

**LawDIS: Language-Window-based  
Controllable Dichotomous Image Segmentation**  
Supplementary Material



Figure 7. Comparisons between the original images and their backgrounds-removed correspondences generated by our LawDIS.

## A. Implementation Details

### A.1. Annotation Pipeline of Language Prompt

We designed a semi-automated pipeline with three steps to generate a language prompt that accurately describes the foreground region of each image. Our pipeline effectively produces language descriptions that semantically align with each image’s ground-truth (GT) mask, offering a high-quality foundation for building LawDIS.

**Step-I: Initial prompt generation.** This step generates three sets of descriptive captions for each image from the DIS5K dataset [32]. Specifically, as illustrated in Step-I of Fig. 8, we meticulously crafted Instruction<sub>1</sub> - Instruction<sub>3</sub> to command the generation of each prompt. We employ two multimodal chatbots, namely MiniGPT-v2 [2] and GPT-4V [1], to generate language descriptions based on the entire image, designated Prompt<sub>1</sub> and Prompt<sub>2</sub>, respectively. Furthermore, under the guidance of Instruction<sub>3</sub>, GPT-4V generates Prompt<sub>3</sub>, focusing solely on the foreground area<sup>2</sup>.

**Step-II: Category-specific prompt optimization.** This step optimizes the three above generated prompts by integrating category-specific instructions. Specifically, in Step-II of Fig. 8, we design Instruction<sub>4</sub> - Instruction<sub>6</sub> to optimize the prompts generated in Step-I by incorporating the foreground object categories provided by the DIS5K dataset for each image. This optimization includes creating versions with synonymous expressions, more concise formulations, and category-specific descriptions. Using this instruction framework, we guide the GPT-4o mini [1] chatbot to generate more language descriptions, generating 12 new prompts: Prompt<sub>4</sub> to Prompt<sub>8</sub> derived from Prompt<sub>1</sub>, Prompt<sub>9</sub> to Prompt<sub>13</sub> derived from Prompt<sub>2</sub>, and Prompt<sub>14</sub> to Prompt<sub>15</sub> derived from Prompt<sub>3</sub>. This not only broadens the descriptive aspects but also enhances the coherence between the prompts and the category-specific content.

**Step-III: Optimal prompt selection.** As illustrated in Step-III of Fig. 8, we stitch the original image and the foreground area side by side to form a single image, and use GPT-4o [1] to select the optimal prompt from the 15 prompts generated earlier (*i.e.*, Prompt<sub>1</sub> to Prompt<sub>15</sub>), following the carefully designed Instruction<sub>7</sub>.

### A.2. Window Selection Strategy

In the process of window-controlled refinement, there are two pipelines to select the window: one is user-selected through clicking and dragging in the semi-automated refinement pipeline, and the other is automatically generated in the fully automated refinement pipeline. Therefore, we design a window selection strategy that can simulate the user’s window selection process in semi-automatic refinement pipelines to facilitate our experimental validation,

<sup>2</sup>The foreground area is obtained by element-wise multiplication of the image and the GT mask.

while also enabling automatic window generation in fully automated refinement pipelines. Algorithm 1 illustrates the pytorch-like pseudocode for the proposed strategy.

Specifically, in semi-automated pipeline, to simulate the process of users manually clicking to set windows, we take GT  $s$  as input and generate windows to be refined around the edges of the foreground objects in GT. This aligns with real-world applications where the edges of foreground objects often require the most refinement. In fully automated pipeline, we use the initial language-controlled segmentation map  $\hat{s}$  as the input for the window selection strategy, allowing the selection of windows around the object boundaries in the initial segmentation results. Notably, in practical applications, the windows in the semi-automated pipeline are generated by user clicks and do not require execution of this window selection strategy, providing greater flexibility.

## B. More Qualitative Comparisons

This section presents additional visual results. Fig. 9 illustrates the qualitative predictions with different prompts. Our approach adaptively segments the corresponding foreground objects based on user-defined language prompts, whereas the other two SOTA methods [45, 49], which lack prompt support, yield a fixed result. Moreover, Fig. 10 presents qualitative comparisons between our method and the four top existing DIS models, further demonstrating its superior performance.

