A. Appendix

A.1. Formulation of Diffusion Model

We define $\mathbf{x}_0$ as the desired refined sailency latent, and $\mathbf{x}_t$ as the saliency latent distribution by adding Gaussian noise distribution sequentially t times. Following the notion of (Ho et al., 2020; Dhariwal & Nichol, 2021; Nichol & Dhariwal, 2021), this is called the diffusion process. Thus, we could continuously add noise into original $\mathbf{x}_0$ through a Markov process sampling variables $\{\mathbf{x}_0, \mathbf{x}_1, ...\mathbf{x}_{t-1}, \mathbf{x}_t, ..., \mathbf{x}_T\}$ until $\mathbf{x}_T$ becomes a normal noise distribution $p(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{x}_T; 0, I)$. Here, we call the $\mathbf{x}_T$ as the initialization of the Starting from a refined salient latent distribution $\mathbf{x}_0$, we define a forward Markovian noising process q as below. In particular, the added noise is scheduled by the variance $\beta_t(I)$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) \coloneqq \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{11}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{12}$$

As noted by Ho (Ho et al., 2020) of the sampling properties, we can directly sample data $\mathbf{x}_t$ at an arbitrary timestep t without the need of applying q repeatedly:

$$q(\mathbf{x}_t|\mathbf{x}_0) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{13}$$

$$\coloneqq \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \epsilon\sqrt{1-\bar{\alpha}_t}, \epsilon(I)\mathcal{N}(0, \mathbf{I}) \tag{14}$$

where $\bar{\alpha}_t \coloneqq \prod_{s=0}^{t} \alpha_s$ and $\alpha_t \coloneqq 1 - \beta_t$ are also a fixed variance coefficient schedule corresponding to β_t. Based on Bayes' theorem, it is found that the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is a Gaussian distribution as well:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}) \tag{15}$$

where

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) \coloneqq \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \tag{16}$$

and

$$\tilde{\beta}_t \coloneqq \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{17}$$

are the mean and variance of this Gaussian distribution.

In practice, the representation of $\mathbf{x}_t$ could be obtained by extending the diffusion process defined in Equation 14 as below.

$$\mathbf{x}_t = \bar{\alpha}_t\mathbf{x}_0 + \alpha_t\bar{\beta}_{t-1}\bar{\boldsymbol{\vartheta}}_{t-1} + \beta_t\boldsymbol{\vartheta}_t \tag{18}$$

where $\boldsymbol{\vartheta_t} \sim \mathcal{N}(0, (I))$ is a gaussian distribution that represents the stochastic property of the diffusion process. It also gives a description of how to represent the diffusion result in $\mathbf{x}_t$ by real sample $\mathbf{x}_0$ and given fixed variance scheduler α_t and β_t. We could get a sample from $q(\mathbf{x}_0)$ by first sampling from $q(\mathbf{x}_T)$ and running the reversing steps $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ until $\mathbf{x}_0$.

Besides, the distribution of $q(\mathbf{x}_T)$ is nearly an isotropic Gaussian distribution with a sufficiently large T and reasonable schedule of β_t ($\beta_t \to 0$), which making it trivial to sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. Moreover, since calculating $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ exactly should depend on the entire data distribution, we could approximate $q_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ using a neural network posterior process, which is optimized to predict a mean μ_θ and a diagonal covariance matrix Σ_θ:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \coloneqq \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_\theta(\mathbf{x}_t, t)) \tag{19}$$

Instead of directly parameterizing $\mu_\theta(\mathbf{x}_t, t)$, Ho (Ho et al., 2020) found learning a network $f_\theta(\mathbf{x}_t, t)$ to predict the ϵ or $\mathbf{x}_0$ from 14 worked best. We choose to predict $\mathbf{x}_0$ in this work.

