Free-Form Motion Control: Controlling the 6D Poses of Camera and Objects in Video Generation

Xincheng Shuai[1] Henghui Ding[1]✉ Zhenyuan Qin[1] Hao Luo[2,3] Xingjun Ma[1] Dacheng Tao[4]
[1]Fudan University, China [2]DAMO Academy, Alibaba group [3]Hupan Lab
[4]Nanyang Technological University, Singapore
henghui.ding@gmail.com dacheng.tao@gmail.com
https://henghuiding.com/SynFMC/

A. Preliminary

• **Diffusion Loss in Text-to-Video Generation.** During training, T2V diffusion models [7, 10, 11, 16, 17, 20, 26, 28] first add noise ϵ to image sequence $\mathbf{z}_0^{1:N}$ to get current latents $\mathbf{z}_t^{1:N}$ in time t.

$$\mathbf{z}_t^{1:N} = \sqrt{\bar{\alpha}_t}\mathbf{z}_0^{1:N} + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i, \qquad \text{(i)}$$

where α_t is the hyperparameter specified by variance scheduler. Then, the estimation network ε_θ is trained to infer the injected noise from current latents and conditions. The goal is to minimize the following item:

$$\arg\min_\theta E_{\mathbf{z}_0^{1:N},t,\epsilon,\mathbf{C}_p}\left[\left\|\varepsilon_\theta\left(\mathbf{z}_t^{1:N}, t, \mathbf{C}_p\right) - \epsilon\right\|^2\right]. \qquad \text{(ii)}$$

• **Plücker Embedding.** Directly feeding the explicit camera information into network [23] is simple but not effective. To this end, CameraCtrl [8] uses plücker embedding to represent camera pose for geometric interpretation. For each pixel (u, v) in the image space, the plücker embedding is calculated as $(\mathbf{o}_c \times \mathbf{d}_{u,v}, \mathbf{d}_{u,v})$, where $\mathbf{o}_c$ is camera location in the global coordinate system. The direction vector $\mathbf{d}_{u,v}$ is computed by $\mathbf{R}\mathbf{K}^{-1}[u, v, 1] + \mathbf{o}_c$, where $\mathbf{R}$ and $\mathbf{K}$ are rotation and intrinsic matrices respectively. The plücker embedding of the frame composes of the embeddings in each location.

B. *SynFMC* Dataset

B.1. Asset Collection and Annotation

The panoramic HDRI maps serve as environments during rendering. We source 88 *ground/near ground*, 25 *sky*, 27 *underwater* and 6 *water surface* HDRIs from Poly-Haven [14], BlenderKit [1] and other websites. In addition, 1-4 texts are provided for each background, facilitating to complete the total descriptions of videos. For object assets, we source the objects with available animations from public datasets [3, 4] and other websites [13], which cover a diverse range of categories. Then, we query InternVL [2] about some basic properties of the object such as class name, environment, speed and size types. Among them, the environment type indicates the object's matching environment assets. Table I represents the distribution of object assets across environments. The definition of speed and size types, and the asset percentage of each category are shown in Table II. These information are useful in trajectory generation and rendering stages for realistic simulation. The comprehensive query texts of MLLM are presented in Figure XIX-Figure XXI. Finally, the human annotators will verify or correct these information. Besides, they require to annotate the motion type and description of each object animation. Specifically, we assign the action type to each animation according to the environment type and visual effect, which corresponds to several motion types as shown in Table III. For example, the in-place animations like dancing are assigned to "Idle". The whole process of asset collection and annotation is presented in Figure I. Figure III presents several examples from each environment, demonstrating various object and background in *SynFMC*. Figure II also indicates the category diversity

Table I. Object asset distribution across environments.

Environment Type	Object Asset Percentage(%)
Ground	35.33
Near Ground	19.09
Water Surface	12.94
Sky	17.17
Underwater	15.47

✉ Henghui Ding (henghui.ding@gmail.com) is the corresponding author with the Institute of Big Data, College of Computer Science and Artificial Intelligence, Fudan University, Shanghai, China.

Table II. Definition of speed and size types and asset percentage of each category.

Speed Type	Speed (m/s)	Object Asset Percentage(%)	Size Type	Size (cm)	Object Asset Percentage(%)
Stationary	0	20.32	Tiny	0.1-5	3.92
Slow	0.01-1.5	6.21	Small	5-30	8.38
Moderate	1.5-10	37.33	Medium	30-150	32.04
Fast	10-60	28.11	Large	150-300	37.66
Very Fast	60-340	5.78	Extra-Large	300-500	16.12
Supersonic	>340	2.25	Gigantic	>500	1.88

Table III. Distribution of object action types.

Action Type	Available Environments	Motion Type	Percentage(%)
Idle	All Environments	Stationary Point	34.83
Move	Ground&Water Surface	Horizontal Line / Curve	26.99
Fly	Near Ground&Sky	(Non-)Horizontal Line / Curve	21.17
Swim	Underwater	(Non-)Horizontal Line / Curve	17.01

of 3D assets in *SynFMC*.

B.2. Data Generation

• **Motion Simulation.** In object motion simulation, we initialize the object's position based on it's corresponding environment. For the *ground* and *water surface*, vertical coordinates are set to 0. Then, we use Bézier curve to simulate the object's trajectory for each motion segment. Specifically, the control points and endpoint positions are randomized, while ensuring that the curve length is less than the product of the object's speed and time. Afterward, we construct the object's rotation matrix using the curve's tangent and normal vectors. For camera simulation, we obtain the camera's location based on the object's position and the offset vector. The direction and magnitude of the vector on the horizontal plane are determined by the camera's *viewpoint type* and *distance type* respectively. For example, for the front viewpoint, the direction corresponds to the projection of the object's velocity onto the horizontal plane. For the magnitude, we determine the current value based on *distance type* and the value in the previous step. The *height type* determines the magnitude of the offset vector along the vertical axis. Finally, the camera orientation is adjusted to focus on a random offset from the object's centroid. Figure III illustrates various object and camera motion patterns, which facilitate the model learning complicated dynamics from videos. Figure IV shows examples from different scenes. Notably, the *multi-object* examples (in the 3rd and 4th rows) simulate the complex, irregular characteristics in the real world, where objects accidentally enter or disappear from the field of view.

• **Video Annotations.** To support diverse research tasks, *SynFMC* provides auxiliary annotations, including instance segmentation maps [5, 6], depth maps, and descriptions of both visual content and motion. An example is shown in Figure V. Notably, we provide an algorithm to automatically generate motion descriptions. For content description, the appearing objects are tracked in each frame to identify which objects enter or exit the view. The following template is used: "[*background description*], [*objects description in the first frame*]. [*entering objects description*], [*exiting objects description*]..." For example: "*With forest backdrop, a man and a woman are walking. Then a cat enters the view, while the man disappears from the image.*" For motion description, key frames are extracted at equal intervals, and we calculate the camera's position change between consecutive frames to describe its motion, *e.g.*, "*The camera moves upward and forward.*"

C. *Free-Form Motion Control*

C.1. Network Architecture of Motion Controllers

For OMC (Object Motion Controller), we use ControlNet-like architecture [27] to receive spatial-form conditions of object pose. CMC (Camera Motion Controller) consists of two parts: *Camera Encoder* and *Camera Adapter*, where the Camera Adapter is introduced into the temporal modules. We adopt the similar architecture from previous works [8].

C.2. Discussion about Interaction of Control Signals

The object and camera motions are controlled independently in our *FMC*, since their control signals are defined in camera space and world space, respectively. Specifically, the object control map identifies the object pose (location, size, and orientation) from camera view. An example in Figure VI (self-attention maps in 2nd row are visualized via PCA) shows the object and camera moving in opposite directions. Camera motion (to left in world space) shifts scene viewpoint, *e.g.*, the background rock moves right, while the moving object's camera-view pose is independently governed by object control signal (to lower-right and disappear).

