Scene Lighting Estimation for Augmented reality with Neural Models

Zegang Wang

***Abstract*—In this article [1], I am going to briefly compare two recent research papers, Xihe and GLEAM, that provide real-time lighting estimation on mobile devices. Lighting estimation, briefly, refers to obtaining lighting information from environment scenes. For augmented reality, the scenes are physical environments (e.g., your living room) and therefore can have natural lighting condition changes, both temporally and spatially. Dynamic scene lighting is a key reason that lighting estimation for physical environments is challenging. As lighting conditions can potentially change from one frame to the next, such rapid changes necessitate fast estimation (e.g., 30fps); additionally, lighting conditions are likely to vary from one corner of the room to the other corner (e.g., due to light source location), making it important to derive lighting estimation for positions of interests. Existing commercial AR platforms including ARCore and ARKit have started to provide lighting estimate APIs, such as the ARLightEstimate. However, existing supports are still at an early stage, only providing ambient light information for an entire scene. In other words, existing commercial AR platforms including ARCore and ARKit still lack supports for accurate lighting estimation and thus often result in unrealistic rendering effects. Below are two diagrams that demonstrate the rendering effects using lighting information provided by ARKit vs. GLEAM/Xihe. Both GLEAM and Xihe produce more visually coherent virtual objects.**

I. Introduction

In a nutshell, GLEAM uses physical light probes to capture environment lighting information which will then be provided to graphic renderers to illuminate virtual objects in the physical world. What is a physical light probe? It is usually a reflective sphere that is placed at a specific physical location for sensing light. The diagram below shows an example light probe in the form of a chrome ball. What is a cubemap? A cubemap is one of the common representations of an environment map. An environment map helps us to model illumination by mapping incoming ray directions to ray intensity. The diagram below shows an example cubemap captured by GLEAM.

GLEAM conceptually consists of the following two key steps. Step 1: generating a radiance sample. To start, a GLEAM light probe (a physical light probe and an AR positioning marker) will be placed in the scene. Then, the AR user will capture images of the GLEAM light probe to generate radiance samples by associating each pixel with a reflected ray. The captured radiance samples will serve as the basis for constructing the environment map. Step 2: compositing a cubemap from multiple radiance samples. As the radiance samples are inherently sparse, e.g., limited by the number of light probes or FoVs, GLEAM uses a modified inverse distance weighting (IDW) interpolation algorithm and a nearest neighbor algorithm to assign values to each cubemap texel. The texel value assignment is a two-stage process.

Fig. 1: screenshot from the original GLEAM paper: AR scene rendering comparison.

Specifically, each texel u(x) will be first populated by corresponding values from eligible radiance samples using the following equation. Here, wi(x) denotes the weight assigned to radiance sample i, and d(x, xi) describes the distance between the radiance sample i and the texel x. GLEAM uses a reliability-based weight function and L1 distance function. Theoretically, one can choose other weight and distance functions. For each texel, an eligible radiance sample has to satisfy a radius check. In the second stage, any empty texels will be filled using their neighboring texel values.

GLEAM presents an end-to-end framework for lighting estimation in mobile AR. As the paper stated, "GLEAM builds on a rich history of illumination estimation from the graphics community." Indeed, the physical light probes-based approach is widely used outside mobile AR; but it is still pleasant to see this idea realized and demonstrated effectiveness for mobile AR. If you are interested in working on physical probe-based lighting estimation, the authors have pointed out a number of limitations. For example, how to achieve better interpolation quality of cubemap from radiance samples while adhering to the real-time goal? What is the network impact on GLEAM's performance, especially under multi-viewport configurations?

