

---

# 3D Visual Illusion Depth Estimation

---

Anonymous Author(s)

Affiliation

Address

email

## 1 A 3D Visual Illusion Dataset

### 2 A.1 Video Collection

3 We collect a large amount of videos from web-source data and generative models, covering five  
4 types of illusions: inpainting illusion (e.g., inpainting on a wall/floor), picture illusion (e.g., picture  
5 printed/drawn on a paper), replay illusion (e.g., video replayed on different screens), holography  
6 illusion, and mirror illusion (e.g., specular or transparent surfaces), as shown in Figure 1. When  
7 collecting videos from generative models, we observe four important considerations in the design of  
8 text prompts. (1) Level of Detail in Prompts: Overly fine-grained control in prompts often leads to  
9 physically unrealistic results, such as requiring the object in the mirror to maintain the same pose  
10 as its real-world counterpart, enforcing perfect mirror symmetry, or specifying excessive positional  
11 details. Instead, less detailed scene descriptions tend to produce more physically accurate and realistic  
12 results. (2) Challenges with Dynamic Objects: Generating videos with dynamic objects proves  
13 particularly difficult. The virtual image in the mirror and the real-world objects often exhibit motion  
14 inconsistencies. As a result, we focus primarily on static scenes or those with only slight object  
15 movement. (3) Layout Complexity: Complex scene layouts frequently lead to mismatches between  
16 the mirror world and the real world, causing spatial inconsistencies. (4) Camera Motion: To ensure a  
17 stable and realistic scene, the camera is required to pan slowly. Excessive camera movement may  
18 result in abrupt rotations or scene transitions, disrupting the illusion.

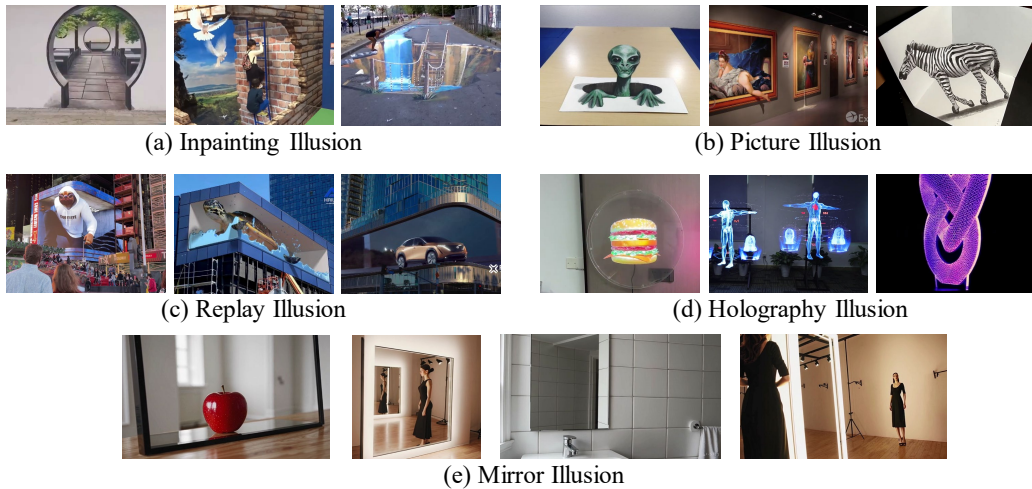


Figure 1: The visualization of 3D visual illusions.

## 19 A.2 Depth Geneation

20 To mitigate the impact of noise in plane fitting, we adopt RANSAC for robust plane estimation:

$$\min_{\alpha, \beta, \delta, \gamma} \sum_{i=1}^N (\alpha \cdot u_i + \beta \cdot v_i + \delta \cdot d_i + \gamma)^2, \quad (1)$$

$$\text{subject to } \alpha^2 + \beta^2 + \delta^2 = 1.$$

21  $(\alpha, \beta, \delta, \gamma)$  are the plane parameters,  $(u, v)$  is image plane coordinate and  $d$  is disparity. As illustrated  
 22 in Algorithm 1., we randomly sample three points to define a candidate plane at each iteration of  
 23 RANSAC. The plane normal  $(\alpha, \beta, \delta)$  is computed as the cross product of vectors formed by these  
 24 three points, and the offset  $\gamma$  is derived by substituting one point into  $\alpha \cdot u + \beta \cdot v + \delta \cdot d + \gamma = 0$ . We  
 25 then compute the distance from each point in the support region to the candidate plane to determine  
 26 the inliers. After all iterations, the candidate plane with the largest number of inliers is selected, which  
 27 are taken as the best inlier set. The plane parameters  $(\alpha, \beta, \delta, \gamma)$  are then estimated by computing the  
 28 eigenvector corresponding to the smallest eigenvalue of the covariance matrix constructed from the  
 29 best inlier set. We also present visualizations of the rendered and rectified depth images in Figure 2  
 30 and 3. After applying plane fitting for rectification, the resulting depth map becomes smoother and  
 31 more geometrically accurate.

---

### Algorithm 1 RANSAC Plane Fitting

---

**Require:** Point set  $\mathbf{P} = \{(u_i, v_i, d_i)\}_{i=1}^N \in \mathbb{R}^{N \times 3}$ , inlier threshold  $\tau_d$ , sub-sample size per iteration  
 $M$ , max iterations  $T_p$   
**Ensure:** Optimal plane parameters  $\pi^* = [\alpha, \beta, \delta, \gamma]$   
 1: Initialize:  $best\_score = 0$ ,  $best\_plane = \mathbf{0}$ ,  $best\_inliers = \emptyset$   
 2: **for**  $t = 1$  **to**  $T_p$  **do**  
 3: Randomly sample  $M$  sets of 3-point tuples:  
 $Q = \{(u_i^0, v_i^0, d_i^0), (u_i^1, v_i^1, d_i^1), (u_i^2, v_i^2, d_i^2)\}_{i=1}^M \in \mathbb{R}^{M \times 3 \times 3}$   
 4: **for**  $b = 1$  **to**  $M$  **do**  
 5:  $\mathbf{v}_{10} = (u_i^1, v_i^1, d_i^1) - (u_i^0, v_i^0, d_i^0)$ ,  $\mathbf{v}_{20} = (u_i^2, v_i^2, d_i^2) - (u_i^0, v_i^0, d_i^0)$   
 6:  $\mathbf{n}_b = \mathbf{v}_{10} \times \mathbf{v}_{20}$  {Normal via cross product}  
 7:  $d_b = -\mathbf{n}_b^\top [u_i^1, v_i^1, d_i^1]$   
 8:  $\pi_t[b] = [\mathbf{n}_b^\top, d_b]$   
 9: **end for**  
 10:  $\mathbf{D}_t = \pi_t[\mathbf{P}, \mathbf{1}]^\top / \|\pi_t[:, 0 : 3]\|_2$  {Batch distance computation}  
 11:  $\mathbf{M}_t = \|\mathbf{D}_t\| < \tau_d$   
 12:  $\mathbf{c}_t = \text{sum}(\mathbf{M}_t, \text{dim}=1)$   
 13:  $k = \arg \max \mathbf{c}_t$ ,  $c_{\max} = \mathbf{c}_t[k]$   
 14: **if**  $c_{\max} > best\_score$  **then**  
 15:  $best\_score = c_{\max}$   
 16:  $best\_plane = \pi_t[k]$   
 17:  $best\_inliers = \mathbf{M}_t[k]$   
 18: **end if**  
 19: **end for**  
 20: **Refinement via Eigen Decomposition**  
 21:  $\mathbf{P}_{inliers} = \mathbf{P}[best\_inliers]$   
 22:  $\mathbf{S} = [\mathbf{P}_{inliers}, \mathbf{1}]^\top [\mathbf{P}_{inliers}, \mathbf{1}]$   
 23:  $(\mathbf{W}, \mathbf{V}) = \text{eigh}(\mathbf{S})$   
 24:  $\pi^* = \mathbf{V}[:, 0]$   
 25: **return**  $\pi^*$