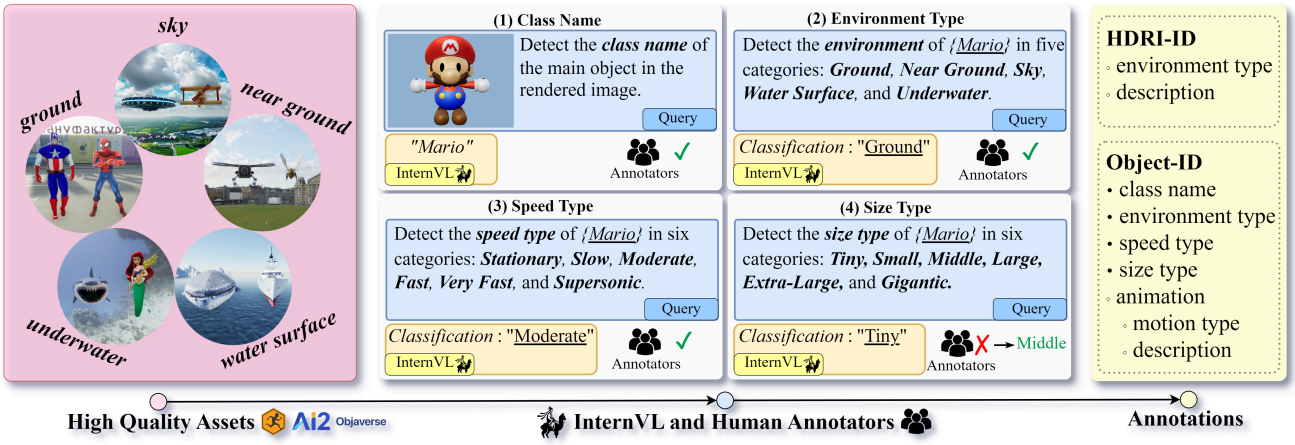
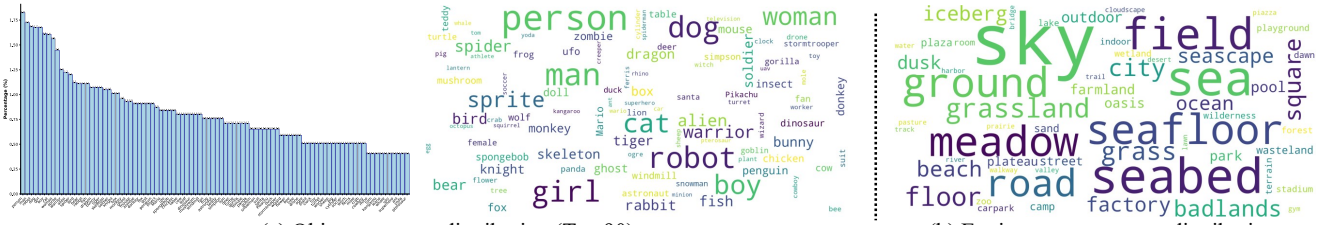


Figure I. Process of asset collection and annotation. The fields marked with hollow circles are fully annotated by human annotators.



(a) Object category distribution (Top 90)

(b) Environment category distribution

Figure II. Statistics of 3D assets in *SynFMC*. (a) and (b) indicate the category distributions of objects and environments respectively.

C.3. Construction of Training Data

We randomly sample frames from the original video at uniform intervals and obtain the corresponding object masks. Since *SynFMC* provides detailed background, object, and action descriptions at each time step, the text descriptions of videos can be automatically constructed using specific templates. The text description, object masks and camera&object pose annotations are then used to train motion controllers.

C.4. Construction of Validation Data

To construct validation samples for qualitative and quantitative experiments, we use different strategies for independent camera/object control and simultaneous control.

• **Independent Camera Control.** GPT-4 is queried to generate 1000 descriptions for scenes with and without dynamic object respectively, accessing the ability of models in different scenarios. For motion conditions, our rule-based algorithm is employed to generate the camera trajectory while omitting the object movements.

• **Independent Object Control.** In this setting, we utilize descriptions of scenes with dynamic object as mentioned above. For motion conditions, we only consider the object trajectories and the camera is fixed in the space.

• **Simultaneous Control.** We query 500 descriptions for each scene: *static single-object*, *static multi-object*, *dynamic single-object* and *dynamic multi-object*. Then the algorithm generates the trajectories based on the scene types.

C.5. More Details of User Study

We employed 42 volunteers to evaluate outputs from each method via a platform shown in Figure VII. To ensure fairness, the method used for each video was hidden from users. Scores were averaged and normalized by the maximum value. We assess following metrics: 1) Quality was evaluated by deviation from the base model's quality. 2) Text similarity was accessed based on coverage of all semantic elements in the prompt. 3) Camera/Object motion fidelity was measured by comparing generated videos with reference videos rendered in Unreal Engine using the same trajectories.

C.6. More Results in Independent Control of Camera

The results of independent camera motion control are presented in Figure VIII. As shown in first five rows, *FMC* is capable of performing basic translational and rotational movements with high accuracy. For more complex camera motions that involve both translation and rotation, *FMC* can still reliably generate the desired movements, maintaining high fidelity to motion condition. Furthermore, as shown

Figure III. Examples of synthetic videos in *SynFMC* from different environments.

in rows 5, 7, 8, when applied to scenes with dynamic objects, *FMC* demonstrates its ability to generate realistic and coherent object motion.

C.7. More Results in Independent Control of Object

Figure IX presents several examples of independent object motion control, spanning both basic (the 1st and 2nd examples) and complicated (the 3rd example) scenarios.

Figure IV. Examples of synthetic videos in *SynFMC* from different scenes.

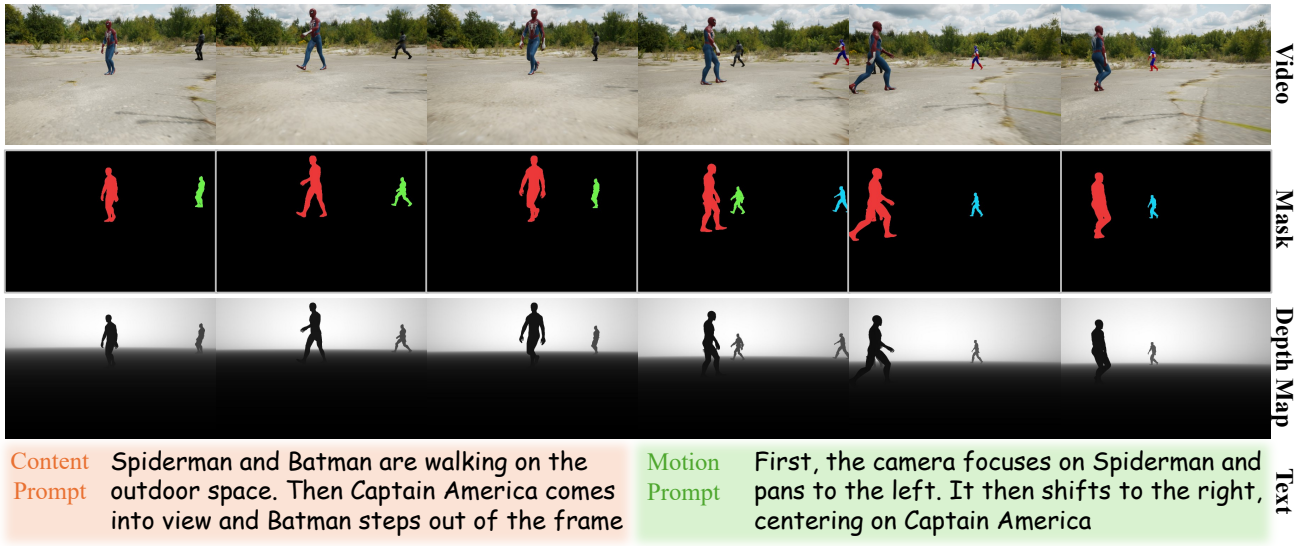


Figure V. An example of auxiliary annotations in *SynFMC*.