## C. Training cost & model complexity

As seen in Tab. 8, LawDIS-S surpasses transformer-based MVANet [45] by 2.1%, requiring fewer (36K) training iterations and achieving a faster runtime of 0.326s/image. Moreover, we select diffusion-based DiffDIS [46] for a fair comparison. Our LawDIS-S (macro mode w/o post-refinement) still has the upper hand in performance (0.925 vs. 0.908) and inference speed (0.326s vs. 0.846s), despite comparable model parameters. All three models can run on an NVIDIA 4090 card, with user-friendly video memory consumption.

Model	Type	Parameters	Memory	Iter.	Time ↓	$F_{\beta}^{max}$ ↑
MVANet [45]	Transformer	369M	10.00G	240K	0.714s	0.904
DiffDIS [46]	Diffusion	3480M	18.38G	67.5K	0.846s	0.908
LawDIS-S	Diffusion	3537M	17.12G	<b>36K</b>	<b>0.326s</b>	<b>0.925</b>

Table 8. Comparison of training cost and model complexity.

## D. User study of controllability

To evaluate user satisfaction with controllability, ten users participated in a rating study involving ten visual cases predicted in macro and micro modes, where a score of 5 means the best controllability. The results are shown in Tab. 9.

	CASE#1	CASE#2	CASE#3	CASE#4	CASE#5	CASE#6	CASE#7	CASE#8	CASE#9	CASE#10	Avg. (macro/micro)
USER#01	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 4	5 / 5	5 / 5	5 / 5	5 / 5	5 / 4.9
USER#02	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5
USER#03	5 / 4	4 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 4	5 / 5	5 / 4	4.9 / 4.7
USER#04	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5
USER#05	4 / 5	5 / 4	5 / 4	5 / 5	5 / 5	5 / 5	4 / 5	5 / 5	5 / 5	5 / 5	4.8 / 4.8
USER#06	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5
USER#07	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5
USER#08	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5
USER#09	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5
USER#10	4 / 5	5 / 5	4 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 5	5 / 4	4.8 / 4.9
<b>Avg. (macro/micro)</b>	4.8 / 4.9	4.9 / 4.9	4.9 / 4.9	5 / 5	5 / 5	5 / 4.9	4.9 / 5	5 / 4.9	5 / 5	5 / 4.8	<b>4.95 / 4.93</b>

Table 9. User study of controllability in macro and micro modes.

## E. Discussion

LawDIS is the first framework for the *controllable* DIS task. Our innovation lies in repurposing the switcher to dynamically control generation at different levels of granularity, enabling a seamless transition between macro and micro parts. To achieve this, we designed new loss functions and a tailored training process. Compared to DPM [17], which is an LDM with a DDPM scheduler designed for general segmentation, our method introduces specific mechanisms—such as the mode switcher and VAE fine-tuning—to better handle fine structures. Moreover, compared to MbG [41], a two-stage diffusion model designed for human matting, our mode switcher connects macro segmentation with micro refinement during training, adaptively unlocking their synergy to enhance performance. Importantly, MbG-stage2 requires multiple patch-level refinements, each involving 50 time-consuming diffusion steps, which further highlights the efficiency advantage of our inference.

## F. Applications

Due to its capability of achieving high-precision segmentation of foreground objects at high resolutions, our LawDIS enables extensive application across a variety of scenarios. Fig. 7 shows the comparison between several original images and their corresponding background-removed versions. As can be seen, compared with the original image, the background-removed image shows higher aesthetic values and good usability, which can even be directly used as: 3D modeling (see the supplementary material file “app1-3D.mp4”), augmented reality (AR) (see the supplementary material file “app2-AR.mp4”), and still image animation (see the supplementary material file “app3-Still-Image-Animation.mp4”).

