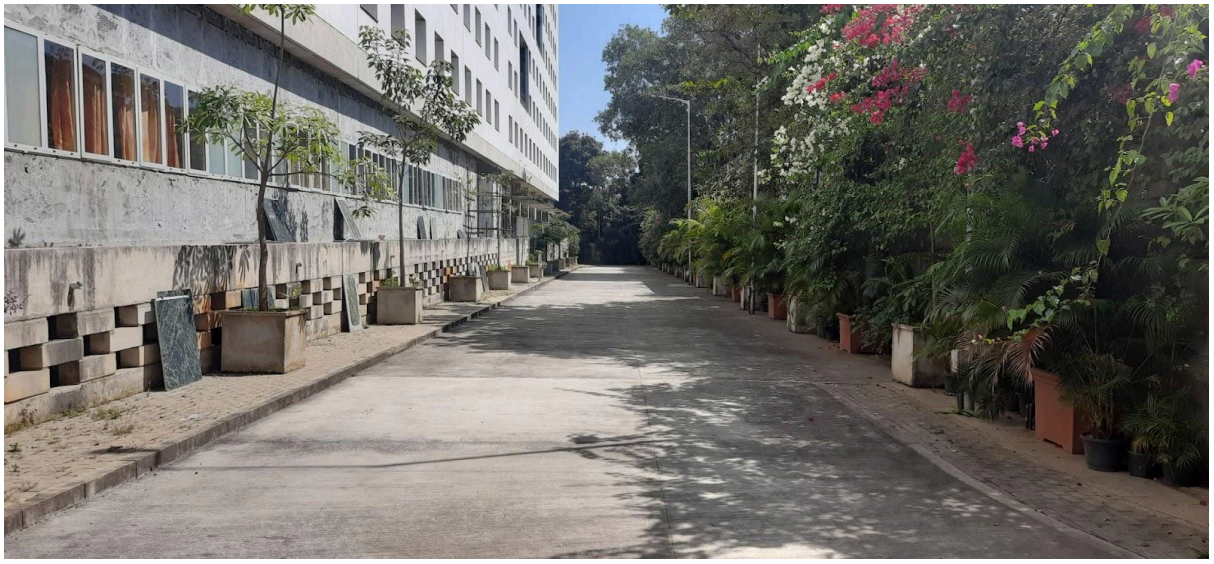
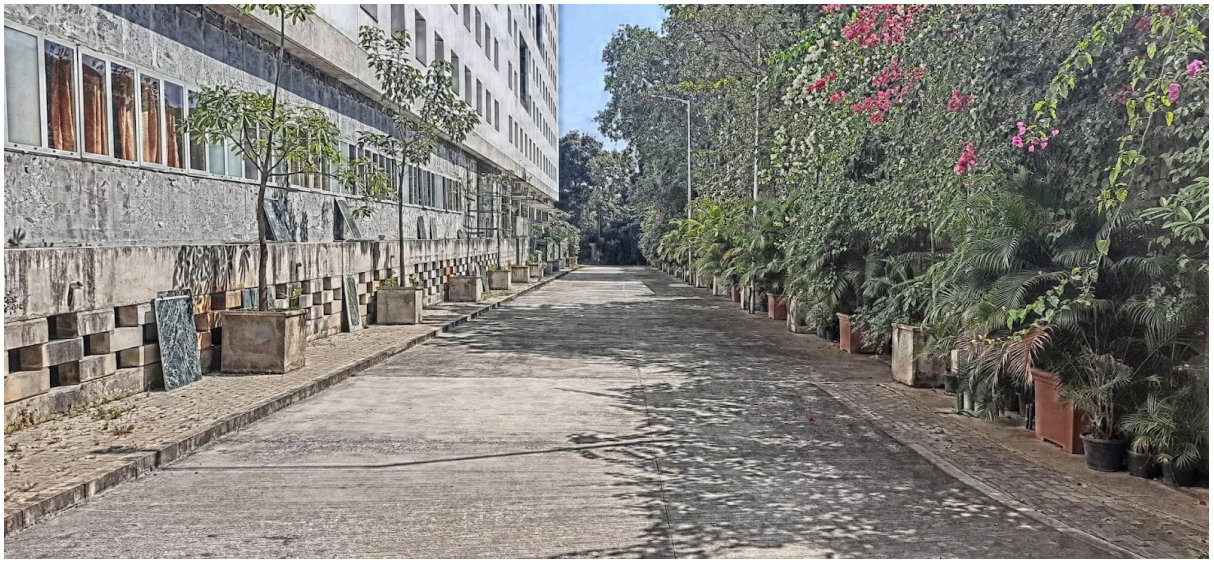
2. Shadow Removal

i) Remove image via LAB image

Aim was to remove shadows from the image.





