# Logo Detection in MP4 Video

# Using Histogram Backprojection Algorithm (Step by step)

import cv2

import numpy as np

def detect\_logo(video\_path, logo\_path, threshold=10):

    # Step 1: Convert images into HSV and calculate  histograms

    cap = cv2.VideoCapture(video\_path)

    roi = cv2.imread(logo\_path)

    hsvr = cv2.cvtColor(roi, cv2.COLOR\_BGR2HSV)

    M = cv2.calcHist([hsvr], [0, 1], None, [180, 256], [0, 180, 0, 256])

    while True:

        # Read a frame from the video

        ret, frame = cap.read()

        if not ret:

            break

        # Preprocess frame (noise reduction)

        blur = cv2.GaussianBlur(frame, (5, 5), 0)

        # Convert preprocessed frame to HSV and calculate histogram

        hsvt = cv2.cvtColor(blur, cv2.COLOR\_BGR2HSV)

        I = cv2.calcHist([hsvt], [0, 1], None, [180, 256], [0, 180, 0, 256])

        # Step 2: Find the ratio R

        R = M / (I + 1)

        # Step 3: Backproject R

        h, s, v = cv2.split(hsvt)

        B = R[h.ravel(), s.ravel()]

        B = np.minimum(B, 1)

        B = B.reshape(hsvt.shape[:2])

        # Step 4: Fine-tune B

        disc = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

        cv2.filter2D(B, -1, disc, B)

        B = np.uint8(B)

        cv2.normalize(B, B, 0, 255, cv2.NORM\_MINMAX)

        # Step 5: Thresholding

# Logo Detection in MP4 Video

# Using Histogram Backprojection Algorithm (Step by step)

import cv2

import numpy as np

def detect\_logo(video\_path, logo\_path, threshold=10):

    # Step 1: Convert images into HSV and calculate  histograms

    cap = cv2.VideoCapture(video\_path)

    roi = cv2.imread(logo\_path)

    hsvr = cv2.cvtColor(roi, cv2.COLOR\_BGR2HSV)

    M = cv2.calcHist([hsvr], [0, 1], None, [180, 256], [0, 180, 0, 256])

    while True:

        # Read a frame from the video

        ret, frame = cap.read()

        if not ret:

            break

        # Preprocess frame (noise reduction)

        blur = cv2.GaussianBlur(frame, (5, 5), 0)

        # Convert preprocessed frame to HSV and calculate histogram

        hsvt = cv2.cvtColor(blur, cv2.COLOR\_BGR2HSV)

        I = cv2.calcHist([hsvt], [0, 1], None, [180, 256], [0, 180, 0, 256])

        # Step 2: Find the ratio R

        R = M / (I + 1)

        # Step 3: Backproject R

        h, s, v = cv2.split(hsvt)

        B = R[h.ravel(), s.ravel()]

        B = np.minimum(B, 1)

        B = B.reshape(hsvt.shape[:2])

        # Step 4: Fine-tune B

        disc = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

        cv2.filter2D(B, -1, disc, B)

        B = np.uint8(B)

        cv2.normalize(B, B, 0, 255, cv2.NORM\_MINMAX)

        # Step 5: Thresholding

        ret, thresh = cv2.threshold(B, threshold, 255, cv2.THRESH\_BINARY)

        # Find contours

        contours, \_ = cv2.findContours(

            thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE

        )

        largest\_contour = None

        max\_area = 0

        for cnt in contours:

            area = cv2.contourArea(cnt)

            if area > max\_area:

                largest\_contour = cnt

                max\_area = area

        # Draw rectangle around the logo

        if largest\_contour is not None:

            x, y, w, h = cv2.boundingRect(largest\_contour)

            cv2.rectangle(

                frame, (x, y), (x + w, y + h), (0, 255, 0), 2

            )  # Green rectangle

        # Display the result

        cv2.imshow("Logo Detection (Improved)", frame)

        # Exit on 'q' key press

        if cv2.waitKey(1) & 0xFF == ord("q"):

            break

    # Release resources

    cap.release()

    cv2.destroyAllWindows()

video\_path = "output.mp4"

logo\_path = "logo\_detection\_img.png"

detect\_logo(video\_path, logo\_path)