However, saliency denoising can't be performed without any given images. We utilize the visual condition c to guide the denoising process. As we assume above, the condition, $\mathbf{c}$ is entirely independent of the denoising and diffusion process. So we can easily change it into conditioned denoising by simply modifying the denoising process as below.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) \coloneqq \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t, \mathbf{c}), \sigma_\theta(\mathbf{x}_t, t)) \tag{20}$$

However, since we desired a deterministic process, which means that given an image pairs, the generated saliency map is uniquely deterministic. This is a pre-request for accurate prediction tasks. In that case, we eliminate randomness according to DDIM (Song & Ermon, 2020). Where the random variance $\boldsymbol{\vartheta_t}$ is set to 0.

B. Extended Implementation details

The implementation and the pre-trained model will be open-source after acceptance. Here we release an anonymous link [1] for review. SimpleDiffusion is implemented with the Pytorch (Paszke et al., 2019) framework. We train the entire model with batch size 32 for 150 epochs iterations on a single node with NVIDIA A100 GPUs. We utilize the AdamW optimizer (Kingma & Ba, 2014) with (β_1, β_2, w) = (0.9, 0.999, 0.01), where w is the weight decay. The linear learning rate warm-up strategy is applied for the first 15% iterations. The cosine annealing learning rate strategy is adopted for the learning rate decay from the initial learning rate of $1e-4$ to $1e-8$. We use L1 and L2 pixel-wise loss at the first 50% training iterations as auxiliary subversion.

Augmentation We randomly crop a patch from the original image and its corresponding depth or thermal map and resize them to the desired input size. This helps to avoid overfitting and focus on learning to refine the details of different regions of the image. We randomly adjust the brightness, contrast, saturation, and hue of the input image. This helps to simulate different lighting conditions and make the network invariant with color changes. We randomly flip the input image and its corresponding depth or thermal map horizontally. This helps to augment the data with different orientations and symmetries. Random rotation: We randomly rotate the input image and its corresponding depth or thermal map by a small angle. This helps to augment the data with different angles and perspectives. Here, we use $-5-5$ degrees as the rotation parameter.

Visual Condition: SimpleDiffusion is compatible with any backbone which could extract multi-scale features. Here, we respectively evaluate our model on the standard convolution-based ResNet (He et al., 2016) backbones and transformer-based PVT backbones. We employ hierarchical aggregation and heterogeneous interaction (HAHI (Li et al., 2022)) neck to enhance features between scales and feature pyramid neck (Lin et al., 2017) to aggregate features into corss-modal salient visual condition. The salient visual condition dimension is equal to the last layer of the neck.

Diffusiong Head: We use the improved sampling process (Song & Ermon, 2020) with 1000 diffusion steps for training and 10 inference steps for inference. The learning rate of the diffusion head is 10 times larger than the backbone parameters. The dimension d of the encoded saliency latent is 16 with shape $\frac{H}{2}, \frac{W}{2}, d$, we conduct detailed ablation to illustrate different inference settings.

B.1. Evaluation Metrics

Suppose the predicted and ground-truth saliency to be $\hat{I} \in \mathbb{R}^{m \times n}$ and $I_{gt} \in \mathbb{R}^{m \times n}$, respectively, and the number of valid pixels to be N. We follow the existing methods (Luo et al., 2024) and utilize the following measures for quantitative evaluation. We listed all potential metrics below: We follow the existing methods (Luo et al., 2024) and utilize the following measures for quantitative evaluation. We list all potential metrics below:

1.Structure Measure (S_m). The structure measure (S_m) evaluates the structural similarity between a predicted saliency map (P) and a ground-truth (G). It is defined as follows:

$$S_m = \alpha \cdot S_o + (1-\alpha) \cdot S_r$$

where S_o measures the object-aware structural similarity and S_r measures the region-aware structural similarity. α balances the two terms, typically set to 0.5. This measure assesses how well the predicted saliency map preserves structural

[1] https://anonymous.4open.science/r/simple-diffusion

information from the ground-truth. S_m aims to align predictions with ground-truth both globally (region-based) and locally (object-based), making it particularly suitable for saliency detection tasks.