II. Related Work

A. Image Generation

Image to Image translation have been around for sometime before the invention of CycleGANs. One really interesting one is the work of Phillip Isola et al in the paper Image to Image Translation with Conditional Adversarial Networks where images from one domain are translated into images in another domain. The dataset for this work consists of aligned pair of images from each domain. Generative image models

[1] https://medium.com/codex/lighting-estimation-for-mobile-ar-part-i-8c548affc7e5

Fig. 2: screenshot from the original Xihe paper: AR scene rendering comparison.

are well studied and fall into two categories: parametric and nonparametric. The non-parametric models often do matching from a database of existing images, often matching patches of images, and have been used in texture synthesis, super-resolution and in-painting. Parametric models for generating images has been explored extensively (for example on MNIST digits or for texture synthesis). However, generating natural images of the real world have had not much success until recently. A variational sampling approach to generating images has had some success, but the samples often suffer from being blurry. Another approach generates images using an iterative forward diffusion process. Generative Adversarial Networks generated images suffering from being noisy and incomprehensible. The maximum mean discrepancy (MMD) [1] is a measure of the difference between two distributions P and Q given by the supremum over a function space $\mathcal{F}$ of differences between the expectations with regard to two distributions. MMD has been used for deep generative models [2], [3], [4] and model criticism [5]. WGAN [6] conducted a comprehensive theoretical analysis of how the Earth Mover (EM) distance behaves in comparison with popular probability distances and divergences such as the total variation (TV) distance, the Kullback-Leibler (KL) divergence, and the Jensen-Shannon (JS) divergence utilized in the context of learning distributions.Based on W-div, Wu et al. [7] introduce a Wasserstein divergence objective for GANs (WGAN-div), which can faithfully approximate W-div by optimization. CramerGAN [8] argues that the Wasserstein distance leads to biased gradients, suggesting the Cramér distance between two distributions. Other papers related to WGAN can be found in [9], [10], [11], [12], [13], [14]. Rather than utilizing a single unstructured noise vector z, InfoGAN [15] proposes to decompose the input noise vector into two parts: z, which is seen as incompressible noise; c, which is called the latent code and will target the significant structured semantic features of the real data distribution. Maximizing $I(c; G(z, c))$ means maximizing the mutual information between c and $G(z, c)$ to make c contain as much important and meaningful features of the real samples as possible. However, $I(c; G(z, c))$ is difficult to optimize directly in practice since it requires access to the posterior $P(c|x)$. Fortunately, we can have a lower bound of $I(c; G(z, c))$ by defining an auxiliary distribution $Q(c|x)$ to approximate $P(c|x)$. The discriminator of original GANs [16] is trained to maximize the log-likelihood that it assigns to the correct source [17]. Discriminators of most cGANs based methods [18], [19], [20], [21], [22], [23], [24] feed conditional information y into the discriminator by simply concatenating (embedded) y to the input or to the feature vector at some middle layer. cGANs with projection discriminator [25] adopts an inner product between the condition vector y and the feature vector. Two-domain I2I can solve many problems in computer vision, computer graphics and image processing, such as image style transfer (f.) [26], [27], which can be used in photo editor apps to promote user experience and semantic segmentation (c.) [28], [29], which benefits the autonomous driving and image colorization (d.) [30], [31]. If low-resolution images are taken as the source domain and high-resolution images are taken as the target domain, we can naturally achieve image super-resolution through I2I (e.) [32], [33]. Indeed, two-domain I2I can be used for many different types of applications as long as the appropriate type and amount of data are provided as the source-target images. Therefore, we refer to the universal taxonomy in machine learning, such as the categorizations used in [34], [35], [36], and classify two-domain I2I methods into four categories based on the different ways of leveraging various sources of information: supervised I2I, unsupervised I2I, semi-supervised I2I and few-shot I2I, as described in following paragraph. We also provide the summary of these two-domain I2I methods including method name, publication year, the type of training data, whether multi-modal or not and corresponding insights. Style transfer is the process of changing a particular characteristic of an image to adopt the style of another one, for example adding color to a black and white image. [37] provides a way to make these modifications while keeping the image realistic by jointly training a discriminator that learns to differentiate between real and generated data and a generator that tries to fool the discriminator. An encoder-decoder with skip-links is used for the generator, and a "PatchGAN" classifier for the discriminator. This discriminator only penalizes high-frequency artefacts, as it classifies each $N \times N$ patch of an image. This method requires to have an available dataset of the wanted style transfer. [38] extends this strategy to unpaired mappings, allowing for example the transformation of a photograph to a painting, and [39] generalizes the GAN by enabling it to learn mappings from and to multiple domains. An application of those techniques can be seen in [40], where the lighting of an image is changed to emulate any time of the day. Instead of learning an image-to-image translation, another way to meaningfully modify an image is by learning an image-to-illumination mapping. This mapping can then be used to relight underexposed photos for instance [41].

