Appendix

A. Visualization

A.1. Qualitative results

Fig. A1 and Fig. A2 showcase qualitative comparisons with Atlantis on the D3 and D5 subsets of the Sea-thru [2] dataset and the SQUID [3] dataset. All models trained on the SynTIDE dataset, including AdaBins [4], NeWCRFs [8], PixerFormer [1], and MIM [7], consistently present better visual results on underwater images compared with those trained on the Atlantis dataset. Especially in the results of the first two close-shot images in Fig. A1, the model trained on the Atlantis dataset fails to clearly show the difference in distance between the ball and the background. In contrast, our results match the ground truth closer, more distinctly displaying the contrast between the ball and the background in the image.

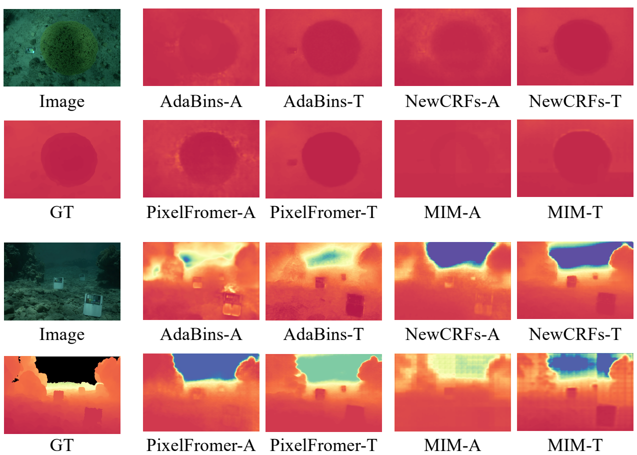

Figure A1. Qualitative results on the Sea-thru dataset [2]. '-A' and '-T' denote models trained on Atlantis [9] and Our SynTIDE dataset, respectively. The depth estimation results are notably improved after training on our dataset. Due to the original 'Image' being extremely dim, the content is hardly visible. To clearly display the content of 'Image', we adjust its contrast and brightness in this figure. These adjustments do not apply to any inference or evaluation processes at the code level.

A.2. Zero-shot underwater depth data generation

Thanks to our training strategy, which fine-tunes the pretrained text-to-image model [5] using LoRA [6] with a minor low rank, we retain its strong generalization ability to a certain extent. This enables TIDE to generate underwater depth data for scenes and objects never seen during training, as shown in Fig. A3. Even when the provided text prompts contain objects that do not exist in the real world, such as Godzilla, TIDE can still generate seemingly reasonable underwater image-depth pairs. However, this capability is particularly challenging for Atlantis [9], which requires the depth map in advance as a condition.

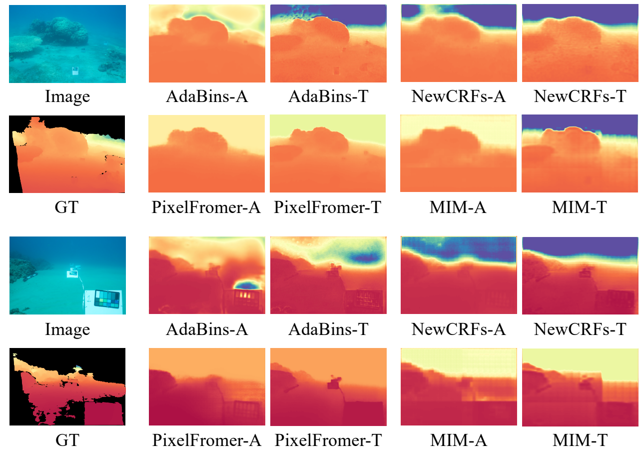

Figure A2. Qualitative results on SQUID dataset [3]. '-A' and '-T' denote models trained on Atlantis [9] and Our SynTIDE dataset, respectively. The depth estimation results are notably improved after training on our dataset.

B. Pseudo-code

To demonstrate the simplicity of TIDE and each component, we provide pseudo code with PyTorch in Listing 1, Listing 2, and Listing 3. These codes are simple and easy to implement. The complete code to reproduce the experiments will be made available before the conference.

