Realistic Object Blending With Shadow using Light Direction

Objective:

This project demonstrates an automated pipeline to realistically composite an object onto a background image. The process includes:

- Background removal
- Shadow-aware object blending
- Light direction estimation
- Depth-aware shadow rendering

Technologies & Libraries Used:

- **Python**
- **OpenCV** – Image processing
- **NumPy** – Array operations
- **PIL** – Image handling
- **rembg** – Background removal
- **PyTorch** – Depth estimation
- **Torchvision** – Model transforms
- **OS module** – File paths

Implementation Breakdown:

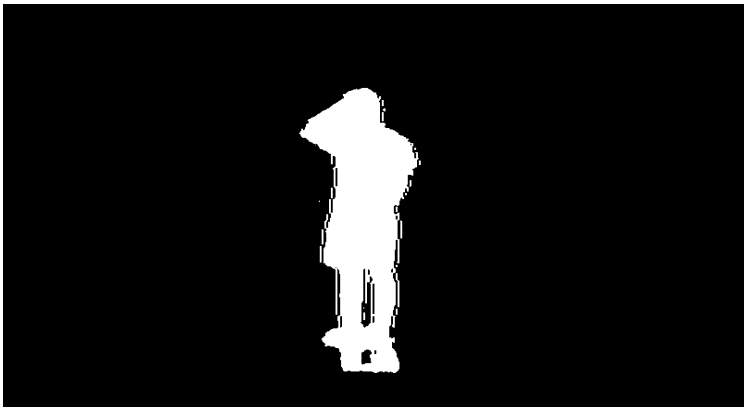
Task 1: Capturing and Preparing the Person's Image

Step 1: Capture a High-Quality Image

- A clear, frontal image of a person is used.
- The person is assumed to be captured under good lighting conditions.

Step 2: Remove the Background

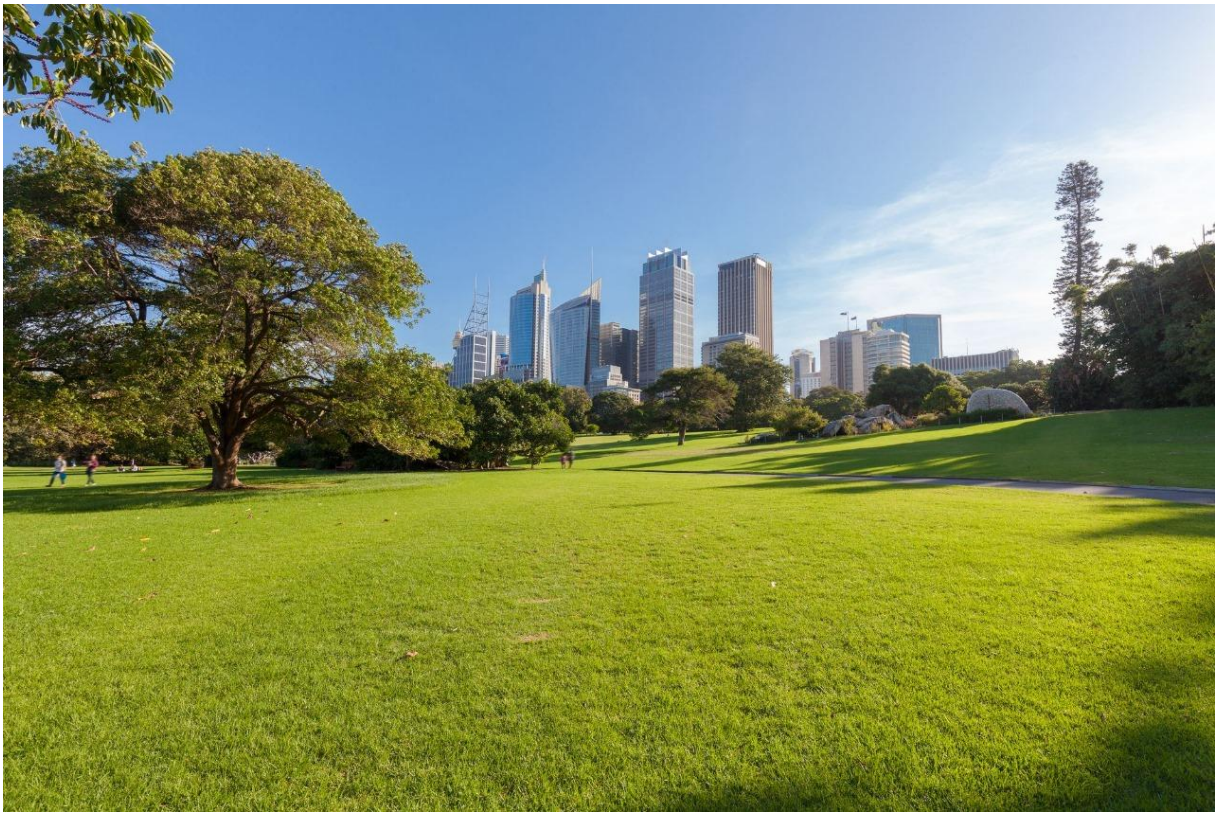
- Script: generate_mask.py
- Background is removed using the rembg library.
- A transparent PNG is generated (RGBA).
- A binary mask is created from the alpha channel.
- Gaussian blur and thresholding are applied to refine the mask.
- Below is the input image and the extracted object after background removal.



