공학석사 학위논문

Autonomous Navigation by Domain Adaptation with Generative Adversarial Networks

생성적 대립 신경망과 도메인 적응 기법을 이용한
자율 주행 방법 연구

2018년 8월

서울대학교 대학원

전기·컴퓨터공학부

홍 용 준

Autonomous Navigation by Domain Adaptation with Generative Adversarial Networks

생성적 대립 신경망과 도메인 적응 기법을 이용한
자율 주행 방법 연구

지도교수 윤 성 로
이 논문을 공학석사 학위논문으로 제출함

2018년 8월

서울대학교 대학원

전기 컴퓨터 공학부

홍 용 준

홍용준의 공학석사 학위 논문을 인준함

2018년 8월

위 원 장: _____
부위원장: _____
위 원: _____

Abstract

Generative adversarial networks (GANs) has received popular attention because of their great potential for important subjects in the machine learning field, such as semi-supervised learning, image translation and domain adaptation. Specifically, GANs are capable of generating more sharp and realistic synthetic data than prior generative models without any probability distribution assumptions.

In this paper, we apply GANs to autonomous navigation, which has been a unique subject for various fields such as robotics, mechanical engineering and machine learning. Prior approaches with deep neural networks to autonomous navigation are mostly gathering lots of labeled real data and training neural networks by supervised-learning method. However, large labeled datasets are indispensable for these approaches, and labeled real data is hard to acquire, laborious and often erroneous.

We address this problem by domain adaptation with GANs. In simulator environment, various kinds of labels can be easily taken. By exploiting labeled synthetic data from simulator environment, we alleviate labeling issue in real environment through domain adaptation. In addition to that, by GANs, we try to make synthetic data look like real data, so that autonomous navigation can be successfully done without label information of real data.

keywords: Generative adversarial Networks, Autonomous navigation, Domain adaptation,

student number: 2014-21634

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Recent big advance in machine learning with deep neural networks can be possible because of enormous amount of data. Particularly, supervised learning method with labeled data shows great success in various fields. However, lots of labeled datasets exist mostly one-sided to specific tasks related to images, and getting corresponding labels to input data is laborious, expensive and often erroneous. Therefore, generating synthetic data look like real data has been a popular subject in that real-like synthetic data can replace real data and attaching corresponding labels artificially is cheaper and easier than at real environment.

Generative adversarial networks (GAN) [19] has been focused on greatly because of its capability of generating more real-like data than other previous generative model such as variational auto-encoder [14]. Specifically, GAN adopt the discriminator, which distinguishing real data and synthetic data. With concept of the discriminator, GAN can produce real-like data without any probability distribution assumptions [28].

Even though GAN are leveraged to various tasks recently, we concentrate on how GAN is applied to domain adaptation. Domain adaptation refers to adapting one domain to another so that one can use the data from the former domain to train a model for the tasks in the latter domain [53]. Intuitively, to successfully transfer knowledges, differences between two domains should be minimized, and this situation is highly ac-

ceptable to a GAN concept in that GAN also aims to minimize distance between real data probability and synthetic data probability [28].

In this dissertation, we utilize domain adaptation with GAN to autonomous navigation [76]. Most prior studies about autonomous navigation required large labeled datasets, corresponding steering information for input image, for example. We address this labeling issue by getting corresponding labels from simulator environment, which is much easier than getting labels from real environment. Since simulator environment and real environment are much different visually and perceptually, we apply GAN for domain adaptation to minimize environments' difference. By doing that, autonomous navigation task can be accomplished without labels of real environment. The source code for the proposed method is available at `https://github.com/yjhong89/DA_CIL`. Note that this dissertation is based on the following papers:

- Jaeyoon Yoo, Yongjun Hong and Sungroh Yoon. Domain Adaptation Using Adversarial Learning for Autonomous Navigation. Under review
- Youngjun Hong, Uiwon Hwang, Jaeyoon Yoo and Sungroh Yoon. How Generative Adversarial Networks and Its Variants Work: An Overview of GAN. Under review

We summarize two major advantages of our approach over previous works.

First, our method reduces the labeling issue through domain adaptation by generating realistic images from simulated images. We apply domain adaption to a simulator environment to train a navigation model that works reasonably well in a real environment using simulated data without any real labeled data. Simulator environments do not suffer from labeling issues [9] because we can make any kinds of labels easily in simulator environment.

Second, our method requires only a monocular camera. Many existing approaches to autonomous navigation depend on various sensors, including global positioning system (GPS) sensors and depth sensors [16, 37, 43, 70]. Such approaches often raise

issues such as the additional cost and complexity, which are the main problem with using multiple sensors. They may increase the actuator burden, especially in the case of micro aerial vehicles. Additionally, sensors have their own limitations. For example, a GPS sensor does not usually operate well in an indoor environment or in a forest. Compared with other sensors, a monocular camera is light and cost-effective. Images can be acquired in real time with a camera in most well-lighted areas. Moreover, the latest deep learning techniques, having shown remarkable achievements in the image domain [41], may compensate for the lack of other sensors during autonomous navigation.

Chapter 2

BACKGROUND

2.1 Generative Adversarial Networks

[19] proposed generative adversarial networks (GANs), which is composed of two components, the generator G and the discriminator D. G produces fake samples from the latent variable z whereas D takes both fake samples and real samples and decides whether its input is real or fake. D produces higher probability as it determines its input is more likely to be real. G and D oppose each other to achieve their individual goals, so the adversarial term is coined. When this adversarial behavior is formulated as the objective function, GAN solves minimax Equation 2.1 with parametrized networks G and D, which both have full capacity. $p_d(x)$ and $p_z(z)$ in Equation 2.1 denote a real data probability distribution defined in the data space X and a probability distribution of the latent variable z defined on the latent space Z, respectively. It should be noted that G maps the latent variable z from Z into the element of X, whereas D takes an input x and distinguishes whether x comes from real samples or from G. The objective function is thus represented as follows:

$$\min_G \max_D V(G,D) = \min_G \max_D \mathbb{E}_{x\sim p_d}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z))] \quad (2.1)$$

where $V(G,D)$ is a binary cross entropy function which is commonly used in binary

classification problems.

As D wants to classify real or fake samples, $V(G, D)$ is a natural choice for an objective function in aspect of the classification problem. From D's perspective, if a sample comes from real data, D will maximize its output and if a sample comes from G, D will minimize its output. Meanwhile, G wants to deceive D so it tries to maximize D's output when a fake sample is presented to D. Consequently, D tries to maximize $V(G, D)$ while G tries to minimize $V(G, D)$, and which is where the minimax relationship in Equation 2.1 comes from. Figure 2.1 shows an outline of the GAN where D produces higher probability (near 1) when it decides its input is more likely to come from real data.

Theoretically, the Nash equilibrium for Equation 2.1 can be achieved through the following procedure. First, D is trained to obtain an optimal discriminator for a fixed G, and then G tries to fool D to enable $D(G(z))$ produce a high probability. By iteratively optimizing such a minimax problem, D cannot discriminate whether its input is real or fake anymore because $p_d(x) = p_\theta(x)$ has been achieved, and $D(x)$ produces a probability of $\frac{1}{2}$ for all real and fake samples where $p_\theta(x)$ stands for a probability distribution of generated samples with a generator parameter θ. Particularly, [19] shows that solving Equation 2.1 is equivalent to minimizing the Jensen Shannon Divergence (JSD) between $p_d(x)$ and $p_\theta(x)$. JSD between $p_d(x)$ and $p_\theta(x)$ is defined as follows:

$$JSD(p_d(x)||p_\theta(x)) = \frac{1}{2}(\mathbb{E}_{x\sim p_d(x)}[\log \frac{p_d(x)}{\frac{p_d(x)+p_\theta(x)}{2}}] + \mathbb{E}_{x\sim p_\theta(x)}[\log \frac{p_\theta(x)}{\frac{p_d(x)+p_\theta(x)}{2}}]) \tag{2.2}$$

Note that JSD is symmetrical (*i.e.*, $JSD(p_d(x)||p_\theta(x)) = JSD(p_\theta(x)||p_d(x))$).

2.1.1 Integral Probability Metric

Integral probability metric (IPM) defines a critic function f, which belongs to a specific function class $\mathcal{F}$, and IPM is defined as a maximal measure between two arbitrary distributions in the frame of f. In a compact space $\mathcal{X} \subset R^d$ where d is a dimension,

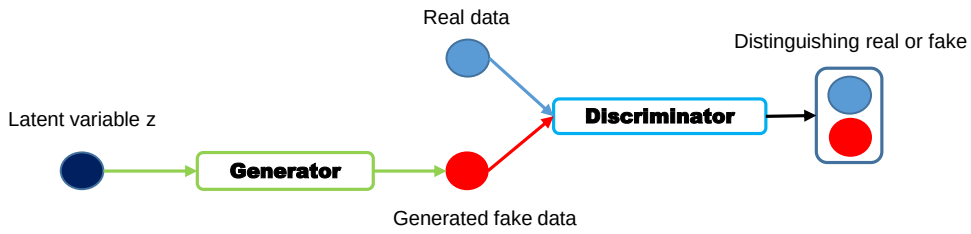

Figure 2.1: A brief overview of the GAN.

let $\mathcal{P}(\mathcal{X})$ denote the probability measures defined on $\mathcal{X}$. We now define IPM metrics between two distributions p_d, $p_\theta \in \mathcal{P}(\mathcal{X})$ as follows:

$$d_{\mathcal{F}}(p_d, p_\theta) = sup_{f \in \mathcal{F}} \ \mathbb{E}_{x \sim p_d}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)] \tag{2.3}$$

As shown in Equation 2.3, IPM metric $d_{\mathcal{F}}(p_d, p_\theta)$ defined on $\mathcal{X}$ determines a maximal distance between $p_d(x)$ and $p_\theta(x)$ with functions belonging to $\mathcal{F}$. From Equation 2.3, we note that $\mathcal{F}$ is a set of measurable, bounded, real-valued functions in that if they are not, $d_{\mathcal{F}}(p_d, p_\theta)$ would not be properly defined. How to define $\mathcal{F}$ determines various distances and their properties. We consider the function class $\mathcal{F}_{v,w}$ whose elements are the critic f, which scores its input as a single value so that it can be defined as an inner product of parameterized neural networks $\Phi_w(x)$ and a linear output activation function v. It should be noted that w belongs to parameter space Ω that forces the function space to be bounded. With the definition of the function class in Equation 2.4, we can reformulate Equation 2.3 as the following equations:

$$\mathcal{F}_{v,w} = \{f(x) =< v, \Phi_w(x) > | v \in R^m, \Phi_w(x) : \mathcal{X} \to R^m\} \tag{2.4}$$

$$d_{\mathcal{F}_{v,w}}(p_d, p_\theta) = sup_{f \in \mathcal{F}_{v,w,p}} \mathbb{E}_{x \sim p_d} f(x) - \mathbb{E}_{x \sim p_\theta} f(x) \tag{2.5}$$

$$= \max_{w \in \Omega, v} < v, \mathbb{E}_{x \sim p_d} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) > \tag{2.6}$$

$$= \max_{w \in \Omega} \max_{v} < v, \mathbb{E}_{x \sim p_d} \Phi_w(x) - \mathbb{E}_{x \sim p_\theta} \Phi_w(x) > \tag{2.7}$$

From Equation 2.7, how we restrict v determines the semantic meanings of corresponding IPM metrics. From now on, we discuss IPM metric variants such as the Wasserstein metric, maximum mean discrepancy (MMD), and the Fisher metric based on Equation 2.7.

