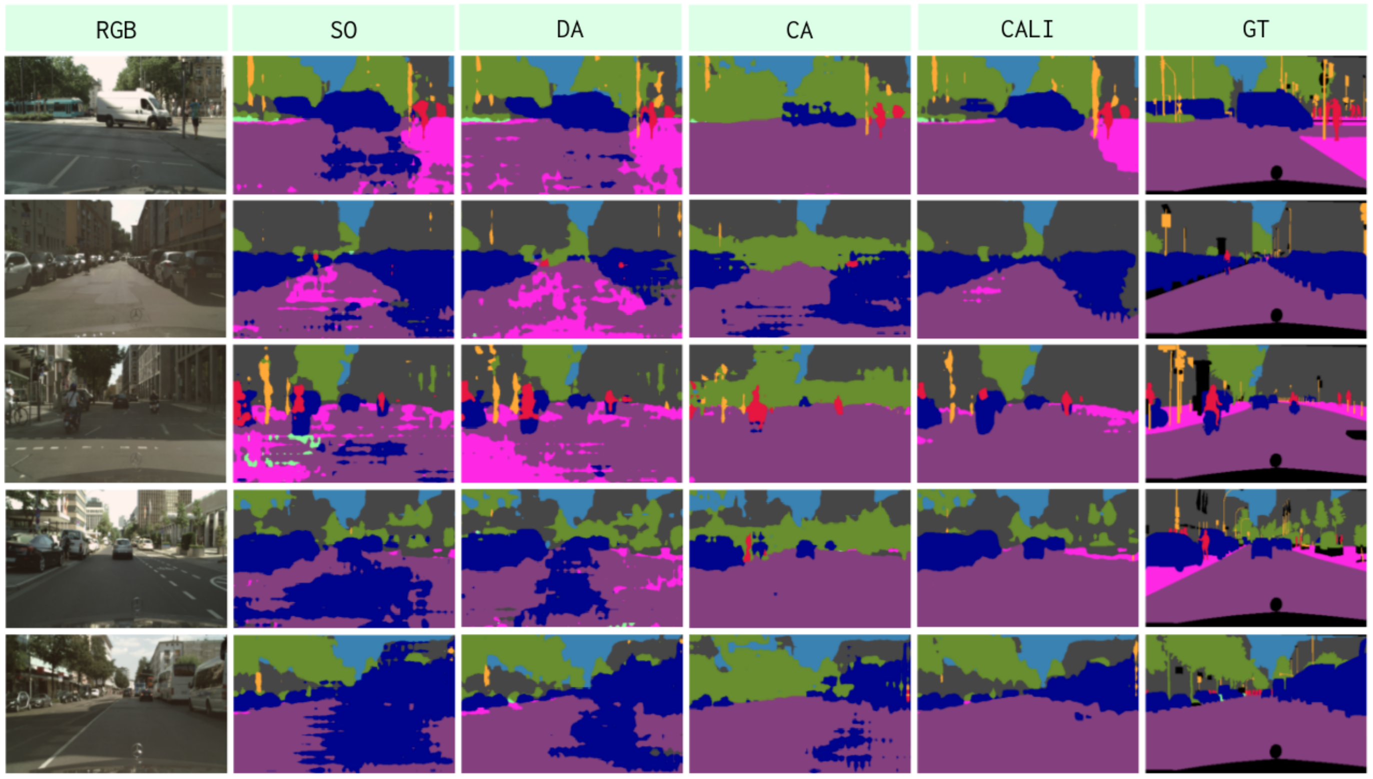


Fig. 5: Qualitative results on adaptation GTA5→Cityscapes. Results of our proposed model is listed in the last second column. GT represents the ground-truth labels.

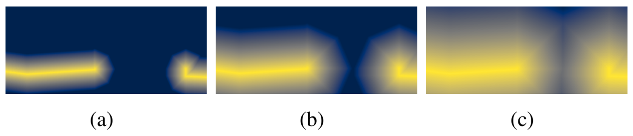


Fig. 6: Different SEDFs with varying scale factors of (a) $\alpha = 0.25$, (b) $\alpha = 0.55$ and (c) $\alpha = 1.00$. Values range from 0 to 1 by the color from blue to yellow.

features like grass, trees (particularly challenging in winter due to foliage loss and monochromatic colors), mulch, etc. It helps us to further validate the performance of our proposed model for traversability prediction in challenging scenes, particularly the off-road environments.