---

## 32 A.3 Right Image Geneation

33 The right-view images for generative-model videos are directly rendered using Gaussian Splatting  
 34 (GS). For web-sourced videos, right views are generated by warping the left images using monocular  
 35 disparity. As shown in Algorithm 2, we generate a right-view image  $\hat{I}_R$  from a given left-view image  
 36  $I_L$  and disparity map  $D$ . It begins by estimating an appropriate disparity scaling factor  $s$  via binary

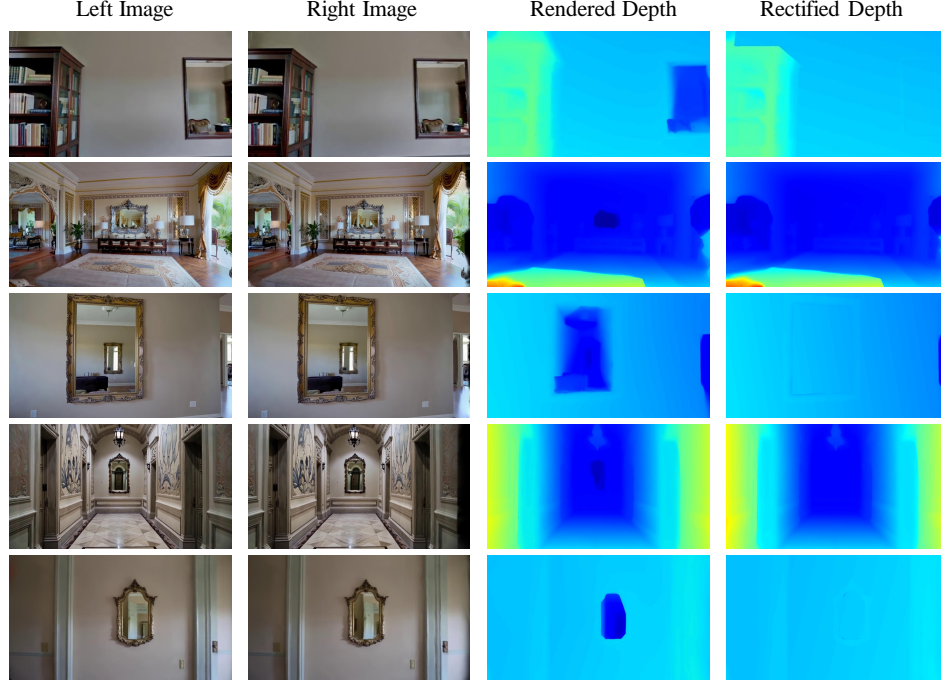


Figure 2: The visualization of results for video from generative models.

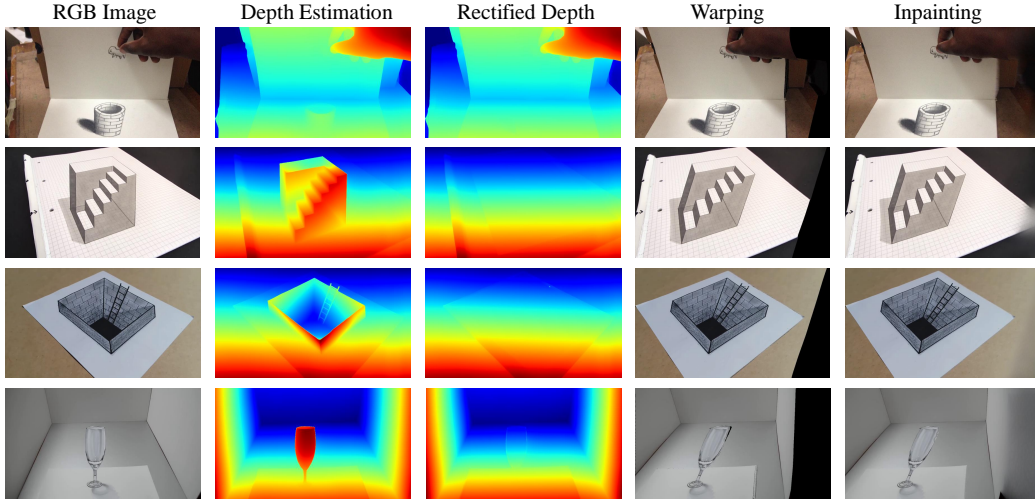


Figure 3: The visualization of results for web-source video.

37 search, ensuring that a sufficient proportion of the projected pixels fall within valid image bounds.  
 38 Using the computed  $s$ , pixel coordinates are mapped from the left to the right view, with invalid  
 39 coordinates filtered out. An initial right-view image is synthesized by transferring valid pixel values  
 40 based on the mapping. Finally, image inpainting is applied to fill missing regions, resulting in the  
 41 completed right-view image  $\hat{I}_R$ . The algorithm outputs both  $\hat{I}_R$  and the estimated scaling factor  $s$ .  
 42 We also present the visualization of the initial warped image and the inpainted image in Figure reffig:  
 43 vis web source. The inpainting process effectively fills in the missing regions, resulting in a more  
 44 complete and visually coherent right-view image.

