21MIA1025 - Abhineswari M - CSE4076

```
In [1]:
import cv2
import matplotlib.pyplot as plt
import os
import numpy as np
```

Loading the video file and extracting individual frames from the video

```
In [2]:
def extract_and_display_frames(video_path, output_folder, display_size=(640, 480)):
    # Create the output folder if it doesn't exist
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Open the video file
    cap = cv2.VideoCapture(video_path)
    frame_count = 0

    while True:
        # Read the next frame
        ret, frame = cap.read()
        if not ret:
            break

        # Resize the frame
        resized_frame = cv2.resize(frame, display_size)

        # Save the frame as an image file
        frame_filename = os.path.join(output_folder, f"frame_{frame_count:04d}.jpg")
        cv2.imwrite(frame_filename, resized_frame)

        # Display the frame
        cv2.imshow('Frame', resized_frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

        frame_count += 1

    # Release the video capture object and close all OpenCV windows
    cap.release()
    cv2.destroyAllWindows()
    print(f"Extracted and displayed {frame_count} frames to {output_folder}")

# Example usage
video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"
output_folder = 'output_frames'
extract_and_display_frames(video_path, output_folder)
```

```
Extracted and displayed 883 frames to output_frames
```

Applying Guassian Noise Reduction

```
In [3]:
from scipy.ndimage import convolve

def gaussian_kernel(size, sigma=1):
    """Generates a Gaussian kernel."""
    kernel = np.fromfunction(
        lambda x, y: (1/(2*np.pi*sigma**2)) * np.exp(-((x**2 + y**2) / (2*sigma**2))),
        (size, size)
    )
    return kernel / np.sum(kernel)

def apply_gaussian_blur(image, kernel_size=5, sigma=1):
    """Applies Gaussian blur to an image."""
    kernel = gaussian_kernel(kernel_size, sigma)

    if len(image.shape) == 2:  # Grayscale image
        blurred_image = convolve(image, kernel)
    else:  # Color image
        blurred_image = np.zeros_like(image)
        for i in range(3):  # Apply to each color channel
            blurred_image[:, :, i] = convolve(image[:, :, i], kernel)

    return blurred_image
```

```python
def apply_gaussian_noise_reduction(input_folder, output_folder, kernel_size=5, sigma=1.0):
    # Create the output folder if it doesn't exist
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Check if the input folder exists
    if not os.path.exists(input_folder):
        print(f"Input folder '{input_folder}' does not exist.")
        return

    # List all files in the input folder
    frame_files = sorted([f for f in os.listdir(input_folder) if f.endswith('.jpg')])

    for frame_file in frame_files:
        # Read the frame
        frame_path = os.path.join(input_folder, frame_file)
        frame = cv2.imread(frame_path)

        # Apply custom Gaussian noise reduction
        denoised_frame = apply_gaussian_blur(frame, kernel_size, sigma)

        # Save the denoised frame
        denoised_frame_path = os.path.join(output_folder, frame_file)
        cv2.imwrite(denoised_frame_path, denoised_frame)

    print(f"Applied Gaussian noise reduction to {len(frame_files)} frames and saved to {output_folder}")

input_folder = 'output_frames'
output_folder = 'denoised_frames'
apply_gaussian_noise_reduction(input_folder, output_folder)
```

```
Applied Gaussian noise reduction to 883 frames and saved to denoised_frames
```

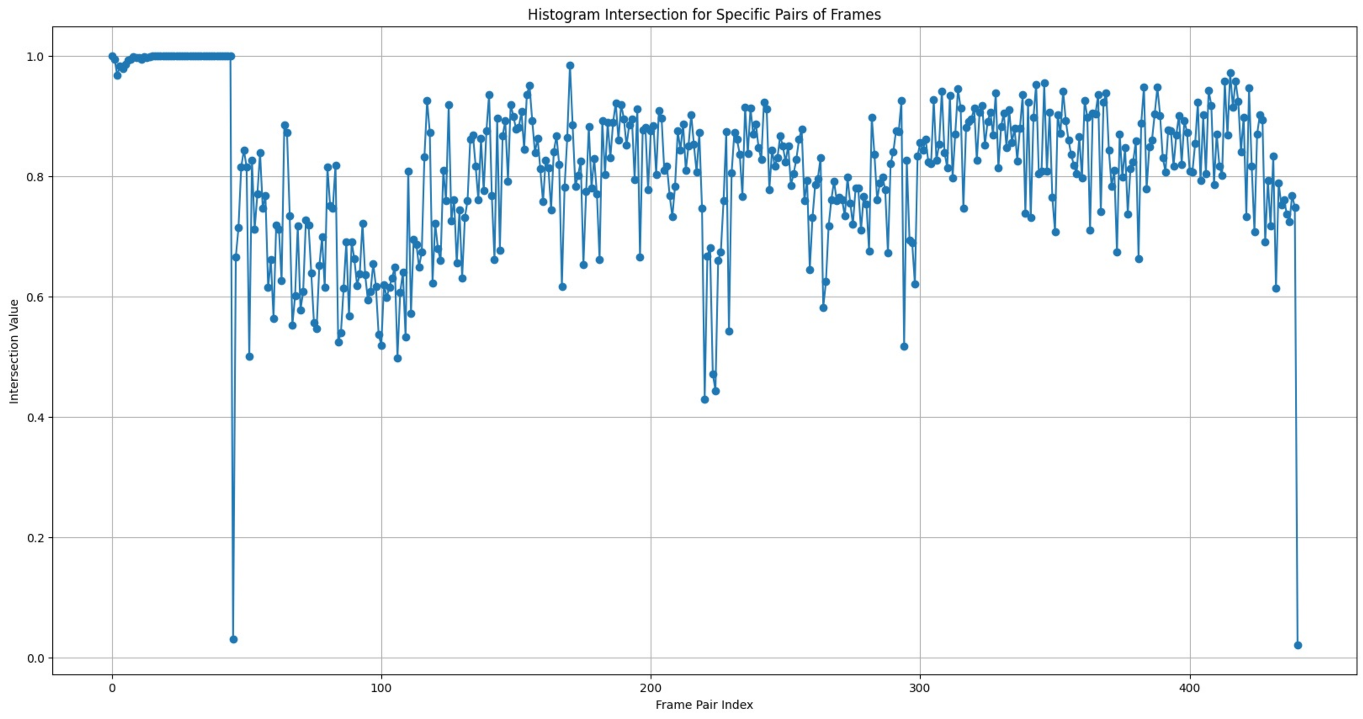
Calculating the Histogram Equalization to each channel of the HSV image along with the histogram intersection score

In [4]:
```python
def histogram_equalization(image):
    equalized_image = np.zeros_like(image)
    for i in range(3):  # Apply to each channel