## *B. Implementation Details*

To be consistent with our theoretical analysis, the implementation of CALI only adopts the necessary indications by Eq. (11). First, Eq. (11) requires that the input of the two upper bounds (one for DA and the other one for CA) should be the same. Second, nothing else but only domain classification and hypotheses discrepancy are involved in Eq. (11) and other related analyses (Eq. (3) - Eq. (9)). Accordingly, we strictly follow the guidance of our theoretical analyses. First, CALI performs DA in the intermediate-feature level ($f$ in Fig. 4), instead of the output-feature level used in [33]. Second, we exclude the multiple additional tricks, e.g., entropy-based and multi-level features based alignment, and class-ratio priors in [33] and multi-steps training for feature extractor in [31]. We also implement baseline methods without those techniques for a fair comparison. To avoid possible degraded performance bought by a class imbalance in the used datasets, we regroup those rare classes into classes with a higher pixel ratio. For example, we treat the building, wall, and fence as the same class; the person and rider as the same class in the adaptation of GTA5→Cityscapes. In the adaptation of RUGD→RELLIS, we treat the tree, bush, and log as the same class, and the rock and rockbed as the same class. Details about remapping can

TABLE I: Quantitative comparison of different methods in UDA of GTA5→Cityscapes. mIoU* represents the average mIoU over all of classes.

| Class | SO | DA | CA | CALI |
|---|---|---|---|---|
| Road | 38.86 | 52.80 | **78.56** | 75.36 |
| Sidewalk | 17.47 | 18.95 | 2.79 | **27.12** |
| Building | 63.60 | 61.73 | 43.51 | **67.00** |
| Sky | 58.08 | 54.35 | 46.59 | **60.49** |
| Vegetation | 67.21 | 64.69 | 41.48 | **67.50** |
| Terrain | 7.63 | 7.04 | 8.37 | **9.56** |
| Person | **16.89** | 15.45 | 13.48 | 15.03 |
| Car | 30.32 | 43.41 | 31.64 | **52.25** |
| Pole | 11.61 | **12.38** | 9.68 | 11.91 |
| mIoU* | 34.63 | 36.76 | 30.68 | **42.91** |

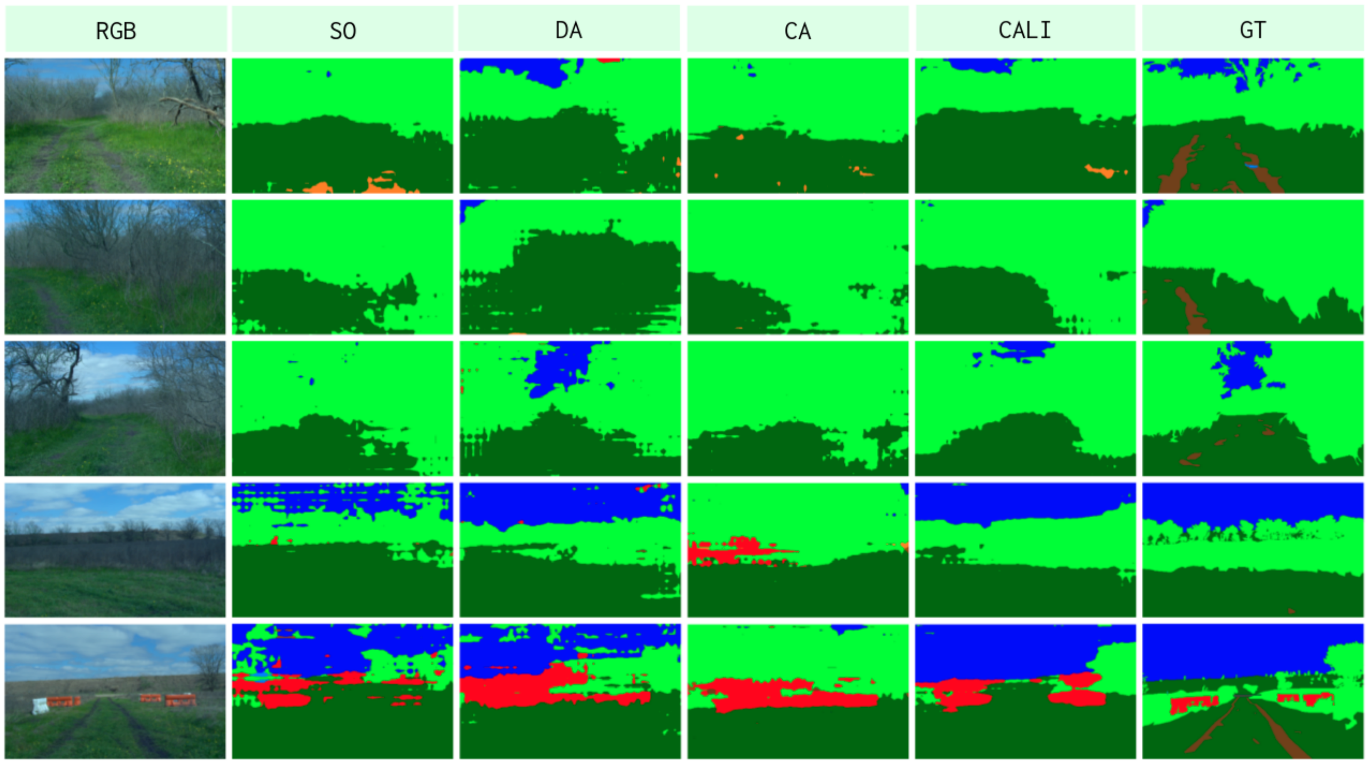


Fig. 7: Qualitative results on adaptation RUGD→RELLIS. Results of our proposed model is listed in the last second column. GT represents the ground-truth labels.