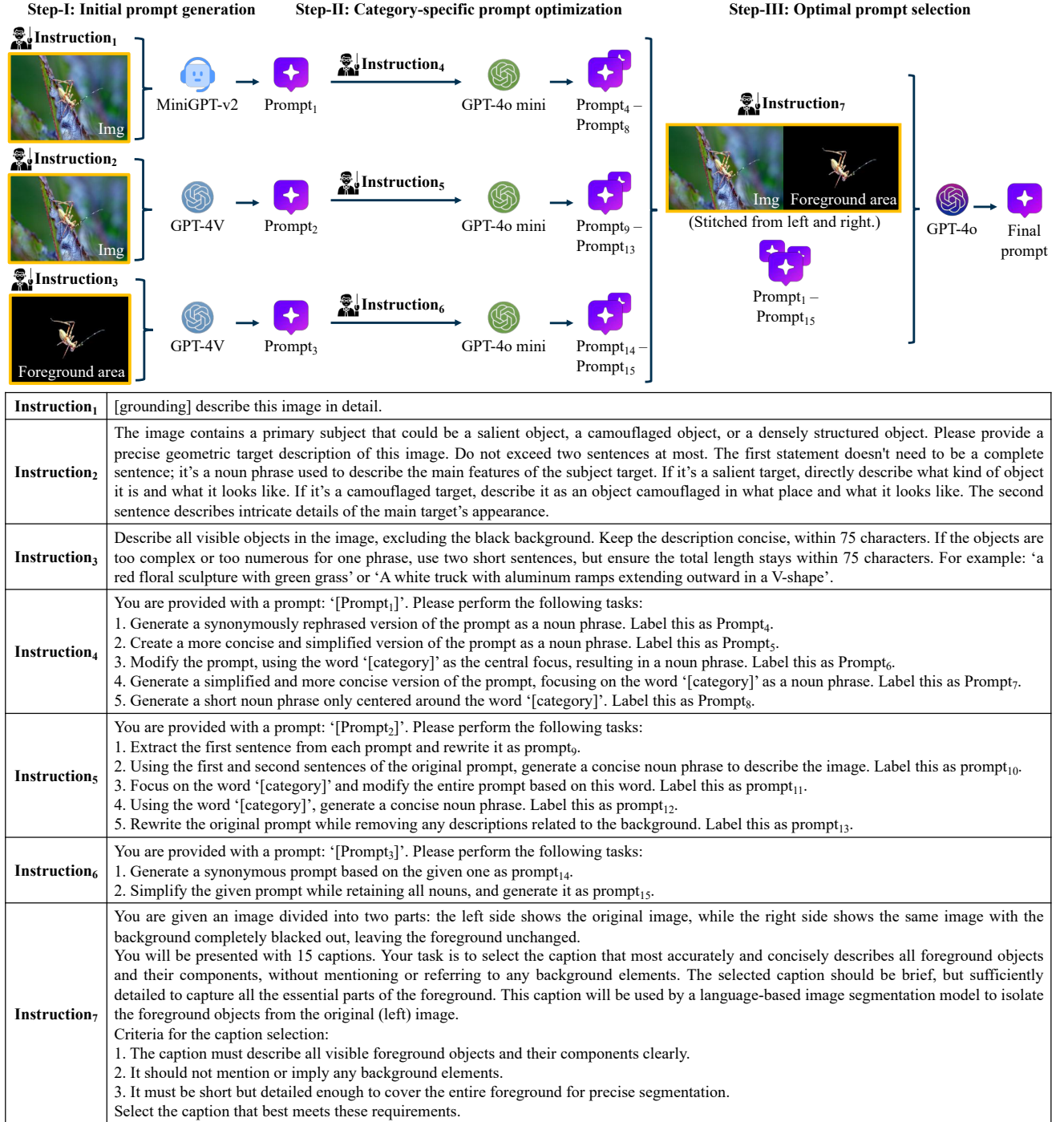


Figure 8. Annotation pipeline of language prompt.