Task 2: Analyzing Lighting in the Background Image

Step 1: Detect Light Source

- Script: location.py
- Function: detect_brightest_direction(image)
- Converts background to grayscale.
- Applies Gaussian blur to smooth noise.
- Locates the brightest point in the image.
- Calculates direction from image center to this point.
- Returns a normalized light direction vector.
- No explicit shadow detection is performed — light is inferred from brightness.



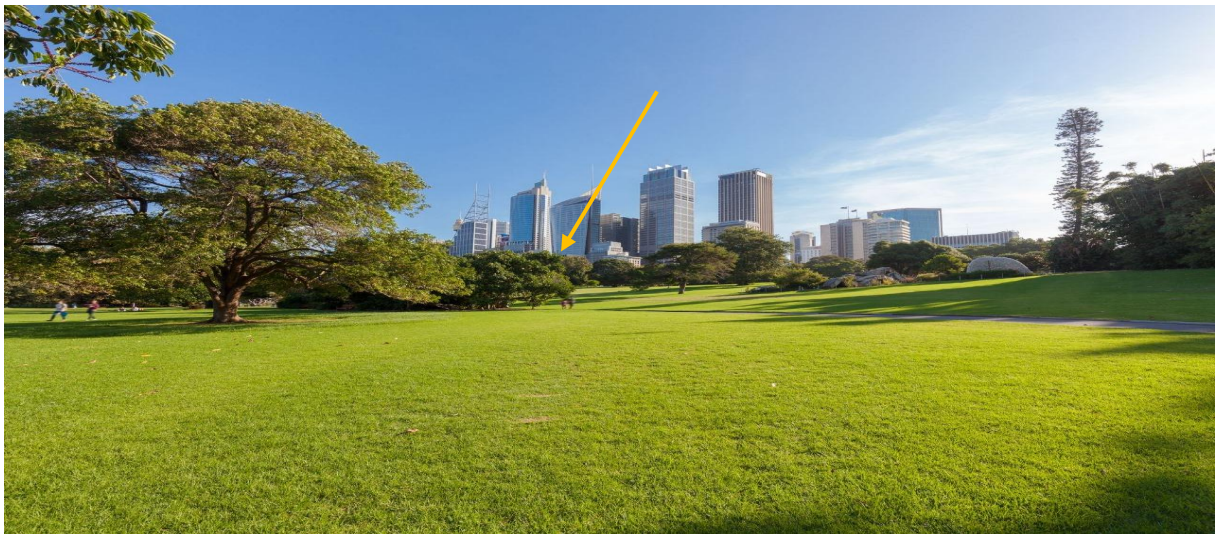
Task 3: Determining Light Direction

Step 1: Compute Light Direction for Outdoor Scenes

- From each valid contour, the furthest point from its centre is used to determine shadow orientation.
- Normalized vector (dx/length, dy/length) indicates direction **from** the object **towards** the light source.
- Fallback direction is (1, 0.2) if none found.
- Optionally visualized using an arrow in visualize_light_direction.

Step 2: Estimate Lighting for Indoor Scenes

- Not explicitly implemented. In this code, a common method is applied regardless of indoor or outdoor.
- Below is the image where the direction of the light is marked by an arrow in the Background image.



Task 4: Colouring and Blending

Step 1: Blending the Object into the Background image

- Shadows are created via affine translation, dilation, and large-radius Gaussian blur.
- Sharpening (sharpen_shadow) enhances the softness using unsharp masking.
- Object is resized and positioned in the lower center of the background.
- Background is darkened at the shadow location using cv2.subtract.
- Object is alpha-blended using the resized alpha mask.
- Below is the Blended object in the Background image without scaling or shadow generation.

Task 5: Generating the Final Output

- The final image (Final_park.jpg) contains the person realistically integrated into the background (uncle.png).
- The light direction and shadow are aligned.
- Color and tone blending is applied via smooth alpha compositing.
- The shadow is softened and partially transparent, mimicking real ambient lighting.
- Here, below is the Final output with the object Blended and Scaled and with shadow generated in the same direction of light.

Functions Overview:

detect_brightest_direction(bg_img)

- Estimates the light source direction in the background image.
- Converts image to grayscale, applies Gaussian blur, and locates the **brightest point**.
- Returns a normalized light direction vector from the center of the image.
- Used in place of traditional shadow detection for faster estimation.

• adjust_object_lighting(obj, obj_mask, light_direction)

- Simulates lighting on the object to match the background scene.
- Applies a directional gradient based on the detected light direction.
- Helps create a more **realistically shaded** object before blending.

• estimate_depth(img)

- Uses the **MiDaS model** via PyTorch to generate a depth map of the background.
- The depth map helps in shaping shadows more accurately with perspective.
- Normalizes the depth to a 0–1 scale.

• create_realistic_shadow(obj_mask, depth_map, light_direction)

- Generates a **soft, displaced shadow** based on:
  - Object mask
  - Scene depth
  - Light direction
    - Adds an elliptical ground contact shadow for better grounding.
    - Uses Gaussian blur for smoothness and transparency.

• enhanced_blend_object_shadow(bg, obj, obj_mask, light_direction)

- Main function that:
  - Places the object in the background
  - Adjusts its lighting
  - Generates and blends realistic shadows
    - Returns the final composited image.

Comparisons With Real Images

Conclusion:

This assignment demonstrates a pipeline that addresses real-world problems in AR and image compositing. The integration of lighting, shadow realism, and blending creates a visually coherent result. Enhancements like color correction and indoor lighting estimation can be added in future iterations.