---

**Algorithm 2** Right Image Generation

---

**Require:** Left image  $I_L \in \mathbb{R}^{H \times W \times 3}$ , disparity map  $D \in \mathbb{R}^{H \times W}$ , valid pixel threshold  $\theta = 0.9$ , maximum iterations  $T_g$

**Ensure:** Synthesized right image  $\hat{I}_R$ , scaling factor  $s$

```
1: Step 1: Compute Scaling Factor
2: Initialize:  $l = 0, r = W/(4 \cdot \max(D)), \epsilon = 10^{-6}, t = 0$ 
3: while  $|r - l| > \epsilon$  and  $t < T_g$  do
4:    $t = t + 1$ 
5:    $s = (l + r)/2$ 
6:   Coordinate projection:  $U' = U - s \cdot D$ 
7:   Compute valid pixel ratio:  $\eta = \frac{1}{HW} \sum \mathbb{I}(U' \in [0, W))$ 
8:   if  $\eta \geq \theta$  then
9:      $l = s$ 
10:  else
11:     $r = s$ 
12:  end if
13: end while
14: Final scaling factor:  $s = (l + r)/2$ 
15: Step 2: Image Coordinate Mapping
16: Generate coordinate grid:  $(u, v) = \text{MESHGRID}(0 : W - 1, 0 : H - 1)$ 
17: Compute projected coordinates:  $u' = u - s \cdot D(u, v)$ 
18: Quantize coordinates:  $\hat{u}' = \{\lfloor u' \rfloor, \lceil u' \rceil\}$ 
19: Filter invalid coordinates:  $\{\hat{u}' \mid \hat{u}' \geq 0 \text{ and } \hat{u}' < W\}$ 
20: Step 3: Right View Image Synthesis
21: Initialize:  $I_R = \mathbf{0}^{H \times W \times 3}$ 
22: for each pixel  $(u, v)$  do
23:   Generate initial right-view image  $I_R$ :  $I_R(u', v) = I_L(u^*, v), u^* = \arg \max_u \{d_{(u, v)} \mid u - s \cdot D_{(u, v)} = u'\}$ .
24: end for
25: Step 4: Image Completion Perform inpainting on  $I_R$  to fill invalid regions and obtain the final right-view image  $\hat{I}_R$ 
26: return  $\hat{I}_R, s$ 
```

---

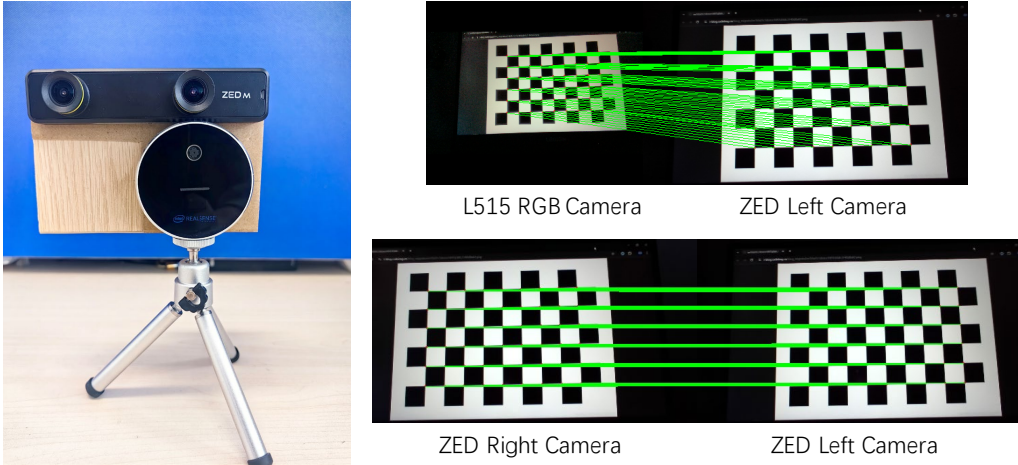


Figure 4: Camera System and Calibration Visualization

## 45 A.4 Real-world Data

### 46 A.4.1 Camera System

47 We collect real-world data using a stereo camera (ZED Mini) and a LiDAR-based depth sensor  
 48 (Realsense L515). The sensors are rigidly mounted and calibrated using a checkerboard to ensure  
 49 accurate alignment, as shown in Figure 4. The ZED Mini captures RGB images, while the L515  
 50 provides depth maps. The intrinsic and extrinsic parameters of both cameras are obtained through  
 51 calibration. The calibration process involves capturing multiple images of the checkerboard pattern  
 52 from different angles and distances, allowing for accurate estimation of the camera parameters.

### 53 A.4.2 Depth Map Projection

54 The L515 depth map is warped to the ZED left camera to construct the ground-truth depth. As  
 55 shown in Algorithm 3, the process begins by upsampling the depth map and scaling the intrinsic  
 56 matrix accordingly. 3D points are then computed and transformed from the L515 frame to the ZED  
 57 frame using calibrated extrinsics, followed by projection onto the ZED image plane. The resulting  
 58 depth values are splatted to the ZED image grid, and missing regions are filled using inpainting and  
 59 guided filtering. To ensure consistency, a backward reprojection step verifies each pixel’s validity by  
 60 comparing it with the original L515 depth. Finally, noise is suppressed using median filtering, and  
 61 valid depth values are converted into disparities based on the ZED stereo baseline and focal length.

## 62 B VLM-Driven Monocular-Stereo Fusion Model

63 Inspired by the strong reasoning ability of vision-language models (VLMs), as shown in Figure 5.  
 64 We propose a VLM-driven monocular-stereo fusion model that leverages commonsense knowledge  
 65 to assess the reliability of monocular and stereo depth in different regions, enabling effective depth  
 66 fusion.