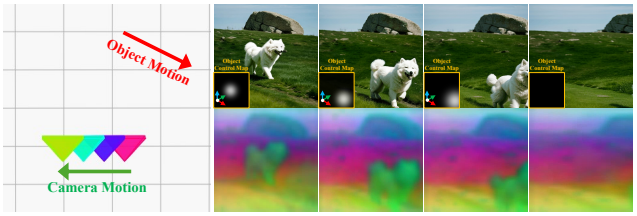


Figure VI. Discussion about the interaction of control signals.

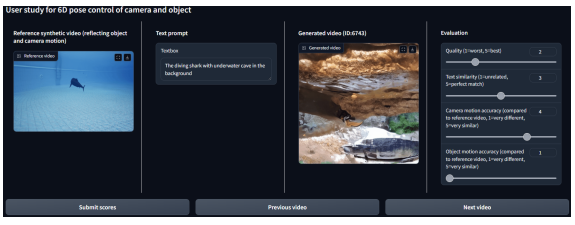


Figure VII. The platform for user study.

The results from MotionCtrl [23] and Direct-a-Video [24] show that the background content in the video often exhibits

random dynamics (rows 1, 2, 7, 8). Furthermore, the object presents low-quality appearance (row 7) or incorrect orientation (rows 5, 7, 8) in several cases. This observation suggests that the image space methods [12, 15, 21, 25] couple the movements of both the objects and the camera, while lacking the orientation information of the object. In contrast, *FMC* addresses this issue by introducing a fixed camera pose sequence as input, thereby constraining the camera to remain stationary effectively. Besides, the 6D object pose also facilitates *FMC* generating high quality appearance and reasonable orientation.

C.8. More Results in Simultaneous Control

For simultaneously controlling motions of camera and objects, Figure X and Figure XI show the results of the proposed *FMC* and MotionCtrl [23] in *single-object* scenes. MotionCtrl [23] often fails to produce correct results due to its inability to access both the camera and object motion annotations during training, where the object frequently disappears from the field of view in cases with complicated camera motion (1st row in Figure X and most cases in Figure XI). In contrast, *FMC* achieves superior performance in both *static* and *dynamic* scenarios. This is primarily due to the comprehensive 6D pose information in *SynFMC*. Additionally, OMC's enhanced perception of object orientation and size, as well as its efficient training objective, ensure robust and high quality outcomes across various settings. Figure XII presents the results in *multi-object* scenes. The figure highlights the capability of *FMC* to control the intricate movements of multiple objects and camera within the same video.

C.9. More Results in Ablation Study

• **More Results in *SynFMC* Dataset.** To thoroughly assess the effectiveness and generalization capabilities of our dataset, we train the MotionCtrl [23] with *SynFMC*. In this experiment, we adopt the same training strategy and configuration outlined in our *FMC*. As shown in the 1st and 3rd rows of Figure XIII, without the camera poses $\mathcal{C}_{RT}$, it becomes difficult to maintain a consistent and stable appearance of the object across frames, leading to suboptimal results. This issue becomes even more pronounced when more complex camera movements are involved. On the other hand, as demonstrated in the 2nd and 4th rows, the integration of camera condition significantly mitigates these problems.

• **More Results in Object Motion Control Modules.** Figure XIV presents a comparative analysis of the results produced by MotionCtrl [23] trained on *SynFMC*, and our method. As shown in the figure, *FMC* accurately captures the changes in object orientation and size within the image, due to the OMC in *FMC* can handle with the 6D pose and coarse mask of the object. In contrast, MotionCtrl can only handle with the trajectory input confined to the image space, which restricts its ability to generate plausible object appearances across varying viewpoints.

• **More Results in Training Objectives.** To illustrate the effectiveness of the training objective at each stage, we use the standard diffusion loss [9, 22] to train CMC and OMC. The impact of L_{cam} defined in Eq.1 is demonstrated in Figure XV. The 1st row reveals that the model struggles to learn complex camera movements without L_{cam} under the same training configuration. Moreover, the 3rd row of Figure XV shows that the model instead moves the object to satisfy the relative movement indicated by the motion conditions (the petals of the lotus are unnaturally stretched toward the camera). Figure XVI underscores the importance of L_{obj} in Eq.2. As seen in the 1st and 3rd rows, the model fails to generate coherent object appearances with the standard diffusion loss. By introducing L_{obj}, the OMC is able to concentrate on the object region, resulting in higher training efficiency and video quality.

• **More Results in Personalized T2I Models.** Figure XVII demonstrates the generation results of *FMC* based on different T2I personalized models [18, 19]. The generated results not only accurately reflect the motion conditions but also retain the distinctive stylistic features inherent to the used backbones. This validates both the effectiveness and the generalizability of our *SynFMC* and corresponding training strategy.

D. Failure Cases

Although *FMC* presents significant advancements, it still exhibits room for improvement in scenarios involving multi-object motion control. As shown in the 1st row of Figure XVIII, *FMC* fails to completely reflect semantics contained in the description (the bird is omitted in the video). This limitation becomes even more apparent when the number of objects or the complexity of movements increases, as shown in the 2nd row. We will solve the issue in our future works.

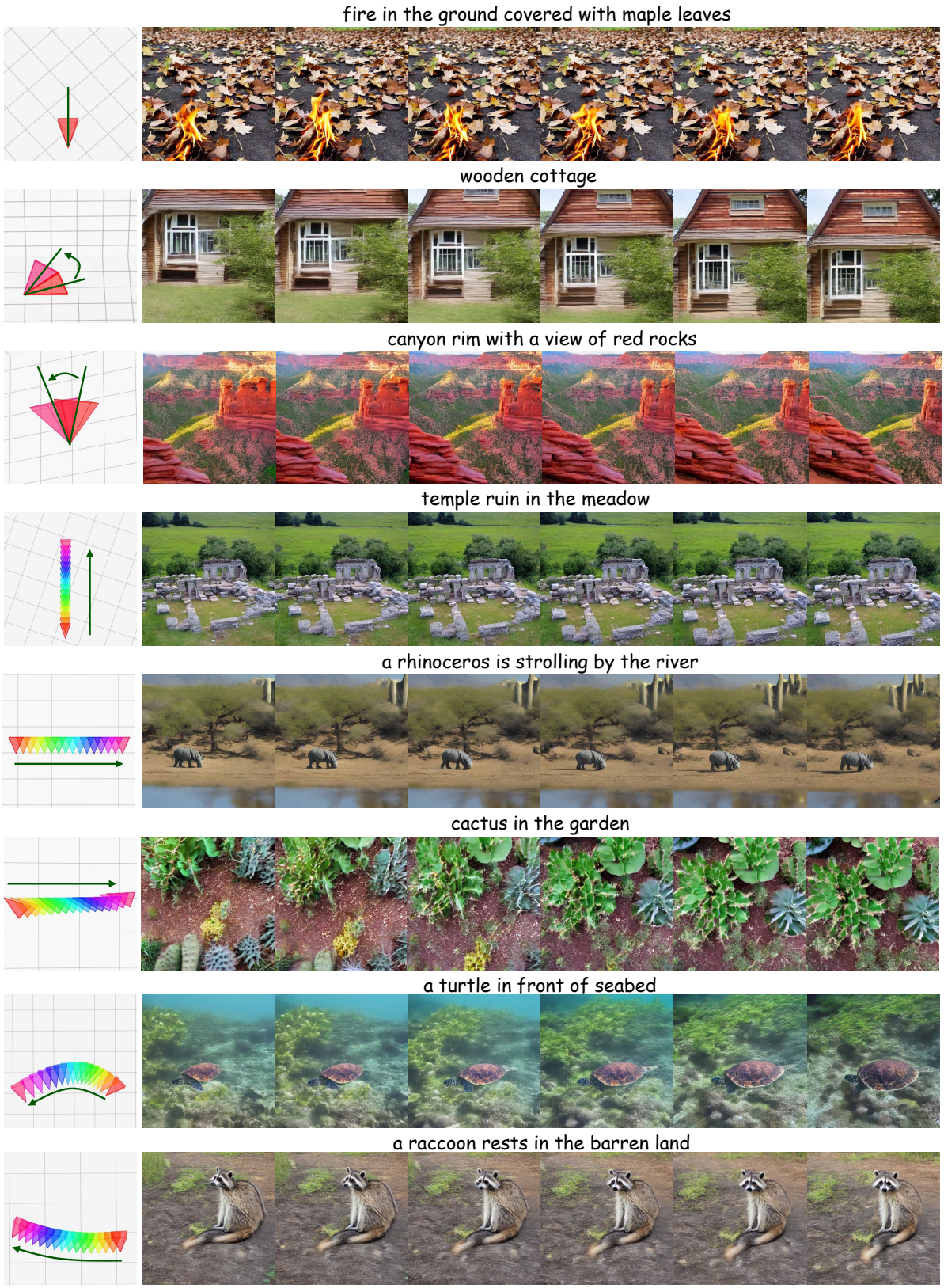


Figure VIII. Independent control over camera motion. The first five rows illustrates basic translational or rotational movements. Other rows present more complex camera motions that involve both translation and rotation. *FMC* can generate high fidelity results in each case.