TABLE II: Quantitative comparison of different methods in UDA of RUGD→RELLIS. mIoU* is the average mIoU over all of classes.

| Class | SO | DA | CA | CALI |
|---|---|---|---|---|
| Dirt | 0.00 | 0.53 | **3.23** | 0.01 |
| Grass | 64.78 | 61.63 | 65.35 | **67.08** |
| Tree | 40.79 | 45.93 | 41.51 | **55.80** |
| Sky | 45.07 | 67.00 | 2.31 | **72.99** |
| Building | 10.90 | **12.29** | 10.91 | 10.28 |
| mIoU* | 32.31 | 37.48 | 24.66 | **41.23** |

be seen in Fig. 14 and Fig. 15 in Appendix. VI-B.

We use the PyTorch [26] framework for implementation. Training images from source and target domains are cropped to be half of their original image dimensions. The batch size is set to 1 and the weights of all batch normalization layers are fixed. We use the ResNet-101 [11] pretrained on ImageNet [9] as the model $G$ for extracting features. We use the ASPP module in DeepLab-V2 [6] as the structure for $C_1$ and $C_2$. We use the similar structure in [27] as the discriminator $D$, which consists of 5 convolution layers with kernel $4 \times 4$ and with channel size $\{64, 128, 256, 512, 1\}$ and stride of 2. Each convolution layer is followed by a Leaky-ReLU [22] parameterized by 0.2, but only the last convolution layer is follwed by a Sigmoid function. During the training, we use SGD [4] as the optimizer for $G, C_1$ and $C_2$ with a momentum of 0.9, and use Adam [18] to optimize $D$ with $\beta_1 = 0.9, \beta_2 = 0.99$. We set all SGD optimizers a weight decay of 5e-4. The initial learning rates of all SGDs for performing domain alignment are set to 2.5e-4 and the one of Adam is set as 1e-4. For class alignment, the initial learning rate of SGDs is set to 1e-3. All of the learning rates are decayed by a poly learning rate policy, where the initial learning rate is multiplied by $(1 - \frac{iter}{max\_iters})^{power}$ with $power = 0.9$. All experiments are conducted on a single Nvidia Geforce RTX 2080 Super GPU.

## C. Comparative Studies

We present comparative experimental results of our proposed model, CALI, compared to different baseline methods – Source-Only (SO) method, Domain-Alignment (DA) [33] method, and Class-Alignment [31] method. Specifically, we first perform evaluations on a sim2real UDA in city-like environments, where the source domain is represented by GTA5 while the target domain is the Cityscapes. Then we consider a transfer of real2real in forest environments, where the source domain and target domain are set as RUGD and RELLIS, respectively. All models are trained with full access to the images and labels in the source domain and with only access to the images in the target domain. The labels in target datasets are only used for evaluation purposes. Finally, we further validate our model performance for adapting from RUGD to our self-collected dataset MESH.