Wasserstein GAN

Wasserstein GAN (WGAN) [3] proposes a significant result regarding the distance between $p_d(x)$ and $p_\theta(x)$. In a GAN, we are likely to learn the generator function g_θ that transforms an existing function z into $p_\theta(x)$ rather than directly learning the probability distribution $p_d(x)$ itself. From this view, a measure between $p_\theta(x)$ and $p_d(x)$ is necessary to train g_θ and WGAN suggests the Earth-Mover (EM) distance which is also called as Wasserstein distance, as a measure of two distributions. The Wasserstein distance is defined as follows:

$$W(p_d, p_\theta) = inf_{\gamma \in J(p_d, p_\theta)} \mathbb{E}_{(x,y) \sim \gamma}[\|x - y\|] \tag{2.8}$$

where $J(p_d, p_\theta)$ denotes the set of all joint distributions and marginals of $\gamma(x, y)$, which are $p_d(x)$ and $p_\theta(x)$ respectively.

Probability distributions can be interpreted as the amount of mass they place at each point, and EM distance is the minimum total amount of work transforming $p_d(x)$ into $p_\theta(x)$. From this view, calculating the EM distance is equal to finding a transport plan $\gamma(x, y)$, which defines how we distribute the amount of mass from $p_d(x)$ over $p_\theta(y)$. Therefore, a marginality condition can be interpreted in that $p_d(x) = \int_y \gamma(x, y) dy$ is the amount of mass to move from point x and $p_\theta(y) = \int_x \gamma(x, y) dx$

is the amount of mass to be transported to point y. Because work is defined as the amount of mass times the distance it moves, we have to multiply the Euclidean distance $\|x-y\|$ by $\gamma(x,y)$ at each point x, y and the minimum amount of work is derived (Equation 2.8).

The benefit of the EM distance over other metrics is that it is a more sensible objective function when learning distributions supported by low-dimensional manifolds. The article on WGAN shows that EM distance is the weakest convergent metric in that the converging sequence under the EM distance does not converge under other metrics and it is continuous and differentiable almost everywhere under the Lipschitz condition, which general feed-forward neural networks satisfy. This benefit can be intuitively thought of in that the EM distance is a more tolerable measure than other distances such as JSD and Kull-back Liebler Divergence (KLD).

As the $\inf$ term in Equation 2.8 is highly intractable, it is converted into a more suitable equation via Kantorovich-Rubinstein duality with the 1-Lipschitz function class [56], [25] for a set of functions such that for the critic f. A duality of Equation 2.8 is as follows:

$$W(p_d, p_\theta) = sup_{|f|_L \leq 1} \mathbb{E}_{x \sim p_d}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)] \tag{2.9}$$

Consequently, if we parametrize the critic f with w to be a 1-Lipschitz function, it transforms to solving the minimax problem in that we train f_w first to approximate $W(p_d, p_\theta)$ by searching for the maximum as in Equation 2.9, and minimize such approximated distance by optimizing the generator g_θ. To guarantee f_w to be a Lipschitz function, weight clipping is conducted for every update of w to ensure the parameter space of w lies in a compact space. It should be noted that $f(x)$ is called the critic because it is not explicitly classifying inputs as the discriminator, rather it scores its input.

[23] points out that weight clipping for the critic while training WGAN incurs a pathological behavior of the discriminator and suggests adding a penalizing term of

the gradient's norm instead of weight clipping. It shows that guaranteeing the Lipschitz condition for the critic via weight clipping represents a very limited subset of all Lipschitz functions; this biases the critic toward a simpler function. Weight clipping also creates a gradient problem as it pushes weights to the extremes of the clipping range. Instead of weight clipping, it suggests adding a gradient penalty term to Equation 2.9 with the purpose of directly constraining the gradient of the critic, which the Lipschitz condition represents.

Fisher GAN

Instead of using standard IPM in Equation 2.3, Fisher GAN [51] emphasizes a standardized mean discrepancy which naturally induces a data-dependent constraint by the following equations:

$$d_{\mathcal{F}}(p_d, p_\theta) = \sup_{f \in \mathcal{F}} \frac{\mathbb{E}_{x \sim p_d} f(x) - \mathbb{E}_{x \sim p_\theta} f(x)}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_d} f^2(x) + \mathbb{E}_{x \sim p_\theta} f^2(x))}} \tag{2.10}$$

$$= \sup_{f \in \mathcal{F}, \frac{1}{2}\mathbb{E}_{x \sim p_d} f^2(x) + \mathbb{E}_{x \sim p_\theta} f^2(x) = 1} \mathbb{E}_{x \sim p_d} f(x) - \mathbb{E}_{x \sim p_\theta} f(x) \tag{2.11}$$

Equation 3.3 is motivated from Fisher linear discriminant analysis (FLDA) [72] in that it not only maximizes the mean difference but also reduces the total with-in class variance of two distributions. Equation 3.5 follows from the constraining numerator of Equation 3.3 to be 1. It is also, as other IPM metrics, interpreted as a mean feature matching problem but with different constraints. With the definition of Equation 2.4, Fisher GAN derives another mean feature matching problem with second order moment constraint. A mean feature matching problem derived from the FLDA concept is

as follows:

$$d_{\mathcal{F}}(p_d, p_\theta) = \max_{w \in \Omega} \max_{v} \frac{< v, \mathbb{E}_{x \sim p_d} \Phi(x) - \mathbb{E}_{x \sim p_\theta} \Phi(x) >}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_d} f^2(x) + \mathbb{E}_{x \sim p_\theta} f^2(x))}} \tag{2.12}$$

$$= \max_{w \in \Omega} \max_{v} \frac{< v, \mathbb{E}_{x \sim p_d} \Phi(x) - \mathbb{E}_{x \sim p_\theta} \Phi(x) >}{\sqrt{\frac{1}{2}(\mathbb{E}_{x \sim p_d} v^T \Phi(x) \Phi(x)^T v + \mathbb{E}_{x \sim p_\theta} v^T \Phi(x) \Phi(x)^T v)}} \tag{2.13}$$

$$= \max_{w \in \Omega} \max_{v} \frac{< v, \mu_w(p_d) - \mu_w(p_\theta) >}{\sqrt{v^T (\frac{1}{2} \sum_w (p_d) + \frac{1}{2} \sum_w (p_\theta) + \gamma I_m) v}} \tag{2.14}$$

$$= \max_{w \in \Omega} \max_{v, v^T (\frac{1}{2} \sum_w (p_d) + \frac{1}{2} \sum_w (p_\theta) + \gamma I_m) v = 1} < v, \mu_w(p_d) - \mu_w(p_\theta) > \tag{2.15}$$

where $\mu_w(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}} \Phi_w(x)$ denotes an embedding mean and $\sum_w(\mathcal{P}) = \mathbb{E}_{x \sim \mathcal{P}} \Phi_w(x) \Phi_w^T(x)$ denotes an embedding covariance for the probability $\mathcal{P}$.

Equation 2.13 can be induced using the inner product of f defined as in Equation 2.4. γI_m of Equation 2.14 is an m by m identity matrix that guarantees a numerator of the above equations not to be zero. In Equation 2.15, Fisher GAN aims to find the embedding direction v which maximizes the mean discrepancy while constraining it to lie in a hyper-ellipsoid as $\frac{1}{2} \sum_w (p_d) + \frac{1}{2} \sum_w (p_\theta) + \gamma I_m) v = 1$ represents. It naturally derives the Mahalanobis distance [13] which is defined as distance between two distributions given a positive definite matrix such as a covariance matrix of each class. More importantly, Fisher GAN has advantages over WGAN in that it does not impose a data independent constraint such as weight clipping which makes training too sensitive on the clipping value and also has computational benefit over the gradient penalty method in Improved WGAN [60] as the latter must compute gradients of the critic while Fisher GAN computes covariances.

2.1.2 Image to Image Translation

Image translation is a task translating images from domain X to images from another domain Y. Mainly, translated images have dominants characteristic of domain Y main-

taining its attributes before being translated. As in classical machine learning, there are supervised and unsupervised techniques to conduct image translation.

Image translation with paired images can be regarded as supervised image translation in that image $x \in$ domain X always has the target image $y \in$ domain Y. [32] suggests an image translation method with paired images using the U-NET architecture [57] which is widely used for biomedical image segmentation. It adopts a conditional GAN framework [50] in that its generator produces a corresponding target image conditioned on an input image. In contrast, [71] adds the perceptual loss between a paired data (x, y) to the generative adversarial loss to transform input image x into ground-truth image y. Instead of using the pixel-wise loss to push the generated image toward the target image, it uses hidden layer discrepancies of the discriminator between an input image x and ground truth image y. It tries to transform x to y to be perceptually similar by minimizing perceptual information discrepancies from the discriminator.

Image translation in an unsupervised manners is that given unpaired data from two domains, it learns a mapping between two domains without supervision. [79] and [38] aim to conduct unpaired image-to-image translation using a cyclic consistent loss term. With a sole translator G: $X \rightarrow Y$, there can be many possible mappings from $X \rightarrow Y$ so that meaningless translation can occur or mode collapse occurs in which several images in X are mapped to one image in Y. They adopt another inverse translator T: $Y \rightarrow X$ and introduce the cyclic consistency loss which encourages $T(G(x)) \approx x$ and $G(T(y)) \approx y$. Cyclic loss term with $L1$ norm reconstructs $T(G(x))$ to be an original x and $G(T(y))$ to be an original y so that each direction of translation finds plausible mapping between the two domains. Figure 2.2 shows the baseline of unpaired image translation where D denotes the discriminator in each domain and R is the reconstruction loss for the cyclic consistency.