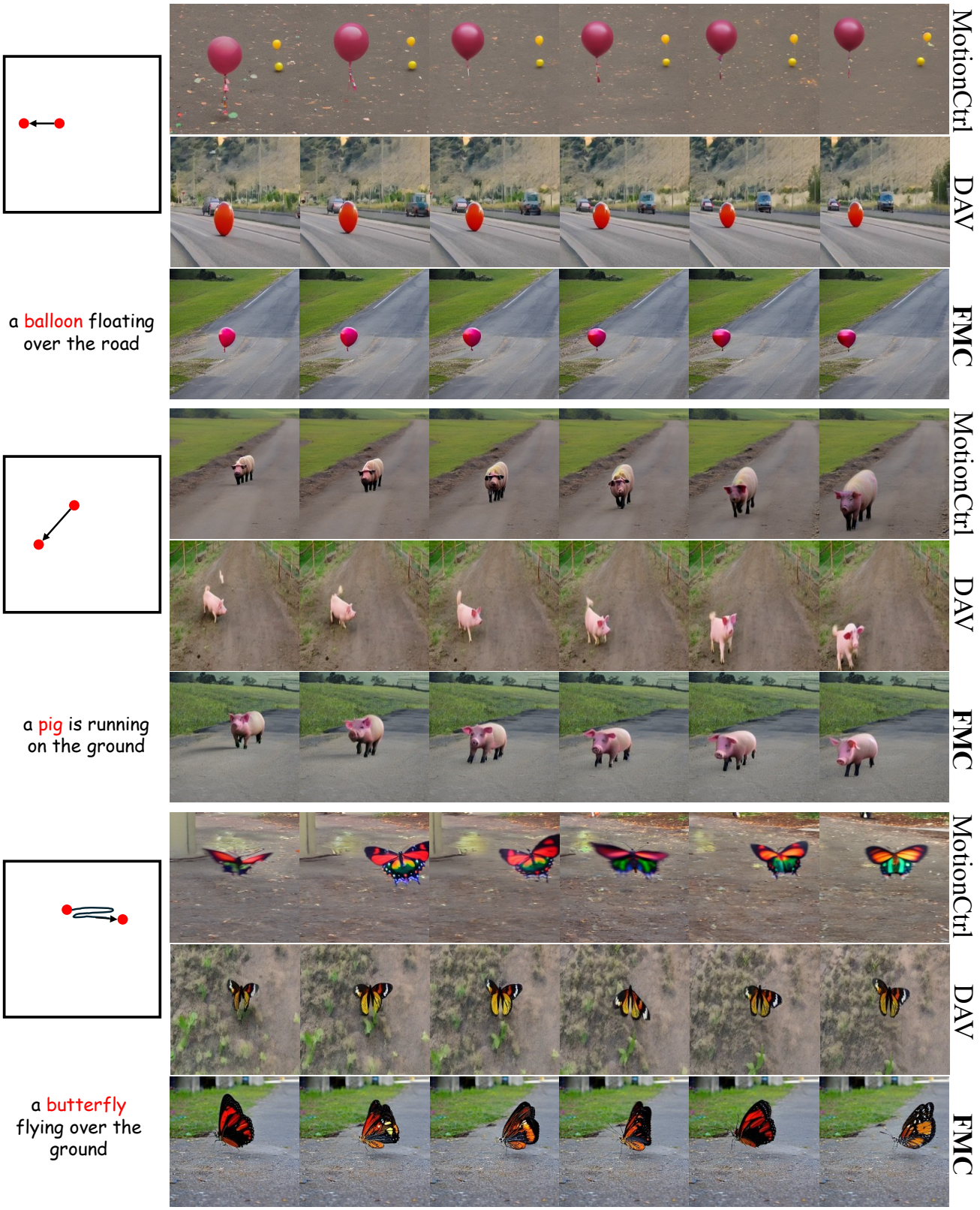


Figure IX. Independent control over object motion. The results from MotionCtrl [23] and Direct-a-Video [24] show that the background content in the video often exhibits random dynamics. Furthermore, the object presents low-quality appearance or incorrect orientation in several cases.

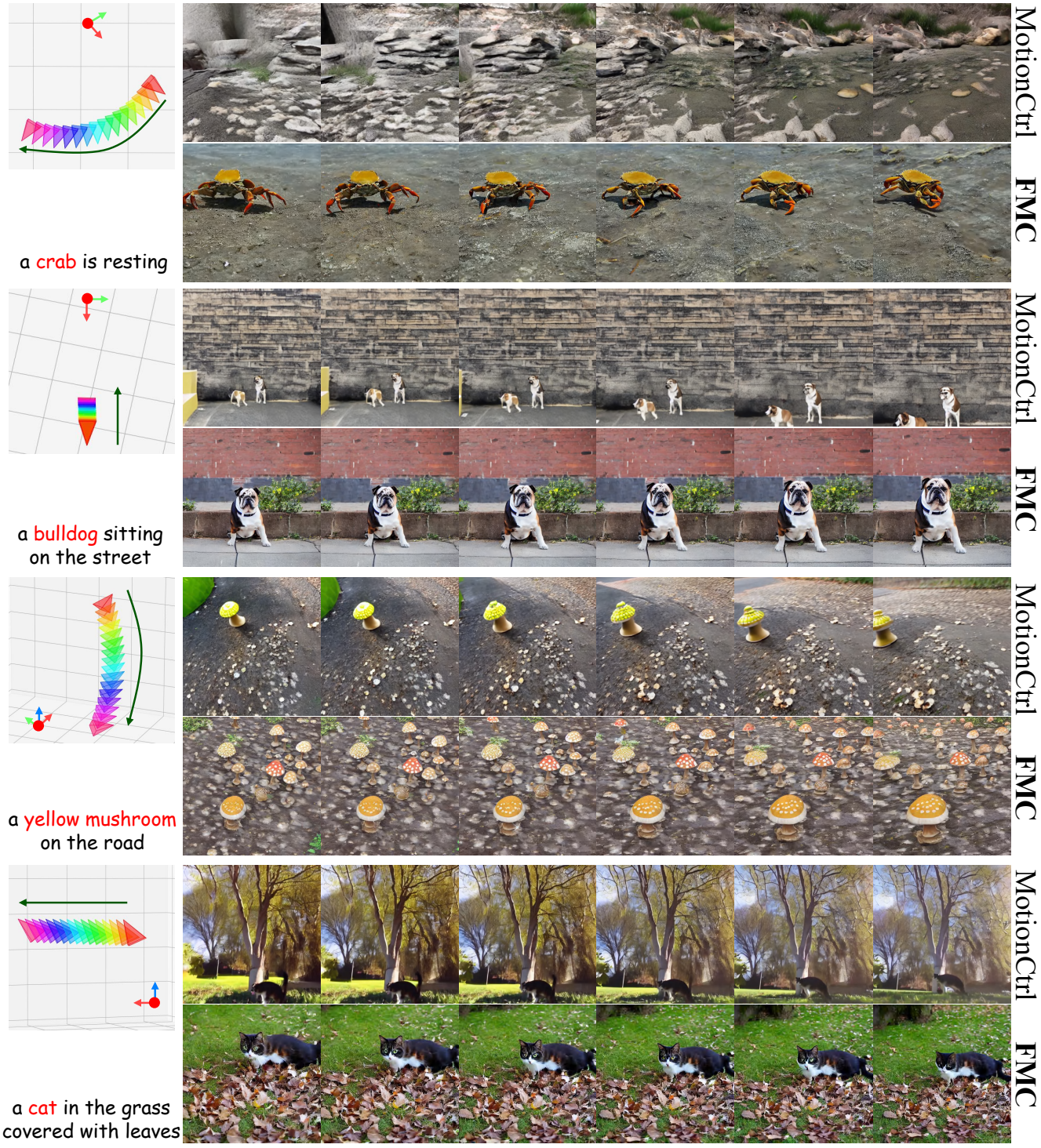


Figure X. Simultaneous control in *static single-object* scene.

