

Marigold & Depth Anything V2



Hanyang University, Department of Automotive engineering



Chungwoo Lee

Paper review

Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation

Bingxin Ke Anton Obukhov Shengyu Huang
Nando Metzger Rodrigo Caye Daudt Konrad Schindler
Photogrammetry and Remote Sensing, ETH Zürich



Figure 1: We present Marigold, a diffusion model and associated fine-tuning protocol for monocular depth estimation. Its core principle is to leverage the rich visual knowledge stored in modern generative image models. Our model, derived from Stable Diffusion and fine-tuned with synthetic data, can zero-shot transfer to unseen datasets, offering state-of-the-art monocular depth estimation results.

Abstract

Monocular depth estimation is a fundamental computer vision task. Recovering 3D depth from a single image is geometrically ill-posed and requires scene understanding, so it is not surprising that the rise of deep learning has led to a breakthrough. The impressive progress of monocular depth estimators has mirrored the growth in model capacity, from relatively modest CNNs to large Transformer architectures. Still, monocular depth estimators tend to struggle when presented with images with unfamiliar content and layout, since their knowledge of the visual world is restricted by the data seen during training, and challenged by zero-shot generalization to new domains. This motivates us to explore whether the extensive priors captured in recent generative diffusion models can enable better, more generalizable depth estimation. We introduce Marigold, a method for affine-invariant

monocular depth estimation that is derived from Stable Diffusion and retains its rich prior knowledge. The estimator can be fine-tuned in a couple of days on a single GPU using only synthetic training data. It delivers state-of-the-art performance across a wide range of datasets, including over 20% performance gains in specific cases. Project page: <https://marigoldmonodepth.github.io>.

1. Introduction

Monocular depth estimation aims to transform a photographic image into a depth map, i.e., regress a range value for every pixel. The task arises whenever the 3D scene structure is needed, and no direct range or stereo measurements are available. Clearly, undoing the projection from the 3D world to a 2D image is a geometrically ill-posed problem and can

<Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation>

Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, Konrad Schindler

Depth Anything V2

Lihe Yang¹ Bingyi Kang^{2†} Zilong Huang²
Zhen Zhao Xiaogang Xu Jiashi Feng² Hengshuang Zhao^{1‡}
¹HKU ²TikTok
[†]project lead [‡]corresponding author
<https://depth-anything-v2.github.io>

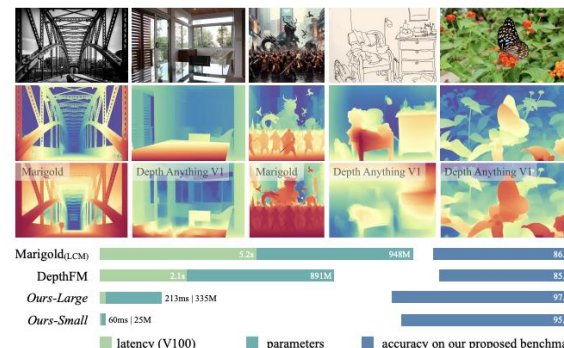


Figure 1: Depth Anything V2 significantly outperforms V1 [89] in robustness and fine-grained details. Compared with SD-based models [31, 25], it enjoys faster inference speed, fewer parameters, and higher depth accuracy. Please refer to Figure 13 and Figure 14 for more qualitative comparisons with V1 and Marigold.

Abstract

This work presents *Depth Anything V2*. Without pursuing fancy techniques, we aim to reveal crucial findings to pave the way towards building a powerful monocular depth estimation model. Notably, compared with V1 [89], this version produces much finer and more robust depth predictions through three key practices: 1) replacing all labeled real images with synthetic images, 2) scaling up the capacity of our teacher model, and 3) teaching student models via the bridge of large-scale pseudo-labeled real images. Compared with the latest models [31] built on Stable Diffusion, our models are significantly more efficient (more than 10× faster) and more accurate. We offer models of different scales (ranging from 25M to 1.3B params) to support extensive scenarios. Benefiting from their strong generalization capability, we fine-tune them with metric depth labels to obtain our metric depth models. In addition to our models, considering the limited diversity and frequent noise in current test sets, we construct a versatile evaluation benchmark with precise annotations and diverse scenes to facilitate future research.

Work done during an internship at TikTok.