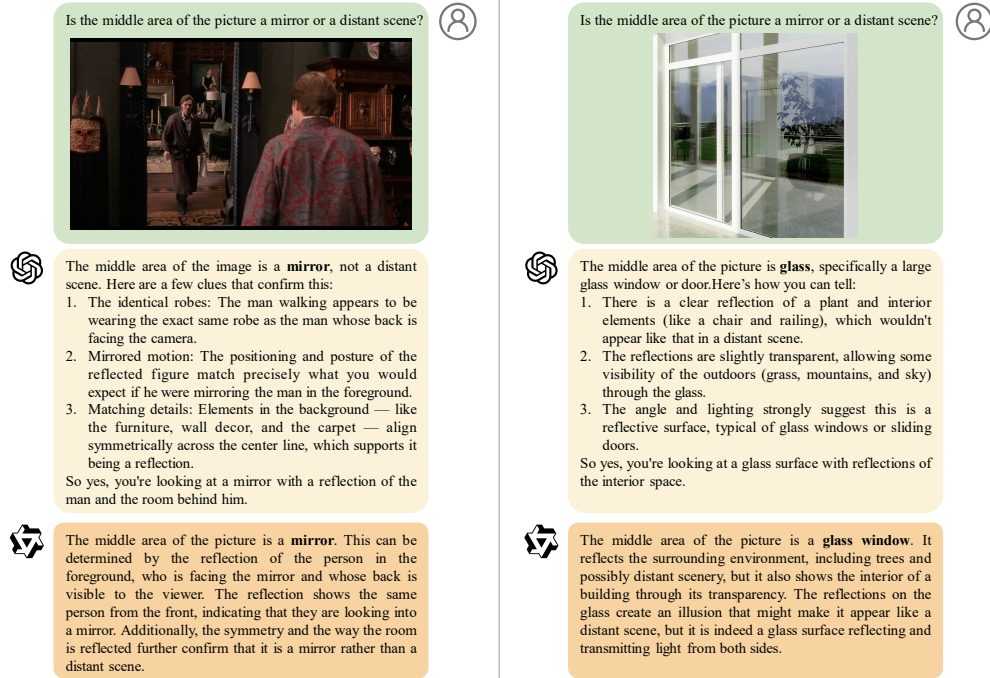


Figure 5: Visualization of commonsense knowledge from VLMs. The left image shows a scene with a mirror. The right image depicts a scene with a transparent object, where the background is visible through the object.

---

**Algorithm 3** Depth Map Reprojection

---

**Require:** Depth map from L515 camera  $\mathcal{Z}_L \in \mathbb{R}^{H_L \times W_L}$ , RGB image from ZED left camera  $I_Z \in \mathbb{R}^{H' \times W' \times 3}$ , intrinsic matrix of L515  $K_L \in \mathbb{R}^{3 \times 3}$ , rotation matrix  $R \in \mathbb{R}^{3 \times 3}$  and translation matrix  $T \in \mathbb{R}^{3 \times 1}$  from L515 to ZED-left, upsampling factor  $s = 3$ , ZED stereo baseline  $B$ , and ZED focal length  $F$

**Ensure:** Disparity map of ZED left camera  $D \in \mathbb{R}^{H' \times W'}$

```
1: Step 1: Depth Upsampling
2:  $\tilde{\mathcal{Z}}_L = \text{resize}(\mathcal{Z}_L, \text{scale} = s, \text{interp} = \text{NEAREST})$ 
3:  $\tilde{K}_L = s \cdot K_L$ 
4: Step 2: Coordinate Transformation
5: for each pixel  $(u_L, v_L)$  in  $\tilde{\mathcal{Z}}_L$  do
6:    $[x_Z, y_Z, z_Z] = R \cdot \tilde{\mathcal{Z}}_L \cdot \tilde{K}_L^{-1} \cdot [u_L, v_L, 1]^T + T$ 
7:    $[u_Z, v_Z, 1] = K_Z \cdot [x_Z/z_Z, y_Z/z_Z, 1]$ 
8: end for
9: Step 3: Depth Projection
10: Initialize  $\mathcal{Z}_Z = \infty^{H' \times W'}$ 
11: for each projected point  $(u_Z^i, v_Z^i, z_Z^i)$  do
12:    $(u_1, v_1) = (\lfloor u_Z^i \rfloor, \lfloor v_Z^i \rfloor), (u_2, v_2) = (\lceil u_Z^i \rceil, \lceil v_Z^i \rceil)$ 
13:   for  $(u, v) \in \{(u_1, v_1), (u_1, v_2), (u_2, v_1), (u_2, v_2)\}$  do
14:     if  $(u, v)$  is within image bounds then
15:        $\mathcal{Z}_Z(u, v) = \min(\mathcal{Z}_Z(u, v), z_Z^i)$ 
16:     end if
17:   end for
18: end for
19: Step 4: Hole Filling
20:  $\mathcal{M}_{\text{invalid}} = (\mathcal{Z}_Z == \infty), \mathcal{Z}_Z = \mathcal{Z}_Z \odot \neg \mathcal{M}_{\text{invalid}} + \mathbf{0} \odot \mathcal{M}_{\text{invalid}}$ 
21:  $\mathcal{M}_{\text{small}} = \text{connectedComponents}(\mathcal{M}_{\text{invalid}}, \text{area\_th} = 100)$ 
22:  $\mathcal{Z}_{\text{repair\_small}} = \text{inpaint}(\mathcal{Z}_Z, \mathcal{M}_{\text{small}}), \mathcal{Z}_{\text{repair\_all}} = \text{inpaint}(\mathcal{Z}_Z, \mathcal{M}_{\text{invalid}})$ 
23:  $\mathcal{Z}_{\text{repair\_all}} = \text{guidedFilter}(I_Z, \mathcal{Z}_{\text{repair\_all}}, \text{radius} = 5, \epsilon = 1e - 3)$ 
24:  $\mathcal{Z}_{\text{repair}} = \mathcal{Z}_Z \odot \neg \mathcal{M}_{\text{invalid}} + \mathcal{Z}_{\text{repair\_all}} \odot \neg (\mathcal{Z}_{\text{repair\_small}} == 0) \odot \mathcal{M}_{\text{invalid}}$ 
25: Step 5: Backward Reprojection for Invalid Region Detection
26: for each pixel  $(u_Z, v_Z)$  in  $\mathcal{Z}_{\text{repair}}$  do
27:    $[x_{Z \rightarrow L}, y_{Z \rightarrow L}, z_{Z \rightarrow L}] = R^{-1} \cdot (z_Z \cdot K_Z^{-1} \cdot [u_Z, v_Z, 1] - T)$ 
28:    $[u_{Z \rightarrow L}, v_{Z \rightarrow L}, 1] = \frac{1}{z_{Z \rightarrow L}} \cdot K_L [x_{Z \rightarrow L}, y_{Z \rightarrow L}, z_{Z \rightarrow L}]$ 
29:   if  $\mathcal{Z}_L(v_{Z \rightarrow L}, u_{Z \rightarrow L}) == 0$  or  $\|\mathcal{Z}_L(v_{Z \rightarrow L}, u_{Z \rightarrow L}) - z_{Z \rightarrow L}\| > \tau$  then
30:      $\mathcal{Z}_{\text{repair}}(u_Z, v_Z) = 0$ 
31:   end if
32: end for
33: Step 6: Noise Suppression
34:  $\mathcal{Z}_{\text{smooth}} = \text{medianFilter}(\mathcal{Z}_{\text{repair}}, \text{size} = 3)$ 
35:  $\mathcal{M}_{\text{noise}} = |\mathcal{Z}_{\text{repair}} - \mathcal{Z}_{\text{smooth}}| > 0.03$ 
36:  $\mathcal{Z}_{\text{final}} = \mathcal{Z}_{\text{repair}} \odot \neg \mathcal{M}_{\text{noise}} + \mathbf{0} \odot \mathcal{M}_{\text{noise}}$ 
37: Step 7: Disparity Computation
38:  $D = B \cdot F / \mathcal{Z}_{\text{final}}$ 
39: return  $D$ 
```

---

## 67 B.1 Loss

68 The supervised loss function consists of two main components: one ( $\mathcal{L}_d$ ) for the disparity maps and  
69 the other ( $\mathcal{L}_c$ ) for the confidence map:

$$\mathcal{L} = \mathcal{L}_d + w\mathcal{L}_c, \quad (2)$$

70 where  $w$  is a manually set weighting factor for balancing the confidence map loss.