2. Maximum Enhanced-alignment Measure (E_m) The maximum enhanced-alignment measure (E_m) evaluates the alignment between the predicted saliency map (P) and the ground-truth (G) by directly comparing their pixel-level distributions. It is formulated as:

$$E_m = \max_{\beta} E_\beta, \quad E_\beta = \frac{2 \cdot \Phi(P, G)}{\Phi(P, P) + \Phi(G, G)}$$

where Φ denotes the alignment function that computes the similarity between the two distributions. β is a scaling factor applied to the prediction prior to computing E_β. This metric emphasizes precise alignment between the saliency prediction and the ground-truth, making it robust against outlier regions. E_m improves on previous evaluation metrics by enhancing the sensitivity to consistent patterns.

3. Maximum F-measure (F_m) The F-measure is a widely used metric in binary classification problems and is defined as the harmonic mean of precision and recall. For saliency detection, the maximum F-measure is computed as:

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

and the maximum value across all thresholds is used:

$$F_m = \max_{\tau} F_\beta(\tau)$$

where β^2 is typically set to 0.3 to emphasize precision over recall, and τ is the threshold applied to the predicted saliency map. F_m provides a trade-off between precision and recall, offering a balanced evaluation of the model's capacity to detect salient regions.

4. Mean Absolute Error (MAE) The mean absolute error (MAE) computes the average pixel-wise absolute difference between the predicted saliency map (P) and the ground-truth (G) as:

$$\text{MAE} = \frac{1}{W \cdot H} \sum_{i=1}^{W} \sum_{j=1}^{H} |P(i, j) - G(i, j)|$$

where W and H are the width and height of the saliency map, respectively. MAE provides a straightforward assessment of the overall accuracy of the model's predictions, measuring how much the predicted saliency map differs from the ground-truth on average. These metrics collectively provide a comprehensive evaluation framework, ensuring both structural alignment and pixel-wise accuracy are taken into account.

C. Extensive Generalized Experiments

Building upon previous work, we conducted additional experiments on the RGB-D Salient Instance Segmentation Dataset. Remarkably, without making any modifications to our SimpleDiffusion framework, we only replaced the input GT mask with segmentation masks, and our model achieved excellent instance segmentation results. This further suggests that our multimodal salient object detection framework, SimpleDiffusion, is a versatile solution applicable to a wide range of saliency-related tasks. Please refer to the figure below for more details.

D. Results of Other Comparative Experiments.

To demonstrate that our multi-scale feature extraction backbone can be seamlessly replaced with other general-purpose backbones, we replaced the PVT backbone with Res2Net50 and conducted extensive comparisons with other methods. The detailed results are presented below.

Table 4. Quantitative Comparison of E-Measure (E_ξ), Weighed F-Measure (F_β^ω), F-Measure (F_β), Mean Absolute Error (MAE) and Frame-Per-Second (FPS) with 17 Different RGBD Methods on seven Testing Datasets, Including SIP, DUT-RGBD (i.e., DUT), SSD, STERE, ReDWeb-S, NLPR, and NJUD. The Best, Second Best and Third Best Results Are Marked with Red and Green, Respectively. '-' Indicates the Code or Result Is Not Available.