To ensure a fair comparison, all the methods use the same

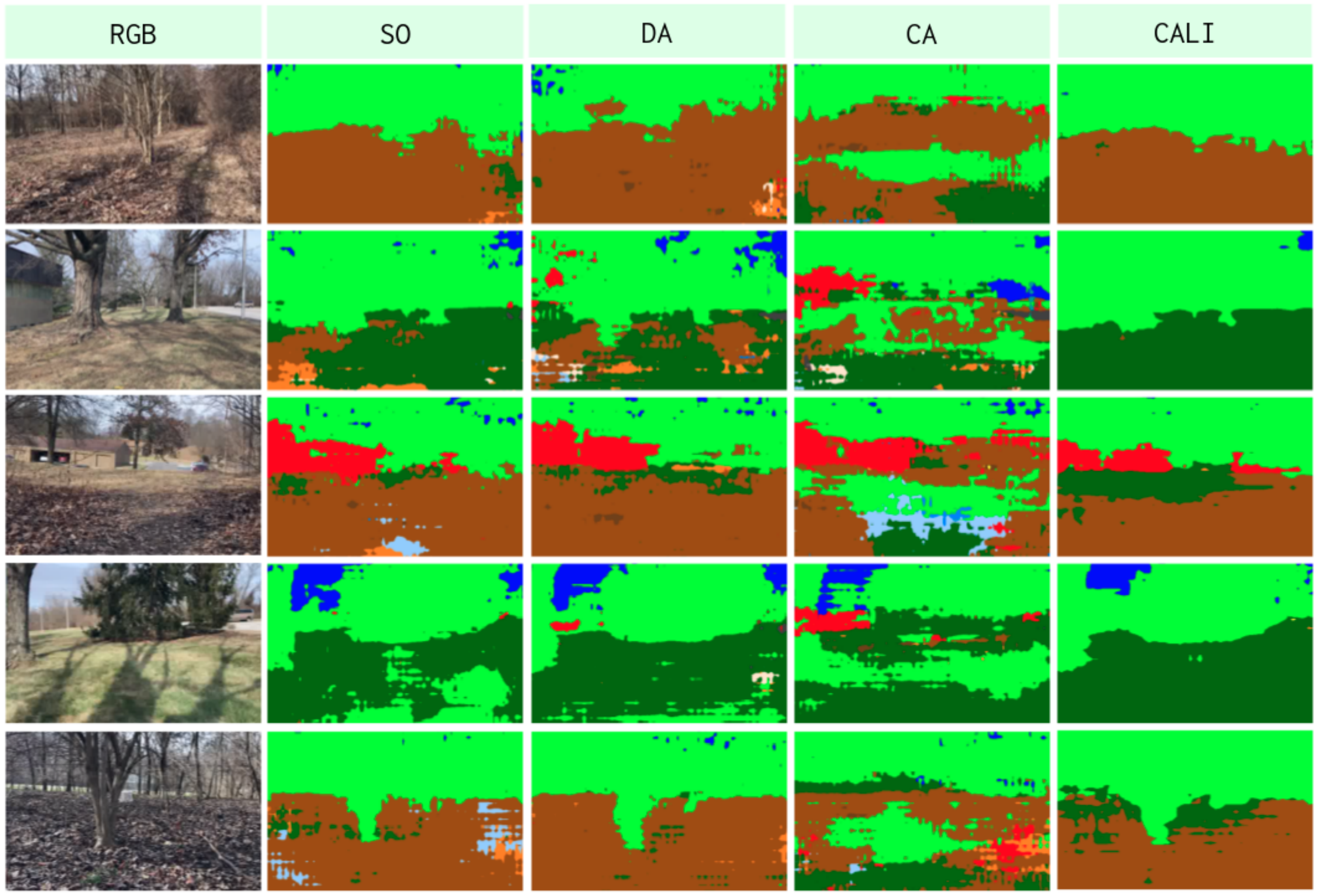


Fig. 8: Qualitative results on adaptation RUGD→MESH. Results of our proposed model is listed in the last column.

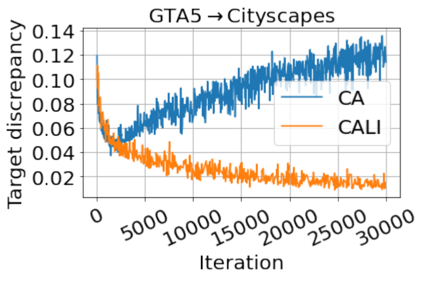
feature extractor $G$; both DA and CALI have the same domain discriminator $D$; both CA and CALI have the same two classifiers $C_1$ and $C_2$. We also use the same optimizers and optimization-related hyperparameters if any is used for models under comparison.

We use the mean of Intersection over Union (mIoU) as the metric to evaluate each class and overall segmentation performance on testing images. IoU is computed as $\frac{n_{tp}}{n_{tp}+n_{fp}+n_{fn}}$, where $n_{tp}, n_{tn}, n_{fp}$ and $n_{fn}$ are true positive, true negative, false positive and false negative, respectively.

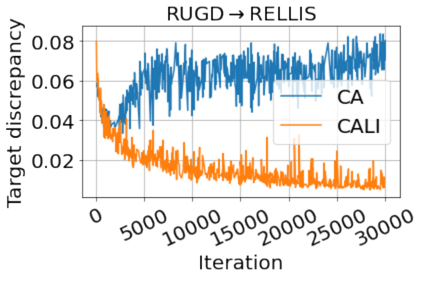
*1) GTA5→Cityscapes:* Quantitative comparison results of GTA5→Cityscapes are shown in Table. I, where segmentations are evaluated on 9 classes (as regrouped in Fig. 14). Our proposed method has significant advantages over multiple baseline methods for most categories and overall performance (mIoU*).

In our testing case, SO achieves the highest score for the class person even without any domain adaptation. One possible reason for this is the deep features of the source person and the target person from the model solely trained on source domain, are already well-aligned. If we try to interfere this well-aligned relation using *unnecessary* additional efforts, the target prediction error might be increased (see the mIoU values of the person from the other three methods). We call this phenomenon as *negative transfer*, which also happens to other classes if we compare SO and DA/CA, e.g., sidewalk, building, sky, vegetation, and so on. In contrast, CALI maintains an improved performance compared to either SO or DA/CA. We validate our analytical method for DA and CA (Section. IV-A) by a comparison between CALI and baselines. This indicates either single DA or CA is problematic for semantic segmentation, particularly when we strictly follow what the theory supports and do not include any other training tricks (that might increase the training complexity and make the training unstable). This implies that integration of DA and CA is beneficial to each other with significant improvements, and more importantly, CALI is well theoretically supported, and the training process is easy and stable.