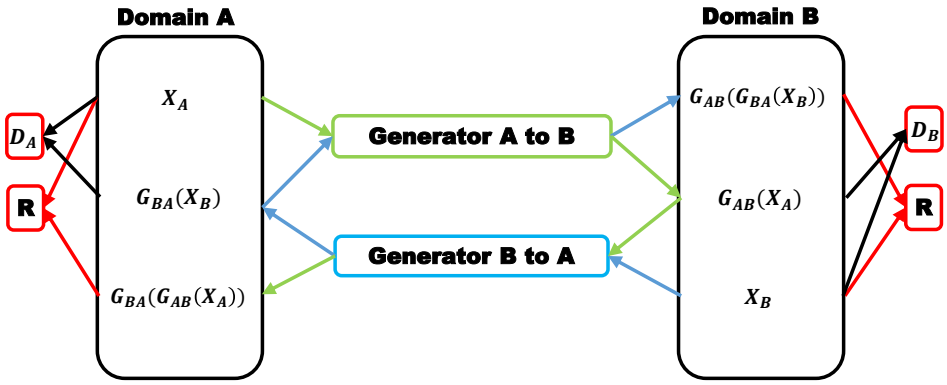

Figure 2.2: A figure of unpaired image translation.

2.2 Domain Adaptation

Domain adaptation is a type of transfer learning which tries to adapt data from one domain (*i.e.*, the source domain) into another domain (*i.e.*, the target domain), while the classification task performance is preserved in a target domain [52]. The reason why domain adaptation gets popular attention is that there are lots of situation such that there are the source domain where input data has corresponding label information and the target domain where input data does not have corresponding label information. These situation widely happens in various field because labeled data is difficult to obtain in the real world. Domain adaptation can be considered as transferring prior knowledge of a known source domain to a target domain for doing tasks (*e.g.* classification) in the target domain without label information of the target domain. Formally, we assume that there is a joint probability distribution of input data, x, with its label, y, in each domain: $\mathbb{P}_S(x, y)$ in the source domain and $\mathbb{P}_T(x, y)$ in the target domain. These two joint distribution are defined over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are sets of a data space and a label space, respectively. Given the labeled source domain data, $(x_s, y_s) \sim \mathbb{P}_S(x, y)$, and unlabeled target domain data x_t, which comes from the

marginal distribution $\mathbb{P}_T(x)$ of $\mathbb{P}_T(x, y)$, domain adaptation aims to learn a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$, which properly fits on the posterior distribution, $\mathbb{P}_T(y|x)$ of $\mathbb{P}_T(x, y)$.

The simplest way to conduct domain adaptation is to not care about differences in the distributions of the two domains, which are called a domain discrepancy [45]. It trains a model only with the source domain data (x_s, y_s) and applies the model directly to the target domain without concerning domain discrepancy. In many cases, this method does not work well in the target domain because of the domain discrepancy [45]. The most intuitive way to overcome this issue is to extract a common representation space where distributions of the two domain are projected [2, 45, 61]. This approach attempts to obtain a domain invariant representation space where a projected distribution is invariant from the source domain to the target domain. It is considered that, as distributions of two domains on the representation space become closer, the domain discrepancy becomes smaller and the learned model performs satisfactorily on the target domain.

To find the common representation space, [7] and [29] proposed methods that performed re-weighting of samples from the source domain. [55] and [20] aimed to find a feature space transformation that maps a source domain distribution to a target domain distribution. While their methods attempted to match the feature distributions from the source and the target domain by re-weighting or geometric transformation, there are some trials that applied a GAN framework to reconstruct the common representation space [2, 8, 61]. These methods focus on leading the feature representations to be indistinguishable between the two domains while maintaining the capability of the task classifier via adversarial learning with a discriminator. These adversarial domain adaptation methods have benefits over [7, 29] in that they do not need to do complex sample re-weighting or feature space transformation. Rather, they change the common representation space itself dynamically by adopting the deep discriminative domain classifier (domain discriminator). By employing a GAN concept which naturally measures the distance between two domains, they aim to merge finding common features

and training the task classifier into one framework.

[2] approaches domain adaptation by maintaining discriminative and attaining domain invariance concepts. Maintaining discriminative means maintaining a label prediction ability to classify well in the target domain. Domain invariance stands for making generated features trained for classifying source data, not distinguishing whether it comes from the source domain or the target domain. As seen in Figure 13a, there are two components sharing a feature extractor. One is a label classifier which classifies labels of source data. The other is a domain discriminator which distinguishes where data comes from. We train a label classifier to classify labels of source data via a cross entropy loss. We also train a domain discriminator adversarially via gradient reversal layer (GRL), which induces the feature generator to extract domain invariant features. Therefore, the feature generator acts like the generator, which produces source-like features from the target domain, and the domain discriminator acts like the discriminator in that it discriminates source domain samples as real and target domain samples as fake with a binary cross entropy loss.

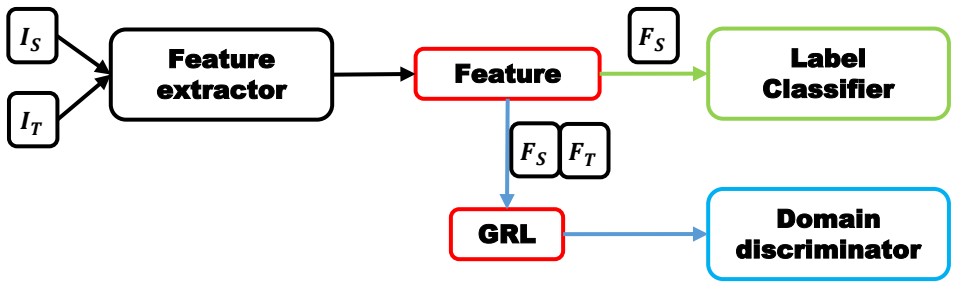

Figure 2.3: A figure of [2]. I_S and I_T stand for a source domain image and a target domain image and F_S, F_T are mean extracted source features, and target features, respectively

[61] aim to solve the target domain unlabeled data classification problem by incor-

porating [24]. As other adversarial approaches in the above paragraphs, it also consists of three parts: the feature generator generating domain invariant features, the critic estimating Wasserstein distance between generated features from source and target domains, and the classifier passing the supervised signal from the source domain to the target domain with cross entropy loss. It differs from [2, 8] in that it accesses the minimax problem with the Wasserstein critic to train the domain discriminator.

[8] take a little different approach to [2]. It attempts to adapt source images to be seen as if they were drawn from the target domain while [2] try to make generated features from both domains similar. There are three components in the suggested model similar to [2]. The generator maps a source image x_s and a noise vector z to a target adapted image x_f. The discriminator distinguishes a real target image x_t and a target adapted fake image x_f. The classifier assigns a class label for x_s. In addition, it adds a content similarity loss which reflects a prior knowledge of adaptation, which guides some important parts of x_s to be maintained while it is being transferred. Content similarity loss is defined as the pixel-wise mean squared error between source image and adapted image only for the masked area, which becomes a prior important knowledge of adaptation. It informs the adapted image that the masked area is necessarily kept during adaptation. More importantly, the method of [8] can check x_f is well transferred during training phase because x_f is an image itself, while [2, 61] can not visually check domain adaptation process because the common representation space can not be easily visualized.

[27] are similar to [8] in that they can also visualize transferred image, but they adopt two-way generators between the source domain and the target domain. Based on the concept of cyclic consistency loss [38, 79], they add semantic consistency loss and feature level adversarial loss from a classifier which pre-trained with source domain data.

[64] focus on situations where a classifier does not perform well in both domains and tackle this non-conservative domain adaptation with a cluster assumption [10],

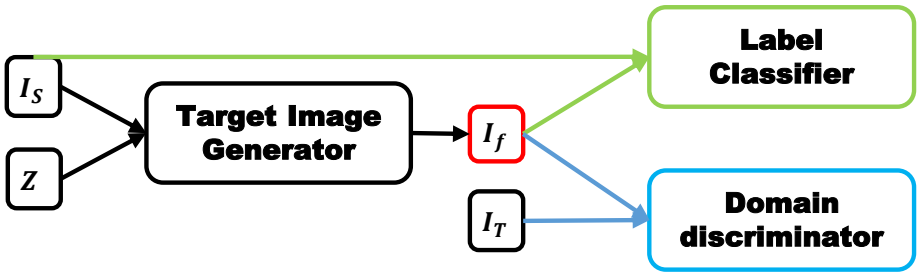

Figure 2.4: A figure of [8]. I_f and Z stand for a fake image and a random noise vector from a uniform distribution $\mathcal{U}[-1, 1]$.

which data samples in the same cluster share same label information. They impose a cluster assumption for target domain data by reducing conditional entropy loss. To control a classifier's behavior to be more robust near the data points, they incorporate locally-Lipschitz via natural gradient [21]. By making a parametrization-invariant classifier through natural gradient, they try to reject classifiers which violate the cluster assumption in the target domain, so reduce a classifier function space to be more feasible.

Domain adaptation has been applied in many tasks. [45] learned transferable policies for the UAV navigation using domain adaptation. They tried to decrease domain discrepancy defined with maximum mean discrepancy (MMD) [22] of a predefined kernel. MMD has shown its applicability in various tasks [15, 36, 54], yet is still highly dependent on the choice of the kernel [44] and may result in biased estimation, because of the sampling variance [22]. [8] and [63] used a simulator for domain adaptation with an adversarial learning framework. [9] adopted this framework for a grasping task. [63] refined eye rendered images to be more realistic by using GANs.

2.3 Autonomous Navigation

2.3.1 Non-learning Based Approaches for Autonomous Navigation

Non-learning based approaches can be interpreted as non-data driven approaches, which needs additional requirements, such as state feedback and state extraction from raw inputs with other sensors, leading to computationally expensive systems [42, 47, 48, 62, 80]. Moreover, these approaches depend on many sensor such as Global Positioning System (GPS), depth sensor and expensive laser (LIDAR) sensor [16, 43, 70], which are incompatible with our purpose. However, learning-based methods or data-driven methods produce a reactive controller by mapping raw inputs into an action. Data-driven methods are also known to seek a robust solution [5].

2.3.2 Learning Based Approaches for Autonomous Navigation

Learning based approaches for autonomous navigation mostly adopt supervised learning method, which incorporate raw input x and its corresponding y to train autonomous navigation system. Supervised learning based methods have been proposed to perform autonomous navigation using real images. [58] and [34] proposed DAGGER algorithm and GPS algorithm, respectively, and each group implemented a variant of supervised learning method to navigate the UAV in the outdoor environment. DAGGER progressively trains a moving policy, that expert iteratively labels appropriate actions for newly unvisited states. By doing thorough data aggregation, DAGGER tries to match a moving policy with a policy of expert. [6] achieved autonomous highway driving using the basic supervised method. [78] ,[75] and [11] improved basic supervised learning methods for a car and remote-control car driving. [17] and [65] viewed navigation as a classification problem instead of a regression problem, simplifying the navigation itself. These approaches have a limitation, that acquiring labels for real images is laborious or expensive. It causes these methods to be less applicable to various environments. Although automated the labeling process, they still needed additional

information that is difficult to acquire in the outdoor environment with only a monocular camera. We addressed this limitation by utilizing a simulator and by not applying supervised learning method on the real image domain directly.