B. Illumination Generation

While relighting a scene captured under a large number of different illuminations is an easy process, doing so with one or very few samples is a complex challenge, as multi-illumination

datasets are hard to capture. [42] developed a system to quickly capture scenes under several lighting conditions using a flash bouncing off the walls to illuminate the scene in different directions. It provides a dataset of real scenes captured under 25 lighting conditions, that can be used to train a relighting network. [43] trains a fully convolutional encoder-decoder to relight a scene from a sparse set of input images. It needs 5 images of a same scene under different directional lighting to produce a relit image in a novel light. Both [44] and [45] attempt to relight a portrait from a single image. [44] uses a U-Net, an encoder-decoder with skip connections, to predict the illumination corresponding to the source image and replace it with the target lighting. The network produces great results on faces but the objective functions are applied only to masked part of the images – people – while the backgrounds are not affected but replaced by the blurred version of the environment map. [45] uses a similar architecture, but the encoder extracts facial features in addition to lighting features. The facial feature stays unchanged while the target lighting replaces the input lighting. A PatchGAN is used to improve the quality of the generated image and removes local artifacts. This network gives good results on portrait images but does not generalizes to other scenes, giving poor results on non-portrait images. The paper [46], [47] introduce propose an illumination estimation framework that leverages a regression network and a neural projector for accurate illumination estimation. The paper [48] aims to identify the relation between different tasks – such as reshading, semantic segmentation, surface normals estimation. By using in one task the encoder trained in another task, they compute a similarity between the two tasks. They have for instance shown that reshading is close to finding occlusion edges, both of which are close to Z-depth and distance estimation, but very far from 2D-segmentation, camera pose estimation or scene classification. The paper [49] presents NeedleLight, a new lighting estimation model that represents illumination with needlets and allows lighting estimation in both frequency domain and spatial domain jointly. The paper [50] aims to produce a network for estimating distance between the camera and the points of the images. Authors of [51] analyze how the manipulation of the latent representation on different depth levels of the generative network influences the semantical properties of the generated scenes. Even though it does not directly address the problem of image relighting, the paper claims that the generative networks learn to divide scene synthesis into a hierarchy of steps and that the representations responsible for the color scheme of the image are localized in the deepest part of the network.

Fig. 3: Screenshot from the original GLEAM paper: an example physical light probe.

Fig. 4: Screenshot from the original GLEAM paper: an example cubemap.

III. Methods

A key part for creating realistic AR experiences is getting the lighting right. When a virtual object is missing a shadow or has a shiny material that doesn't reflect the surrounding space, users can sense that the object doesn't quite fit, even if they can't explain why. This is because humans subconsciously perceive cues regarding how objects are lit in their environment. The Lighting Estimation API analyzes given images for such cues, providing detailed information about the lighting in a scene. You can then use this information when rendering virtual objects to light them under the same conditions as the scene they're placed in, keeping users grounded and engaged.

In a nutshell, Xihe leverages 3D vision data (RGB-D) to generate point clouds at the rendering position which are then passed to a lightweight neural network to predict lighting represented in Spherical Harmonics (SH) coefficients. (You might also be interested in other mobile AR-related papers presented at MobiSys'21. Check out my previous post.) What are SH and SH coefficients? SH are functions defined over the sphere and were introduced by Ramamoorthi et al. for creating realistic rendering. SH coefficients can be used as a compact lighting representation of diffuse irradiance map. What is the