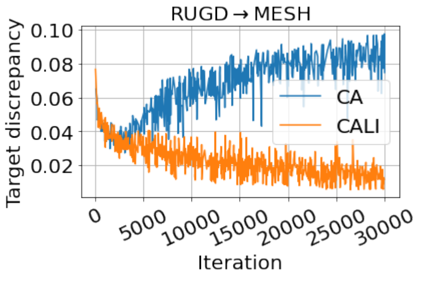
Fig. 5 shows the examples of qualitative comparison for UDA of GTA5→Cityscapes. We find that CALI prediction is less noisy compared to the baselines methods as shown in the second and third columns (sidewalk or car on-road), and shows better completeness (part of the car is missing, see the fourth column).



(a)

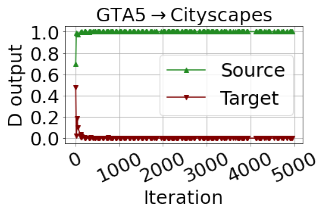


(b)

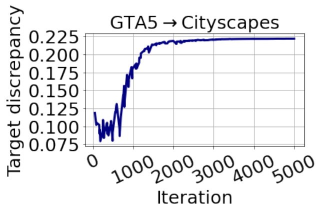


(c)

Fig. 9: Target discrepancy changes during training process of (a) GTA5→Cityscapes; (b) RUGD→RELLIS; and (c) RUGD→MESH.



(a)



(b)

Fig. 10: Using minmax can cause the collapse of training.

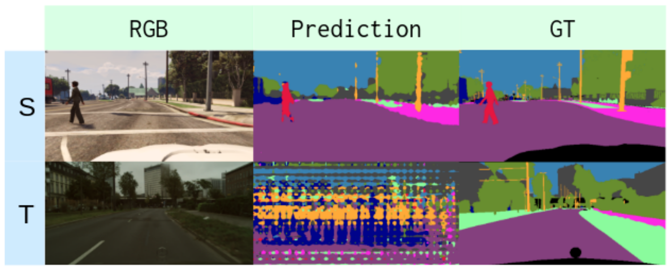


Fig. 11: An example of collapsed trained model using minmax.

*2) RUGD→RELLIS:* We show quantitative results of RUGD→RELLIS in Table. II, where only 5 classes[1] are evaluated. It shows the same trend as Table. I. Both tables show that CA has the negative transfer (compared with SO) issue for either sim2real or real2real UDA. However, if we constrain the training of CA with DA, as in our proposed model, then the performance will be remarkably improved. Some qualitative results are shown in Fig. 7.

*3) RUGD→MESH:* Our MESH dataset contains only unlabeled images that restrict us to show only a qualitative comparison for the UDA of RUGD→MESH, as shown in Fig. 8. We have collected data in winter forest environments, which are significantly different than the images in the source domain (RUGD) - collected in a different season, e.g., summer or spring. These cross-season scenarios make the prediction more challenging. However, it is more practical to evaluate the UDA performance of cross-season scenarios, as we might have to deploy our robot at any time, even with extreme weather conditions, but our available datasets might be far from covering every season and every weather condition. From Fig. 8, we can still see the obvious advantages of our proposed CALI model over other baselines.

[1]This is because other classes (in Fig. 15) frequently appearing in source domain (RUGD) are extremely rare in target domain (RELLIS), hence no prediction for those classes occurs especially considering the domain shift.

### *D. Discussions*

In this section, we aim to discuss our model behaviors in more details. Specifically, first we will explain the advantages of CALI over CA from the perspective of training process. Second, we will show the vital influence of mistakenly using wrong order of adversarial training.

The most important part in CA is the discrepancy between the two classifiers, which is the only training force for the functionality of CA. It has been empirically studied in [31] that the target prediction accuracy will increase as the target discrepancy is decreasing, hence the discrepancy is also an indicator showing if the training is on the right track. We compare the target discrepancy changes of CALI and our baseline CA in Fig. 9, where the curves for the three UDA scenarios are presented from (a) to (c) and we only show the data before iteration 30k. It can be seen that before around iteration 2k, the target discrepancy of both CALI and CA are drastically decreasing, but thereafter, the discrepancy of CA starts to increase. On the other hand, if we impose a DA constraint over the same CA (iteratively), leading to our proposed CALI, then the target discrepancy will be decreasing as expected. This validates that integrating DA and CA will make the training process of CA more stable, thus improving the target prediction accuracy.