2.3.3 Employing a Simulator for Autonomous Navigation

Using a simulator for training an autonomous navigation system has advantages. One can measure any physical property, including location and velocity in freedom. One need not care about damage to a vehicle. However, there is a domain discrepancy issue. To address this issue, [49] exploited a depth image, which is more invariant across the two environments. They used a reconstructed depth from a 2D image or a depth sensor to obtain the depth image. [59] suggested randomization be applied to the simulator with highly various features, such as light conditions, textures, and objects. They showed that their navigation model worked reasonably well in an indoor environment although they did not provide any real images during the training. However, using a depth sensor increases the cost. Reconstructing a depth image incurs additional error and high computing cost. In addition, the indoor environment is also relatively simple and plain, compared to the outdoors. Thus, it remains doubtful whether the method of [59] would work satisfactorily in the outdoors.

Chapter 3

METHOD

3.1 Problem Setup

Our goal is for a vehicle to navigate autonomously along train. We view navigation as a multi-class classification problem. As in [65], we train a deep learning model that classifies where the UAV is located with respect to the road (*i.e.*, left/center/right-lateral class) and where the UAV heads for along the road (*i.e.*, , left/straight/right head class), based on a raw image from a camera on the UAV. Figure 3.1 represents how we define lateral and head classes. For autonomous navigation, we utilize simulator images, real images, and correct lateral/head labels for simulator images. To address the gap between simulator images and real images and the absence of labels for real images, we adopt domain adaptation with an adversarial learning framework. We attempt to transfer a simulator image (*i.e.*, source domain) into a transferred image (*i.e.*, transferred domain) to look like a real image (*i.e.*, target domain) while maintaining characteristics of each class. Correct classification in the target domain can thus be achieved without any labels for the target domain images.

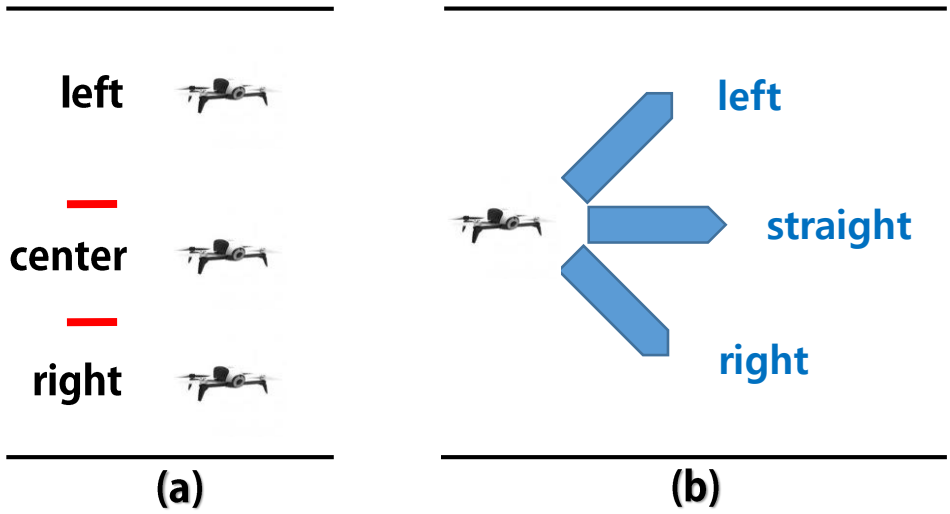

Figure 3.1: Illustration for (a) lateral classes and (b) head classes with respect to a road. Note that the lateral center class is slightly skewed to the right.

3.2 Domain Adaptation with Adversarial Learning

For autonomous navigation, we utilize simulator images with the labels for classification and real images without corresponding labels. There are two major obstacles to do the task. One is the absence of the labels for real images, and the other is the domain discrepancy caused by a gap between simulator images and real images. We try to address both issues at the same time by domain adaptation using the GAN framework.

An outline of our model is illustrated in Fig. 3.2. Our model is composed of two generators $G_{T\to S}, G_{S\to T}$, two discriminators D_T, D_S, and a task classifier C with trainable parameters $\theta_{g^T}, \theta_{g^S}, \theta_{d^T}, \theta_{d^S}, \theta_c$ respectively. The subscript T and S indicate the domain of the input image (source,target). The generator converts the input image from one domain to fit to the other domain. The discriminator takes the transferred images and the real images, and distinguishes the inputs between them. The classifier takes the transferred image and the ground-truth labels of the source image then learns

to classify the lateral and head classes of the transferred image, respectively during training. Though it is not shown in Figure 3.2, the classifier takes target domain images then classifies their classes during the test. These components are combined into one model, and they are trained simultaneously in an end-to-end manner. No labels for target images is required.

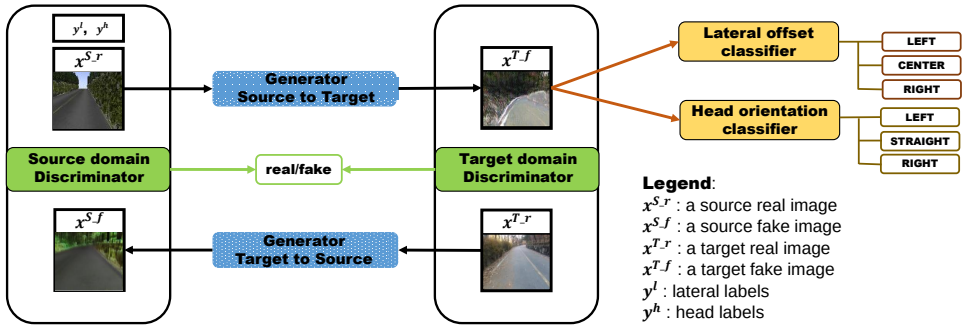

Figure 3.2: An overview of the proposed method.

3.3 Network Architecture

We adopt similar approach with [38, 79]. As shown in Figure 3.2, our model is composed of two generators and the two discriminator for the source domain and the target domain, and the classifier. A pair of the source domain generator and the source domain discriminator is responsible for making source domain images look like target domain images, and the other pair of the generator and the discriminator is responsible for making target domain images look like source domain images. We present technical details of each component of our model. Specification of each component is detailed at Section 4.2

3.3.1 Generator

To translate images across the source domain and the target domain, we adopt the encoder-decoder based architecture with skip connections for the generator [32]. The encoder-decoder network gradually down-samples the input image until a bottleneck layer, and then, upscales down-sampled input to the original resolution of the input via up-sampling or transposed-convolution. However, without skip connections, such architecture results that all information signal should pass through all layers, and this behavior causes low-level features from the input not to be shared across the network. Since the input image and the translated output image share underlying structure to some degree, we add skip connections so the low-level information near the input can be connected to layers near the output, as similar with the U-NET [57]. Concretely, for the total number of layers n, we connect layer i to the layer $n - i$ except the middle bottleneck layer. We use 4×4 spatial filters for all convolution and transposed convolution layers with the Relu activation function. As different to the U-NET, we remove the max-pooling layer and down-sample with the convolution layer with stride 2, and we utilize transposed-convolution layers for up-sampling.

3.3.2 Discriminator

PatchGAN

As we adopt the reconstruction L1 loss term for the cyclic consistency, this loss term fails to reconstruct high-frequency parts of the original image. Using L2 loss term, instead of L1 loss term, leads to more blurry images because minimizing L2 loss is equivalent to maximizing log-likelihood of a Gaussian so fails to capture different modes of real samples. Even though L1 loss term results in less blurry images, it also fails to reconstruct sharp parts of the image and only able to capture low-frequency parts of the image. However, the GAN is able to represent more sharp images than L1 and L2 reconstruction loss terms because the generator is trained to fool the discrim-

inator so the generator is encouraged to represent more accurately as real samples. In the standard GAN [19], the discriminator produces a single output from the whole input image. To facilitate modeling high-frequency regions for local patches in the input image, we adopt a PatchGAN [32], that the discriminator produces N by N grid output. For one element of the discriminator's output, its receptive field in the input image should be one small local patch in the input image, so the discriminator aims to distinguish each patch in the input image. To achieve this, we remove the fully-connected layer in the last part of the discriminator in the standard GAN. As the discriminator becomes fully convolution networks, so it can be applied to arbitrary size of the input image. As a matter of fact, a PatchGAN is equivalent to adopting multiple discriminators for every patches of the image, so makes the discriminator help the generator to represent more sharp images locally. In addition, we average out all the discriminator's value into a single scalar for the objective function [32].

Mini-batch Discrimination

We also adopt the mini-batch discrimination technique to encourage the generator to capture sample's diversity [35]. The mini-batch discrimination was firstly proposed by [60], allowing the discriminator look at not only individual examples, but also multiple examples in a mini-batch to avoid the mode collapse of the generator. To make the discriminator process each example with the correlation of other examples, it models a mini-batch layer in an intermediate layer of the discriminator, which calculates L1-distance based statistics of samples in a mini-batch. By adding such statistics to the discriminator, each example in a mini-batch can see how far or close to other examples in the mini-batch and this information can be internally utilized by the discriminator, which helps the discriminator reflect samples' diversity to the output. For the aspect of the generator, it tries to make similar statistics as those of real samples in the discriminator by adversarial learning procedure. In this paper, we simplify the mini-batch discrimination as in [35]. We do not adopt a tensor which projects statistics as in [60].

Rather, we use the mean of the standard deviation for each features (channels) in each spatial location over the mini-batch [35].

3.3.3 Classifier

We use the Resnet [26] based networks, which is widely known to be effective in deep convolution neural networks for the classification problem by making residual connection [26]. In the Resnet, the input image is progressively down-sampled to a coarse feature map. However, many down-samplings to a tiny feature map make the spatial information almost crushed. We highly need a spatial structure of the input image such as the location of a road and the border line of a road from the background. Motivated by [77], we partially adopt dilation convolution, which increases resolution of the feature map without decreasing receptive fields of latter layers. Originally, we apply down-sampling three times at Resnet based classifier. We replace last down-sampling layer with dilation convolution. By doing that, we increase a final resolution of the output feature map without reducing receptive fields of latter layers.

3.4 Objective Functions

The objective function, $\boldsymbol{L}$, for training our UAV navigation model, is as follows:

$$\boldsymbol{L} = \kappa L_A + \mu L_C \tag{3.1}$$

where κ, μ, and γ are the weights of an adversarial loss term, L_A, and a classification loss term, L_C, respectively.