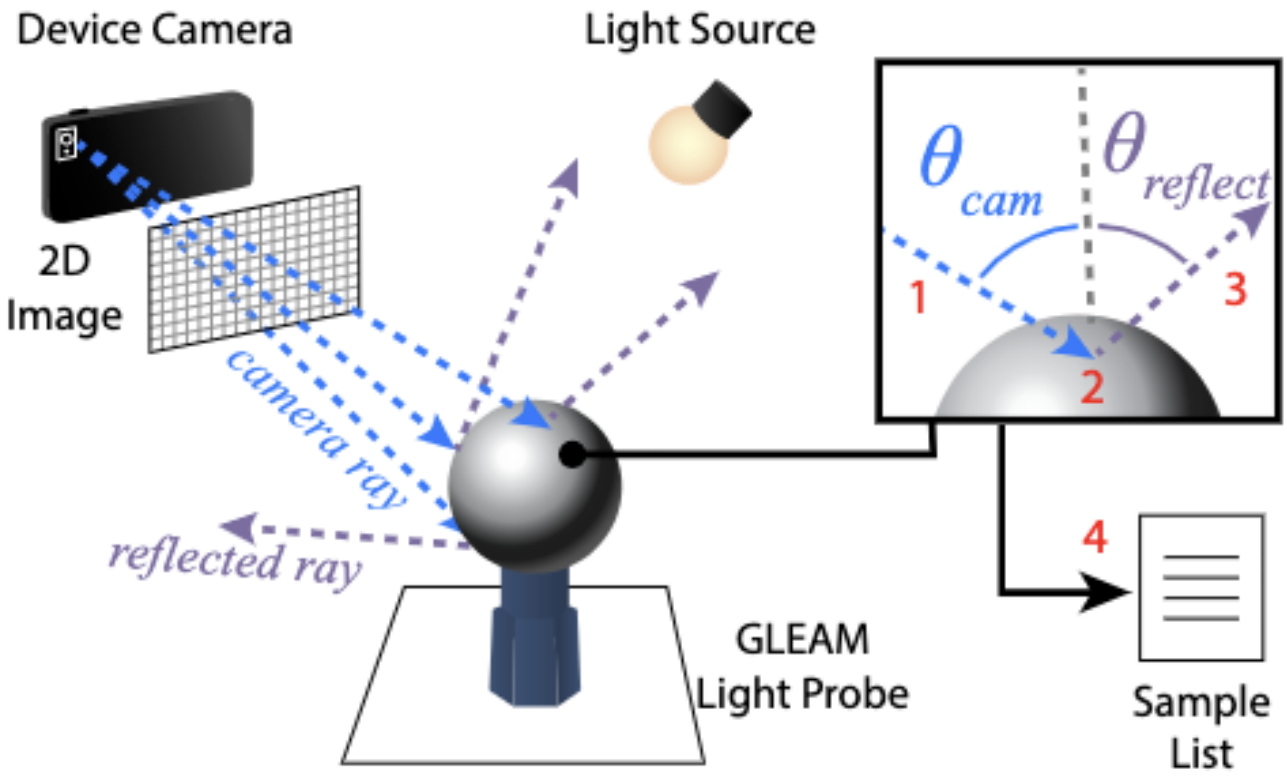

Fig. 5: Screenshot from the original GLEAM paper.

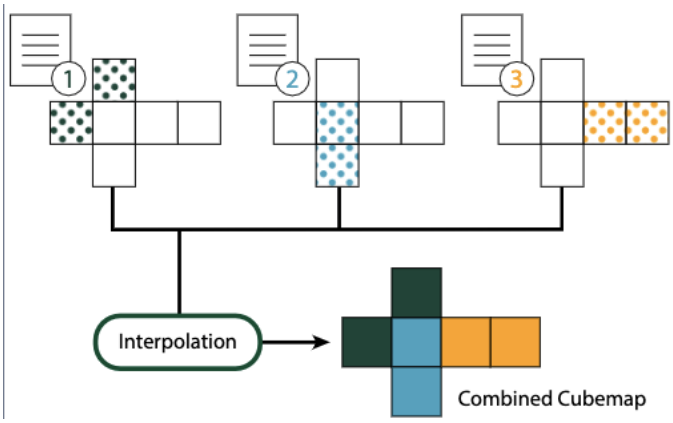

Fig. 6: Screenshot from the original GLEAM paper.

difference between the placement position, estimation position, and observation position? The observation position is defined as where the camera (and mobile user) is at; the placement position refers to the geometric center of a rendered virtual object while the estimation position corresponds to lighting representations. In other words, larger objects can benefit from having more than one estimation position.

