Seamlessly Integrating a Person into a Scene

Objective:

The goal of this assignment is to seamlessly integrate a person into a background scene so that the result appears photorealistic. This involves capturing the person's image, removing its background, analysing lighting and shadows in the background image, estimating light direction, generating realistic shadows, and finally blending the person into the background scene.

Technologies & Libraries Used:

- **Python**
- **OpenCV (cv2)**: For image processing tasks
- **NumPy**: For efficient numerical and array operations
- **OS module**: For file path handling

Implementation Breakdown:

Task 1: Capturing and Preparing the Person's Image

Step 1: Capture a High-Quality Image

- A frontal image of a person is used.
- The person image is assumed to be already captured under a well-lit environment.

Step 2: Remove the Background

- The code handles RGBA images.
- If alpha channel exists, it is used as a binary mask to isolate the foreground (person).
- If not, the image is converted to grayscale, and thresholding is applied to create a binary mask.
- Below is the Input Image of a person and then removal of the background and the person is only left as an object.

Fig (1.1)-

Fig (1.2)-

Task 2: Analysing Shadows and Lighting of the Background Image

Step 1: Detect and Classify Shadows

- Function: detect_light_direction(bg_img)
- Converts background image to grayscale.
- Applies Gaussian blur to smooth noise.
- Uses thresholding to isolate dark regions (possible shadows).
- Morphological closing is used to clean up the binary mask.
- Contours are extracted, and their moments and extents are analysed to infer direction.
- Shadows aren't explicitly classified as "hard" or "soft," but this could be added using gradient analysis.

Fig (1.3)-

Task 3: Determining Light Direction

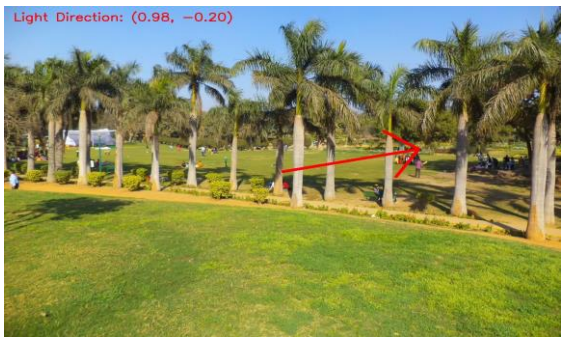
Step 1: Compute Light Direction for Outdoor Scenes

- 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.

Fig(1.4)-



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.

Fig (1.5)-

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.

Fig (1.6)-

Functions Overview:

- visualize_light_direction(bg_img, light_dir)

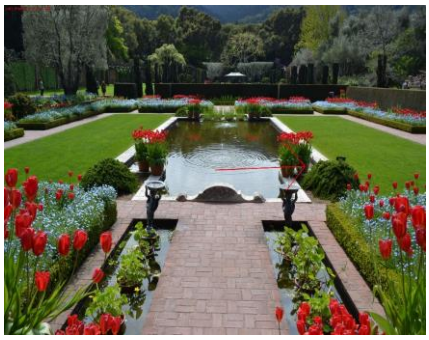
    - Adds arrow showing light direction.
- detect_light_direction(bg_img, visualize=False)
    - Computes light direction based on shadows in the background.
- sharpen_shadow(shadow_mask)
    - Enhances shadow clarity without losing softness.
- create_realistic_shadow(mask, light_dir)
    - Creates a smooth, offset, and slightly blurred shadow.
- blend_object(background_path, object_path, output_path)
    - Main pipeline combining background, object, light direction, and compositing.

Highlights and Innovations:

- Light direction detection is dynamic based on shadow analysis.
- Realistic soft shadows are generated based on lighting.
- Positioning is flexible and ensures the object fits within bounds.
- Alpha blending ensures soft edges and smooth integration.
- Code handles both RGBA and RGB inputs.
- Code can access well on different background images from different light directions as well below are some examples –

Fig (1.7)-

Fig(1.8)-



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.