For the adversarial loss, L_A, we adopt loss terms of the Fisher GAN [51] for each domain. The Fisher GAN belongs to the IPM framework [67] as described in Section 2.1.1, which is known to be strongly and consistently convergent [28, 66]. The Fisher GAN has useful properties: it is computationally efficient, and the estimated distance is naturally bounded while maintaining stable training property of the WGAN [3, 51]. As mentioned in Section 3.3, we also add cyclic L1 loss term for L_A for

each domain [38, 79]. In addition, we note that $G_{\theta_{g^s}}/D_{\theta_{d^s}}$ stand for source domain generator/discriminator/noise vector, and $G_{\theta_{g^t}}/D_{\theta_{d^t}}$ stand for target domain generator/discriminator/noise vector. $G_{\theta_{g^t}}$ is a generator which transfers the source domain image into the target domain, while $G_{\theta_{g^s}}$ transfers the target domain image into the source domain. We only describe object function for the target domain for simplicity.

The adversarial loss L_A is defined as follows:

$$L_A = \alpha_{A_1} l_{A_f} + \alpha_{A_2} l_{A_c} \tag{3.2}$$

where α_{A_1} and α_{A_2} refer to weights of each loss term of L_A, respectively.

To train two generators and two discriminators from each domain, we use Fisher GAN objective term l_{A_f} as follows:

$$l_{A_f} = \Phi(\theta_{d^t}, \theta_{g^t}) + \lambda(1 - \Omega(\theta_{d^t}, \theta_{g^t})) - \frac{\rho}{2}(\Omega(\theta_{d^t}, \theta_{g^t}) - 1)^2 \tag{3.3}$$

where λ is the Lagrange multiplier and ρ is the quadratic penalty weight coefficient [51]. $\Phi(\theta_d^t, \theta_g^t)$ and $\Omega(\theta_d^t, \theta_g^t)$ for each domain are the IPM equation inspired from augmented Lagrangian [18] and a constraint of the Fisher IPM metric [28, 51], respectively. $\Phi(\theta_d^t, \theta_g^t)$ and $\Omega(\theta_d^t, \theta_g^t)$ are defined as follows:

$$\Phi(\theta_d^t, \theta_g^t) = \mathbb{E}_{x \sim \mathbb{P}_{\mathrm{T}}} D_{\theta_d^t}(x) - \mathbb{E}_{x \sim \mathbb{P}_{\mathrm{S}}} D_{\theta_d}(G_{\theta_g}(x) \tag{3.4}$$

$$\Omega(\theta_d^t, \theta_g^t) = \frac{1}{2}(\mathbb{E}_{x \sim \mathbb{P}_{\mathrm{T}}} D^2_{\theta_d^t}(x) + \mathbb{E}_{x \sim \mathbb{P}_{\mathrm{S}}} D^2_{\theta_d^t}(G_{\theta_g^t}(x))) \tag{3.5}$$

To impose cyclic consistency for each domain, we use cyclic loss term l_{A_c} for the target domain as follows:

$$l_{A_c} = \mathbb{E}_{x \sim \mathbb{P}_{\mathrm{S}}} \|G_{\theta_{g^t}}(G_{\theta_{g^s}}(x)) - x\|_1 \tag{3.6}$$

This cyclic consistency role has an important role in the learning process. Since this term forces the generator to reconstruct to the original domain, major parts of the image such as road, should be maintained during adversarial learning process. Without this

term, there is no reason for the generator to preserve the content and shape of the image. It should be noted that object loss terms for the source domain can be derived in a similar manner. Object functions for the both domain are jointly used for training.

The classification loss term, L_C, is defined by

$$L_C = \alpha_{C_1} l_{C_e} + \alpha_{C_2} l_{C_n} + \alpha_{C_3} l_{C_p} \tag{3.7}$$

where l_e is a soft-max cross entropy term. l_n and l_p are a negative entropy regularization term and a penalty loss term, respectively, for an extreme failure [65]. α_{C_1}, α_{C_2}, and α_{C_3} refer to weights of each loss term of L_C, respectively.

The soft-max cross entropy term for our multi-class classification problem is as follows:

$$l_{C_e} = -\sum_i y_i \log(p_i) - \sum_j y_j \log(p_j) \tag{3.8}$$

where p_i and p_j are classifier predictions of head position, i $\in$ {left, straight, right}, and classifier predictions of lateral position, j $\in$ {left, center, right}, respectively. y_i and y_j are the ground truth labels of head and lateral positions, respectively.

Additionally, we add the negative entropy regularization term, l_n, and the penalty loss term, l_p, motivated by [65]. l_n prevents the classifier from producing extremely sharp results. l_p penalizes the classifier so that it will not swap left and right labels of lateral and head classes, respectively. l_n and l_p are defined as follows:

$$l_{C_n} = \sum_i p_i \log(p_i) + \sum_j p_j \log(p_j) \tag{3.9}$$

$$l_{C_p} = p_{l/r}^{\text{head}} + p_{r/l}^{\text{head}} + p_{l/r}^{\text{lateral}} + p_{r/l}^{\text{lateral}} \tag{3.10}$$

where $p_{l/r}$ and $p_{r/l}$ refer to a left label prediction probability when the ground truth is right and a right label prediction probability when the ground truth is left respectively.

Chapter 4

EXPERIMENT

4.1 Experiment setup

We generated about 8000 images for each lateral and head class by randomizing the position of the UAV in the simulator and by fixing the titled angle by -45,0,45 degrees, with respect to the road. Aside from the simulator images to be used as the source domain data, we gathered around total 30,000 images for the target domain data by recording on a local mountain trail. In both cases, we fixed the altitude of UAV to approximately 1.5 m to maintain the UAV's frame of view consistently.

We built our simulator environment using Gazebo [40], a widely known robot simulator. Because our navigation model targeted a local road made of the asphalt surrounded by bushes and tall trees, we built a simulated environment that was similar to the targeted real environment. We built our simulator environment in two phases. First, we generated various curved and straight roads by determining road's curvature and length heuristically. We then created many objects, such as a maple tree, a ginkgo tree, bushes, fences, etc., using an open source CAD tool Blender so as to make simulator image as much as look like outdoor environment. We arranged these objects along the road, randomly.

We used the parrot Bebop2 drone for our experiment. To send a command from

the trained model to a drone, we utilized parrot Bebop ROS package with a joystick connection. We implemented our UAV navigation system with [1]. During the test, we calculated our model's prediction output every 50ms with NVIDIA GTX 1060. We set our commanding interval to every one second for robust navigation.

4.1.1 Steering Command

Our model gives no steering commands; it provides probabilities of each class. Thus, we need an additional steering controller module when we navigate the UAV in the inference phase. When our model provides soft-max output $[p_{\text{left}}^{\text{lateral}}, p_{\text{center}}^{\text{lateral}}, p_{\text{right}}^{\text{lateral}}]$ for the lateral classes and $[p_{\text{left}}^{\text{head}}, p_{\text{straight}}^{\text{head}}, p_{\text{right}}^{\text{head}}]$ for the head classes, our steering controller sends two velocity commands as follows:

$$\begin{aligned} v_y &= \beta_1(p_{\text{right}}^{\text{lateral}} - p_{\text{left}}^{\text{lateral}}) \\ w_z &= \beta_2(p_{\text{right}}^{\text{head}} - p_{\text{left}}^{\text{head}}) \end{aligned} \tag{4.1}$$

where v_y is a y-axis linear velocity and w_z is a z-axis angular velocity with coefficients, β_1 and β_2. A positive y-axis linear velocity cause the UAV to move left and a positive z-axis angular velocity causes the UAV to rotate counter-clockwise. Additionally, the UAV flies forward with a constant velocity, by default. Unlike the steering controller of Smolyanskiy et al. [65], we send the median of five successive commands to prevent abrupt behavior of the UAV.

4.2 Implementation Details

We implemented our model with Tensorflow [1], which is a popular deep learning library. We resized input image as 256 by 256, which thought to be most suitable size for generation. We initialize all weight variables with truncated normal initialization of standard deviation of 0.02. We used batch size of 4 to compensate large usage of memory. We used adam optimizer [39] with learning rate 0.0002 and did not decay

learning rate. We used a data augmentation technique such as image contrast, brightness, saturation and injecting gaussian noise with probability 0.5 to diverse sample images.

Related to training of GANs, we intentionally reduce the power of the discriminator. Practically, the discriminator often overwhelms the generator so makes training imbalanced. We inject additive gaussian noise of standard deviation 0.1 to the hidden layers of the discriminator. In addition, we also add dropout to the discriminator to prevent over-fitting to the generated data.

We adapt our architecture from [8, 33]. Table 4.1, 4.2 and 4.3 show architecture details of the generator, discriminator and classifier, respectively. We denote a filter size, a stride, the number of feature map and a dilation rate as **F**, **S**, **C**, and **D**, respectively. For example, **F3S1C64** stands for a convolution layer where 3×3 spatial filter are applied with stride 1, generating 64 feature maps. If **D** is appended, the dilation convolution is applied with a corresponding dilation rate. In Tab. 4.1, arrows represent concatenation to latter layers of the generator. In Tab. 4.3, layers with two multi rows involve a residual connection. In Tab. 4.2, $Lrelu$ denotes a leaky relu function with a slope of 0.2. It should be noted that we do not apply non-linear activations and normalization to the last layer of each component.

4.3 Result

4.3.1 Off-line Test Result

We conducted off-line tests for our trained model prior to testing outside. We not only measured classification accuracy, we also monitored steering commands on the test target domain data which was not used in the training phase. We checked both because the navigation depends on steering commands, not on classification accuracy. Therefore, though the classifying accuracy for the center/straight class is quite low, UAV may goes forward well because the commander does not let UAV move. Table 4.4 and

Table 4.5 list classification accuracies and average steering commands, respectively. Because our approach focuses on using a simulator and unlabeled real data, not the architecture of the model, we compared our model to a supervised learning (SL) model with source domain data and another with target domain data.

Table 4.4 lists the off-line test results (see captions for detail). Our model succeeded in giving correct commands in all cases, as like the target image SL model, whereas both source SL models failed. Furthermore, even though classification accuracies of some classes listed in Table 4.4 were relatively low in our model's result, the UAV could reasonably move because a steering command with a correct sign guides it to move properly. One may argue that the probability of a right lateral in the target image SL model is lower than other classes in Tab. 4.4. Although we have another target image SL model which has 99 % accuracy in the right lateral label, this model performed worse on outdoor navigation than the target image SL model of Table 4.4.

4.3.2 Outdoor Navigation On-line Test Results

In the outdoor on-line test, we verified the performance of our approach based on two criteria, as is commonly carried out in this field [65]. First, we ask, "how many human interrupts are required to finish navigation in certain trail courses?" to see navigation performance. Second, we ask, "how well UAV recovers from intentional disturbance?" to see the ability to handle unexpected situation. Because the source mask SL model showed the lowest performance in the off-line test, we omitted that case from navigation results.

Autonomous navigation

We tested our model on three courses. The length of each was approximately 190 m, 60 m, and 110 m, respectively.