        equalized_image[:, :, i] = cv2.equalizeHist(image[:, :, i])
    return equalized_image

def calculate_histogram_intersection(hist1, hist2):
    minima = np.minimum(hist1, hist2)
    intersection = np.true_divide(np.sum(minima), np.sum(hist2))
    return intersection

def apply_histogram_equalization_and_intersection(input_folder, output_folder):
    # Create the output folder if it doesn't exist
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # List all files in the input folder
    frame_files = sorted([f for f in os.listdir(input_folder) if f.endswith('.jpg')])

    histograms = []
    intersections = []

    for frame_file in frame_files:
        # Read the frame
        frame_path = os.path.join(input_folder, frame_file)
        frame = cv2.imread(frame_path)

        # Apply histogram equalization
        equalized_frame = histogram_equalization(frame)

        # Save the equalized frame
        equalized_frame_path = os.path.join(output_folder, frame_file)
        cv2.imwrite(equalized_frame_path, equalized_frame)

        # Calculate histogram for the equalized frame
        hist = cv2.calcHist([equalized_frame], [0], None, [256], [0, 256])
        histograms.append(hist)

    # Calculate histogram intersections for specific pairs of frames
    for i in range(0, len(histograms) - 1, 2):
        intersection = calculate_histogram_intersection(histograms[i], histograms[i + 1])
        intersections.append(intersection)
        print(f"Histogram intersection for frame {i} with frame {i + 1}: {intersection}")
    print(f"Applied histogram equalization to {len(frame_files)} frames and saved to {output_folder}")

    return intersections

input_folder = 'output_frames'
equalized_output_folder = 'equalized_frames'
intersections = apply_histogram_equalization_and_intersection(input_folder, equalized_output_folder)
```

```
Histogram intersection for frame 0 with frame 1: 0.9994401335716248
Histogram intersection for frame 2 with frame 3: 0.994615912437439
Histogram intersection for frame 4 with frame 5: 0.96749347448349
Histogram intersection for frame 6 with frame 7: 0.9830077886581421
Histogram intersection for frame 8 with frame 9: 0.9784603118896484
Histogram intersection for frame 10 with frame 11: 0.9850976467132568
Histogram intersection for frame 12 with frame 13: 0.99271160364151
Histogram intersection for frame 14 with frame 15: 0.9945540428161621
Histogram intersection for frame 16 with frame 17: 0.9983952045440674
Histogram intersection for frame 18 with frame 19: 0.99727863073349
Histogram intersection for frame 20 with frame 21: 0.9969921708106995
Histogram intersection for frame 22 with frame 23: 0.9934732913970947
Histogram intersection for frame 24 with frame 25: 0.9980859160423279
Histogram intersection for frame 26 with frame 27: 0.9964550733566284
Histogram intersection for frame 28 with frame 29: 0.9983658790588379
Histogram intersection for frame 30 with frame 31: 1.0
Histogram intersection for frame 32 with frame 33: 0.9996809959411621
Histogram intersection for frame 34 with frame 35: 1.0
Histogram intersection for frame 36 with frame 37: 1.0
Histogram intersection for frame 38 with frame 39: 1.0
Histogram intersection for frame 40 with frame 41: 1.0
Histogram intersection for frame 42 with frame 43: 1.0
Histogram intersection for frame 44 with frame 45: 1.0
Histogram intersection for frame 46 with frame 47: 1.0
Histogram intersection for frame 48 with frame 49: 1.0
Histogram intersection for frame 50 with frame 51: 1.0
Histogram intersection for frame 52 with frame 53: 1.0
Histogram intersection for frame 54 with frame 55: 1.0
Histogram intersection for frame 56 with frame 57: 1.0
Histogram intersection for frame 58 with frame 59: 1.0
Histogram intersection for frame 60 with frame 61: 1.0