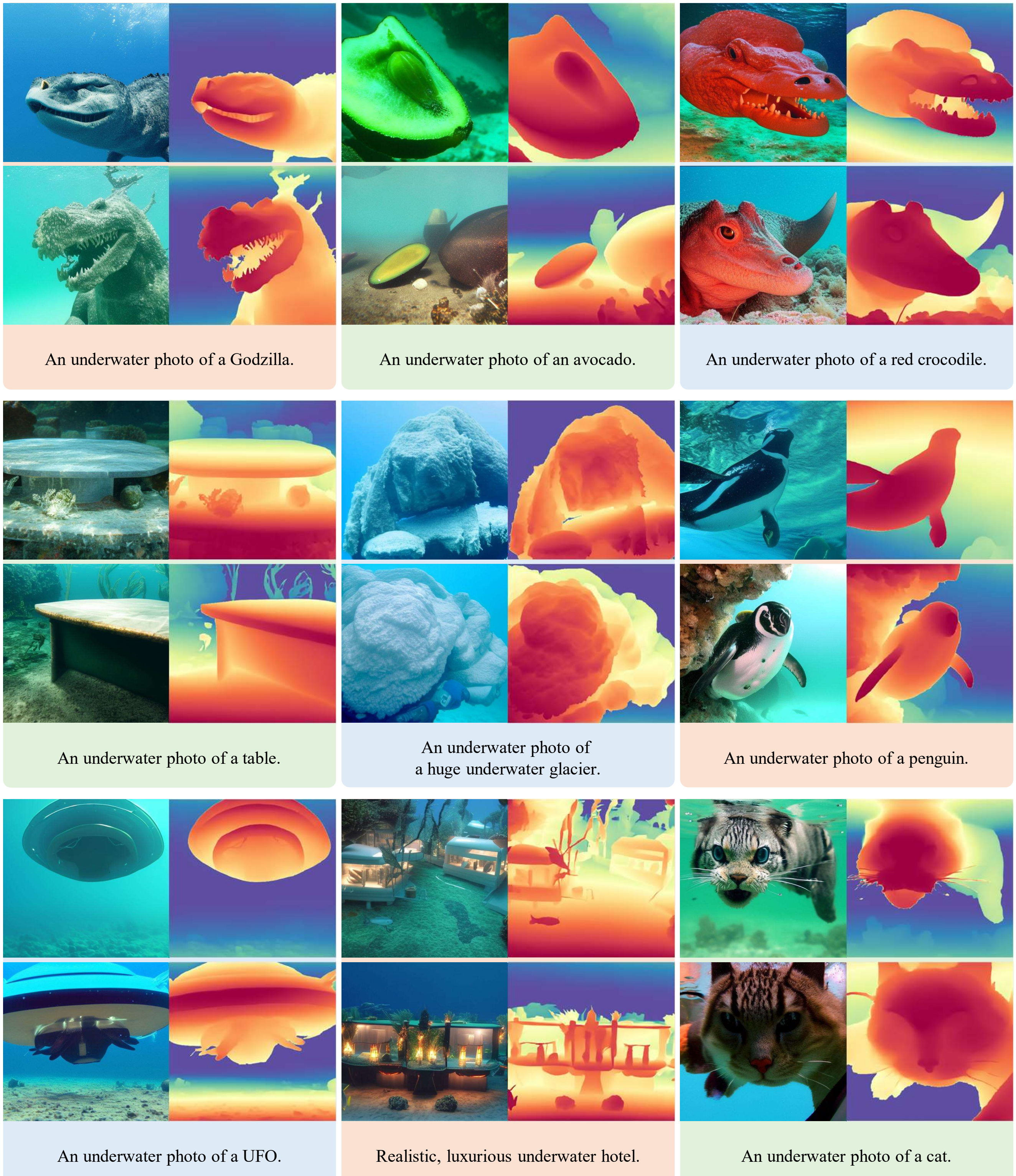

Figure A3. Representative zero-shot image-depth pairs synthesized by TIDE present strong consistency, diversity, and generalization. Images of relevant categories are not included in the training data.

```python
from torch import nn

class TAN(nn.Module):
    def __init__(self, nhidden=1152, hidden_dim=256, time_hidden_dim=6912):
        super().__init__()

        self.gamma_mlp = nn.MLP(input_dim=nhidden, hidden_dim=hidden_dim, output_dim=nhidden)
        self.beta_mlp = nn.MLP(input_dim=nhidden, hidden_dim=hidden_dim, output_dim=nhidden)
        self.time_adaptive_scale = nn.Sequential(
            nn.Linear(time_hidden_dim, 1),
            nn.Sigmoid(),
        )

    def forward(self, x, time_embed, modal1_feats, modal2_feats=None):
        if modal2_feats is not None:
            gamma1, beta1 = self._forward(modal1_feats)
            gamma2, beta2 = self._forward(modal2_feats)

            gamma = (gamma1 + gamma2) / 2
            beta = (beta1 + beta2) / 2
        else:
            gamma, beta = self._forward(modal1_feats)

        sigma = self.time_adaptive_scale(time_embed)
        out = x * (1 + sigma * gamma) + sigma * beta
        return out

    def _forward(self, modal_feats):
        gamma = self.gamma_mlp(modal_feats)
        beta = self.beta_mlp(modal_feats)
        return gamma, beta
```

