Project 2 Part 1

Onkar Kher

UID: 120407062

Directory ID: okher

Mounting the current directory to google drive path where the video is stored.

```
from google.colab import drive
drive.mount('/content/drive/', force_remount=True)
```

```
Mounted at /content/drive/
```

Accessing the specific folder in which the video has been stored.

```
path_to_folder = "ENPM673/Project2/"
%cd /content/drive/My\ Drive/{path_to_folder}
```

```
/content/drive/My Drive/ENPM673/Project2
```

Video Processing piepline.

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow  # For displaying images in Colab

def variance_of_laplacian(gray_frame):    #Manually applying the laplacian kernel t
    laplacian_kernel = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])
    laplacian = cv2.filter2D(gray_frame, -1, laplacian_kernel)
    return np.var(laplacian)    #Returning the variance of each frame (low variance

def line_length(line):      #Defining a function to compute the length of a line sc
    x1, y1, x2, y2 = line[0]
    return np.sqrt((x2 - x1)**2 + (y2 - y1)**2)

def find_intersection(line1, line2):      #Defining a function to find the intersec
    # Calculating the determinants
    x1, y1, x2, y2 = line1[0]
    x3, y3, x4, y4 = line2[0]
    det = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4)
```

```
    det = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4)

    if det == 0:
        return None  # Lines are parallel, no intersection

    # Calculating the x, y intersection coordinates (corner coordinates)
    det_inv = 1 / det
    x = det_inv * ((x1*y2 - y1*x2) * (x3 - x4) - (x1 - x2) * (x3*y4 - y3*x4))
    y = det_inv * ((x1*y2 - y1*x2) * (y3 - y4) - (y1 - y2) * (x3*y4 - y3*x4))

    return (int(x), int(y))

video_path = 'Source/proj2_v2.mp4'   #Accessing the video file from inside folder
cap = cv2.VideoCapture(video_path)

sharp_frames_count = 0    #Initialising counts for sharp and blurry frames
blurry_frames_count = 0
threshold_sharpness = 53    #Setting the threshold for sharpness as 53 which satisf
threshold_white = 220       #Setting the threshold for detecting white objects.

# Parameters for Hough Line Transform
rho = 1
theta = np.pi / 180
threshold_hough = 50
min_line_length = 115
max_line_gap = 10

# Parameters for Harris corner detection
block_size = 2
ksize = 3
k = 0.04

# Define the codec and create VideoWriter object to save the output video
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Codec definition
output_path = 'Source/output_proj2_v2.mp4'  # Define a new path for the output vide
out = cv2.VideoWriter(output_path, fourcc, 10.0, (int(cap.get(3)), int(cap.get(4))))


# Processing loop
while True:
    ret, frame = cap.read()   #To extract frames from the video.
    if not ret:
        break

    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)    #Converting all the fra
    sharpness = variance_of_laplacian(gray_frame)     #Calculating the sharpness of

    #Conditonal to separate the sharp frames from the blurry frames
    if sharpness > threshold_sharpness:
        sharp_frames_count += 1
        _, white_regions = cv2.threshold(gray_frame, threshold_white, 255, cv2.THRE
```

```
        edges = cv2.Canny(white_regions, 150, 450)    #Using canny edge detector to

        lines = cv2.HoughLinesP(edges, rho, theta, threshold_hough, minLineLength=m

        if lines is not None:
            dominant_lines = [line for line in lines if line_length(line) > min_lir

            for line in dominant_lines:
                x1, y1, x2, y2 = line[0]
                cv2.line(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

            intersections = []
            for i, line1 in enumerate(dominant_lines):
                for line2 in dominant_lines[i+1:]:
                    intersect = find_intersection(line1, line2)
                    if intersect:
                        intersections.append(intersect)


            # Applying the Harris corner detection
            harris_corners = cv2.cornerHarris(np.float32(gray_frame), block_size, k
            harris_corners = cv2.dilate(harris_corners, None)


            #Thresholding to get the coordinates of the Harris corners
            corners = np.where(harris_corners > 0.01 * harris_corners.max())
            corners = list(zip(*corners[::-1]))  # Reversing to (x,y) and make a li

            # Verifying if Hough intersections are close to Harris corners
            for intersect in intersections:
              x, y = intersect
              if any(np.linalg.norm(np.array(intersect) - np.array(corner)) < 10 fo
                # This intersection is verified by Harris, mark it
                cv2.circle(frame, (x, y), 5, (0, 0, 255), -1)  # Mark verified corn


        out.write(frame)  # Write the processed frame to the output video

    else:
        blurry_frames_count += 1

cap.release()
out.release()
cv2.destroyAllWindows()
print(f"Total sharp (non-blurry) frames: {sharp_frames_count}")
print(f"Total number of blurry frames skipped: {blurry_frames_count}")
print("Output video has been generated in the location where the input video is sto
```

```
Total sharp (non-blurry) frames: 199
Total number of blurry frames skipped: 187
```

```
Total number of blurry frames skipped: 137
Output video has been generated in the location where the input video is store
```