Course 1 in Figure 4.1, where we gathered real images, has relatively slight curves. Courses 2 and 3, where we did not gather images, have more sharp curves compared to

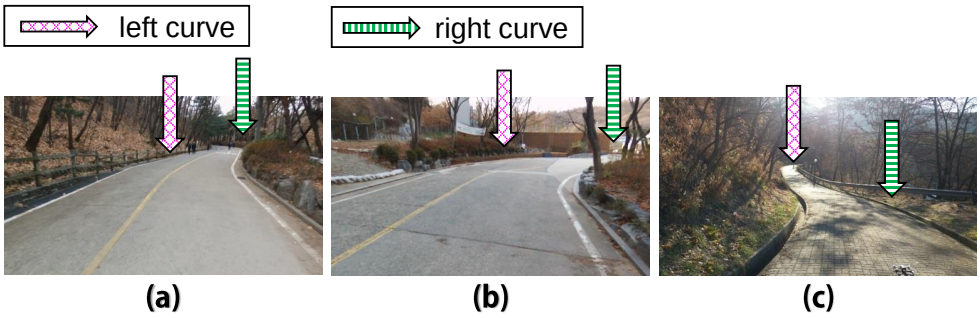

Figure 4.1: Navigation test courses. (a) Course 1: two slight corners. (b) Course 2: two sharp curves, unseen street during the training phase. (c) Course 3: different environment, not asphalt but sidewalk road with two corners. These courses are thought to be sufficient to test our model because they contain road curving to the left and right.

Course 1. We argue that tests on these three courses were appropriate to demonstrate the effectiveness of our model. We observed not only navigation performance but also robustness of our model by testing on the course of which scene was used during training (*i.e.*, Course 1) and courses unseen prior (*i.e.*, Courses 2 and 3) even with different environments.

Table 4.6 shows the average number of human interrupts until the UAV finished each course with three models. The source image SL models failed to finish the course within reasonable trials. We omitted the performance of source image SL on Courses 2 and 3, because Course 1 could be regarded as the easiest course. Our model showed successful performance compared to the target image SL model.

Given the performance of the source image SL models, we could conclude that applying the model trained only with the simulator data into outdoor environment is prone to failure. This means that domain adaptation is essential.

Recovery from disturbance

We tested our model for successful recovery from disturbances. A vehicle should avoid objects, including humans. In the case of a UAV, the wind blows in the outdoor and strong air currents are caused by the UAV, itself. Thus, the UAV tends to lose its stability and deviate from the desirable location or path. It is very important to recover from these deviations during autonomous navigation. We placed the UAV in a non-desirable position intentionally to measure its ratio of successful returns to the desired position. Tab. 4.7 shows the ratios in extreme cases (i.e., lateral left/head left and lateral right/head right). We set 20 undesirable positions for each case and measured the number of UAV that returned to the normal path (i.e., lateral center/head straight) in a reasonable time. Our model successfully recovered as much as the target SL model.

4.3.3 Image Transfer

Since we train the classifier of our model with transferred images, which are paired with corresponding head/lateral labels, a visual quality of transferred images to target domain directly affects the performance of the classifier. Figure 4.2 shows source images, transferred images and target images. Though transferred images in Figure 4.2 are not sharp, they look like target images at some degree.

Since we can not judge the quality of image transfer only by looking images, we show the similarity between the two distributions by embedding images from each to a low dimensional space, using t-SNE [46], isomap [68], and principal component analysis (PCA) [73]. Figure 4.3, Figure 4.5 and 4.4 represent the embedding results by isomap , PCA and t-SNE, respectively. Each figure shows the embedded points for random sampled 1000 images from the source domain **S**, the transferred domain **F** and the target domain **T**, respectively. The transferred images were closer to the real images than the source images, meaning that our model successfully transferred source images as similar as target images.

Figure 4.2: A figure of source images in row (a), transferred images in row (b) and target images in row (c). Note that target images are picked up as most similar ones as transferred images.

Layer Index	UNET
Input	256×256×3 Image
L - 1	Conv, F4S2C64, Relu
L - 2	Conv, F4S2C128, Relu
L - 3	Conv, F4S2C256, Relu
L - 5	Conv, F4S2C512, Relu
L - 6	Conv, F4S2C512, Relu
L - 7	Conv, F4S2C512, Relu
L - 8	Conv, F4S2C512, Relu
L - 9	Conv, F4S2C512, Relu
L - 10	Deconv, F4S2C512, Relu
L - 11	Deconv, F4S2C512, Relu
L - 12	Deconv, F4S2C512, Relu
L - 13	Deconv, F4S2C512, Relu
L - 14	Deconv, F4S2S256, Relu
L - 15	Deconv, F4S2C128, Relu
L - 16	Deconv, F4S2C128, Relu
L - 17	Deconv, F4S2C3

Table 4.1: Architecture details of the generator.

Layer Index	Discriminator
Input	96×96×3 Image
L - 1	Conv, F4S1C64, Lrelu Dropout, $\rho = 0.5$ Gaussian noise, $\sigma = 0.2$
L - 2	Conv, F4S2C128, Lrelu Dropout, $\rho = 0.5$ Gaussian noise, $\sigma = 0.2$
L - 3	Conv, F4S2C256, Lrelu Dropout, $\rho = 0.5$ Gaussian noise, $\sigma = 0.2$
L - 4	Conv, F4S2C512, Lelu Dropout, $\rho = 0.5$ Gaussian noise, $\sigma = 0.2$
L - 14	Conv, F4S2C1024, Lrelu
L - 15	Conv, F1S1C1

Table 4.2: Architecture details of the discriminator.

Layer Index	Classifier
Input	96×96×3 Image
L - 1	Conv, F7S2C64, Relu
L - 2	Conv, F3S1C64, Relu Conv, F3S1C64, Relu
L - 3	Conv, F3S1C64, Relu Conv, F3S1C64, Relu
L - 6	2×2 max-pool, S2
L - 4	Conv, F3S1C128, Relu Conv, F3S1C128, Relu
L - 5	Conv, F3S1C128, Relu Conv, F3S1C128, Relu
L - 6	Conv, F3S1C256D1, Relu Conv, F3S1C256D2, Relu
L - 7	Conv, F3S1C256D2, Relu Conv, F3S1C256D2, Relu
L - 8	Conv, F3S1C512D2, Relu Conv, F3S1C512D4, Relu
L - 9	Conv, F3S1C512D4, Relu Conv, F3S1C512D4, Relu
L - 10	Conv, F3S1C512D2, Relu
L - 11	Conv, F3S1C512D2, Relu
L - 12	Conv, F3S1C512, Relu
L - 13	Conv, F3S1C512, Relu
L - 14	Global average pooling
	Branches for command
L - 15	FC, C5

Table 4.3: Architecture details of the classifier.

Table 4.4: **The off-line test result.** Source image supervised model (SL) model is supervised-trained with unmasked source images. The source mask SL model is supervised-trained with masked source images. The target image SL model is supervised-trained with real images and their lateral/head labels. The steering commands for images having a left/straight/right lateral label and a left/center/right head label should be negative, zero, and positive, respectively.

Model	Lateral offset label			Head orientation label		
	left	center	right	left	straight	right
Source image SL	0.0 %	65.5 %	55.2 %	76.8 %	64.1 %	45.0 %
Source mask SL	31.4 %	0.0 %	30.3 %	1.7 %	45.2 %	46.4 %
Target image SL	90.1 %	99.8 %	55.4 %	91.7 %	100.0 %	96.9 %
Our model	92.0 %	52.8 %	46.5 %	70.6 %	55.8 %	57.7 %

Table 4.5: **Average steering command.** The steering commands for images having a left/straight/right lateral label and a left/center/right head label should be negative, zero, and positive, respectively. As absolute values of a steering command get larger, the UAV experiences bigger movements. It should be noted that probability of right lateral in target image SL model is quite low than other classes in Table 4.4. Even though we have another target image SL model which has 99% accuracy in right lateral, this model does much poor performance on outdoor navigation than Table 4.4 target SL model. We list up more well-behaved target image SL model into this table. Despite of low accuracy, appropriate command can lead to good navigation.

Model	Lateral offset label			Head orientation label		
	left	center	right	left	straight	right
Source image SL	0.07±0.19	0.35±0.40	0.55±0.44	-0.70±0.52	0.09±0.55	0.04±0.89
Source mask SL	0.35±0.83	-0.28±0.79	-0.32±0.75	0.29±0.34	0.52±0.29	0.57±0.28
Target image SL	-0.83±0.22	0.002±0.04	0.52±0.37	-0.89±0.23	-0.004±0.01	0.94±0.16
Our model	-0.87±0.37	-0.14±0.59	0.31±0.66	-0.68±0.46	0.03±0.59	0.40±0.70

Table 4.6: **Average number of human interrupts to finish.** We counted the total number of interrupts in five navigation trials on each course. Our model showed comparable performance with a target image SL model.

	Course 1	Course 2	Course 3
Source image SL	> 10	·	·
Target image SL	0.2 (1/5)	1.2 (6/5)	0.2 (1/5)
Our model	0.8 (4/5)	1.4 (7/5)	0.0 (0/5)

Table 4.7: **Success rate to recover.** We counted the number of successful recoveries after 20 intentional disturbances at extreme cases. Left/left and right/right refer to lateral left/head left and lateral right/head right, respectively.

	left/left	right/right
Target image SL	95 % (19/20)	60 % (12/20)
Our model	95 % (19/20)	60 % (12/20)