---

**Algorithm 1** Window selection strategy (PyTorch-like pseudocode)

---

```
# src: source. In semi-automated pipeline:  $src = GTs$ ; in fully automated pipeline:  $src = \text{initial segmentation map } \hat{s}$ .
#  $(h_p, w_p)$ : window size, defaulting to (1024, 1024).
#  $K$ : maximum number of selection iterations, defaulting to 50.
#  $(\tau_1, \tau_2)$ : the thresholds for edge strength in Canny edge detection, defaulting to (50, 150).
#  $\tau_{co}$ : the pixel connectivity to be considered when finding connected components, defaulting to 8.
#  $(\tau_{\text{white-pixel}}, \tau_{\text{black-pixel}})$ : the threshold values used to classify pixels as white or black, defaulting to (250, 5).
#  $(\tau_{\text{white-region}}, \tau_{\text{black-region}})$ : the threshold values for the proportion of white and black regions, defaulting to (0.9, 0.95).

i = 0, P = [] # Initialize an iteration count and an empty list of window coordinates.

E = cv2.Canny(src,  $\tau_1$ ,  $\tau_2$ ) # Perform edge detection.
# Compute connected components. stats contains bounding box info and area, centroids contains the center points.
stats, centroids = cv2.connectedComponentsWithStats(E,  $\tau_{co}$ )
stats = stats[1:], centroids = centroids[1:] # Remove background components.
# Compute the maximum side length for each connected component.
side_lengths = max(stats[:, cv2.CC_STAT_WIDTH], stats[:, cv2.CC_STAT_HEIGHT])

# Iterate until there are no more components or maximum iterations are reached.
while len(side_lengths) > 0 and i < K:
    j = argmax(side_lengths),  $(x_c, y_c) = \text{centroids}[j]$  # Find the largest connected component.
    # Create a window centered at the center points of the selected connected component.
     $x_1 = \max(x_c - w_p/2, 0)$ ,  $y_1 = \max(y_c - h_p/2, 0)$ 
     $x_2 = \min(x_c + w_p/2, W)$ ,  $y_2 = \min(y_c + h_p/2, H)$ 
    R = src[y1:y2, x1:x2] # Extract the window region.
    # Compute the black and white pixel ratios.
    white_ratio = Count(R >  $\tau_{\text{white-pixel}}$ ) / Size(R)
    black_ratio = Count(R >  $\tau_{\text{black-pixel}}$ ) / Size(R)

    # If the proportion of white or black pixels exceeds the set threshold, ignore this window.
    if white_ratio >  $\tau_{\text{white-region}}$  or black_ratio >  $\tau_{\text{black-region}}$ :
        # Update the edge map by masking the current region.
        M = zeros_like(E) # Create a mask of the same size as the edge map.
        M[y1:y2, x1:x2] = E[y1:y2, x1:x2] # Set the mask region to the current window.
        E = BitwiseAnd(E, BitwiseNot(M)) # Update the edge map by removing the masked region.

        # Recompute connected components.
        stats, centroids = cv2.connectedComponentsWithStats(E,  $\tau_{co}$ )[1:]
        continue

    P = P.append(( $(x_1, y_1), (x_2, y_2)$ )) # Record the window coordinates.
    # Update the edge map by masking the current region.
    M = zeros_like(E)
    M[y1:y2, x1:x2] = E[y1:y2, x1:x2]
    E = BitwiseAnd(E, BitwiseNot(M))

    # Recompute connected components.
    stats, centroids = cv2.connectedComponentsWithStats(E,  $\tau_{co}$ )[1:]

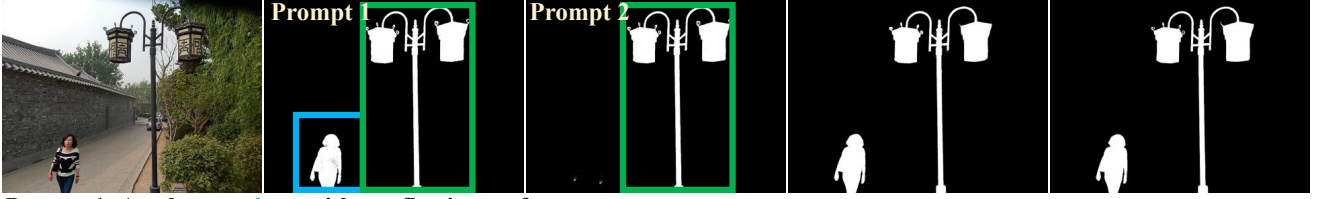
    i = i + 1 # Increment the iteration count.

return P # Return the list of window coordinates.
```

---

Prompt 1: A **lady** in a striped top and jeans under a **streetlamp**.

Prompt 2: A traditional Chinese lantern-style street **lamp**.



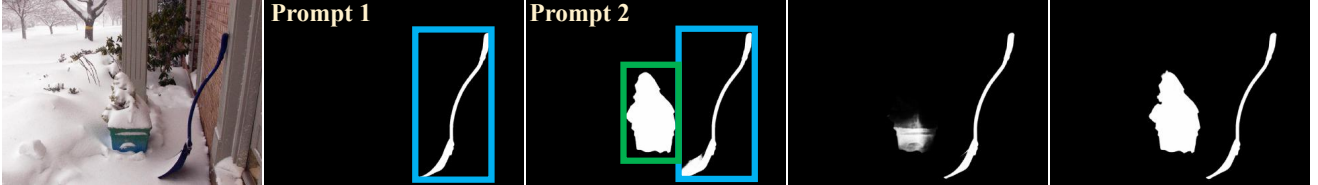
Prompt 1: An elegant **piano** with a reflective surface.

Prompt 2: A **saxophonist** and **pianist** with a grand **piano**.