71 For disparity supervision, we use the  $L_1$  loss to supervise each iteratively updated disparity  $D_s^t$ , the  
72 aligned monocular disparity  $\tilde{D}_m$ , and the final predicted disparity  $D_f$ . The loss function is defined  
73 as:

$$\begin{aligned} \mathcal{L}_d = & \sum_{t=1}^T \gamma_d^{T+2-t} \|D_s^t - D_G\|_1 \\ & + \gamma_d \|\tilde{D}_m - D_G\|_1 + \|D_f - D_G\|_1. \end{aligned} \quad (3)$$

74 Here,  $D_G$  denotes the ground-truth disparity, and  $\gamma_d$  is a weighting coefficient to balance contributions  
75 from intermediate predictions.

76 For confidence map supervision, we adopt the Focal Loss, where the ground-truth for confidence is  
77 derived based on the disparity difference between the final stereo prediction  $D_s^T$  and the ground-truth  
78  $D_G$ :

$$\begin{aligned} \mathcal{L}_c = & \frac{1}{N} \sum_i \alpha_c \cdot \left(1 - e^{-\mathcal{L}_b(i)}\right)^{\gamma_c} \cdot \mathcal{L}_b(i), \\ \mathcal{L}_b = & -\bar{I}_c \log I_c - (1 - \bar{I}_c) \log(1 - I_c), \\ \bar{I}_c = & \mathbb{I}(\text{Interpolate}(|D_G - D_s^T|, \text{scale} = \frac{1}{4}) < \frac{5}{4}), \end{aligned} \quad (4)$$

79 In this formulation,  $\alpha_c$  and  $\gamma_c$  are the hyperparameters of the Focal Loss,  $\mathbb{I}$  is the indicator function,  
80 and Interpolate denotes the downsampling operation that resizes the supervision signal to  $\frac{1}{4}$  of the  
81 original resolution, matching the resolution used in intermediate network outputs.

## 82 C Experiments

### 83 C.1 Evaluation Metric

84 We evaluate model performance in both disparity space and depth space. In disparity space, two  
85 commonly used metrics are adopted. (1) End-Point Error (EPE):  $EPE = \frac{1}{N} \sum_i |r_i - \bar{r}_i|$ , where  
86  $r$  and  $\bar{r}$  denote the predicted and ground-truth disparity values, respectively. EPE measures the  
87 average absolute disparity error in pixels. (2) Bad- $x$  Error:  $\text{bad-}x = \frac{1}{N} \sum_i \mathbb{I}(|r_i - \bar{r}_i| > x)$ ,  
88 which indicates the percentage of pixels where the disparity error exceeds  $x$  pixels. This metric is  
89 especially useful for evaluating the robustness of the model under boundary conditions. In depth space,  
90 four standard evaluation metrics are employed. (1) Absolute Relative Error (AbsRel):  $AbsRel =$   
91  $\frac{1}{N} \sum_i \frac{|r_i - \bar{r}_i|}{\bar{r}_i}$ , which evaluates the relative difference between predictions and ground-truth values,  
92 normalized to mitigate the impact of scale and unit differences, making it suitable for datasets  
93 with diverse depth ranges. (2) Root Mean Squared Error (RMS):  $RMS = \sqrt{\frac{1}{N} \sum_i (r_i - \bar{r}_i)^2}$ .  
94 (3) Log10 Error:  $\log10 = \frac{1}{N} \sum_i |\log_{10}(r_i) - \log_{10}(\bar{r}_i)|$ . (4) Threshold Accuracy ( $\delta_1$ ):  $\delta_1 =$   
95  $\frac{1}{N} \sum_i \mathbb{I}\left(\max\left(\frac{y_i}{\bar{y}_i}, \frac{\bar{y}_i}{y_i}\right) < 1.25\right)$ , which measures the proportion of pixels for which the predicted  
96 depth falls within a certain ratio (e.g., 1.25) of the ground truth. The model is initially trained on the  
97 SceneFlow dataset, then fine-tuned on the 3D-Visual-Illusion training set, and finally evaluated on  
98 the 3D-Illusion test set and the Booster training set.

### 99 C.2 Implementation Details

100 For dataset construction, we use Qwen2-VL-72B [1, 14] to perform initial screening, reducing the  
101 dataset from 5226 videos with 52M frames to 4,519 videos with 1.4M frames. We then built a  
102 Flask[13]-based web tool to manually reduce data to 1384 videos with 236k frames. We further  
103 developed a more convenient flask-based web app for SAM2 [12] to acquire the semantic mask of



104 illusion and support regions. During the semantic segmentation, we delete frames with redundant  
 105 content and illusions imperceptible to humans, reducing data from 236k frames to 176,530 frames.  
 106 Later, we rectify the depth values of illusion regions from the reference of support regions, which is  
 107 further used to generate the right images for web-source data. Besides web-source data, We also use  
 108 large generative models to generate 234 videos with 2382 frames. The video generation is achieved  
 109 via Sora [10] and Kling [6], and a small part of the data is generated from HunyuanVideo [5]. We  
 110 then use InstantSplat [2], DUS3R [15], and GS [4] to generate the right images and depth map,  
 111 followed by similar depth post-processing.

112 As for our VLM-driven monocular-stereo fusion network, we benefit from the vision and language  
 113 foundation model and use Depthanything V2 [16, 17] as a pre-trained monocular model, QwenVL2-7B  
 114 [14, 1] as pre-trained VLM, and FLUX [7] as a diffusion model. We use Lora [3] to fine-tune the last  
 115 layer of QwenVL2-7B and the Q\$V projection layer of FLUX on  $4 \times H100$  with a batch size of 6 on  
 116 each GPU. The entire training takes almost 20 days.