Listing 1. TAN PyTorch code. When multiple modal features are input, they share the MLP weights and average the multiple sets of gamma and beta.

```python
from torch import einsum
import torch.nn.functional as F

def ILS_Attention(attn, image_feats, text_feats, cross_attn_map=None):
    query = attn.to_q(image_feats)
    key = attn.to_k(text_feats)
    value = attn.to_v(text_feats)

    if cross_attn_map is not None:
        hidden_states = einsum('b h l n, b h n c -> b h l c', cross_attn_map, value)
    else:
        hidden_states, cross_attn_map = F.scaled_dot_product_attention(query, key, value)

    return hidden_states, cross_attn_map
```

Listing 2. The ILS Attention PyTorch code. ILS_Attention is a component within the Transformer/MiniTransformer block in Listing 3.

```python
from torch import nn
from pixart import PixartTransfomer, Transformer, MiniTransformer

class TIDE(PixartTransfomer):
    def __init__(self, transfomer_layer=28, mini_transformer_layer=10):
        super().__init__()
        self.t2i_transformer = Transformer(num_layer=transfomer_layer)
        self.t2d_transformer = MiniTransformer(num_layer=mini_transformer_layer)
        self.t2m_transformer = MiniTransformer(num_layer=mini_transformer_layer)

        self.D2M_tan_blocks = nn.ModuleList(
            [
                TAN(nhidden=1152, hidden_dim=256)
                for _ in range(mini_transformer_layer)
            ]
        )
        self.M2D_tan_blocks = nn.ModuleList(
            [
                TAN(nhidden=1152, hidden_dim=256)
                for _ in range(mini_transformer_layer)
            ]
        )
        self.DM2I_tan_blocks = nn.ModuleList(
            [
                TAN(nhidden=1152, hidden_dim=256)
                for _ in range(mini_transformer_layer)
            ]
        )

        self.dense_blocks_inject_pos = [0, 3, 6, 9, 12, 15, 18, 21, 24, 27]

    def forward(self, hidden_states_image, hidden_states_depth, hidden_states_mask,
            hidden_states_text, time_embed
    ):
        for block_index, t2i_block in enumerate(self.t2i_transformer.blocks):
            hidden_states_image, implicit_layout = t2i_block(
                hidden_states_image,
                encoder_hidden_states=hidden_states_text,
                time_embed=time_embed,
            )
            if block_index in self.dense_blocks_inject_pos:
                id = self.dense_blocks_inject_pos.index(block_index)

                hidden_states_mask = self.D2M_tan_blocks[id](
                    hidden_states_mask, time_embed, hidden_states_depth
                )

                hidden_states_depth, _ = self.t2d_transformer.blocks[id](
                    hidden_states_depth,
                    encoder_hidden_states=hidden_states_text,
                    time_embed=time_embed,
                    implicit_layout=implicit_layout,
                )

                hidden_states_mask, _ = self.t2m_transformer.blocks[id](
                    hidden_states_mask,
                    encoder_hidden_states=hidden_states_text,
                    time_embed=time_embed,
                    implicit_layout=implicit_layout,
                )

                hidden_states_depth = self.M2D_tan_blocks[id](
                    hidden_states_depth, time_embed, hidden_states_mask
                )

                hidden_states_image = self.DM2I_tan_blocks[id](
                    hidden_states_image, time_embed, hidden_states_depth, hidden_states_mask
                )

        image_noise_output, depth_noise_output, mask_noise_output = self.output(
            hidden_states_image, hidden_states_depth, hidden_states_mask
        )
        return image_noise_output, depth_noise_output, mask_noise_output
```

Listing 3. TIDE PyTorch code. The entire code of TIDE will be made available before the conference.

References

[1] Ashutosh Agarwal and Chetan Arora. Attention attention everywhere: Monocular depth prediction with skip attention. In *Proc. of IEEE Winter Conf. on Applications of Computer Vision*, pages 5861–5870, 2023. 1

[2] Derya Akkaynak and Tali Treibitz. Sea-thru: A method for removing water from underwater images. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 1682–1691, 2019. 1

[3] Dana Berman, Deborah Levy, Shai Avidan, and Tali Treibitz. Underwater single image color restoration using haze-lines and a new quantitative dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(8):2822–2837, 2020. 1

[4] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021. 1

[5] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2024. 1

[6] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *Proc. of Intl. Conf. on Learning Representations*, 2022. 1

[7] Zhenda Xie, Zigang Geng, Jingcheng Hu, Zheng Zhang, Han Hu, and Yue Cao. Revealing the dark secrets of masked image modeling. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 14475–14485, 2023. 1

[8] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Neural window fully-connected crfs for monocular depth estimation. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 3916–3925, 2022. 1

[9] Fan Zhang, Shaodi You, Yu Li, and Ying Fu. Atlantis: Enabling underwater depth estimation with stable diffusion. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 11852–11861, 2024. 1