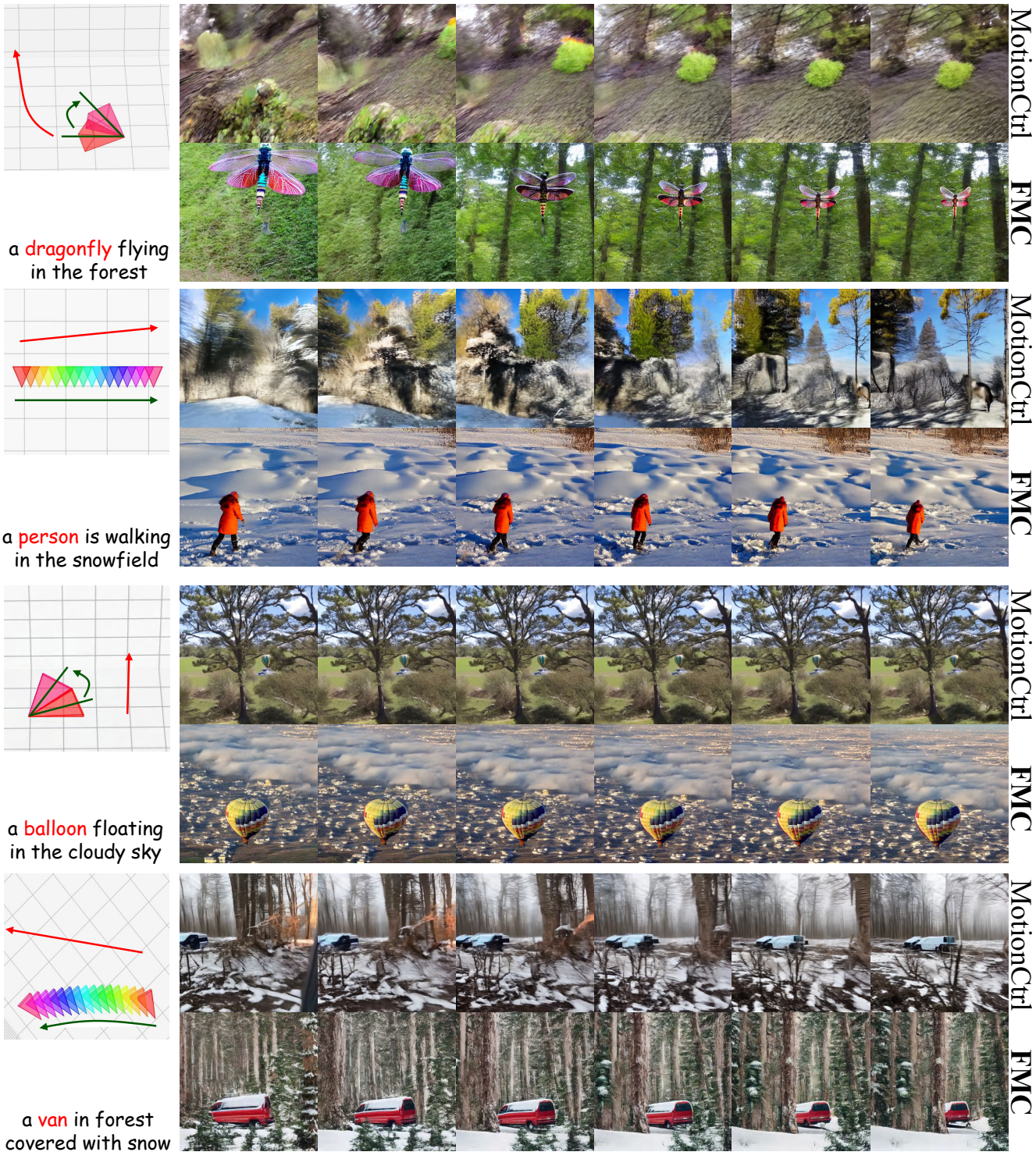


Figure XI. Simultaneous control in *dynamic single-object* scene.

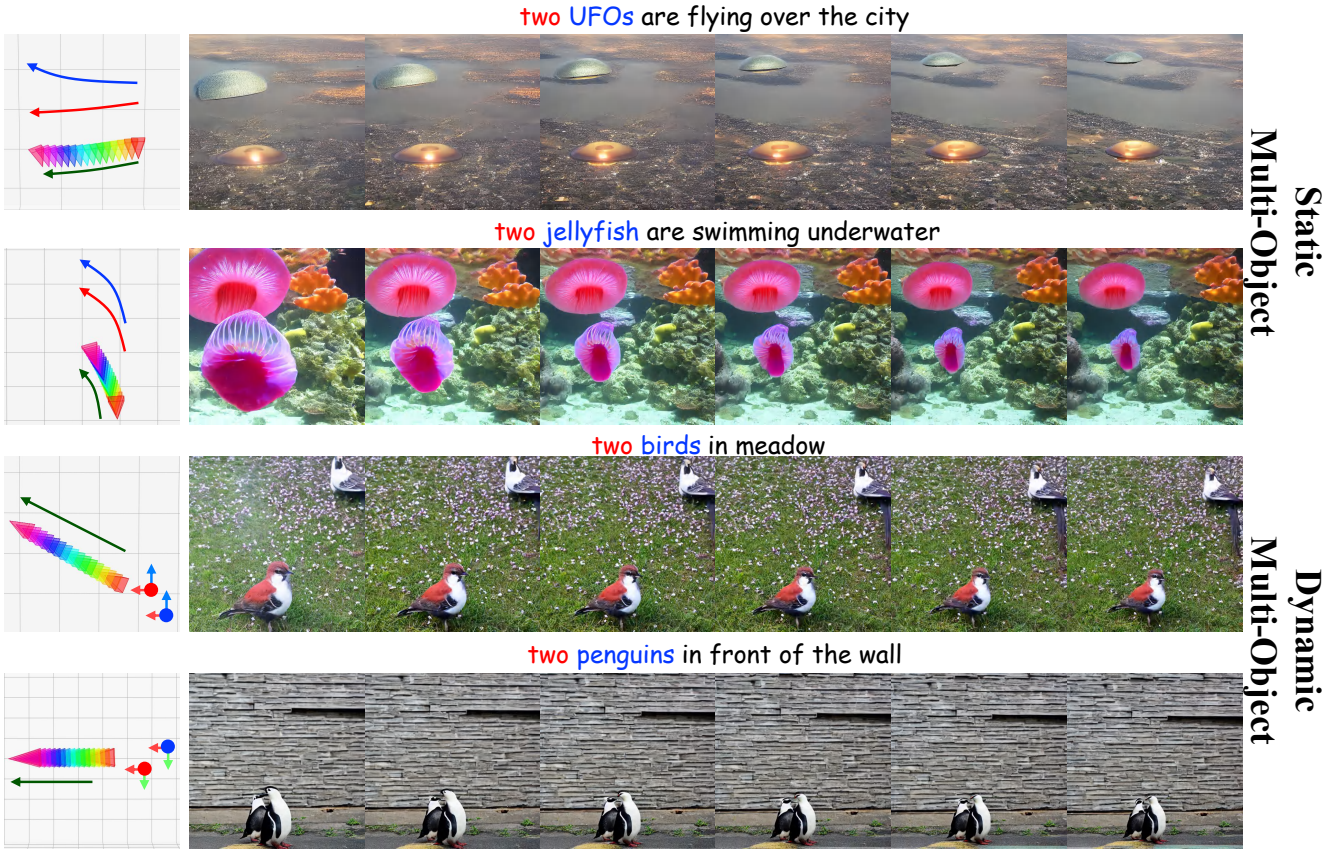


Figure XII. Simultaneous control in *multi-object* scenes.