As mentioned in Algorithm 1, we have to use adversarial training order of $\max_{\psi_D} \min_{\phi_G}$, instead of $\min_{\phi_G} \max_{\psi_D}$. The reason for this is related to our designed net structure. Following the guidance of Eq. (11), we use the same input to the two classifiers and the domain discriminator, hence the discriminator has to receive the intermediate-level feature as the input. If we use the order of $\min_{\phi_G} \max_{\psi_D}$ in CALI, then the outputs of the discriminator will be like Fig. 10(a), where the domain discriminator of CALI will quickly converge to the optimal state and it can accurately discriminate if the feature is from source or target domain. In this case, the adversarial

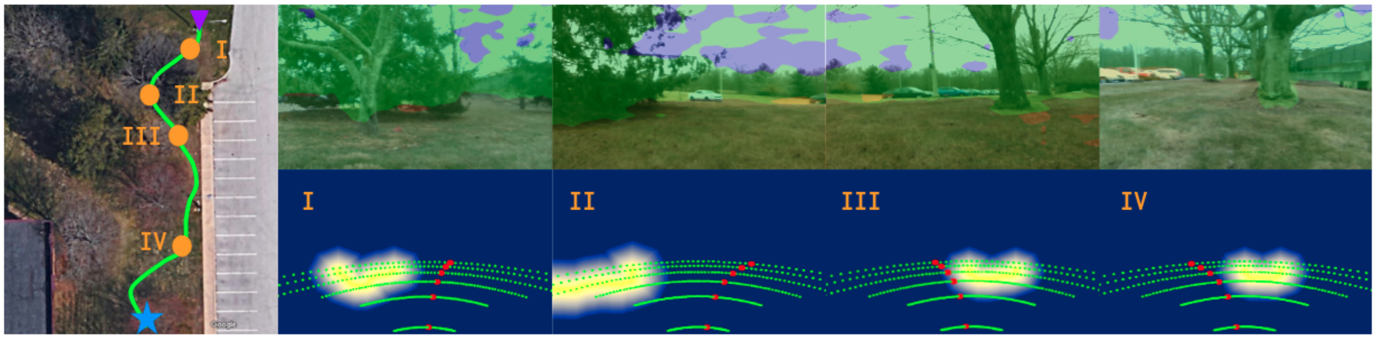


Fig. 12: Navigation behaviors in MESH#1 environment. The left-most column: top-down view of the environment; Purple triangle: the starting point; Blue star: the target point; We also show the segmentation (top row) and planning results (bottom row) at four different moments during the navigation, as shown from the second column to the last one.

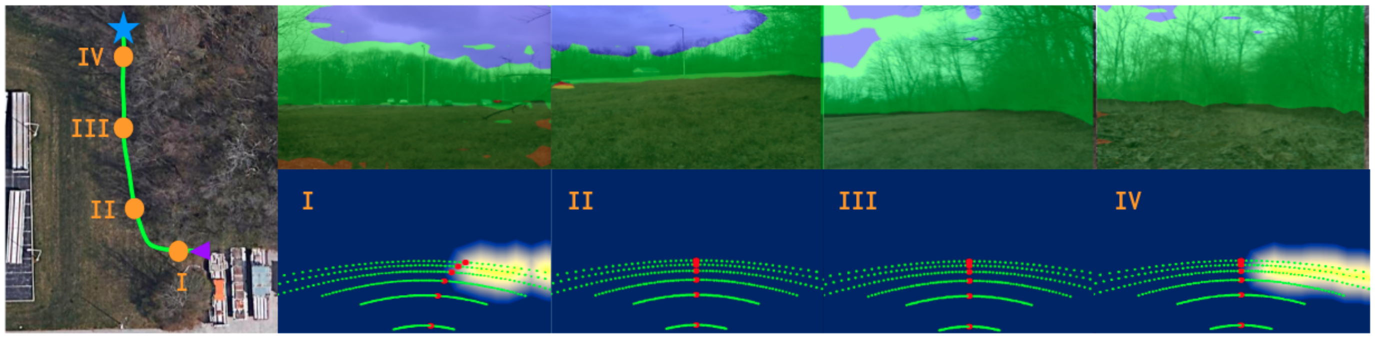


Fig. 13: Navigation behaviors in MESH#2 environment. Same legends with Fig. 12.

loss for updating the feature extractor will be near 0, hence the whole training fails, which is validated by changes of the target discrepancy curve, as shown in Fig. 10(b), where the discrepancy value is decreasing in a small amount in the first few iterations and then quickly increase to a high level that shows the training is divergent and the model is collapsed. This is also verified by the prediction results at (and after) around iteration 1k, as shown in Fig. 11, where the first row is the source images while the second row is the target images.

### E. Navigation Missions

To further show the effectiveness of our proposed model for real deployments, we build a navigation system by combining the proposed CALI (trained with RUGD→MESH set-up) segmentation model with our visual planner. We test behaviors of our navigation system in two different forest environments (named MESH#1 in Fig. 12 and MESH#2 in Fig. 13), where our navigation system shows high reliability. In navigation tasks, the image resolution is $[400, 300]$, and the inference time for pure segmentation inference is around 33 frame per second (FPS). However, since a complete perception system requires several post-processing steps, such as navigability definition, noise filtering, Scaled Euclidean Distance Field computation, motion primitive evaluation and so on, the response time for the whole perception pipeline (in python) is around 8 FPS without any engineering optimization. The inference of segmentation for navigation is performed on an Nvidia Tesla T4 GPU. We set the linear velocity as $0.3m/s$ and control the angular velocity to track the selected motion primitive. The path length is $32.26m$ in Fig. 12 and $28.63m$ in Fig. 13. Although the motion speed is slow in navigation tasks, as a proof of concept and with a very basic motion planner, the system behavior is as expected, and we have validated that the proposed CALI model is able to well accomplish the navigation tasks in unstructured environments.