Methods	Backbone	SIP				DUT				SSD				STERE				ReDWeb-S				NLPR				NJUD				FPS $\uparrow$	Param $\downarrow$	GFLOPs $\downarrow$
		$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$	$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$	$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$	$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$	$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$	$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$	$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$			
HDFNet_{20}	VGG19	0.920	0.848	0.862	0.048	0.937	0.873	0.887	0.040	0.902	0.823	0.831	0.047	0.924	0.863	0.860	0.041	0.760	0.623	0.704	0.133	0.949	0.873	0.874	0.028	0.911	0.879	0.856	0.039	77	164.19	111.02
CoNet_{20}	ResNet101	0.909	0.814	0.842	0.063	0.948	0.896	0.909	0.033	0.896	0.792	0.806	0.059	0.928	0.874	0.885	0.037	0.762	0.618	0.688	0.147	0.934	0.850	0.848	0.031	0.912	0.856	0.873	0.046	154	162.13	13.05
CCAFNet_{21}	ResNet50	0.915	0.839	0.864	0.054	0.940	0.884	0.903	0.037	0.915	0.839	0.864	0.054	0.921	0.853	0.869	0.044	0.687	0.515	0.614	0.173	0.951	0.883	0.880	0.026	0.920	0.883	0.896	0.037	88	159.45	189.47
CDNet_{21}	VGG16	0.913	0.839	0.870	0.056	0.936	0.878	0.901	0.039	0.849	0.706	0.742	0.073	0.929	0.871	0.884	0.039	0.730	0.588	0.678	0.146	0.951	0.886	0.879	0.025	0.903	0.828	0.856	0.054	86	123.65	72.07
HAINet_{21}	VGG16	0.924	0.860	0.883	0.049	0.937	0.887	0.905	0.038	0.843	0.682	0.735	0.101	0.930	0.877	0.888	0.038	0.766	0.654	0.713	0.132	0.951	0.884	0.884	0.025	0.917	0.882	0.895	0.039	11	228.20	181.62
DFM_{21}	MobileNetV2	0.919	0.844	0.871	0.051	0.898	0.795	0.830	0.062	0.871	0.733	0.755	0.076	0.912	0.850	0.858	0.045	0.753	0.610	0.690	0.136	0.945	0.876	0.868	0.026	0.913	0.868	0.885	0.042	252	8.45	5.43
SPNet_{21}	Res2Net50	0.930	0.873	0.891	0.043	0.876	0.747	0.843	0.085	0.910	0.831	0.852	0.044	0.930	0.879	0.886	0.037	0.759	0.637	0.712	0.129	0.957	0.899	0.898	0.021	0.931	0.909	0.915	0.029	50	573.39	68.10
RD3D_{21}	I3DResNet50	0.919	0.852	0.873	0.049	0.949	0.913	0.925	0.031	0.905	0.794	0.812	0.052	0.926	0.877	0.885	0.038	0.701	0.487	0.595	0.177	0.957	0.894	0.890	0.022	0.918	0.890	0.900	0.037	94	110.30	43.51
DSA2F_{21}	VGG19	0.908	0.838	0.867	0.057	0.950	0.914	0.926	0.030	0.904	0.836	0.852	0.047	0.928	0.877	0.895	0.038	-	-	-	-	0.950	0.889	0.897	0.024	0.923	0.889	0.901	0.039	-	-	-
DCF_{21}	ResNet50	0.920	0.850	0.877	0.051	0.952	0.913	0.925	0.030	0.898	0.800	0.829	0.053	0.931	0.880	0.890	0.037	0.755	0.632	0.710	0.135	0.956	0.892	0.893	0.023	0.922	0.884	0.897	0.038	57	111.51	59.38
MobileSal_{21}	MobileNetV2	0.914	0.837	0.860	0.054	0.936	0.869	0.912	0.044	0.898	0.804	0.815	0.052	0.916	0.865	0.848	0.041	0.671	0.455	0.608	0.186	0.950	0.878	0.872	0.025	0.914	0.874	0.894	0.040	268	24.97	1.96
SSL_{22}	VGG16	0.921	0.851	0.875	0.049	0.927	0.859	0.888	0.046	0.833	0.638	0.696	0.100	0.923	0.864	0.875	0.042	-	-	-	-	0.954	0.885	0.884	0.027	0.881	0.786	0.821	0.065	52	282.95	272.20
DIGRNet_{22}	ResNet50	0.918	0.849	0.878	0.053	0.948	0.902	0.919	0.033	0.889	0.804	0.823	0.053	0.927	0.877	0.888	0.038	-	-	-	-	0.955	0.895	0.888	0.023	0.928	0.909	0.916	0.028	33	635.79	68.16
CIRNet_{22}	ResNet50	0.917	0.848	0.874	0.053	0.951	0.908	0.926	0.031	0.898	0.791	0.821	0.054	0.921	0.836	0.875	0.049	0.725	0.519	0.630	0.171	0.955	0.889	0.883	0.023	0.922	0.881	0.896	0.040	93	393.50	42.60
MoADNet_{22}	MobileNetV3	0.908	0.828	0.850	0.058	0.949	0.911	0.923	0.031	0.894	0.801	0.824	0.057	0.914	0.861	0.868	0.042	-	-	-	-	0.945	0.875	0.874	0.027	0.909	0.881	0.892	0.041	167	19.19	1.32
LSNet_{23}	MobileNetV2	0.927	0.856	0.882	0.049	0.891	0.775	0.831	0.074	0.902	0.796	0.820	0.055	0.913	0.827	0.854	0.054	0.691	0.458	0.566	0.193	0.955	0.881	0.882	0.024	0.922	0.885	0.899	0.038	316	17.41	3.04
CAVER_{23}	ResNet50	0.927	0.874	0.884	0.043	0.955	0.920	0.919	0.029	0.915	0.826	0.828	0.044	0.931	0.887	0.871	0.034	0.760	0.663	0.724	0.121	0.959	0.899	0.894	0.022	0.922	0.903	0.874	0.032	67	212.83	218.64
Ours	Res2Net50	0.939	0.887	0.908	0.040	0.959	0.921	0.932	0.025	0.926	0.845	0.867	0.040	0.933	0.892	0.905	0.035	0.769	0.675	0.738	0.129	0.960	0.912	0.914	0.021	0.927	0.918	0.929	0.028	55	24.04	29.42