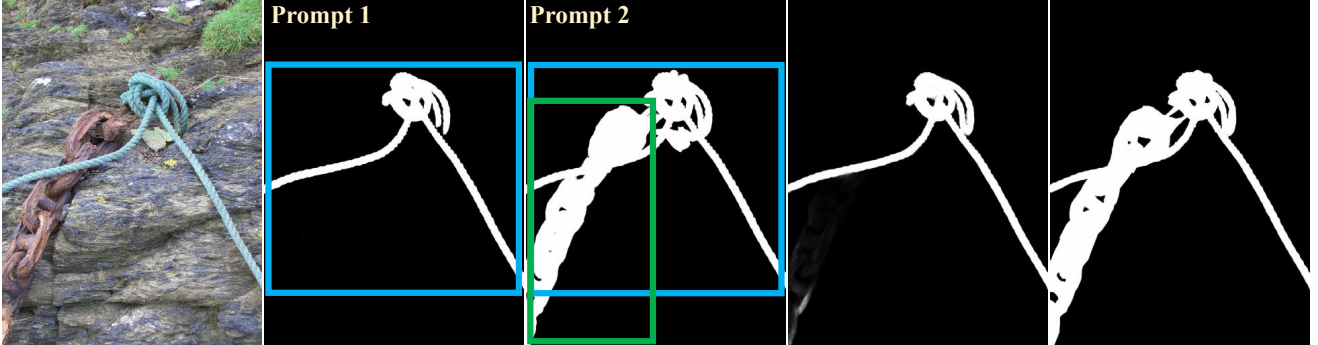
Prompt 1: A **spade** with a long handle and a shiny metal blade.

Prompt 2: Blue snow **shovel** with a curved handle and a black hand grip, near a **planter** with snow piled.



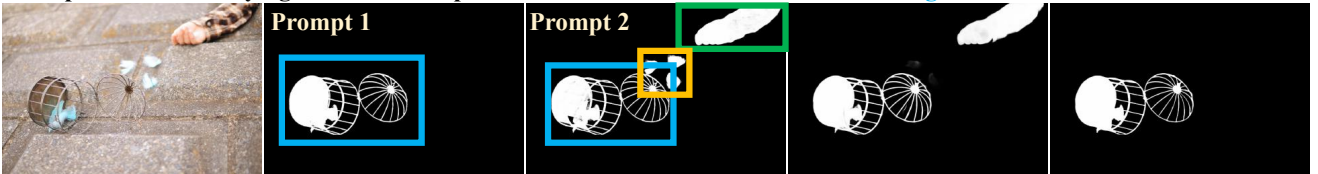
Prompt 1: A light blue **rope** knotted at the top, trailing downwards.

Prompt 2: A rusty **anchor** is tied to a **rope**.

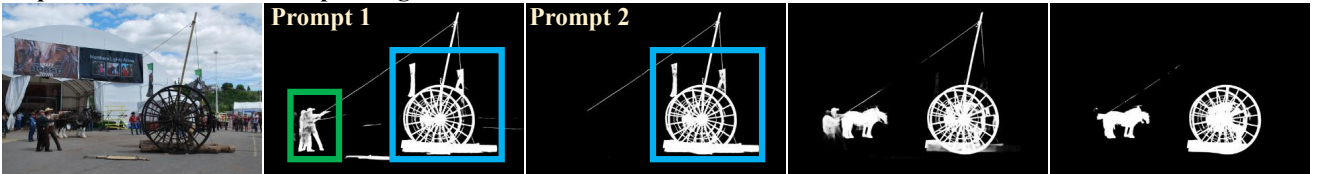


Prompt 1: A round **metal enclosure** with a removable top, showing a blue item inside.

Prompt 2: A **hand** is trying to touch a few pieces of azure **feathers** which is next to a **cage**.



Prompt 1: A grand **timber cart** with a sizable wheel and **two individuals** pulling. One individual directs, while the other helps with the exertion. Prompt 2: A grand **timber cart** with a sizable wheel.



(a) Image

(b) Ours-S

(d) MVANet [45]

(e) BiRefNet [49]

Figure 9. Qualitative predictions under different prompt guidance. Our method flexibly segments various foreground objects based on prompts, while other methods that cannot accept prompts yield only a fixed result.



Figure 10. Qualitative comparison of our model with four leading models. Local zoomed-in patches are evaluated with  $\mathcal{M}$  score for clarity.