## VI. Conclusion

We present CALI, a novel unsupervised domain adaptation model specifically designed for semantic segmentation, which requires fine-grained alignments in the level of class features. We carefully investigate the relationship between a coarse alignment and a fine alignment in theory. The theoretical analysis guides the design of the model structure, losses, and training process. We have validated that the coarse alignment can serve as a constraint to the fine alignment and integrating the two alignments can boost the UDA performance for segmentation. The resultant model shows significant advantages over baselines in various challenging UDA scenarios, e.g., sim2real and real2real. We also demonstrate the proposed segmentation model can be well integrated with our proposed visual planner to enable highly efficient navigation in off-road environments.

## Acknowledgement

This work has been partially supported by NSF grant No. 2006886 and ARL grant W911NF-20-2-0099. We are also grateful for the computational resources provided by the Amazon AWS Machine Learning Research Award.

## References

[1] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[3] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. 2008.

[4] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[5] Romain Brégier, Frédéric Devernay, Laetitia Leyrit, and James L Crowley. Defining the pose of any 3d rigid object and an associated distance. *International Journal of Computer Vision*, 126(6):571–596, 2018.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40 (4):834–848, 2017.

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[13] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018.

[14] Thomas M Howard and Alonzo Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, 26(2): 141–166, 2007.

[15] Thomas M Howard, Colin J Green, Alonzo Kelly, and Dave Ferguson. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *Journal of Field Robotics*, 25(6-7):325–345, 2008.

[16] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. *arXiv preprint arXiv:2111.14887*, 2021.

[17] Peng Jiang, Philip Osteen, Maggie Wigness, and Srikanth Saripalli. Rellis-3d dataset: Data, benchmarks and analysis, 2020.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen Change Loy, and Xiaoou Tang. Deep learning markov random field for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 40(8):1814–1828, 2017.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[21] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2507–2516, 2019.

[22] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[23] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16*, pages 415–430. Springer, 2020.

[24] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[25] Frank C Park. Distance metrics on the rigid-body

motions with applications to mechanism design. 1995.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[27] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[28] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[29] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[31] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3723–3732, 2018.

[32] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7472–7481, 2018.

[33] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2517–2526, 2019.

[34] Haoran Wang, Tong Shen, Wei Zhang, Ling-Yu Duan, and Tao Mei. Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In *European Conference on Computer Vision*, pages 642–659. Springer, 2020.

[35] Maggie Wigness, Sungmin Eum, John G Rogers, David Han, and Heesung Kwon. A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments. In *International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[36] Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

[37] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *arXiv preprint arXiv:2105.15203*, 2021.

[38] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[39] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 2020–2030, 2017.

[40] Youshan Zhang. A survey of unsupervised domain adaptation for visual recognition. *arXiv preprint arXiv:2112.06745*, 2021.

[41] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[42] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021.

[43] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.

## Appendix

### A. *Proof of **Theorem 2***

For a hypothesis $h$,

$$\begin{aligned}
\epsilon_T(h) &\leq \epsilon_T(h^*) + \epsilon_T(h,h^*) \\
&= \epsilon_T(h^*) + \epsilon_S(h,h^*) - \epsilon_S(h,h^*) + \epsilon_T(h,h^*) \\
&\leq \epsilon_T(h^*) + \epsilon_S(h,h^*) + |\epsilon_T(h,h^*) - \epsilon_S(h,h^*)| \\
&\leq \epsilon_T(h^*) + \epsilon_S(h,h^*) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T) \\
&\leq \epsilon_T(h^*) + \epsilon_S(h) + \epsilon_S(h^*) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T) \\
&= \epsilon_S(h) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T) + \epsilon_S(h^*) + \epsilon_T(h^*) \\
&= \epsilon_S(h) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T) + \lambda \\
&= \epsilon_S(h) + \sup_{h,h^{'}\in\mathcal{H}} |\mathrm{P}_{\mathbf{x}\sim\mathcal{D}_S}\left[h(\mathbf{x}) \neq h^{'}(\mathbf{x})\right] - \\
&\qquad\qquad \mathrm{P}_{\mathbf{x}\sim\mathcal{D}_T}\left[h(\mathbf{x}) \neq h^{'}(\mathbf{x})\right]| + \lambda \\
&= \epsilon_S(h) + \sup_{g\in\mathcal{H}\Delta\mathcal{H}} |\mathrm{P}_{\mathbf{x}\sim\mathcal{D}_S}[g(\mathbf{x}) = 1)] - \\
&\qquad\qquad \mathrm{P}_{\mathbf{x}\sim\mathcal{D}_T}[g(\mathbf{x}) = 1]| + \lambda \\
&= \epsilon_S(h) + \sup_{g\in\mathcal{H}\Delta\mathcal{H}} |\mathrm{P}_{\mathbf{x}\sim\mathcal{D}_S}[g(\mathbf{x}) = 1)] + \\
&\qquad\qquad \mathrm{P}_{\mathbf{x}\sim\mathcal{D}_T}[g(\mathbf{x}) = 0] - 1| + \lambda \\
&\leq \epsilon_S(h) + \sup_{g\in\mathcal{H}\Delta\mathcal{H}} |\mathrm{P}_{\mathbf{x}\sim\mathcal{D}_S}[g(\mathbf{x}) = 1)] + \\
&\qquad\qquad \mathrm{P}_{\mathbf{x}\sim\mathcal{D}_T}[g(\mathbf{x}) = 0]| - \inf_{g\in\mathcal{H}\Delta\mathcal{H}} 1 + \lambda \\
&= \epsilon_S(h) + \sup_{g\in\mathcal{H}\Delta\mathcal{H}} |\mathrm{P}_{\mathbf{x}\sim\mathcal{D}_S}[g(\mathbf{x}) = 1)] + \\
&\qquad\qquad \mathrm{P}_{\mathbf{x}\sim\mathcal{D}_T}[g(\mathbf{x}) = 0]| + \lambda - 1 \\
&= \epsilon_S(h) + \frac{1}{2}d_{\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T) + 1 + \lambda - 1 \\
&= \epsilon_S(h) + \frac{1}{2}d_{\mathcal{H}}(\mathcal{D}_S,\mathcal{D}_T) + \lambda,
\end{aligned} \tag{22}$$