Table 5. Quantitative Comparison of E-measure (E_ξ), Weighed F-measure (F_β^ω), F-measure (F_β), Mean Absolute Error (MAE) and Frame-Per-Second (FPS) with 14 Different RGBT Methods on Three Testing Datasets, Including VT5000, VT1000 (Tu et al., 2019) and VT821. The Best, Second Best and Third Best Results Are Marked with Red and Green, Respectively

Methods	Backbone	VT5000				VT1000				VT821				FPS $\uparrow$	Param $\downarrow$	GFLOPs $\downarrow$
		$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$	$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$	$E_\xi \uparrow$	$F_\beta^\omega \uparrow$	$F_\beta \uparrow$	$MAE \downarrow$			
MTMR_{18} (Wang et al., 2018)	-	0.795	0.397	0.595	0.114	0.836	0.485	0.715	0.119	0.815	0.462	0.662	0.108	-	-	-
M3S-NIR_{19}	-	0.780	0.327	0.575	0.168	0.827	0.463	0.717	0.145	0.859	0.407	0.734	0.407	-	-	-
SGDL_{19} (Tu et al., 2019)	-	0.824	0.559	0.672	0.089	0.856	0.652	0.764	0.090	0.847	0.583	0.730	0.085	-	-	-
ADF_{20}	VGG16	0.891	0.722	0.778	0.048	0.921	0.804	0.847	0.034	0.842	0.627	0.077	0.716	27	254.67	191.45
MIDD_{21}	VGG16	0.897	0.763	0.801	0.043	0.933	0.856	0.882	0.027	0.895	0.760	0.804	0.045	33	200	216.72
CSRNet_{21} (Huo et al., 2021)	ESPNetV2	0.905	0.796	0.811	0.042	0.925	0.878	0.877	0.024	0.909	0.821	0.831	0.038	-	-	-
CGFNet_{21}	VGG16	0.922	0.831	0.851	0.035	0.944	0.900	0.906	0.023	0.912	0.829	0.845	0.038	18	266.73	347.78
MMNet_{21}	Res2Net50	0.887	0.770	0.780	0.043	0.923	0.863	0.861	0.027	0.892	0.783	0.794	0.040	-	-	-
ECFFNet_{21} (Zhou et al., 2021)	ResNet34	0.906	0.802	0.807	0.038	0.93	0.885	0.876	0.021	0.902	0.801	0.810	0.034	-	-	-
MIA-DPD_{22}	ResNet50	0.893	0.780	0.793	0.040	0.926	0.864	0.868	0.025	0.850	0.720	0.741	0.070	-	-	-
OSRNet_{22}	ResNet50	0.908	0.807	0.823	0.040	0.935	0.891	0.892	0.022	0.896	0.801	0.814	0.043	142	59.67	51.27
DCNet_{22}	VGG16	0.920	0.819	0.847	0.035	0.948	0.902	0.911	0.021	0.912	0.823	0.841	0.033	43	91.57	207.21
LSNet_{23}	MobileNetV2	0.915	0.806	0.825	0.037	0.935	0.887	0.885	0.023	0.911	0.809	0.825	0.033	314	17.41	3.04
CAVER_{23} (Pang et al., 2023)	ResNet50	0.924	0.835	0.841	0.032	0.945	0.909	0.903	0.017	0.919	0.835	0.839	0.033	67	451.8	137.68
Ours	Res2Net50	0.936	0.857	0.868	0.030	0.948	0.915	0.914	0.020	0.924	0.828	0.863	0.033	55	24.04	29.42