The first key novelty is a new point cloud sampling technique that is both fast and accuracy-preserving. This new sampling technique is called unit-sphere point cloud sampling which effectively compresses raw point clouds by projecting them to a unit-sphere. The diagram below illustrates an example of projecting three points (p1, p2, p3) to the unit-sphere and leads to two colored anchors (a1, a2). In essence, this sampling technique leverages the sphere geometric information to sample points based on their viewing directions to preserve observation coverage as much as possible. Putting theory aside, supporting this sampling technique on mobile devices (even on high-end devices such as iPad Pro) can be challenging in large part due to the sheer number of points and nearest neighbor algorithm. How does Xihe satisfy the real-time goal? Xihe designed a tailored GPU pipeline for managing captured point cloud samples and a densely sampled 2D acceleration grid to trade-off storage and computation. The second key novelty is a co-designed neural network called XiheNet that takes the downsampled point cloud and generates lighting information, presented as SH coefficients. Due to the lack of some operator supports for mobile devices, XiheNet will be hosted at the edge devices running Ubuntu distribution. To minimize network impact on the edge-based inference performance, Xihe uses the following scheme to encode the downsampled point clouds. Rather than using float32 to encode (R, G, B, X, Y, Z), each point will be encoded with lower precision data formats and an anchor index. Further, Xihe removes all uninitialized anchors in a striping step before sending the encoded version from the device to the edge.

The last novelty is an adaptive triggering strategy that skips sending lighting estimation requests to the edge based on the unit sphere-based point cloud difference between consecutive

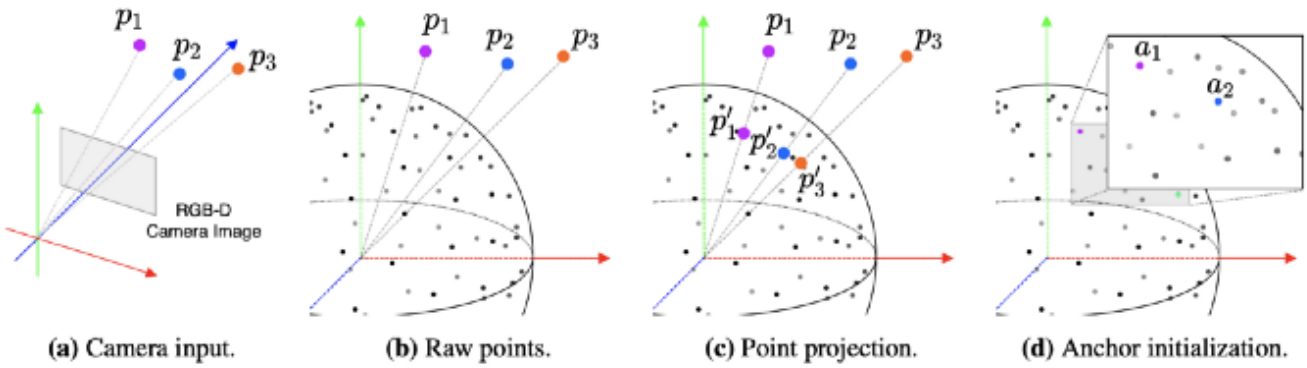

Fig. 7: Screenshot from the original Xihe paper.