<Depth Anything V2>

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, Hengshuang Zhao

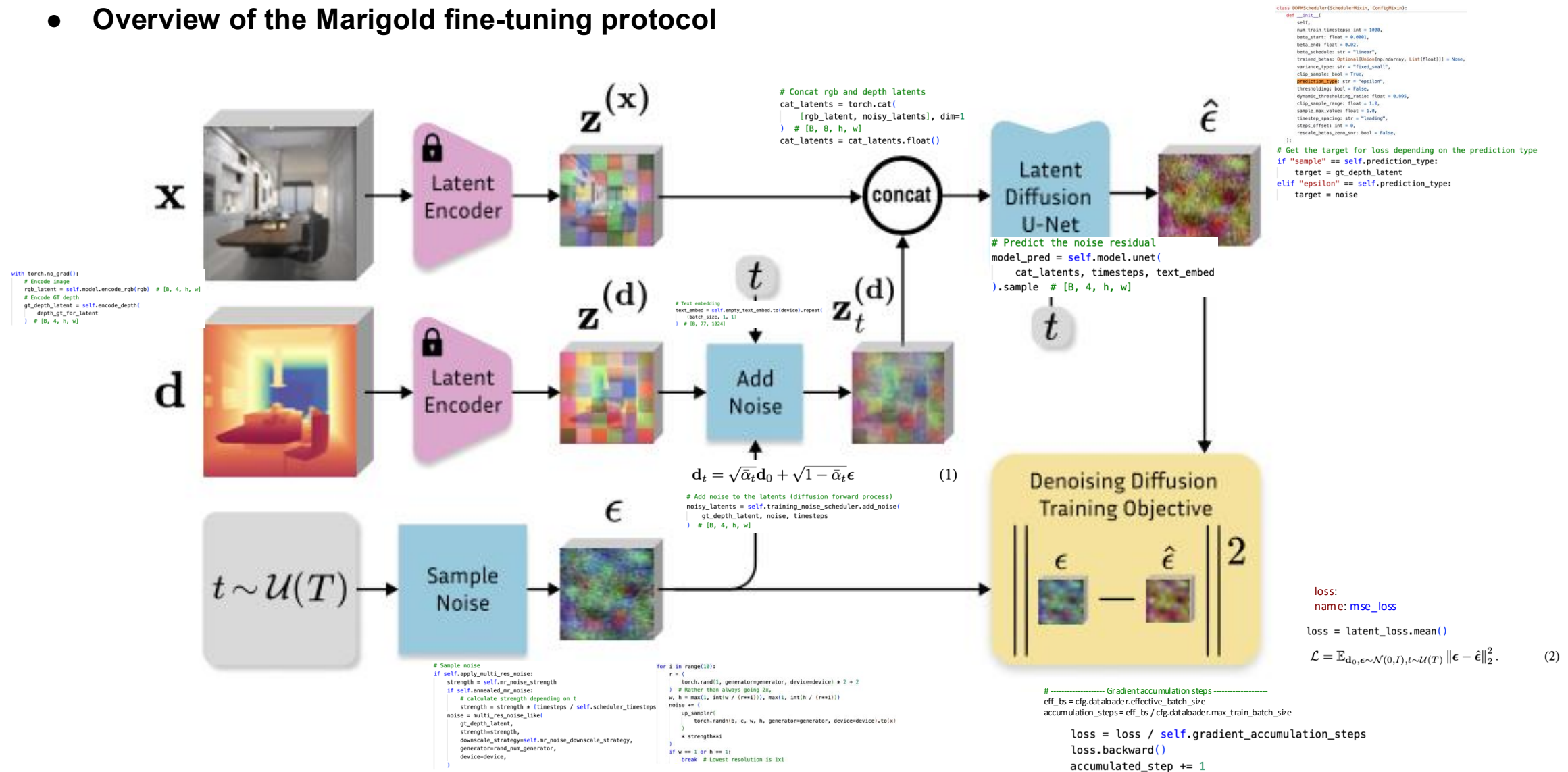
Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation

Abstract

Monocular depth estimation is a fundamental computer vision task. Recovering 3D depth from a single image is geometrically ill-posed and requires scene understanding, so it is not surprising that the rise of deep learning has led to a breakthrough. The impressive progress of monocular depth estimators has mirrored the growth in model capacity, from relatively modest CNNs to large Transformer architectures. Still, monocular depth estimators tend to struggle when presented with images with unfamiliar content and layout, since their knowledge of the visual world is restricted by the data seen during training, and challenged by zero-shot generalization to new domains. This motivates us to explore whether the extensive priors captured in recent generative diffusion models can enable better, more generalizable depth estimation. We introduce Marigold, a method for affine-invariant monocular depth estimation that is derived from Stable Diffusion and retains its rich prior knowledge. The estimator can be fine-tuned in a couple of days on a single GPU using only synthetic training data. It delivers state-of-the-art performance across a wide range of datasets, including over 20% performance gains in specific cases. Project page: <https://marigoldmonodepth.github.io>.

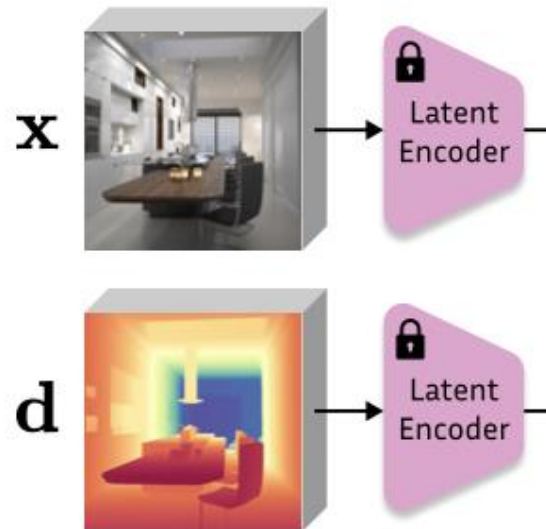
Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation

- Overview of the Marigold fine-tuning protocol



Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation

- Encoding image and GT depth

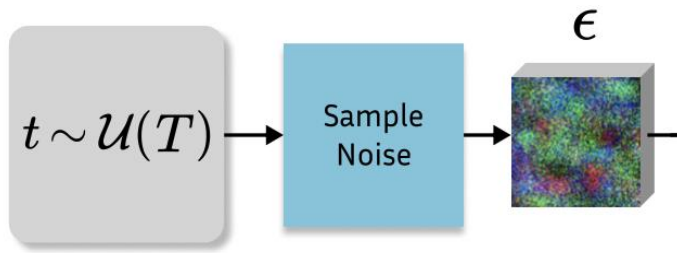


```
with torch.no_grad():  
    # Encode image  
    rgb_latent = self.model.encode_rgb(rgb) # [B, 4, h, w]  
    # Encode GT depth  
    gt_depth_latent = self.encode_depth(  
        depth_gt_for_latent  
    ) # [B, 4, h, w]
```

```
def encode_rgb(self, rgb_in: torch.Tensor) -> torch.Tensor:  
    # encode  
    h = self.vae.encoder(rgb_in)  
    moments = self.vae.quant_conv(h)  
    mean, logvar = torch.chunk(moments, 2, dim=1)  
    # scale latent  
    rgb_latent = mean * self.rgb_latent_scale_factor  
    return rgb_latent
```

```
def decode_depth(self, depth_latent: torch.Tensor) -> torch.Tensor:  
    # scale latent  
    depth_latent = depth_latent / self.depth_latent_scale_factor  
    # decode  
    z = self.vae.post_quant_conv(depth_latent)  
    stacked = self.vae.decoder(z)  
    # mean of output channels  
    depth_mean = stacked.mean(dim=1, keepdim=True)  
    return depth_mean
```


Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation



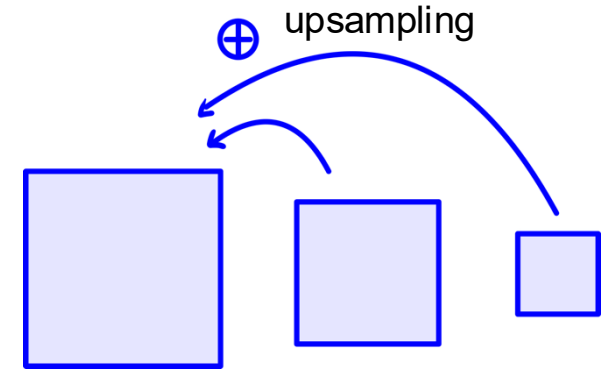
```
if self.apply_multi_res_noise:
    strength = self.mr_noise_strength
    if self.annealed_mr_noise:
        # calculate strength depending on t
        strength = strength * (timesteps / self.scheduler_timesteps)
    noise = multi_res_noise_like(
        gt_depth_latent,
        strength=strength,
        downscale_strategy=self.mr_noise_downscale_strategy,
        generator=rand_num_generator,
        device=device,
    )

if "original" == downscale_strategy:
    for i in range(10):
        r = (
            torch.rand(1, generator=generator, device=device) * 2 + 2
        ) # Rather than always going 2x,
        w, h = max(1, int(w / (r**i))), max(1, int(h / (r**i)))
        noise += (
            up_sampler(
                torch.randn(b, c, w, h, generator=generator, device=device).to(x)
            )
            * strength**i
        )
    if w == 1 or h == 1:
        break # Lowest resolution is 1x1
```

Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation

- **Multi-resolution noise**

- ◆ 고정된 noise를 추가할 때 고주파 정보가 저주파 정보보다 빠르게 소멸
 - ▶ 고주파 정보: 픽셀 변화가 빠른 것
(머리카락, 옷감의 질감 등등...)
 - ▶ 저주파 정보 이미지의 전반적인 형태나 큰 구조
(하늘과 같이 균일하고 큰 면적을 차지하는 것, 점진적으로 변하는 색상 구역 등등...)



- **Annealed schedule**

- ◆ 최종 noise image 에 가까울 수록 noise의 weight를 더 줌

calculate strength depending on t

strength = strength * (timesteps / self.scheduler_timesteps)

Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation

Can be sampled from Gaussian by reparameterization

$$\begin{aligned} q(x_t | x_{t-1}) &= \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t \mathbf{I}) \\ x_t &= \sqrt{1 - \beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \\ &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\ &\quad \text{and } \alpha_t := 1 - \beta_t \\ \mathbf{d}_t &= \sqrt{\bar{\alpha}_t}\mathbf{d}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \end{aligned} \tag{1}$$

Sample x_t at any t in a closed form using reparameterization trick

$$\begin{aligned} x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1 - \alpha_t}\epsilon_{t-1} \\ &= \sqrt{\alpha_t} \left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1 - \alpha_{t-1}}\epsilon_{t-2} \right) + \sqrt{1 - \alpha_t}\epsilon_{t-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1 - \alpha_{t-1}}\sqrt{\alpha_t}\epsilon_{t-2} + \sqrt{1 - \alpha_t}\epsilon_{t-1} \\ &= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \quad \text{where } \bar{\alpha}_t = \prod_{s=1}^t \alpha_s \end{aligned}$$

$z_1 \sim \mathcal{N}(0, \sigma_1^2 \mathbf{I})$ and $z_2 \sim \mathcal{N}(0, \sigma_2^2 \mathbf{I})$, then $z_1 + z_2 \sim \mathcal{N}(0, (\sigma_1^2 + \sigma_2^2) \mathbf{I})$

$$\left(\sqrt{\alpha_t} \sqrt{1 - \alpha_{t-1}} \right)^2 + \left(\sqrt{1 - \alpha_t} \right)^2 = \alpha_t(1 - \alpha_{t-1}) + (1 - \alpha_t) = 1 - \alpha_t\alpha_{t-1}$$