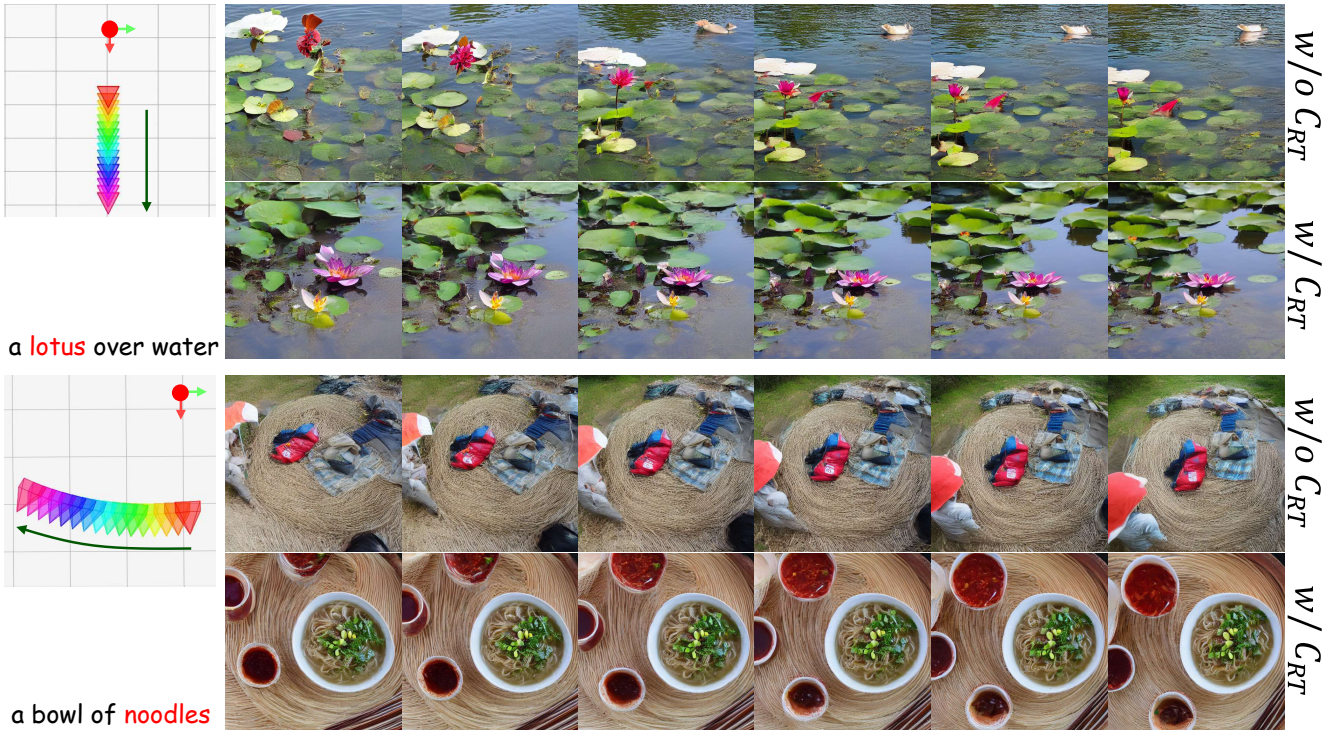


Figure XIII. Ablation study of MotionCtrl [23] trained on our *SynFMC*. The integration of camera condition makes the model generate reasonable results in simultaneous control.

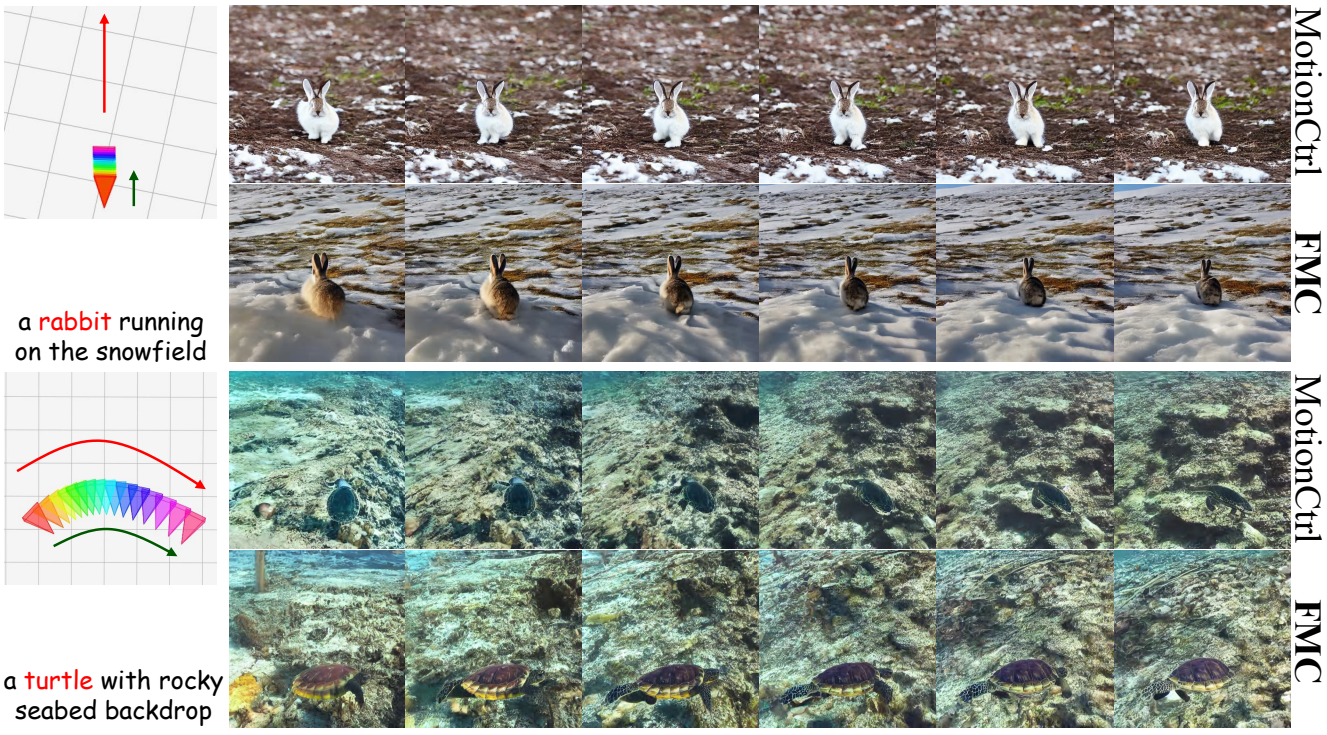


Figure XIV. Ablation study of object motion control modules from MotionCtrl [23] and our *FMC*. MotionCtrl struggles to generate plausible object appearances across varying viewpoints since it can only handle with the trajectory input confined to the image space, being insensitive to the orientation and size of the object within the image.