Fig. 8: Screenshot from the original Xihe slides.

frames. By using this adaptive strategy, Xihe can effectively respond to the dynamic environments while avoiding unnecessary network communication. The strategy is currently implemented using sphere pooling and empirically derived threshold and pooling window size. One of the challenges lies in how to evaluate the effectiveness of the adaptive triggering strategy. Why is it difficult? Largely because (1) we don't have access to lighting ground truth and (2) lighting conditions are hard to control. To solve this challenge, Xihe developed an AR session recorder that allows record-and-replay!

Xihe presents a 3D vision-based framework that achieves real-time lighting estimation. Xihe is fully open-sourced and consists of both client and server APIs as shown below. Additionally, Xihe also provides an accompanying iOS Application with features that facilitate debugging. There are so many open challenges and opportunities in leveraging 3D vision to provide a feature-rich mobile AR experience. We are currently working on improving Xihe to more photorealistically render 3D objects of different shapes and materials! Stay tuned! If you are interested in Augmented Reality, consider following me on Twitter to get updates from both commercial and research worlds!

Lighting cues The Lighting Estimation API provides detailed data that lets you mimic various lighting cues when rendering virtual objects. These cues are shadows, ambient light, shading, specular highlights, and reflections. Shadows Shadows are often directional and tell viewers where light sources are coming from. Ambient light Ambient light is the overall diffuse light that comes in from around the environment, making everything visible. Shading Shading is the intensity of the light. For example, different parts of the same object can have different levels of shading in the same scene, depending on angle relative to the viewer and its proximity to a light source. Specular highlights Specular highlights are the shiny bits of surfaces that reflect a light source directly. Highlights on an object change relative to the position of a viewer in a scene. Reflections Light bounces off of surfaces differently depending on whether the surface has specular (highly reflective) or diffuse (not reflective) properties. For example, a metallic ball will be highly specular and reflect its environment, while another ball painted a dull matte gray will be diffuse. Most real-world objects have a combination of these properties — think of a scuffed-up bowling ball or a well-used credit card. Reflective surfaces also pick up colors from the ambient environment. The coloring of an object can be directly affected by the coloring of its environment. For example, a white ball in a blue room will take on a bluish hue. Environmental HDR mode These modes consist of separate APIs that allow for granular and realistic lighting estimation for directional lighting, shadows, specular highlights, and reflections. Environmental HDR mode uses machine learning to analyze the camera images in real time and synthesize environmental lighting to support realistic rendering of virtual objects.

This lighting estimation mode provides: Main directional light. Represents the main light source. Can be used to cast shadows. Ambient spherical harmonics. Represents the remaining ambient light energy in the scene. An HDR cubemap. Can be used to render reflections in shiny metallic objects. You can use these APIs in different combinations, but they're designed to be used together for the most realistic effect. Main directional light The main directional light API calculates

the direction and intensity of the scene's main light source. This information allows virtual objects in your scene to show reasonably positioned specular highlights, and to cast shadows in a direction consistent with other visible real objects. To see how this works, consider these two images of the same virtual rocket. In the image on the left, there's a shadow under the rocket but its direction doesn't match the other shadows in the scene. In the rocket on the right, the shadow points in the correct direction. It's a subtle but important difference, and it grounds the rocket in the scene because the direction and intensity of the shadow better match other shadows in the scene.

When the main light source or a lit object is in motion, the specular highlight on the object adjusts its position in real time relative to the light source. Directional shadows also adjust their length and direction relative to the position of the main light source, just as they do in the real world. To illustrate this effect, consider these two mannequins, one virtual and the other real. The mannequin on the left is the virtual one. In addition to the light energy in the main directional light, ARCore provides spherical harmonics, representing the overall ambient light coming in from all directions in the scene. Use this information during rendering to add subtle cues that bring out the definition of virtual objects. Consider these two images of the same rocket model. The rocket on the left is rendered using lighting estimation information detected by the main directional light API. The rocket on the right is rendered using information detected by both the main direction light and ambient spherical harmonics APIs. The second rocket clearly has more visual definition, and blends more seamlessly into the scene.