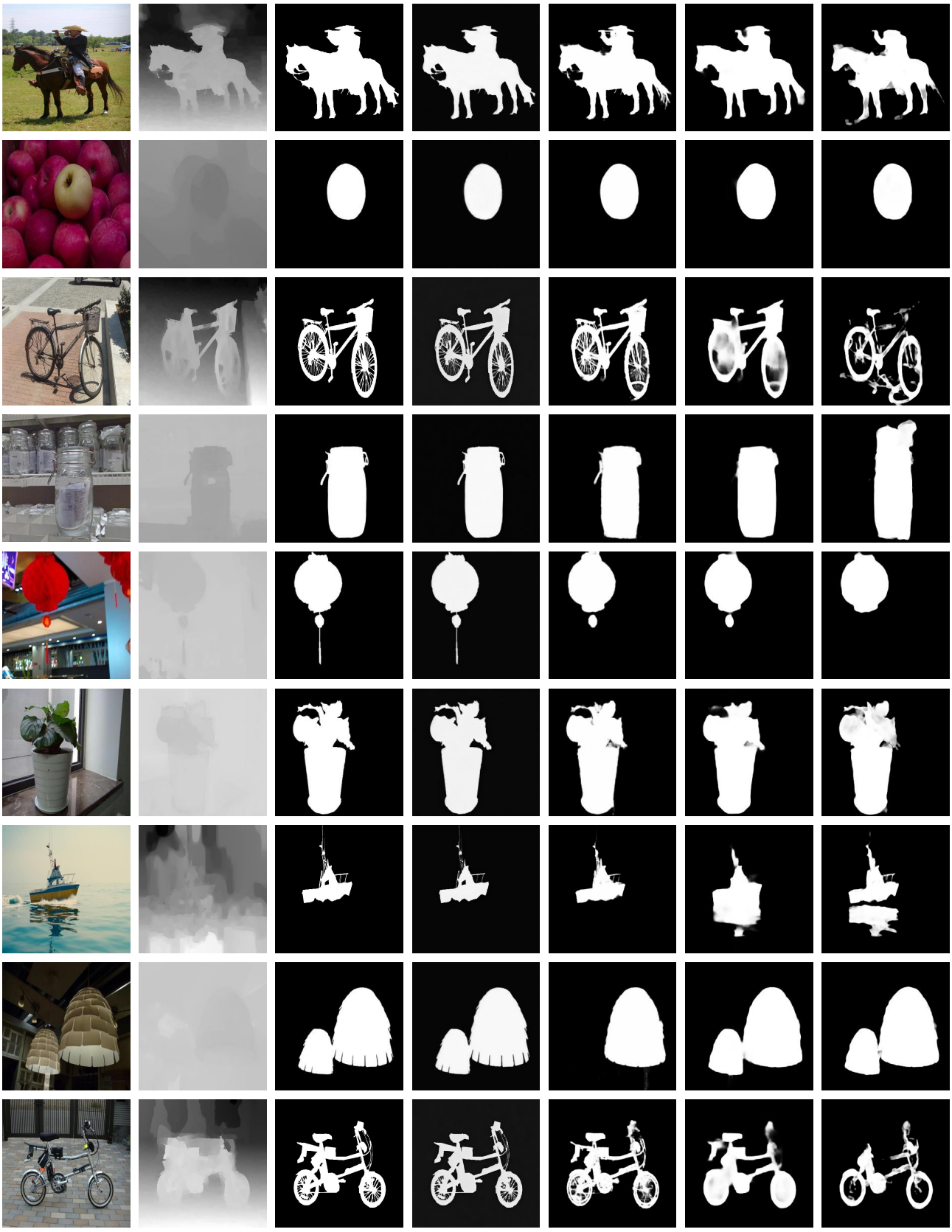

Figure 9. Detailed visual comparison of saliency map results generated by various advanced methods for RGB-D SOD.