In the above image we can see that the lightness is increased where shadows are lightened but not removed.

```
import cv2
import numpy as np

def remove_shadows(image):
   lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)

   l, a, b = cv2.split(lab)

   clahe = cv2.createCLAHE(clipLimit=1.0, tileGridSize=(150, 150))
   cl = clahe.apply(l)
```

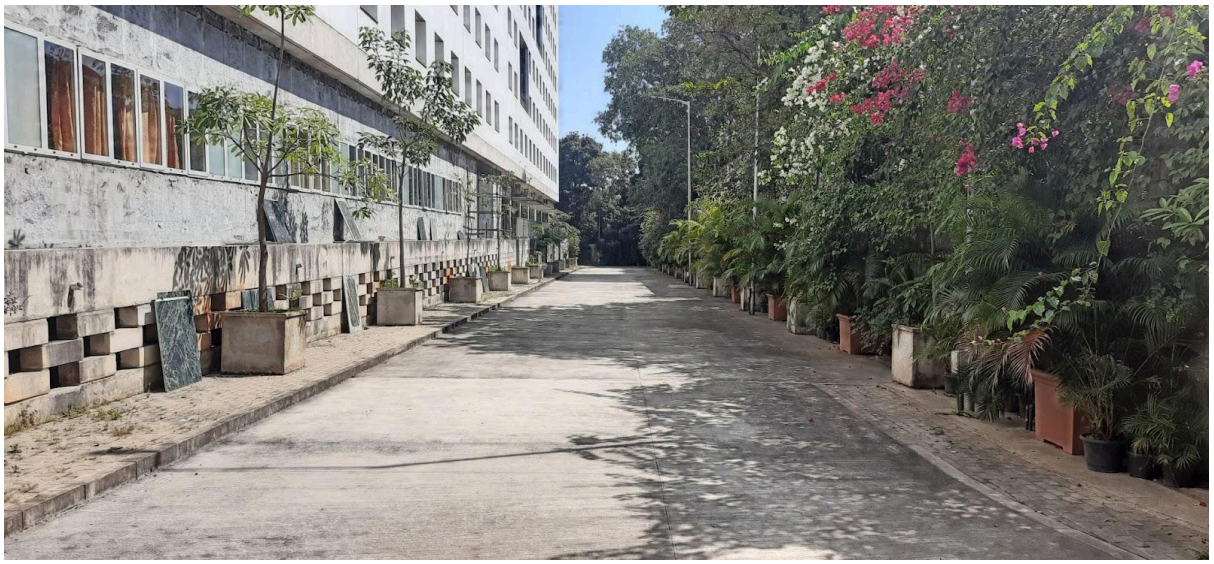
```
    limg = cv2.merge((cl, a, b))
    result = cv2.cvtColor(limg, cv2.COLOR_LAB2BGR)

    return result

input_path = '/content/ShadowRemoval1.jpg'
output_path = '/content/ShadowRemoval1_ans.jpg'

input_image = cv2.imread(input_path)

output_image = remove_shadows(input_image)

cv2.imwrite(output_path, output_image)
```

Firstly I convert RGB image into LAB format where LAB is a color space that separates the image into three channels: L (lightness), A (green to magenta), and B (blue to yellow).

Then I created CLAHE object with specified parameters. CLAHE is a variant of histogram equalization that adapts to local image regions, preventing over-amplification of noise in homogeneous areas. clipLimit controls the amount of contrast enhancement, and tileGridSize defines the size of the grid for histogram equalization. This also enhances the contrast of the lightness channel while preserving details in different parts of the image.

The modified L channel is merged back with the A and B channels to reconstruct the LAB image. The LAB image is converted back to the BGR color space to obtain the final result.

This method helps in reducing the impact of shadows and improving overall image quality.



Another Approach:

```
image_path='/content/ShadowRemoval2.jpg'
output_path='/content/ShadowRemoval2_ans.jpg'
img = cv2.imread(image_path)

lab_img = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
l_channel, a_channel, b_channel = cv2.split(lab_img)

mean_l = np.mean(l_channel)
mean_a = np.mean(a_channel)
mean_b = np.mean(b_channel)
std_l = np.std(l_channel)
```

```
if mean_a + mean_b <= 256:
 shadow_mask = l_channel <= (mean_l - std_l / 3)
else:
 shadow_mask = (l_channel <= mean_l) & (b_channel <= mean_b)

shadow_mask = shadow_mask.astype(np.uint8) * 255
cv2.imwrite(output_path, shadow_mask)
```

I converted BGR image to the LAB color space using cv2.cvtColor. LAB separates the image into three channels: L (lightness), A (green to magenta), and B (blue to yellow). The LAB image is then split into its three channels.

I calculated the mean and standard deviation of the L channel. These values will be used in subsequent steps to define conditions for identifying shadows. Then we check whether the sum of mean_a and mean_b is less than or equal to 256. Depending on this condition, it creates a shadow mask. If true, the mask is created based on the condition l_channel <= (mean_l - std_l / 3). If false, the mask is created based on the condition (l_channel <= mean_l) & (b_channel <= mean_b).

After this, I tried getting pixels where white part is there and then filled the image with neighbouring pixels but still image was not good and sketchy lines were noticed in the original image.

Updated Approach:

Firstly, I converted the input image to the HSV color space and created a binary shadow mask by thresholding within specified HSV bounds. The shadows are identified based on the low saturation and value components, allowing for effective segmentation of shadow regions. This approach in the HSV color space is advantageous as it separates color information, making it easier to distinguish shadows from other elements in the image.

Then main area is used to isolate the shadows specifically on the road. This region, outlined by vertices in the exact_part list, ensures that the shadow removal process is focused on areas of interest. Then I iterated through each pixel in the image, considering whether it belongs to the shadow mask or lies outside the defined road shadow region. For pixels identified as shadows, their color values are not changed and for pixels outside the road shadow region, a neighbour pixel value is used. This used averaging the color values of non-shadow neighbors to estimate almost the color of the shadowed pixel, contributing to a better appealing result.