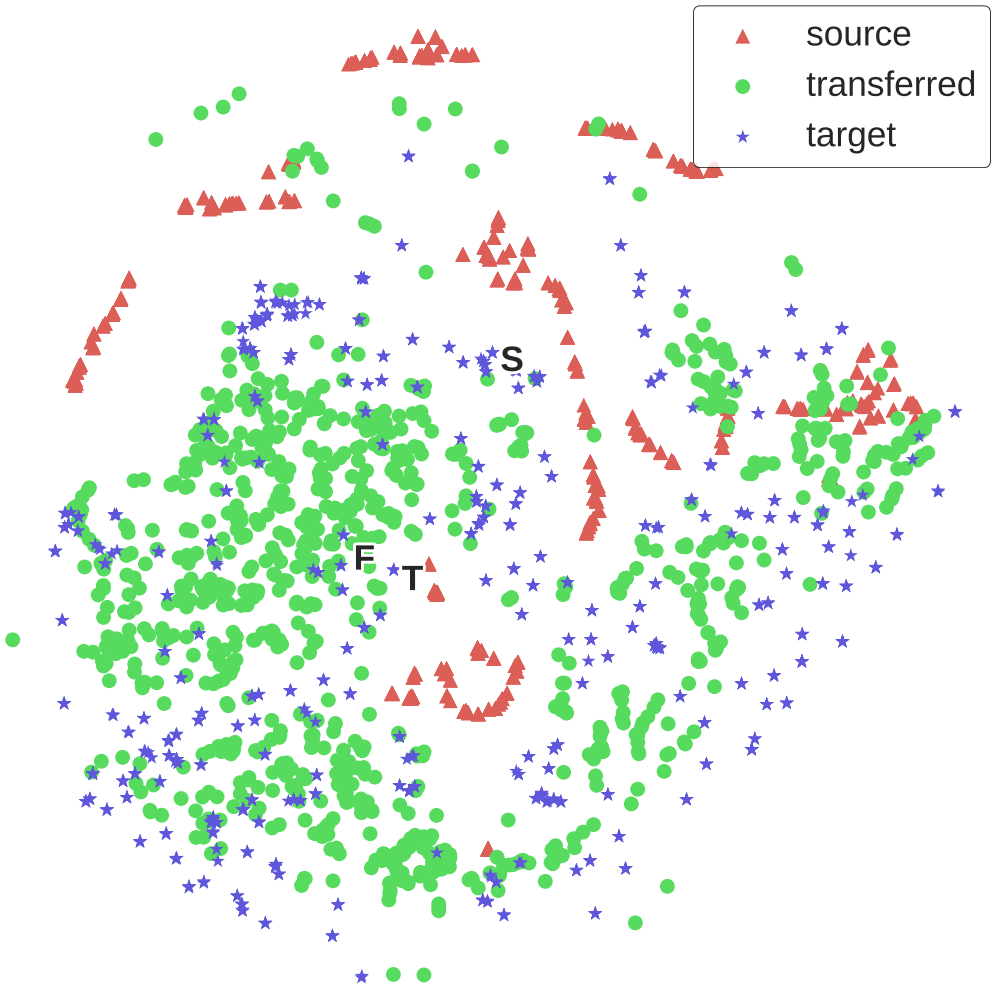

Figure 4.3: A figure of t-SNE result with 1000 images are randomly sampled from the source domain **S**, the transferred domain **F** and the target domain **T**.

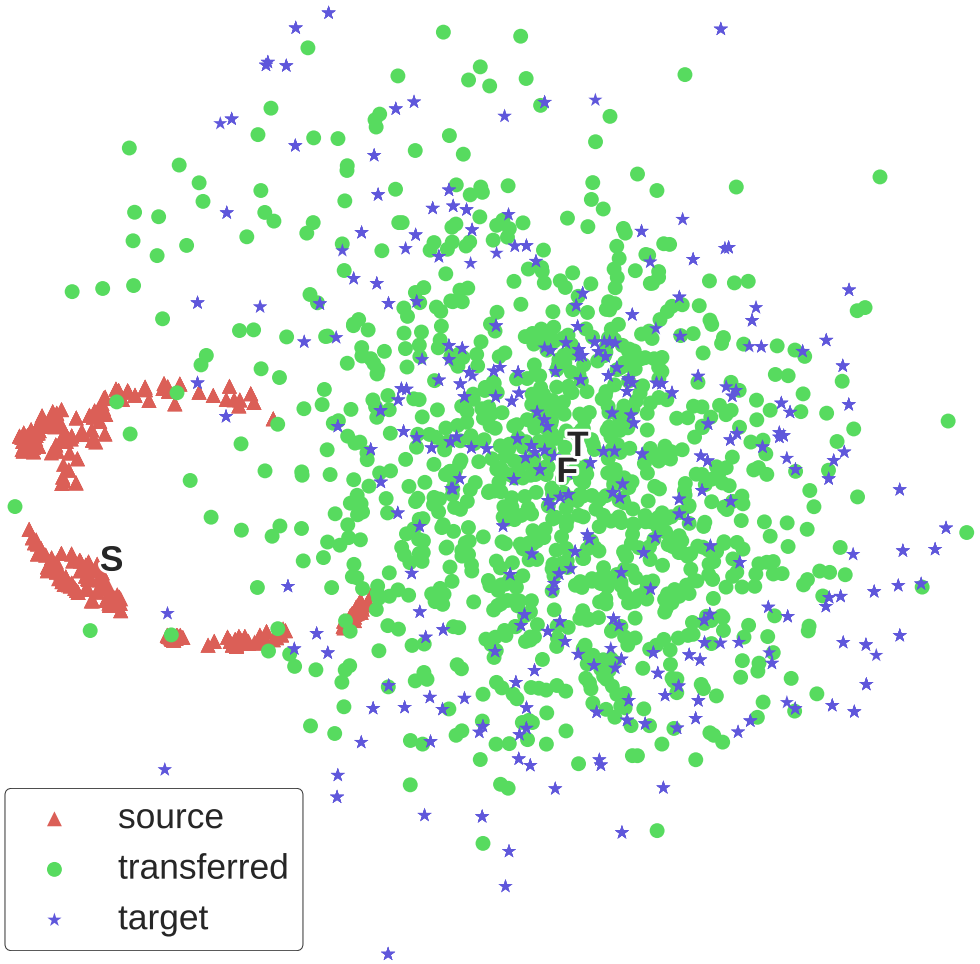

Figure 4.4: A figure of PCA result with 1000 images are randomly sampled from the source domain **S**, the transferred domain **F** and the target domain **T**.

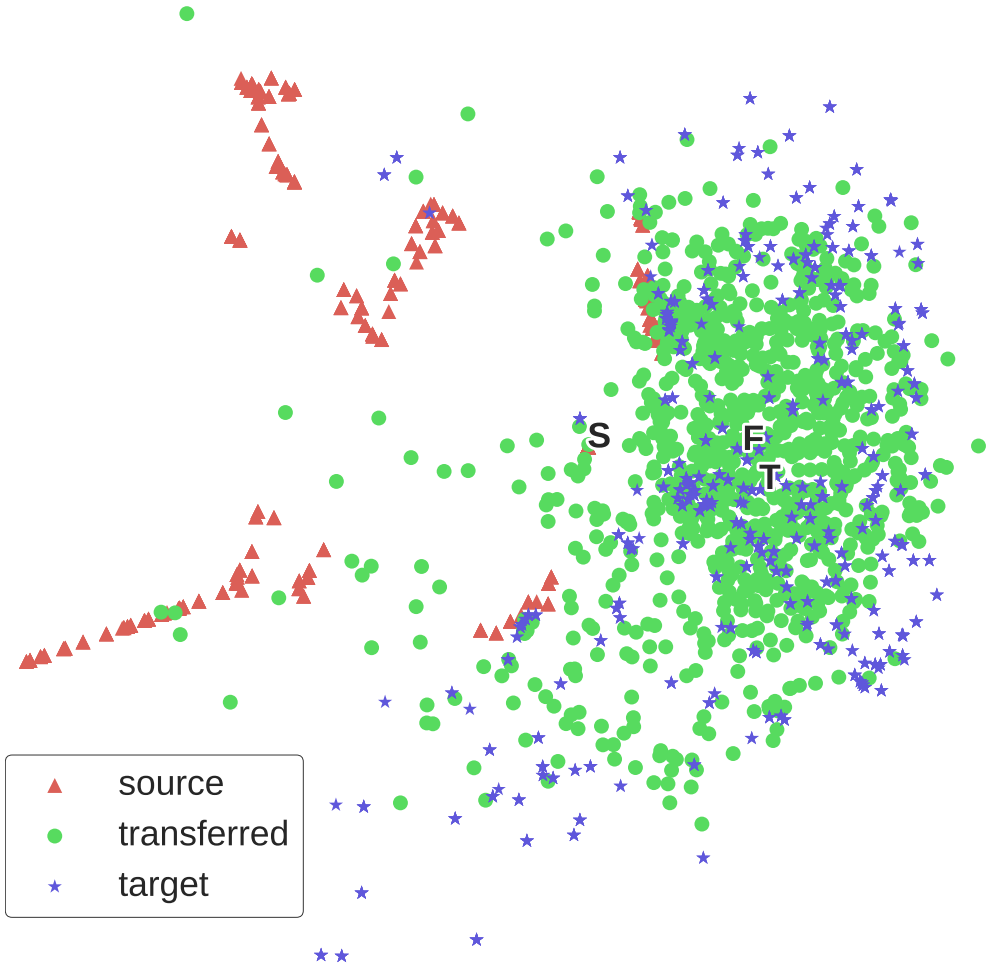

Figure 4.5: A figure of isomap result with 1000 images are randomly sampled from the source domain **S**, the transferred domain **F** and the target domain **T**.

Chapter 5

DISCUSSION

5.1 Training

5.1.1 Content Similarity Loss

In [8], they adopt the content similarity loss L_S to make the location of a road in an image should be invariant between the source and transferred images for successful domain adaptation, which guides the road part of an image to be invariant. Given the mask of the road in a simulator image, we use a masked mean squared error term for the content similarity loss term L_S as follows:

$$L_S = \mathbb{E}_{(x,z)\sim(\mathbb{P}_S,p_z)}\left[\frac{1}{n}\|(x - G_{\theta_{g_S}}(x,z)) \circ m\|_2^2\right] \tag{5.1}$$

where n is the number of pixels of x, $\|\cdot\|_2^2$ is the squared L2 norm, m is a binary road mask, and $\circ$ refers to element-wise multiplication. Intuitively, it is highly similar to cyclic consistency term in Equation 3.6 in that Equation 5.1 tries to reconstruct the masked part of the input image. Therefore, we do not use L_S for our objective function since cyclic consistency term takes a role of reconstruction including the important content of the input image.

5.1.2 Normalization

Adversarial components

We experiment various normalization techniques such as batch normalization [31], instance normalization [30, 64, 69] and pixel normalization [35] to our model. Instance normalization is a specific normalization technique for the style transfer suggested by [69], and aims to impose contrast normalization of the content image. It can be thought as instance specific normalization, differs from batch normalization in that the batch normalization applies normalization to a whole input batch. This makes neural networks to be invariant channel-wide shifts and scaling (contrast normalization) [64]. On the other hand, pixel normalization do not compute across spatial dimensions (width and height). Rather, it normalizes the feature vector in each pixel in spatial dimensions to unit length, so it can preserve local response across spatial dimensions and make signal magnitudes of the generator and the discriminator not increase [35]. Among three of them, we adopt pixel normalization for style transfer between the source domain and the target domain. Batch normalization and instance normalization almost show similar result for transferring and not seem to help transferred images to be sharp and acute. In contrast, pixel normalization maintains transferred images to be sharp because it normalizes with respect to each pixel, so maintaining local responses. We add pixel normalization to every layer before a non-linear activation except the last layer to the generator and the discriminator.