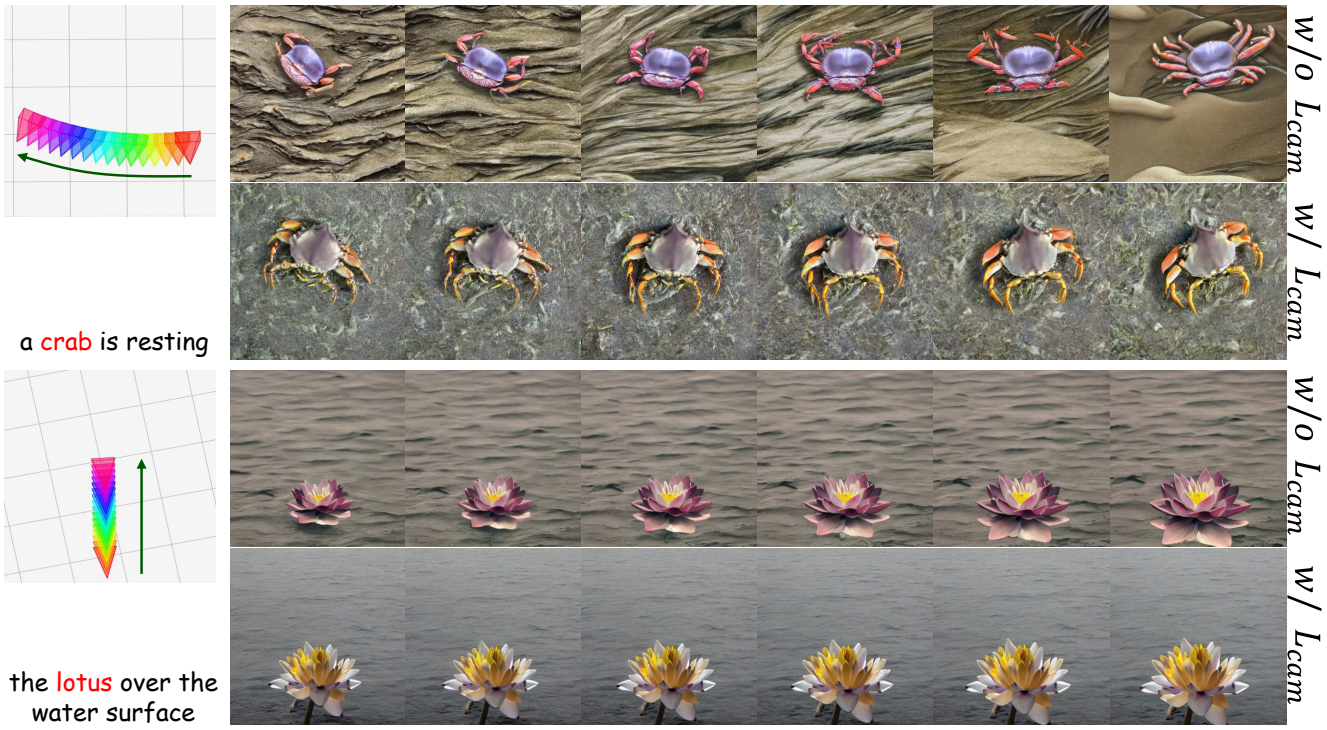


Figure XV. Ablation study of camera loss L_{cam}. The 1st row reveals that the model struggles to learn complex camera movements without L_{cam} under the same training configuration. The 3rd row shows that the model tends to move the object to satisfy the relative movement indicated by the motion conditions (the petals of the lotus are unnaturally stretched toward the camera).

Figure XVI. Ablation study of object loss L_{obj}. The model fails to generate coherent object appearance with the standard diffusion loss.

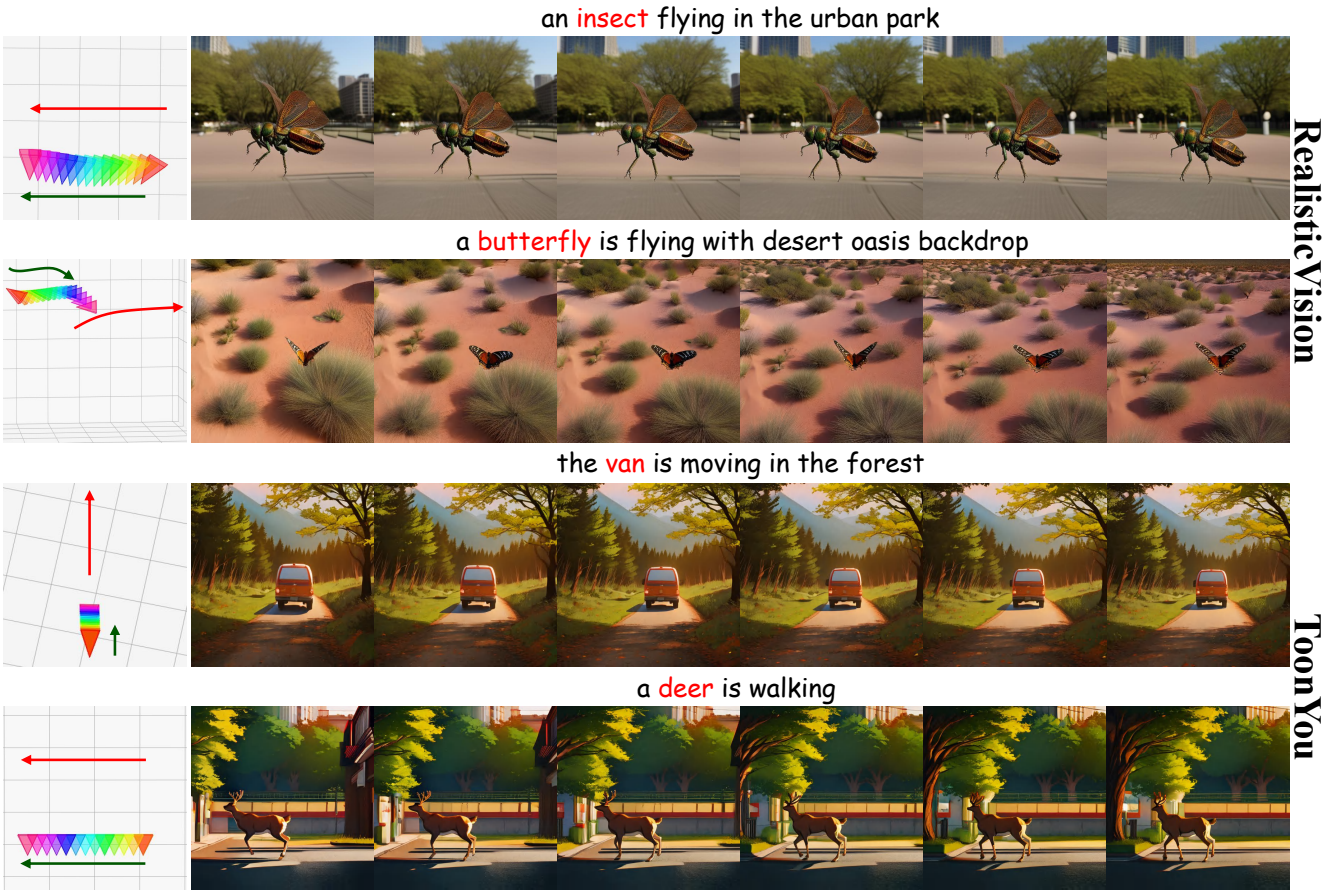


Figure XVII. Results of *FMC* based on different T2I personalized backbones [18, 29].

Figure XVIII. Failure cases of our *FMC*. In some challenging scenarios involving multi-object motion control, *FMC* fails to completely reflect semantics contained in the description.

LLM System: You are an assistant specialized in deducing the habitat classification of objects based on their class names. You will classify them into one of the following categories based on the typical environment in which they are commonly found:
1. *Ground*: Objects that are primarily found on or associated with the ground, such as animals (human, character, dog, snake, monkey, deer, etc.), ground vehicles (car, motorbike, bike, etc.), plants, buildings, etc.
2. *Near Ground*: Objects that are primarily flying or hovering close to the ground, such as small birds, insects like bees or moths, and drones.
3. *Sky*: Objects that are primarily found in the sky, such as airplanes, eagles, or large birds.
4. *Water Surface*: Objects that are primarily found on the surface of water or associated with water activities, such as duck, boats, water lily, floating devices, or bridges.
5. *Underwater*: Objects that are primarily found underwater or associated with underwater activities, such as fish, dolphin, coral, diving equipment, submarines, etc.