Background knowledge

베이시안 rule, 마르코프 법칙

$$q(x_{t-1} | x_t, x_0) = \frac{q(x_t | x_{t-1}, x_0) q(x_{t-1} | x_0)}{q(x_t | x_0)}$$

$$= \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I) \propto \exp \left(-\frac{(x_t - \sqrt{\alpha_t} x_{t-1})^2}{2\beta_t} - \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}} x_0)^2}{2(1 - \bar{\alpha}_{t-1})} + \frac{(x_t - \sqrt{\bar{\alpha}_t} x_0)^2}{2(1 - \bar{\alpha}_t)} \right)$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t} (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t$$

$$= \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right)$$

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

x0를 대입 후 식을 정리

$$\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$$

cf)

$$q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, \beta_t I)$$

$$q(x_t | x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)$$

$$q(x_{t-1} | x_0) = \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}} x_0, (1 - \bar{\alpha}_{t-1}) I)$$

Learning을 위한 파라미터들을
constant로 설정

Background knowledge

$$\begin{aligned} D_{KL} [q(x_{t-1} | x_t, x_0) \parallel p_{\theta}(x_{t-1} | x_t)] &= \frac{1}{2} \left(D \frac{\tilde{\beta}_t}{\sigma_t^2} + \frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_{\theta}(x_t, t)\|^2 - D + \ln \frac{\sigma_t^2}{\tilde{\beta}_t} \right) \\ &= \frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_{\theta}(x_t, t)\|^2 + \text{constant (independent of } \theta) \end{aligned}$$

Learning을 위한 파라미터들을
constant로 설정

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right)$$

$$\begin{aligned} \frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_{\theta}(x_t, t)\|^2 &= \frac{1}{2\sigma_t^2} \left\| \frac{\beta_t}{\sqrt{\alpha_t} \sqrt{1 - \bar{\alpha}_t}} (\epsilon - \epsilon_{\theta}(x_t, t)) \right\|^2 \\ &= \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_{\theta}(x_t, t)\|^2 \\ &= \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2 \end{aligned}$$

Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation

- **Affine-invariant depth normalization**

- ◆ Affine 변환에 대한 불변성은 깊이 값이 scale 이나 shift 에 의해 변하더라도 결과가 같게 유지 하기 위해.
- ◆ Stable diffusion VAE와 호환되기 위한 범위 설정.

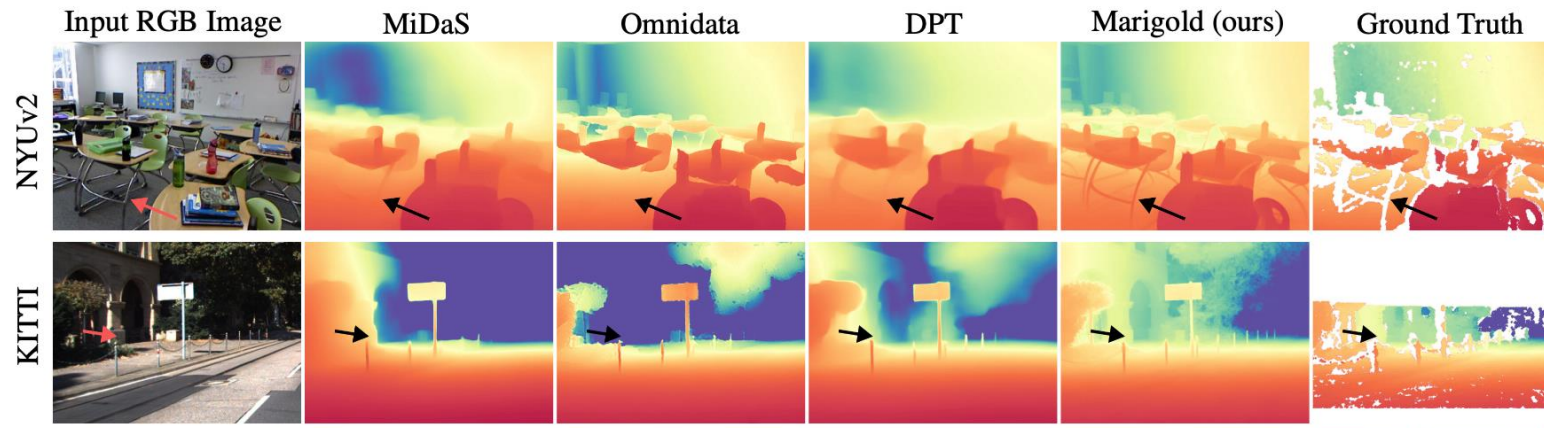
$$\tilde{\mathbf{d}} = \left(\frac{\mathbf{d} - \mathbf{d}_2}{\mathbf{d}_{98} - \mathbf{d}_2} - 0.5 \right) \times 2 \quad [-1,1]$$

- **Test-time ensembling**

$$\min_{\substack{s_1, \dots, s_N \\ t_1, \dots, t_N}} \left(\sqrt{\frac{1}{b} \sum_{i=1}^{N-1} \sum_{j=i+1}^N \|\hat{\mathbf{d}}'_i - \hat{\mathbf{d}}'_j\|_2^2} + \lambda \mathcal{R} \right) \quad \text{where } \hat{\mathbf{d}}' = \hat{\mathbf{d}} \times \hat{s} + \hat{t}.$$

Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation

Method	# Training samples		NYUv2		KITTI		ETH3D		ScanNet		DIODE		Avg. Rank
	Real	Synthetic	AbsRel↓	$\delta 1\uparrow$	AbsRel↓	$\delta 1\uparrow$	AbsRel↓	$\delta 1\uparrow$	AbsRel↓	$\delta 1\uparrow$	AbsRel↓	$\delta 1\uparrow$	
DiverseDepth [56]	320K	—	11.7	87.5	19.0	70.4	22.8	69.4	10.9	88.2	37.6	63.1	7.6
MiDaS [35]	2M	—	11.1	88.5	23.6	63.0	18.4	75.2	12.1	84.6	33.2	71.5	7.3
LeReS [57]	300K	54K	9.0	91.6	14.9	78.4	17.1	77.7	9.1	91.7	27.1	76.6	5.2
Omnidata [13]	11.9M	310K	7.4	94.5	14.9	83.5	16.6	77.8	7.5	93.6	33.9	74.2	4.8
HDN [60]	300K	—	6.9	94.8	11.5	86.7	12.1	83.3	8.0	93.9	<u>24.6</u>	78.0	3.2
DPT [36]	1.2M	188K	9.8	90.3	<u>10.0</u>	90.1	7.8	94.6	8.2	93.4	18.2	75.8	3.9
Ours (w/o ensemble)	—*	74K	<u>6.0</u>	<u>95.9</u>	10.5	<u>90.4</u>	<u>7.1</u>	<u>95.1</u>	<u>6.9</u>	<u>94.5</u>	31.0	77.2	<u>2.5</u>
Ours (w/ ensemble)	—	74K	5.5	96.4	9.9	91.6	6.5	96.0	6.4	95.1	30.8	<u>77.3</u>	1.4



Depth Anything V2

Abstract

This work presents *Depth Anything V2*. Without pursuing fancy techniques, we aim to reveal crucial findings to pave the way towards building a powerful monocular depth estimation model. Notably, compared with V1 [89], this version produces much finer and more robust depth predictions through three key practices: 1) replacing all labeled real images with synthetic images, 2) scaling up the capacity of our teacher model, and 3) teaching student models via the bridge of large-scale pseudo-labeled real images. Compared with the latest models [31] built on Stable Diffusion, our models are significantly more efficient (more than $10\times$ faster) and more accurate. We offer models of different scales (ranging from 25M to 1.3B params) to support extensive scenarios. Benefiting from their strong generalization capability, we fine-tune them with metric depth labels to obtain our metric depth models. In addition to our models, considering the limited diversity and frequent noise in current test sets, we construct a versatile evaluation benchmark with precise annotations and diverse scenes to facilitate future research.

Depth Anything V2

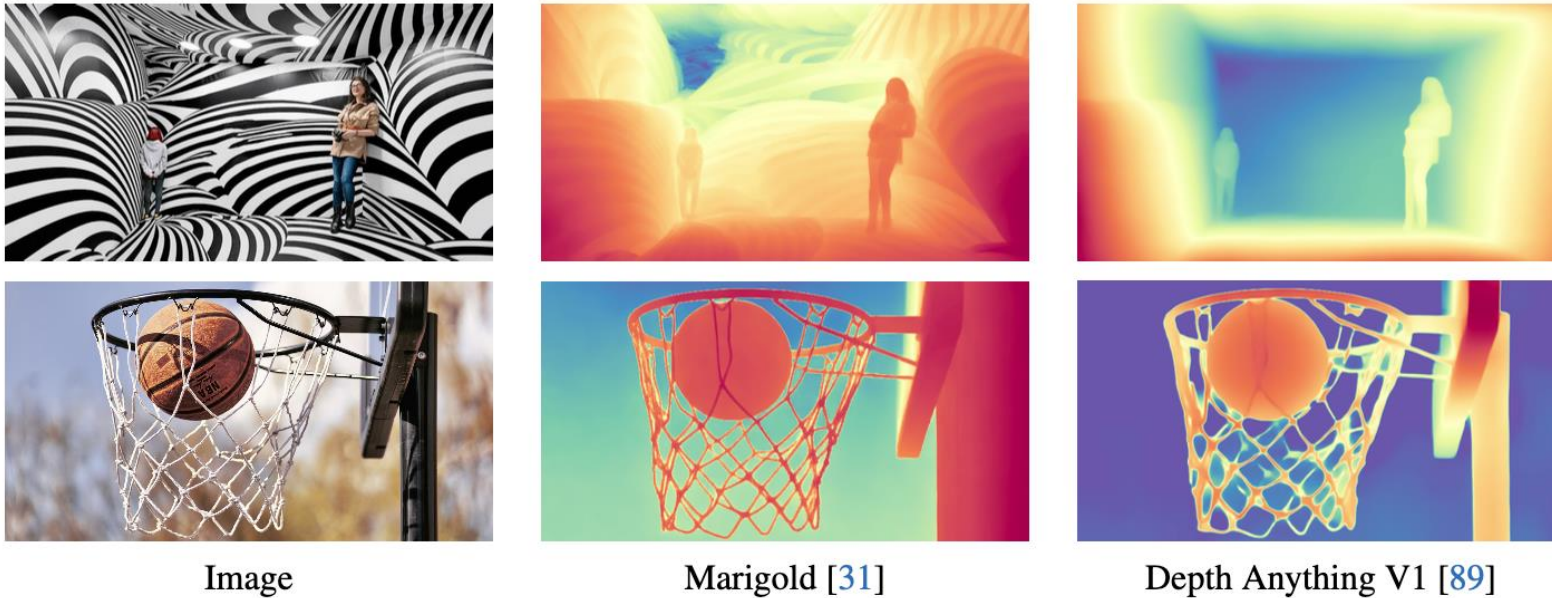


Figure 2: *Robustness* (1st row, the mislading room layout) of Depth Anything V1 and *Fine-grained detail* (2nd row, the thin basketball net) of Marigold.