### 117 C.3 Prompts

118 In dataset construction, we design a prompt to prefilter bad frames using Qwen2-VL-72B [1, 14] due  
 119 to the large amount of videos collected from the Internet. The prompts are as follows:

Reply to me in the format of a string concatenating ‘yes’ or ‘no’ with ‘;’. Each ‘yes’ or ‘no’  
 is an answer to each following question. Does this image feature any flat artistic creation of  
 landscapes where the surface of the creation is flat and has no ups and downs? Does this image  
 contain any areas with perspective illusions? Does this image contain any optical-illusion  
 graffiti or artwork? Does this image contain any transparent or high-reflective areas? Does  
 this image show a display screen playing 3D objects or scenes? Does the image contain areas  
 that make you mistake them for 3D objects? Does this image contain excessive watermarks  
 or captions that seriously affect its quality? Does this image contain small watermarks or  
 captions in the corners? Is this image too blurry? Are most regions of the artistic creation  
 covered by a single/two hands? Is this image a software interface? Is only the figure of the  
 artist clear, but the others are blurry, like artwork, screen, or areas that make you mistake them  
 for 3D objects?

120  
 121 We use the answer from Qwen2-VL-72B to filter out bad frames. We reduce the data from 5,226  
 122 videos and 52 million frames to 4,519 videos and 1.4 million frames.

123 In addition to web-sourced data, we also use videos produced by generative models, resulting in 234  
 124 videos comprising a total of 2,382 frames. The primary generative models used are Sora and Kling,  
 125 with a small portion of the data sourced from HunyuanVideo [5]. The initial prompts were generated  
 126 using ChatGPT, with the prompt used for generation as follows:

Please provide 100 unique and detailed bilingual (Chinese and English) prompts, each with  
 an index number, for generating text-to-video scenes that include mirror reflections. The  
 prompts must meet the following requirements: 1. Specify the mirror type and describe the  
 entities in the scene, the overall layout, and their spatial relationship to the mirror. 2. Include a  
 diverse range of mirror types: dressing mirrors, vanity mirrors, full-length mirrors, bathroom  
 mirrors, car rearview mirrors, polished stainless steel, etc. 3. Ensure varied scene distributions:  
 residential settings, commercial spaces, and public areas. 4. The combination of mirror type  
 and scene context must be reasonable (e.g., polished stainless steel is appropriate in a kitchen  
 but not in a study). 5. Entity configuration: some scenes should include people in front of the  
 mirror (e.g., a woman combing her hair or a customer trying on clothes), some should feature  
 objects (e.g., plants, cosmetics, books), and others should show only the mirror reflecting  
 surfaces like walls. 6. Each prompt must describe the physical correspondence between the  
 real object and its reflection. 7. Avoid overly complex layouts in individual scenes. 8. Ensure  
 a balance of richly textured and minimally textured elements within the same scene. 9. All  
 objects in the scene must remain static, with only slow camera panning; descriptions implying  
 motion (e.g., “a moving car”) are inappropriate. 10. Descriptions should be as precise and  
 detailed as possible.

127



128 The generated prompts were subsequently refined to avoid producing low-quality video outputs, as  
129 pointed in Section A.1. Below are some examples of the prompts:

130 Generate a video showing a cozy, modern living room. A single minimalist-designed mirror is  
mounted on the wall, with clearly defined edges and realistic reflections. The scene combines  
intricate furniture textures with a monochromatic background, and the camera pans slowly.

131 Generate a video set in a creative art space. A uniquely shaped mirror hangs on the wall,  
featuring accurate reflections and distinct boundaries. The scene includes complex graffiti  
textures and smooth surfaces, with slow camera panning.

132 A static and art-deco inspired living room with a framed mirror above a tufted velvet sofa,  
reflecting physical laws accurately, geometric patterns, sleek metal finishes, and glamorous  
lighting. Realistic, glamorous lighting, retro.

133 A static and rustic farmhouse dining area with a reclaimed wood-framed mirror on a weathered  
brick wall, highlighting a crisp realistic reflection, a sturdy wooden table, vintage chairs, and  
warm pendant lighting. Realistic, warm lighting, rustic.

134 Our VLM-driven monocular-stereo fusion framework employs Depthanything V2 [16, 17] as the  
135 pre-trained monocular network, QwenVL2-7B [14, 1] as the pre-trained visual-language network, and  
136 FLUX [8, 11] as the flow matching network. The language prompt for the pre-trained visual-language  
137 network is:

138 Are there any transparent or reflective objects? Like mirror, glass, window, showcase, and  
so on? If true, reply to me with the list of corner coordinates of each object in the format of  
(x1,y1,x2,y2,x3,y3,x4,y4) in the image. If false, reply with an empty list of corners.

139 The language prompt for the pre-trained flow matching network is:

140 Using the provided features extracted by QwenVL2, generate a binary segmentation mask  
for the image. Highlight all transparent or reflective objects (e.g., mirrors, glass, windows,  
showcases) in white (255), while marking all other regions in black (0).

#### 141 C.4 Visualization

142 We present more visualization on the 3D-Visual-Illusion dataset and Booster dataset in Figure 6,  
143 7, and 8. The results demonstrate that our method can effectively handle various types of visual  
144 illusions. The depth maps generated by our model exhibit high fidelity and accuracy, even in  
145 challenging scenarios with complex visual illusions. The depth maps from VGGT and Dust3R mirror  
146 the significance of fusing monocular priors and multi-view matching.

147 We also present the visualization of 3D detection on the real data of the 3D-Visual-Illusion dataset  
148 in Figure 9. We obtain the results from YOLO3D [9], and the results show that 3D visual illusions  
149 can seriously affect the performance of 3D detection. We believe that the 3D visual illusion will  
150 become more and more important as the vision foundation models become more and more powerful,  
151 especially when used in downstream applications, like 3D detection, occupancy and planning.