where $\lambda = \epsilon_S(h^*)+\epsilon_T(h^*)$ and $h^*$ is the ideal joint hypothesis (see the **Definition 2** in Section 4.2 of [2]).

We have the $4^{th}$, and the $8^{th}$ line because of the **Lemma 3** [2]; the $5^{th}$ line because of the **Theorem 2** [2]; the last second line because of the **Lemma 2** [2]. We have the $11^{th}$ line because $\sup|f_1 - f_2| = \sup f_1 - \inf f_2 \leq \sup|f_1| - \inf f_2$. ∎

### B. *Remapping of Label Space*

We regroup the original label classes according to the semantic similarities among classes. In GTA5 and Cityscapes, we cluster the building, wall and fence as the same category; traffic light, traffic sign and pole as the same group; car, train. bicycle, motorcycle, bus and truck as the same class; and treat the person and rider as the same one. See Fig. 14. Similarly, we also have regroupings for classes in RUGD and RELLIS, as can be seen in Fig. 15.

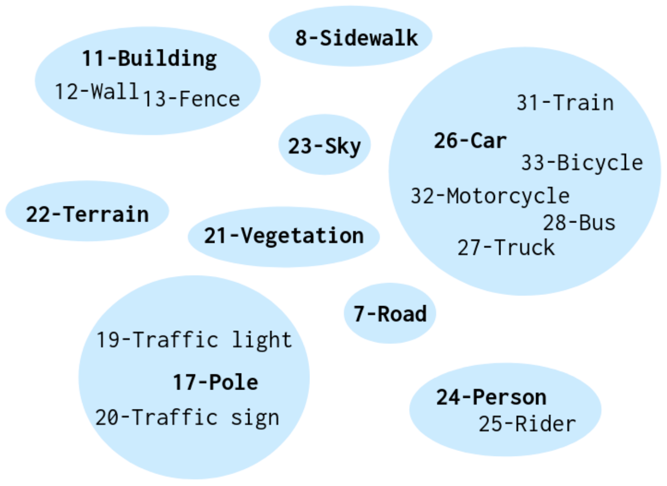


Fig. 14: Lable remapping for GTA5→Cityscapes. Name of each new group is marked as bold.

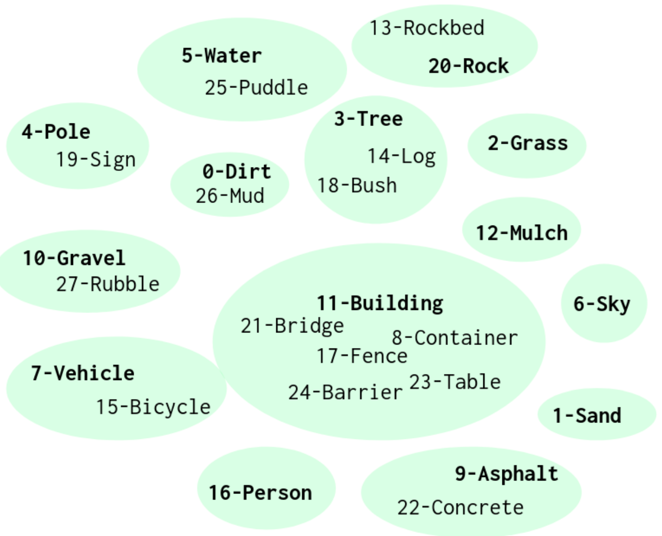


Fig. 15: Lable remapping for RUGD→RELLIS and RUGD→MESH. Name of each new group is marked as bold.