LLM Task: Based on the class name of the object, deduce the environment in which it is typically found and classify it into one of the categories above. Please directly return the category without other words.

LLM Examples: Here are some examples:
1. *Class name of object*: Human
   Response: Ground
2. *Class name of object*: Car
   Response: Ground
3. *Class name of object*: Bee
   Response: Near Ground
4. *Class name of object*: Butterfly
   Response: Near Ground
5. *Class name of object*: Eagle
   Response: Sky
6. *Class name of object*: Airplane
   Response: Sky
7. *Class name of object*: Duck
   Response: Water Surface
8. *Class name of object*: Boat
   Response: Water Surface
9. *Class name of object*: Fish
   Response: Underwater
10. *Class name of object*: Submarine
    Response: Underwater

Figure XIX. The query text for deducing the object's environment type in MLLM.

LLM System: You are an assistant specialized in deducing the speed classification of objects based on their class names. You will classify them according to their typical speed in daily life into one of the following categories:
1. *Stationary*: Objects that do not move or move very little, typically 0 m/s, such as buildings, trees.
2. *Slow*: Objects that move very slowly, typically between 0.01-1.5 m/s, such as snails, turtles, or slow-moving objects like conveyor belts.
3. *Moderate*: Objects with moderate speed, typically between 1.5-10 m/s, such as humans, dogs, or bicycles.
4. *Fast*: Objects that move quickly, typically between 10-60 m/s, such as cars, fast animals like cheetahs, or trains.
5. *Very Fast*: Objects that move at high speeds, typically between 60-340 m/s, such as airplanes, high-speed trains, or fast birds like peregrine falcons.
6. *Supersonic*: Objects moving faster than the speed of sound, typically over 340 m/s, such as fighter jets or rockets.

LLM Task: Based on the class name of the object, deduce its speed and classify it into one of the categories above. Please directly return the category without other words.

LLM Examples: Here are some examples:
1. *Class name of object*: Tree
   Response: Stationary
2. *Class name of object*: Stone
   Response: Stationary
3. *Class name of object*: Snail
   Response: Slow
4. *Class name of object*: Car
   Response: Fast
5. *Class name of object*: Balloon
   Response: Moderate
6. *Class name of object*: Human
   Response: Moderate
7. *Class name of object*: Cheetah
   Response: Fast
8. *Class name of object*: Airplane
   Response: Very Fast
9. *Class name of object*: Fighter Jet
   Response: Supersonic

Figure XX. The query text for deducing the object's speed type in MLLM.

LLM System: You are an assistant specialized in deducing the size classification of objects based on their class names. You will classify them according to their typical size in daily life into one of the following categories:

1. *Tiny*: Objects typically between 0.1-5 cm in length, such as nails, coins, buttons, insects (ants, bees, etc.).
2. *Small*: Handheld or smaller objects, or small animals or plants typically between 5-30 cm in length, such as phones, keys, cups, small birds, hamsters, grass etc.
3. *Medium*: Larger than small objects but still easily carried by one person, typically between 30-150 cm in length, such as chairs, computer monitors, microwaves, flowers, dogs, cats, duck and large birds.
4. *Large*: Objects that usually require two or more people to move, or are difficult to transport, typically between 150-300 cm in length, such as humans, characters, robots, sofas, refrigerators, beds, large dogs, deer, and sheep.
5. *Extra-Large*: Objects between 300-500 cm that require specialized equipment to move, such as trees, cars, parts of ships, cows, horses, elephants and giraffes.
6. *Gigantic*: Large public objects, or very large animals, typically over 500 cm in length, such as airplanes, bridges, large buildings, and whales.

LLM Task: Based on the class name of the object, deduce its size and classify it into one of the categories above. Please directly return the category without other words.

LLM Examples: Here are some examples:

1. *Class name of object*: Phone
   Response: Small
2. *Class name of object*: Dog
   Response: Medium
3. *Class name of object*: Car
   Response: Extra-Large
4. *Class name of object*: Key
   Response: Small
5. *Class name of object*: Elephant
   Response: Extra-Large
6. *Class name of object*: Human
   Response: Large
7. *Class name of object*: Robot
   Response: Large
8. *Class name of object*: Cat
   Response: Medium
9. *Class name of object*: Airplane
   Response: Gigantic

Figure XXI. The query text for deducing the object's size type in MLLM.

References

[1] BlenderKit. *https://www.blenderkit.com*, 2022. 1

[2] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024. 1

[3] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. In *NeurIPS*, 2023. 1

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023. 1

[5] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, and Chen Change Loy. MeViS: A large-scale benchmark for video segmentation with motion expressions. In *ICCV*, 2023. 2

[6] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip HS Torr, and Song Bai. MOSE: A new dataset for video object segmentation in complex scenes. In *ICCV*, 2023. 2

[7] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. AnimateDiff: Animate your personalized text-to-image diffusion models without specific tuning. In *ICLR*, 2024. 1

[8] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. CameraCtrl: Enabling camera control for text-to-video generation. *arXiv*, 2024. 1, 2

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 6

[10] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, et al. Imagen Video: High definition video generation with diffusion models. *arXiv*, 2022. 1

[11] Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *NeurIPS*, 2022. 1

[12] Yash Jain, Anshul Nasery, Vibhav Vineet, and Harkirat S. Behl. Peekaboo: Interactive video generation via masked-diffusion. In *CVPR*, 2024. 6

[13] Mixamo. *https://mixamo.com*, 2022. 1

[14] PolyHaven. *https://polyhaven.com*, 2022. 1

[15] Haonan Qiu, Zhaoxi Chen, Zhouxia Wang, Yingqing He, Menghan Xia, and Ziwei Liu. FreeTraj: Tuning-free trajectory control in video diffusion models. *arXiv*, 2024. 6

[16] Xincheng Shuai, Henghui Ding, Xingjun Ma, Rongcheng Tu, Yu-Gang Jiang, and Dacheng Tao. A survey of multimodal-guided image editing with text-to-image diffusion models. *arXiv preprint arXiv:2406.14555*, 2024. 1

[17] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. In *ICLR*, 2023. 1

[18] Bradcatt: Toonyou. *https://civitai.com/models/30240/toonyou*, 2024. 6, 14

[19] Realistic Vision. *https://civitai.com/models/4201/realistic-vision-v20*, 2023. 6

[20] Mengyu Wang, Henghui Ding, Jianing Peng, Yao Zhao, Yunpeng Chen, and Yunchao Wei. CharaConsist: Fine-grained consistent character generation. In *ICCV*, 2025. 1

[21] Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. VideoComposer: Compositional video synthesis with motion controllability. In *NeurIPS*, 2023. 6

[22] Zihao Wang. Score-based generative modeling through backward stochastic differential equations: Inversion and generation. In *ICLR*, 2021. 6

[23] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. MotionCtrl: A unified and flexible motion controller for video generation. In *SIGGRAPH*, 2024. 1, 5, 6, 8, 12

[24] Shiyuan Yang, Liang Hou, Haibin Huang, Chongyang Ma, Pengfei Wan, Di Zhang, Xiaodong Chen, and Jing Liao. Direct-a-Video: Customized video generation with user-directed camera movement and object motion. In *SIGGRAPH*, 2024. 5, 8

[25] Shengming Yin, Chenfei Wu, Jian Liang, Jie Shi, Houqiang Li, Gong Ming, and Nan Duan. DragNUWA: Fine-grained control in video generation by integrating text, image, and trajectory. *arXiv*, 2023. 6

[26] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G. Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. MAGVIT: masked generative video transformer. In *CVPR*, 2023. 1

[27] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 2

[28] Shiwei Zhang, Jiayu Wang, Yingya Zhang, Kang Zhao, Hangjie Yuan, Zhiwu Qin, Xiang Wang, Deli Zhao, and Jingren Zhou. I2VGen-XL: High-quality image-to-video synthesis via cascaded diffusion models. *arXiv*, 2023. 1

[29] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM TOG*, 2018. 14