Program

#Diffusion Model

```
import numpy as np
from PIL import Image
from ultralytics import YOLO
# Load YOLOv8 segmentation
model model =
YOLO("yolov8n-seg.pt") def
prompt_to_direction(prompt):
prompt = prompt.lower()
if "left" in prompt:
return (-2, 0)
elif "right" in prompt:
return (2, 0)
elif "up" in prompt or "forward" in prompt:
return (0, -2)
elif "down" in prompt or "backward" in prompt:
return (0, 2)
else:
return (2, 0) # default right
def
extract_main_object(image):
results = model(image)
masks = results[0].masks.data.cpu().numpy() if results[0].masks else []
if len(masks) == 0:
return None
# Use the largest mask (main object)
largest_mask = masks[np.argmax([m.sum() for m in masks])]
return largest_mask
object_mask = extract_main_object(image)
if object_mask is None:
print("⚠ No object detected. Generating basic shift animation.")
return [image for _ in range(num_frames)]
# Resize mask to match image (256x256)
from PIL import Image as PILImage
mask_resized = PILImage.fromarray((object_mask *
255).astype(np.uint8)).resize((256,
256))
```

```
object_mask = (np.array(mask_resized) > 128).astype(np.uint8)
img_np = np.array(image)
background = img_np.copy()
if remove_object:
# Remove the object by replacing it with white background
for c in range(3):
background[:, :, c][object_mask == 1] = 255
frames = [Image.fromarray(background.astype(np.uint8)) for _ in range(num_frames)]
return frames
object_region = img_np * object_mask[..., None]
frames = []
for i in range(num_frames):
dx, dy = direction
shifted_obj = np.roll(object_region, shift=i * dx, axis=1)
shifted_obj = np.roll(shifted_obj, shift=i * dy, axis=0)
frame_np = background.copy()
for c in range(3):
frame_np[:, :, c][object_mask == 1] = shifted_obj[:, :, c][object_mask == 1]
frame = Image.fromarray(frame_np.astype(np.uint8))
frames.append(frame)
return frames
def replace_object_with_dummy(image):
object_mask = extract_main_object(image)
if object_mask is None:
return image
from PIL import Image
# Resize mask to match image
mask_resized = Image.fromarray((object_mask * 255).astype(np.uint8)).resize((256,
256))
object_mask = (np.array(mask_resized) > 128).astype(np.uint8)
img_np = np.array(image)
# Replace the object with a red block (placeholder logic)
for c in range(3):
img_np[:, :, c][object_mask == 1] = [0,255,0][c] # red color
return Image.fromarray(img_np.astype(np.uint8))
```

#NoiseWarp Model

```
import numpy as np
def estimate_flow(image):
```

```
H, W = image.size
return np.ones((H, W, 2)) * 2 # move 2px right
def warp_noise(shape, flow):
H, W = shape
noise = np.random.randn(H, W, 3)
warped = np.roll(noise, shift=2, axis=1) # simulate simple warp
return warped
```

#Run Model

```
from utils import load_image, read_prompt, save_gif, save_image
from noise_wrap import estimate_flow, warp_noise
from diffusion import generate_video, replace_object_with_dummy
# Load input image and prompt
image = load_image("input/image.jpg")
prompt = read_prompt("input/prompt.txt")
# Ask user what they want to do
print("\nWhat would you like to do?")
print("1. Create animated GIF (move object)")
print("2. Remove object")
print("3. Replace object (with placeholder)")
choice = input("Enter 1, 2, or 3: ").strip()
# Common preprocessing
flow = estimate_flow(image)
warped_noise = warp_noise(image.size, flow)
# Settings
num_frames = 25
fps = 5
if choice == "1":
# Move object and create GIF
frames = generate_video(image, prompt, warped_noise, num_frames=num_frames,
remove_object=False)
save_gif(frames, "output/result.gif", fps=fps)
print("✅ GIF created at output/result.gif")
elif choice == "2":
# Remove object
frames = generate_video(image, prompt, warped_noise, num_frames=1,
remove_object=True)
save_image(frames[0], "output/removed_object.png")
print("✅ Object removed image saved at
```

output/removed_object.png") elif choice == "3":
# Replace object (with dummy image — real AI model could be used later)
replaced = replace_object_with_dummy(image)
save_image(replaced, "output/replaced_object.png")
print("✅ Object replaced image saved at
output/replaced_object.png") else:
print("❌ Invalid input. Please enter 1, 2, or 3.")

#Utils

from PIL import Image
import imageio
def load_image(path):
return Image.open(path).convert("RGB").resize((256, 256))
def read_prompt(path):
with open(path, "r") as f:
return f.read().strip()
def save_gif(frames, path, fps=5):
imageio.mimsave(path, frames, fps=fps)
def save_image(image, path):
image.save(path)