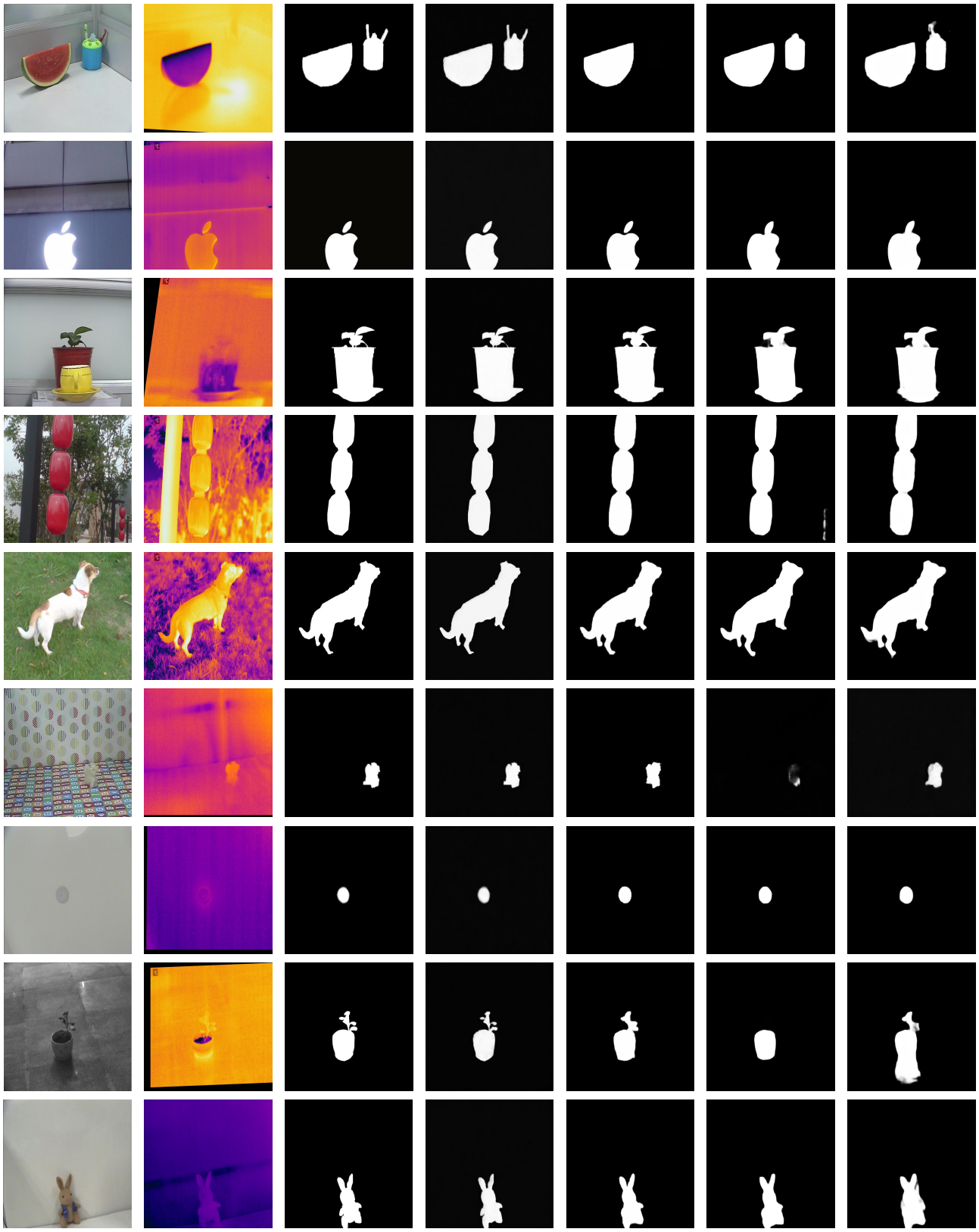

Figure 10. Detailed visual comparison of saliency map results generated by various advanced methods for RGB-T SOD.