Preferable Properties	Fine Detail	Transparent Objects	Reflections	Complex Scenes	Efficiency	Transferability
Marigold [31]	✓	✓	✓	✗	✗	✗
Depth Anything V1 [89]	✗	✗	✗	✓	✓	✓
Depth Anything V2 (Ours)	✓	✓	✓	✓	✓	✓

Table 1: Preferable properties of a powerful monocular depth estimation model.

Depth Anything V2

- **Disadvantages of real labeled data**

- ◆ Label noise

- ▶ 깊이 센서가 투명한 물체의 깊이를 정확하게 측정하지 못함.
 - ▶ Stereo matching algorithms 이 반복적인 패턴을 처리하는데 취약함.

- ◆ Ignored details

- ▶ 나무나 의자와 같은 객체의 깊이 표현이 세밀하지 않다.
 - ▶ 얇은 구멍과 같은 부분에서 세밀한 정보를 주지 못한다.

- **Advantages of synthetic images**

- ◆ Precise Depth Labels for all fine details
 - ◆ Fast Data Expansion

- **Limitations of synthetic images**

- ◆ Bias in data distribution
 - ◆ Incomplete Data
 - ◆ Insufficient Noise Level
 - ◆ Neglecting Temporal and Dynamic aspect

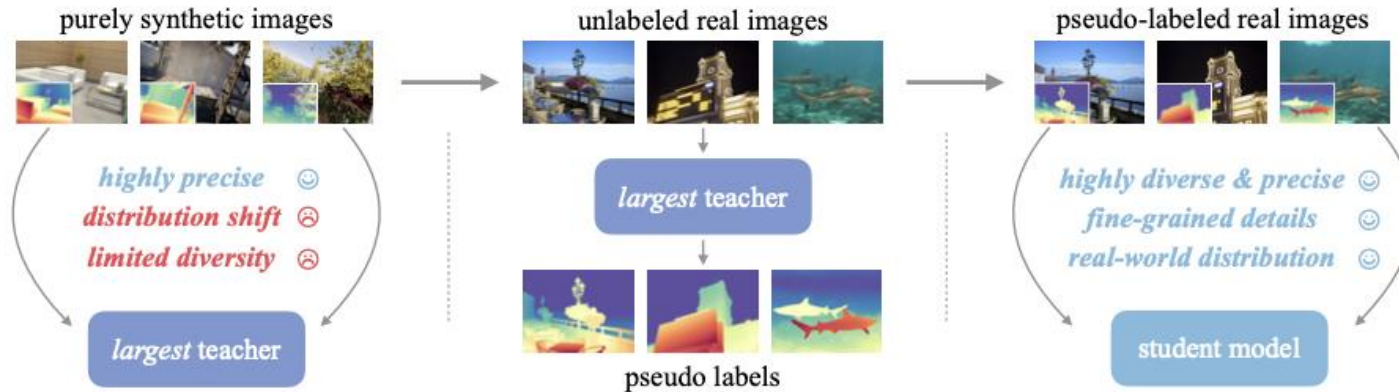
Depth Anything V2

- Pretrained encoder로 합성한 이미지만으로 학습한 모델로 depth 추론에 심각한 일반화 문제 발생함.
- 더 많은 데이터를 모아 학습할 수는 있지만, 현실적으로 모든 시나리오를 흉내 낼 수 있는 그래픽 엔진을 만드는 것은 비효율적임.
- 합성데이터와 실제 이미지와 함께 사용하여 학습을 진행했으나, 오히려 성능 저하
실제 이미지의 coarse한 depth map이 세밀한 예측에 악영향을 미침.
- 합성 이미지를 더 많이 사용하여 Dataset 양을 늘리려 했지만 효과적이지 않음.
- 합성 데이터와 실제 데이터가 trade-off 관계임. 해결하기 위해 더 나은 접근이 필요함.



Figure 12: Adding real training dataset, *e.g.*, HRWSI, to synthetic training datasets, will ruin the original fine-grained depth predictions.

Depth Anything V2



1. 합성 datasets에서만 model 학습 진행
2. Unlabeled real datasets에 합성 Datasets로만 학습한 모델을 통해 Pseudo label 생성
 - ◆ Real image에 pseudo label을 matching 시킬 때,
Scale- and shift-invariant loss와 gradient matching loss 사용
 - ◆ Top-n-largest-loss 영역 10% 있는 data를 빼고 loss 계산
 - ▶ 10%를 pseudo label의 potentially noise로 간주했기 때문.
3. Pseudo labeled real datasets 로 model 학습 진행