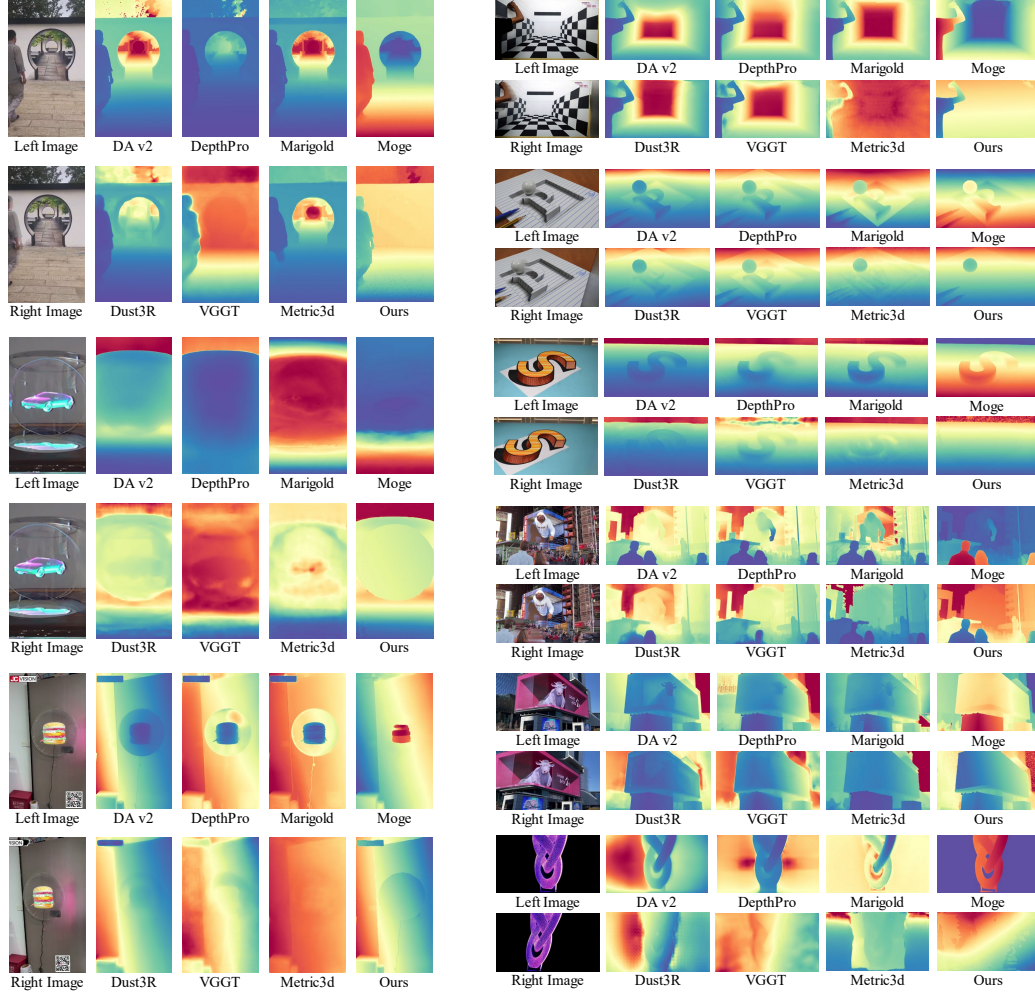


Figure 6: The visualization of results on virtual data of the 3D-Visual-Illusion dataset.

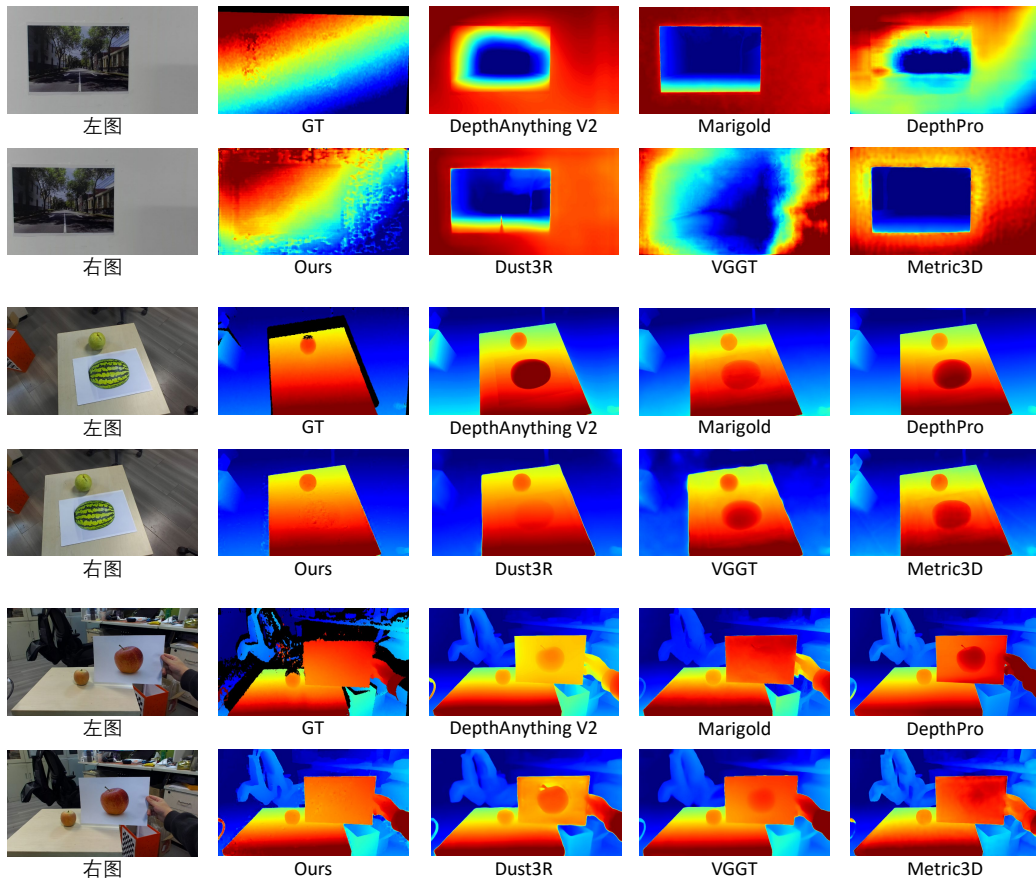


Figure 7: The visualization of results on real data of the 3D-Visual-Illusion dataset.



Figure 8: The visualization of results on the Booster dataset.