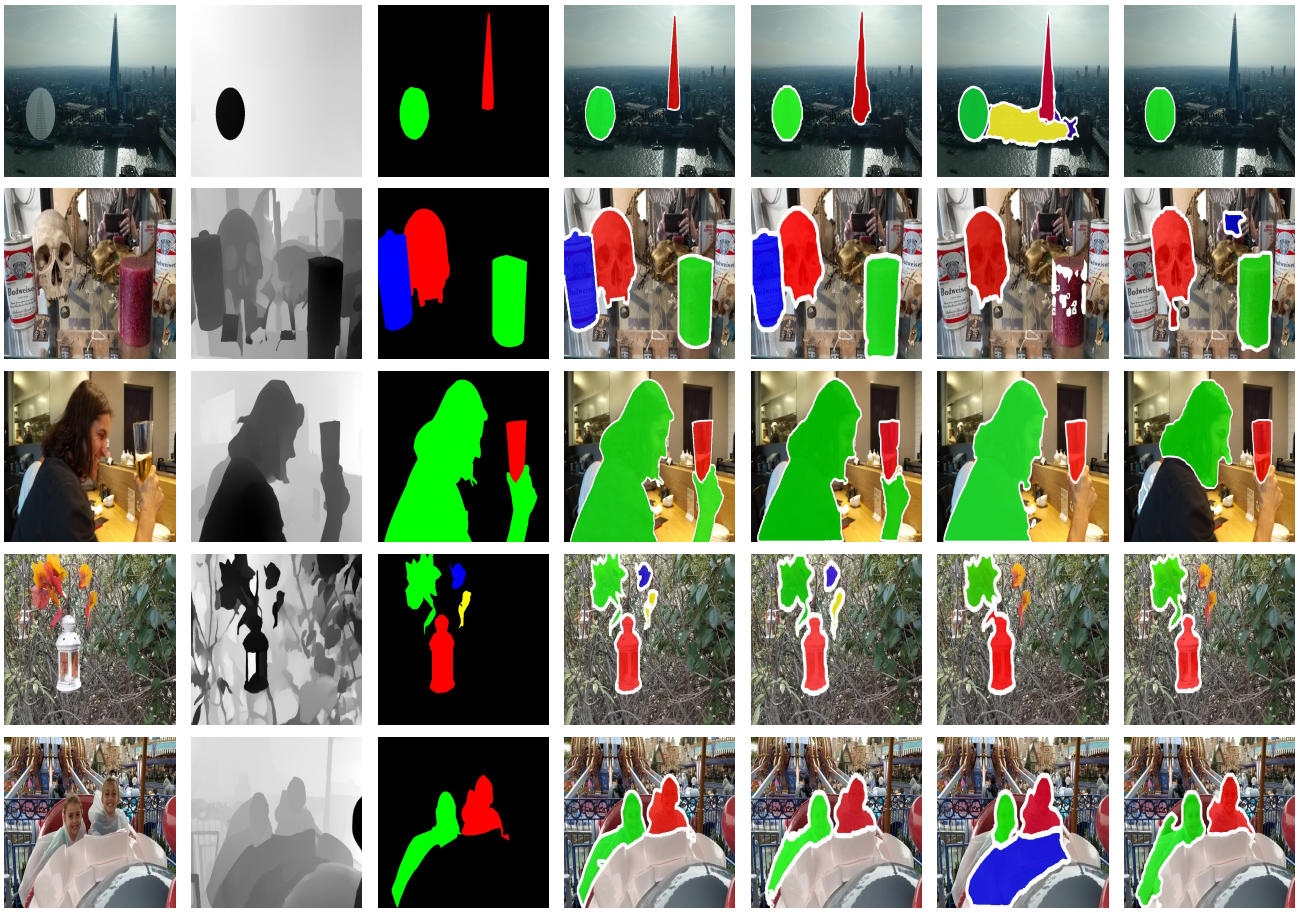

Figure 11. Visual Comparison Results of RGB-D SIS.