Depth Anything V2

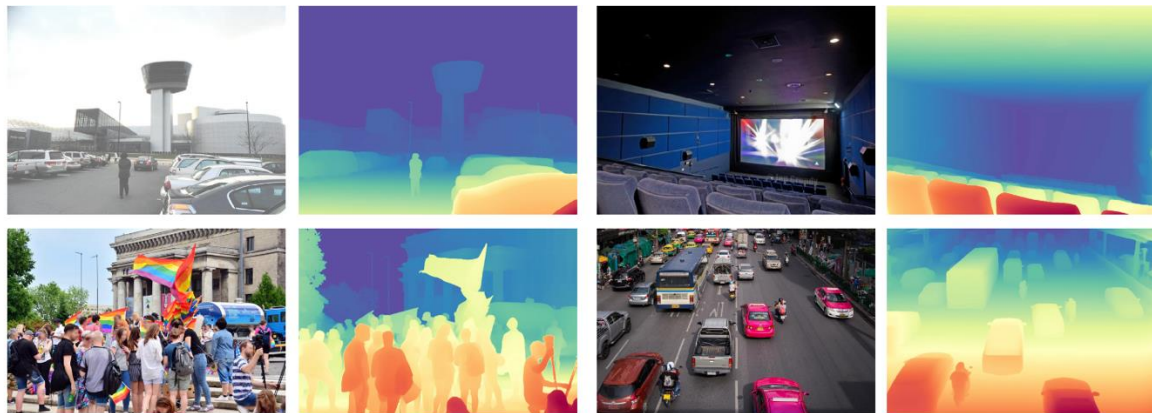


Figure 17: Visualization of our produced pseudo depth labels. From top to bottom, the highly diverse images are sampled from BDD100K [95], Google Landmarks [81], ImageNet-21K [60], LSUN [96], Objects365 [65], Open Images V7 [35], Places365 [101], and SA-1B [33] datasets, respectively.

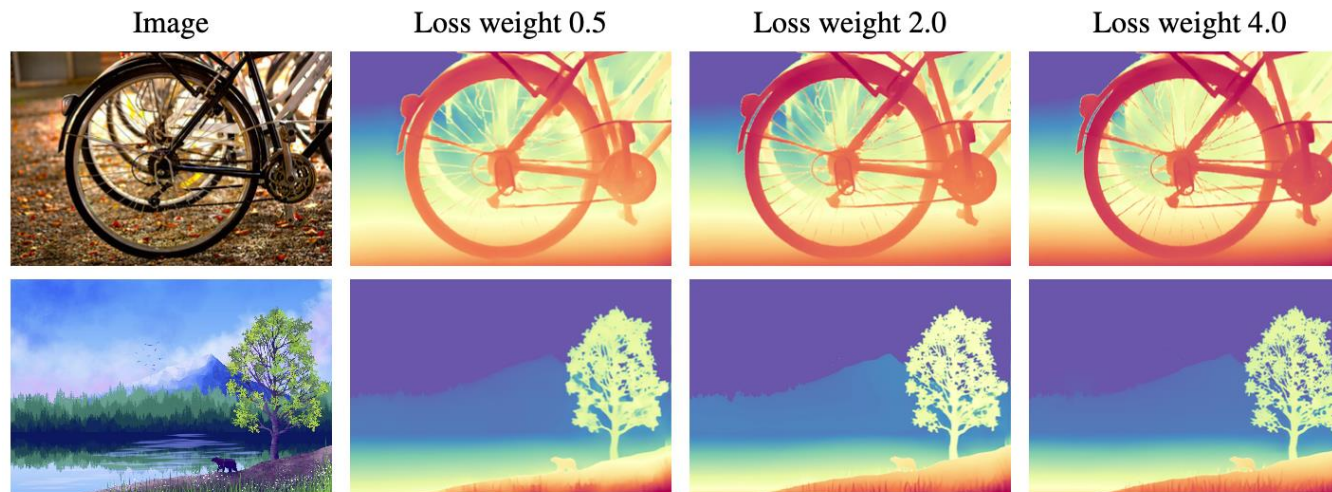


Figure 10: Effect of the gradient matching loss \mathcal{L}_{gm} in terms of fine-grained details.

Depth Anything V2

Method	Encoder	KITTI [24]		NYU-D [70]		Sintel [8]		ETH3D [62]		DIODE [76]	
		AbsRel	δ_1	AbsRel	δ_1	AbsRel	δ_1	AbsRel	δ_1	AbsRel	δ_1
MiDaS V3.1 [7]	ViT-L	0.127	0.850	0.048	0.980	0.587	0.699	0.139	0.867	0.075	0.942
Depth Anything V1 [89]	ViT-S	0.080	0.936	0.053	0.972	0.464	0.739	0.127	0.885	0.076	0.939
	ViT-B	0.080	0.939	0.046	0.979	0.432	0.756	0.126	0.884	0.069	0.946
	ViT-L	0.076	0.947	0.043	0.981	0.458	0.760	0.127	0.882	0.066	0.952
Depth Anything V2	ViT-S	0.078	0.936	0.053	0.973	0.500	0.718	0.142	0.851	0.073	0.942
	ViT-B	0.078	0.939	0.049	0.976	0.495	0.734	0.137	0.858	0.068	0.950
	ViT-L	0.074	0.946	0.045	0.979	0.487	0.752	0.131	0.865	0.066	0.952
	ViT-G	0.075	0.948	0.044	0.979	0.506	0.772	0.132	0.862	0.065	0.954

<Zero-shot relative depth estimation>

Method	Community Models				Depth Anything V2 (Ours)			
	Marigold [31]	Geowizard [20]	DepthFM [25]	Depth Anything V1 [89]	ViT-S	ViT-B	ViT-L	ViT-G
Accuracy (%)	86.8	88.1	85.8	88.5	95.3	97.0	97.1	97.4

Table 3: Performance on our proposed DA-2K evaluation benchmark, which encompasses eight representative scenarios. Even our most lightweight model is superior to all other community models.