Use the HDR cubemap to render realistic reflections on virtual objects with medium to high glossiness, such as shiny metallic surfaces. The cubemap also affects the shading and appearance of objects. For example, the material of a specular object surrounded by a blue environment will reflect blue hues. Calculating the HDR cubemap requires a small amount of additional CPU computation. Whether you should use the HDR cubemap depends on how an object reflects its surroundings. Because the virtual rocket is metallic, it has a strong specular component that directly reflects the environment around it. Thus, it benefits from the cubemap. On the other hand, a virtual object with a dull gray matte material doesn't have a specular component at all. Its color primarily depends on the diffuse component, and it wouldn't benefit from a cubemap. All three Environmental HDR APIs were used to render the rocket below. The HDR cubemap enables the reflective cues and further highlighting that ground the object fully in the scene.

Ambient Intensity mode determines the average pixel intensity and the color correction scalars for a given image. It's a coarse setting designed for use cases where precise lighting is not critical, such as objects that have baked-in lighting. Pixel intensity Captures the average pixel intensity of the lighting in a scene. You can apply this lighting to a whole virtual object. Color Detects the white balance for each individual frame. You can then color correct a virtual object so that it integrates more smoothly into the overall coloring of the scene. Environment probes Environment probes organize 360-degree camera views into environment textures such as cube maps. These textures can then be used to realistically light virtual objects, such as a virtual metal ball that "reflects" the room it's in.

REFERENCES

[1] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[2] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via maximum mean discrepancy optimization," *arXiv preprint arXiv:1505.03906*, 2015.

[3] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network," in *Neural Information Processing Systems*, 2017, pp. 2203–2213.

[4] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *International Conference on Machine Learning*, 2015, pp. 1718–1727.

[5] D. J. Sutherland, H.-Y. Tung, H. Strathmann, S. De, A. Ramdas, A. Smola, and A. Gretton, "Generative models and model criticism via optimized maximum mean discrepancy," *arXiv preprint arXiv:1611.04488*, 2016.

[6] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.

[7] J. Wu, Z. Huang, J. Thoma, D. Acharya, and L. Van Gool, "Wasserstein divergence for gans," in *European Conference on Computer Vision*, 2018, pp. 653–668.

[8] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The cramer distance as a solution to biased wasserstein gradients," *arXiv preprint arXiv:1705.10743*, 2017.

[9] H. Petzka, A. Fischer, and D. Lukovnicov, "On the regularization of wasserstein gans," in *International Conference on Learning Representations*, 2018, pp. 1–24.

[10] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Gang of gans: Generative adversarial networks with maximum margin ranking," *arXiv preprint arXiv:1704.04865*, 2017.

[11] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks," in *Interspeech*, 2017, pp. 3364–3368.

[12] J. Adler and S. Lunz, "Banach wasserstein gan," in *Neural Information Processing Systems*, 2018, pp. 6754–6763.

[13] Y.-S. Chen, Y.-C. Wang, M.-H. Kao, and Y.-Y. Chuang, "Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6306–6314.

[14] S. Athey, G. Imbens, J. Metzger, and E. Munro, "Using wasserstein generative adversial networks for the design of monte carlo simulations," *arXiv preprint arXiv:1909.02210*, 2019.

[15] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Neural Information Processing Systems*, 2016, pp. 2172–2180.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Neural Information Processing Systems*, 2014, pp. 2672–2680.

[17] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International Conference on Machine Learning*, 2017, pp. 2642–2651.

[18] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using laplacian pyramid of adversarial networks," in *Neural Information Processing Systems*, 2015, pp. 1486–1494.

[19] G. Perarnau, J. Van De Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional gans for image editing," *arXiv preprint arXiv:1611.06355*, 2016.

[20] M. Saito, E. Matsumoto, and S. Saito, "Temporal generative adversarial nets with singular value clipping," in *IEEE International Conference on Computer Vision*, 2017, pp. 2830–2839.

[21] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[22] K. Sricharan, R. Bala, M. Shreve, H. Ding, K. Saketh, and J. Sun, "Semi-supervised conditional gans," *arXiv preprint arXiv:1708.05789*, 2017.