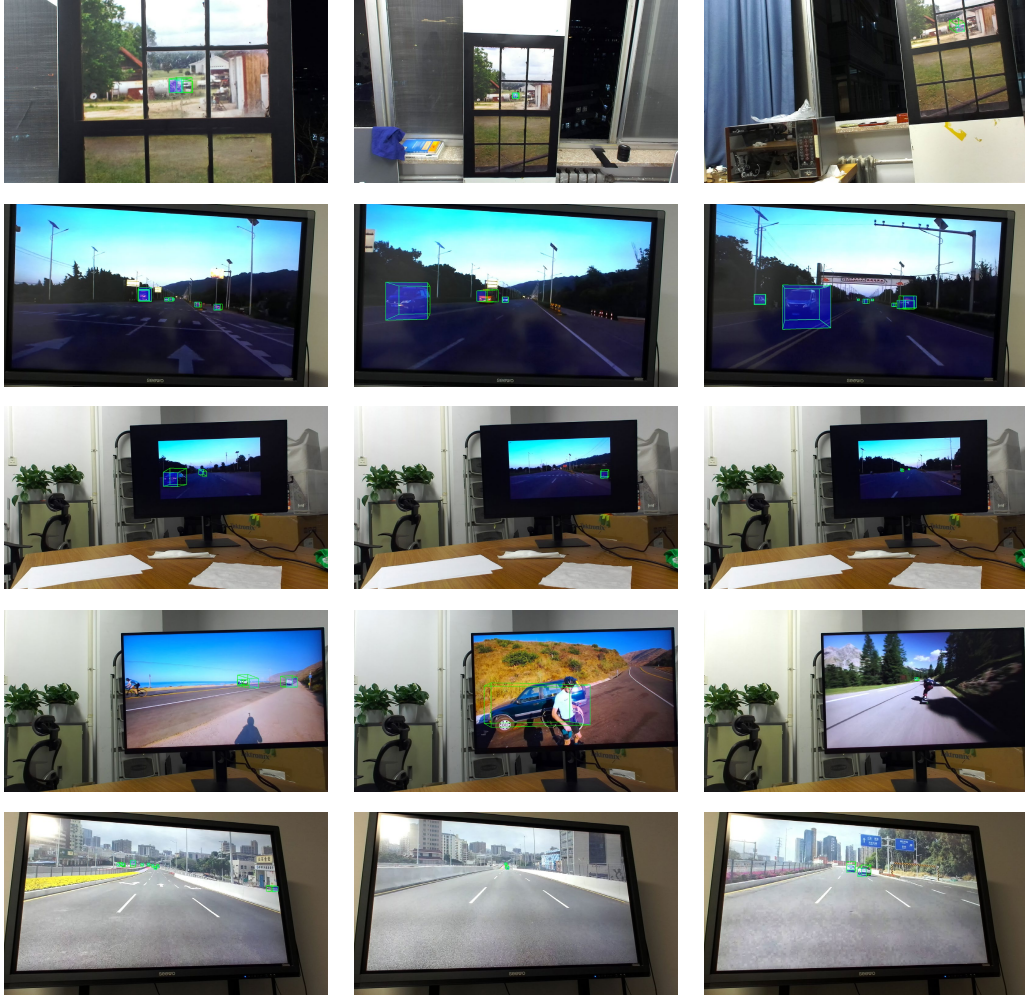


Figure 9: The visualization of 3D detection on real data of 3D-Visual-Illusion dataset.

## References

- [1] Bai Jinze, Bai Shuai, Yang Shusheng, Wang Shijie, Tan Sinan, Wang Peng, Lin Junyang, Zhou Chang, Zhou Jingren. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond // arXiv preprint arXiv:2308.12966. 2023.
- [2] Fan Zhiwen, Cong Wenyan, Wen Kairun, Wang Kevin, Zhang Jian, Ding Xinghao, Xu Danfei, Ivanovic Boris, Pavone Marco, Pavlakos Georgios, others . Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds // arXiv preprint arXiv:2403.20309. 2024. 2. 4.
- [3] Hu Edward J, Wallis Phillip, Allen-Zhu Zeyuan, Li Yuanzhi, Wang Shean, Wang Lu, Chen Weizhu, others . LoRA: Low-Rank Adaptation of Large Language Models // Proceedings of the International Conference on Learning Representations (ICLR). 2022.
- [4] Kerbl Bernhard, Kopanas Georgios, Leimkühler Thomas, Drettakis George. 3d gaussian splatting for real-time radiance field rendering. // ACM Trans. Graph. 2023. 42. 139–1.
- [5] Kong Weijie, Tian Qi, Zhang Zijian, Min Rox, Dai Zuozhuo, Zhou Jin, Xiong Jiangfeng, Li Xin, Wu Bo, Zhang Jianwei, others . Hunyuanvideo: A systematic framework for large video generative models // arXiv preprint arXiv:2412.03603. 2024.
- [6] Kuaishou . Kling: AI Video Generation Tool. 2024. Accessed: 2025-03-08.
- [7] Labs Black Forest. FLUX. 2024.
- [8] Lipman Yaron, Chen Ricky TQ, Ben-Hamu Heli, Nickel Maximilian, Le Matt. Flow Matching for Generative Modeling // Proceedings of the International Conference on Learning Representations (ICLR). 2023.
- [9] 3D Bounding Box Estimation Using Deep Learning and Geometry. // . 2017.
- [10] OpenAI . Sora: AI Video Generation Model. 2024. Accessed: 2025-03-08.
- [11] Peebles William, Xie Saining. Scalable diffusion models with transformers // Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2023. 4195–4205.
- [12] Ravi Nikhila, Gabeur Valentin, Hu Yuan-Ting, Hu Ronghang, Ryali Chaitanya, Ma Tengyu, Khedr Haitham, Rädle Roman, Rolland Chloe, Gustafson Laura, Mintun Eric, Pan Junting, Alwala Kalyan Vasudev, Carion Nicolas, Wu Chao-Yuan, Girshick Ross, Dollár Piotr, Feichtenhofer Christoph. SAM 2: Segment Anything in Images and Videos // arXiv preprint arXiv:2408.00714. 2024.
- [13] Team Pallets. Flask: A lightweight WSGI web application framework. 2025. Accessed: 2025-03-08.
- [14] Wang Peng, Bai Shuai, Tan Sinan, Wang Shijie, Fan Zhihao, Bai Jinze, Chen Keqin, Liu Xuejing, Wang Jialin, Ge Wenbin, Fan Yang, Dang Kai, Du Mengfei, Ren Xuancheng, Men Rui, Liu Dayiheng, Zhou Chang, Zhou Jingren, Lin Junyang. Qwen2-VL: Enhancing Vision-Language Model’s Perception of the World at Any Resolution // arXiv preprint arXiv:2409.12191. 2024.
- [15] Wang Shuzhe, Leroy Vincent, Cabon Yohann, Chidlovskii Boris, Revaud Jerome. Dust3r: Geometric 3d vision made easy // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2024. 20697–20709.
- [16] Yang Lihe, Kang Bingyi, Huang Zilong, Xu Xiaogang, Feng Jiashi, Zhao Hengshuang. Depth anything: Unleashing the power of large-scale unlabeled data // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2024. 10371–10381.
- [17] Yang Lihe, Kang Bingyi, Huang Zilong, Zhao Zhen, Xu Xiaogang, Feng Jiashi, Zhao Hengshuang. Depth Anything V2 // arXiv preprint arXiv:2406.09414. 2024.