Running an inference model on Jetson

ChungWoo-Lee / Getting-Started-with-ZED2-on-Jetson-Xavier-NX

Q Type to search

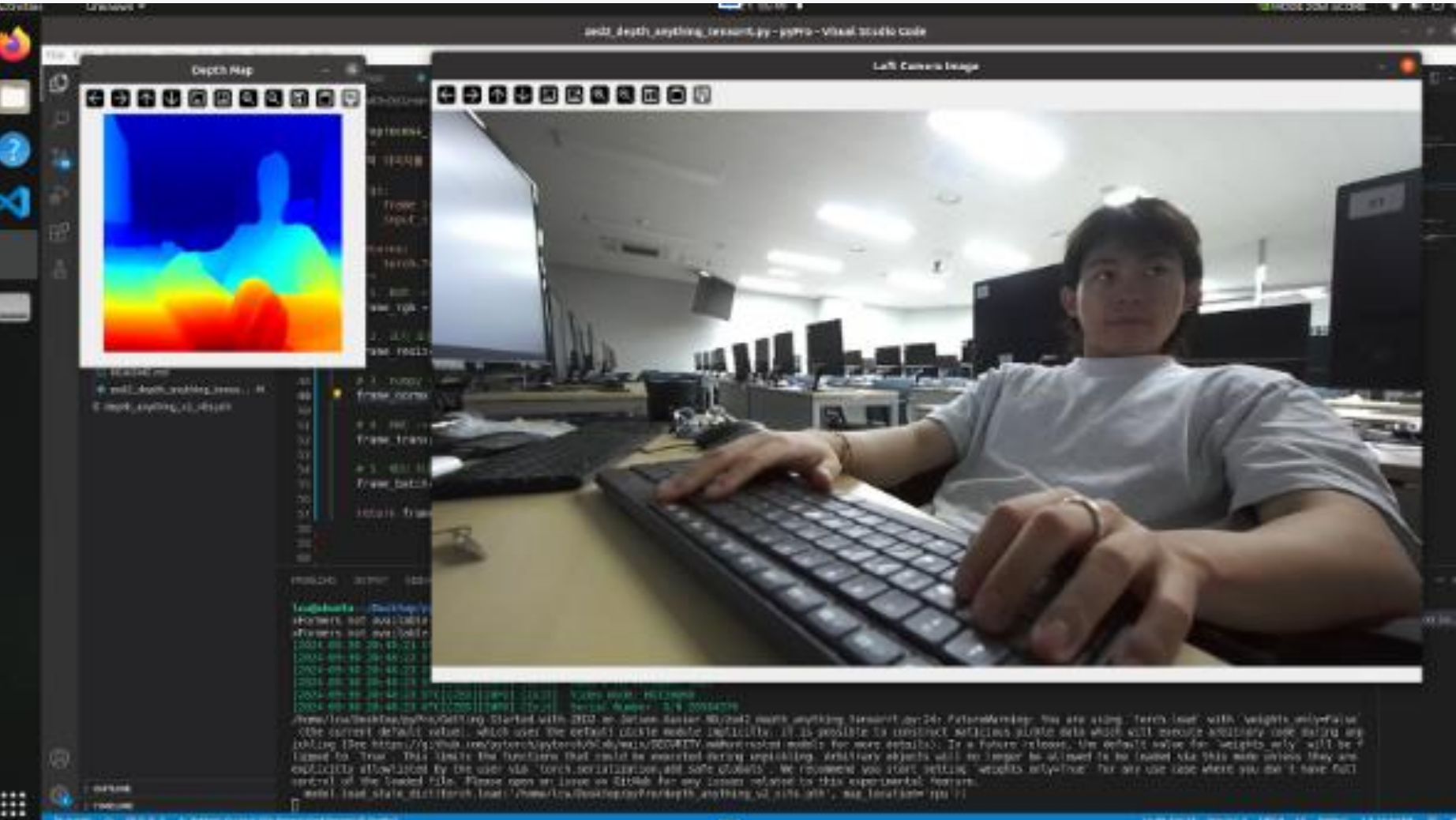
+ -

Depth Map

Left Camera Image

Python 100.0%

the steps below to configure your environment and run the code successfully.



The screenshot shows a terminal window on a Jetson Xavier NX. The terminal is running a Python script named 'zed_depth_anything_inference.py'. The script is using the ZED2 camera to capture images and generate depth maps. The 'Depth Map' window shows a person sitting at a desk, with colors representing depth (blue for far, red for near). The 'Left Camera Image' window shows the same person from the camera's perspective. The terminal window also shows a command prompt and some output text.

- **대학원 생활에서 힘들었던 점**

- ◆ 자율주행 프로젝트에 1년 동안 전념했으나, 개인 연구 주제를 잡지 못해 어려움을 겪음.
- ◆ 논문 리젝을 당했을 때 "기여 부족"이라는 한 줄 평가가 매우 충격적이었음.

- **논문 작성 시 중요한 점**

- ◆ 단순히 "내 알고리즘이 잘된다"로 접근해서는 안 됨. 성능을 넘어서 왜 좋은지를 증명해야 함.
- ◆ 기존 연구와 철저한 비교 분석을 통해 차별화된 성과를 어필해야 함.
- ◆ 새로운 접근을 시도할 때는 그 방법의 성능을 꼼꼼히 분석하고 설명해야 함.

- **협업에서 얻은 교훈**

- ◆ 명확한 인터페이스 약속을 통해 서로의 역할을 분명히 하면, 협업이 훨씬 원활해짐.
- ◆ GitHub를 잘 활용하는 것이 필수적임.
- ◆ 유능한 팀원과의 협력이 성과를 극대화할 수 있음.

- **개인 연구에서 중요한 점**

- ◆ 공부에만 몰두하지 말고, 실제로 코드를 돌려보며 문제를 해결하는 것이 중요함.
- ◆ 선배들에게 논문 작성 조언을 구하면 큰 도움이 됨.