Task component

We originally add the batch normalization to every layer before a non-linear activation except the last layer to the task classifier. However, the performance of batch normalization is degenerated as batch size gets smaller, since a small number of samples in a batch is not guaranteed to represent whole dataset. Moreover, with a small size of batch, the mean and covariance are not consistent, which exacerbates the stabil-

ity of training. Our model is composed of two generator, discriminator and one task classifier, which require large memory size. To compensate for large memory usage, we inevitably have to reduce batch size. Motivated by this reason, we adopt group normalization [74] for the task classifier. Group normalization is similar to instance normalization and layer normalization [4] in that they do normalization independent of batch size. Group normalization can be interpreted as the fusion of layer normalization and instance normalization, where it performs normalization by grouping some part of channel dimension. We add group normalization to every layer before a non-linear activation except the last layer to the task classifier, so try to avoid the performance degeneration of batch normalization with a small batch size.

5.2 Failure Cases

We succeeded in navigating a UAV automatically without any labels for real images. To the best of the authors' knowledge, this is the first attempt to apply domain adaptation with adversarial learning to an autonomous vehicle navigation task. This approach may extend the scope of autonomous navigation to various outdoor environments that can be simulated. Additionally, we showed the potential robustness of our approach by demonstrating reasonable navigation performance in courses where no scene was used during training.

Despite our successful results, there were some failure cases. First, our model was not robust to weather conditions such as strong sunlight, mist, and snow. We collected real images during the hours of 3-5 pm in the fall and winter. Thus, almost all images were exposed to small amounts of sunlight. Thus, the transferred image was necessarily biased to the similar state. When the sun was strong, the road in a real image changed its color from gray to yellow, causing navigation to fail. To be robust to weather conditions, many images under various conditions may be required.

Second, both our model and the target image SL model failed when branches

from trees protruded the road from above. Though a UAV was on the road, protruding branches collided with the wing of the UAV, leading to failure. Because the road remained visible through pine leaves, it was difficult for the UAV to avoid those obstacles.

5.3 Future Works

5.3.1 Regression Problem

As mentioned in Section 3.1, we defined autonomous navigation as a head/lateral classification problem. From probabilities for each class, we gave proper commands to a UAV at head orientation and lateral offset. However, though this classification based control can approximate real steering command, it is not accurate. In fact, estimating steering command by a regression problem is more proper than a classification problem if corresponding accurate label can be taken. [65] approached autonomous navigation with a steering command regression problem and [12] also tackled autonomous navigation with a regression problem, which estimates steering commands and acceleration. However, getting continuous accurate labels for a regression problem is difficult even at simulator environment. Moreover, labels are mostly zero with sparse non-zero signals, which make a probability distribution of labels discontinuous. Therefore, we treated autonomous navigation with a classification aspect. With classification approach, we can get labels relatively easy by grouping a specific range of labels as one class and accomplish autonomous navigation successfully. If we can acquire more delicate steering labels, a regression problem can be a better solution for autonomous navigation.

5.3.2 Command Condition

Our task is to follow a trail without an intersection. However, if a vehicle encounters to an intersection, where should a vehicle go to? With the sole image input, a

vehicle would fail navigation because it simply does not know where to go. In such non-functional situation, we need to feed additional command input which reflects a driver's intention to the goal. [12] suggest branched architecture where each branch estimates steering commands for each command such as go straight, turn left or turn right. As [12], our approach can be further extended to refer a driver's intention at an intersection.

5.3.3 Other Vehicles

Although we used a UAV to validate our method of autonomous navigation, it would be possible to apply our method to other types of vehicles. The task we conducted in this study is a version of lane-tracking, which is one of many components of autonomous driving. As we have shown with the validity of our method to use a simulator in autonomous navigation, we are planning to perform more complex navigation tasks by expanding our method and incorporating other machine learning techniques, including reinforcement learning.

Chapter 6

CONCLUSION

In this dissertation, we demonstrated the method of autonomous navigation, using domain adaptation with generative adversarial networks. We succeeded the trail tracking task with the only sole image input. Our method do not depend on any kinds of sensors. Most importantly, our method does not need label information of real environment image as other supervised learning based approaches. We acquired label information in simulator environment, and transferred those knowledge to real environment by domain adaptation.

In chapter 2, we described the background of generative adversarial networks and domain adaptation. Particularly, among lots of application where GANs are utilized, we focused on image translation and domain adaptation, which are specific subjects for our approach. In addition, we commented prior approaches of autonomous navigation, where most of them are based on supervised learning method or its variants.

In chapter 3, we demonstrated our method. To transfer knowledge of simulator environment, we adopt cycle concept of image translation with task classifier. We also detailed the architecture and objective functions of our model, which are highly crucial for successful training.

In chapter 4, we show the performance of our method by navigating the UAV along trails. We succeeded in navigating the UAV automatically in three outdoor courses of

up to about 300 meters by training our deep learning model using a myriad of simulator images with auto-generated labels and real images without any label. We note that our approach performed as much as pure supervised method model, which tells us the possibility of truly autonomous navigation in various environments using enormous and accessible simulator images.

In chapter 5, we discussed the performance of our model in detail. Particularly, though our method accomplished the trail following task, it is not robust to various weather condition and can not avoid obstacles. We remain these advanced tasks as future works. In addition, we demonstrated possible application of our approach, which can improve our method in aspect of practicality.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[7] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[8] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016.

[9] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857*, 2017.

[10] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *AISTATS*, pages 57–64. Citeseer, 2005.

[11] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *arXiv preprint arXiv:1710.02410*, 2017.

[12] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *arXiv preprint arXiv:1710.02410*, 2017.

[13] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1): 1–18, 2000.

[14] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[15] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training

generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[17] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2): 661–667, 2016.

[18] Ronald Glowinski and Patrick Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*, volume 9. SIAM, 1989.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[20] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 999–1006. IEEE, 2011.

[21] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[22] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[23] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[24] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[25] Leonid G Hanin. Kantorovich-rubinstein norm and its application in the theory of lipschitz spaces. *Proceedings of the American Mathematical Society*, 115(2): 345–352, 1992.

[26] 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.

[27] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[28] Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo, and Sungroh Yoon. How generative adversarial nets and its variants work: An overview of gan. *arXiv preprint arXiv:1711.05914*, 2017.

[29] Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007.

[30] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR, abs/1703.06868*, 2017.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating

deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[32] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[33] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[34] Gregory Kahn, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. Plato: Policy learning using adaptive trajectory optimization. *arXiv preprint arXiv:1603.00622*, 2016.

[35] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[36] Beomjoon Kim and Joelle Pineau. Maximum mean discrepancy imitation learning. In *Robotics: Science and systems*, 2013.

[37] Jong-Hyuk Kim, Salah Sukkarieh, and Stuart Wishart. Real-time navigation, guidance, and control of a uav using low-cost sensors. In *Field and Service Robotics*, pages 299–309. Springer, 2003.

[38] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

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

[40] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an

open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[42] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.

[43] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE, 2011.

[44] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. *arXiv preprint arXiv:1705.08584*, 2017.

[45] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105, 2015.

[46] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[47] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[48] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

[49] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[50] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[51] Youssef Mroueh and Tom Sercu. Fisher gan. *arXiv preprint arXiv:1705.09675*, 2017.

[52] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[53] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[54] Sinno Jialin Pan, James T Kwok, and Qiang Yang. Transfer learning via dimensionality reduction. In *AAAI*, volume 8, pages 677–682, 2008.

[55] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[56] Svetlozar Todorov Rachev et al. Duality theorems for kantorovich-rubinstein and wasserstein functionals. 1990.

[57] 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.

[58] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning

monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1765–1772. IEEE, 2013.

[59] Fereshteh Sadeghi and Sergey Levine. cad^2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[60] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[61] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Adversarial representation learning for domain adaptation. *arXiv preprint arXiv:1707.01217*, 2017.

[62] Shaojie Shen, Nathan Michael, and Vijay Kumar. Autonomous multi-floor indoor navigation with a computationally constrained mav. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 20–25. IEEE, 2011.

[63] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016.

[64] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018.

[65] Nikolai Smolyanskiy, Alexey Kamenev, Jeffrey Smith, and Stan Birchfield. Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness. *arXiv preprint arXiv:1705.02550*, 2017.

[66] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. On integral probability metrics,\phi-divergences and binary classification. *arXiv preprint arXiv:0901.2698*, 2009.

[67] Bharath K Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert RG Lanckriet. Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11(Apr):1517–1561, 2010.

[68] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[69] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proc. CVPR*, 2017.

[70] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[71] Chaoyue Wang, Chang Xu, Chaohui Wang, and Dacheng Tao. Perceptual adversarial networks for image-to-image transformation. *arXiv preprint arXiv:1706.09138*, 2017.

[72] Max Welling. Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto*, 3(1), 2005.

[73] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[74] Yuxin Wu and Kaiming He. Group normalization. *arXiv preprint arXiv:1803.08494*, 2018.

[75] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint*, 2016.

[76] Jaeyoon Yoo, Yongjun Hong, and Sungrho Yoon. Autonomous uav navigation with domain adaptation. *arXiv preprint arXiv:1712.03742*, 2017.

[77] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition*, volume 1, 2017.

[78] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

[79] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

[80] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Mav navigation through indoor corridors using optical flow. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3361–3368. IEEE, 2010.

초 록

대립적 생성 신경망은 (GANs) 중간 지도 학습, 이미지 전이, 도메인 적응 등과 같은 머신 러닝의 중요한 문제들에 대한 중요한 해법으로 각광받고 있다. 특히, 대립적 생성 신경망은 기존의 생성 모델에 비해 확률 분포에 대한 가정 없이 더욱 구체적이고 실제같은 데이터를 생성할 수 있다.

본 논문에서는 로보틱스, 기계 공학, 머신 러닝 등에서 중요한 문제였던 자율 주행에 대립적 생성 신경망을 적용하였다. 깊은 인공신경망을 자율 주행에 이용한 기존 접근 방법들은 데이터에 대한 라벨 정보을 많이 모은 후, 인공신경망을 지도 학습 방법으로 훈련시켰다. 하지만 라벨 정보가 있는 거대한 데이터 집합은 이러한 방법에 필수 불가결하며 라벨 정보가 있는 실제 데이터는 얻기 어렵고 부정확한 경우가 많다.

우리는 이 문제를 대립적 생성 신경망을 이용한 도메인 적응 기법으로 해결하였다. 시뮬레이션 환경에서는 다양한 종류의 라벨 정보들을 쉽게 얻을 수 있다. 라벨 정보가 있는 인공 데이터를 시뮬레이터로부터 추출하고 도메인 적응 기법으로 실제 환경에서의 라벨 문제를 완화하였다. 더불어, 대립적 생성 신경망을 통해 인공 데이터를 실제 데이터와 비슷하게 만들어 실제 환경에서의 라벨이 없이도 자율 주행이 성공하도록 하였다.

주요어: 대립적 생성 신경망, 자율 주행, 도메인 적응

학번: 2014-21634

TO MY FAMILY