[23] F. Zhan, S. Lu, C. Zhang, F. Ma, and X. Xie, "Adversarial image composition with auxiliary illumination," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[24] F. Zhan and C. Zhang, "Spatial-aware gan for unsupervised person re-identification," *Proceedings of the International Conference on Pattern Recognition*, 2020.

[25] T. Miyato and M. Koyama, "cgans with projection discriminator," *arXiv preprint arXiv:1802.05637*, 2018.

[26] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *International Conference on Computer Vision*, 2017, pp. 2223–2232.

[27] H.-Y. Lee, H.-Y. Tseng, Q. Mao, J.-B. Huang, Y.-D. Lu, M. Singh, and M.-H. Yang, "Drit++: Diverse image-to-image translation via disentangled representations," *International Journal of Computer Vision*, pp. 1–16, 2020.

[28] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[29] P. Zhu, R. Abdal, Y. Qin, and P. Wonka, "Sean: Image synthesis with semantic region-adaptive normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[30] P. L. Suárez, A. D. Sappa, and B. X. Vintimilla, "Infrared image colorization based on a triplet dcgan architecture," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 18–23.

[31] J. Lee, E. Kim, Y. Lee, D. Kim, J. Chang, and J. Choo, "Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[32] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[33] Y. Zhang, S. Liu, C. Dong, X. Zhang, and Y. Yuan, "Multiple cycle-in-cycle generative adversarial networks for unsupervised image super-resolution," *IEEE transactions on Image Processing*, vol. 29, pp. 1101–1112, 2019.

[34] R. Navigli, "Word sense disambiguation: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 2, pp. 1–69, 2009.

[35] G.-J. Qi and J. Luo, "Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[36] L. Schmarje, M. Santarossa, S.-M. Schröder, and R. Koch, "A survey on semi-, self- and unsupervised learning for image classification," 2020.

[37] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5967–5976, code: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.

[38] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[39] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.

[40] I. Anokhin, P. Solovev, D. Korzhenkov, A. Kharlamov, T. Khakhulin, G. Sterkin, A. Silvestrov, S. Nikolenko, and V. Lempitsky, "High-resolution daytime translation without domain labels," *arXiv preprint arXiv:2003.08791*, 2020.

[41] R. Wang, Q. Zhang, C.-W. Fu, X. Shen, W.-S. Zheng, and J. Jia, "Underexposed photo enhancement using deep illumination estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6849–6857.

[42] L. Murmann, M. Gharbi, M. Aittala, and F. Durand, "A dataset of multi-illumination images in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4080–4089.

[43] Z. Xu, K. Sunkavalli, S. Hadap, and R. Ramamoorthi, "Deep image-based relighting from optimal sparse samples," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–13, 2018.

[44] T. Sun, J. T. Barron, Y.-T. Tsai, Z. Xu, X. Yu, G. Fyffe, C. Rhemann, J. Busch, P. Debevec, and R. Ramamoorthi, "Single image portrait relighting," *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 2019.

[45] H. Zhou, S. Hadap, K. Sunkavalli, and D. W. Jacobs, "Deep single-image portrait relighting," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7194–7202.

[46] F. Zhan, C. Zhang, Y. Yu, Y. Chang, S. Lu, F. Ma, and X. Xie, "Emlight: Lighting estimation via spherical distribution approximation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 4, 2021, pp. 3287–3295.

[47] F. Zhan, Y. Yu, R. Wu, C. Zhang, S. Lu, L. Shao, F. Ma, and X. Xie, "Gmlight: Lighting estimation via geometric distribution approximation," *arXiv preprint arXiv:2102.10244*, 2021.

[48] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3712–3722.

[49] F. Zhan, C. Zhang, W. Hu, S. Lu, F. Ma, X. Xie, and L. Shao, "Sparse needlets for lighting estimation with spherical transport loss," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 830–12 839.

[50] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[51] C. Yang, Y. Shen, and B. Zhou, "Semantic hierarchy emerges in deep generative representations for scene synthesis," *arXiv preprint arXiv:1911.09267*, 2019.