Histogram intersection for frame 62 with frame 63: 1.0
Histogram intersection for frame 64 with frame 65: 1.0
Histogram intersection for frame 66 with frame 67: 1.0
Histogram intersection for frame 68 with frame 69: 1.0
Histogram intersection for frame 70 with frame 71: 1.0
Histogram intersection for frame 72 with frame 73: 1.0
Histogram intersection for frame 74 with frame 75: 1.0
Histogram intersection for frame 76 with frame 77: 1.0
Histogram intersection for frame 78 with frame 79: 1.0
Histogram intersection for frame 80 with frame 81: 1.0
Histogram intersection for frame 82 with frame 83: 1.0
Histogram intersection for frame 84 with frame 85: 1.0
Histogram intersection for frame 86 with frame 87: 1.0
Histogram intersection for frame 88 with frame 89: 1.0
Histogram intersection for frame 90 with frame 91: 0.030439453199505806
Histogram intersection for frame 92 with frame 93: 0.6657552123069763
Histogram intersection for frame 94 with frame 95: 0.7138965129852295
Histogram intersection for frame 96 with frame 97: 0.8146582245826721
Histogram intersection for frame 98 with frame 99: 0.8430501222610474
Histogram intersection for frame 100 with frame 101: 0.8150618672370911
Histogram intersection for frame 102 with frame 103: 0.4998307228088379
Histogram intersection for frame 104 with frame 105: 0.8262890577316284
Histogram intersection for frame 106 with frame 107: 0.7117382884025574
Histogram intersection for frame 108 with frame 109: 0.7695345282554626
Histogram intersection for frame 110 with frame 111: 0.8387858271598816
Histogram intersection for frame 112 with frame 113: 0.7458561062812805
Histogram intersection for frame 114 with frame 115: 0.7669759392738342
Histogram intersection for frame 116 with frame 117: 0.6143489480018616
Histogram intersection for frame 118 with frame 119: 0.6610254049301147
Histogram intersection for frame 120 with frame 121: 0.5634765625
Histogram intersection for frame 122 with frame 123: 0.7190592288970947
Histogram intersection for frame 124 with frame 125: 0.7113541960716248
Histogram intersection for frame 126 with frame 127: 0.6255598664283752
Histogram intersection for frame 128 with frame 129: 0.8853157758712769
Histogram intersection for frame 130 with frame 131: 0.8727213740348816
Histogram intersection for frame 132 with frame 133: 0.7345019578933716
Histogram intersection for frame 134 with frame 135: 0.5523014068603516
Histogram intersection for frame 136 with frame 137: 0.6013150811195374
Histogram intersection for frame 138 with frame 139: 0.7166829705238342
Histogram intersection for frame 140 with frame 141: 0.5770279765129089
Histogram intersection for frame 142 with frame 143: 0.60794597864151
Histogram intersection for frame 144 with frame 145: 0.7266731858253479
Histogram intersection for frame 146 with frame 147: 0.7179133892059326
Histogram intersection for frame 148 with frame 149: 0.6386165618896484
Histogram intersection for frame 150 with frame 151: 0.5565527081489563
Histogram intersection for frame 152 with frame 153: 0.5470019578933716
Histogram intersection for frame 154 with frame 155: 0.6506184935569763
Histogram intersection for frame 156 with frame 157: 0.6984310150146484
Histogram intersection for frame 158 with frame 159: 0.6152604222297668
Histogram intersection for frame 160 with frame 161: 0.8152180910110474
Histogram intersection for frame 162 with frame 163: 0.7509732842445374
Histogram intersection for frame 164 with frame 165: 0.7460872530937195
Histogram intersection for frame 166 with frame 167: 0.8180533647537231
Histogram intersection for frame 168 with frame 169: 0.5245410203933716
Histogram intersection for frame 170 with frame 171: 0.538867175579071
Histogram intersection for frame 172 with frame 173: 0.6129785180091858
Histogram intersection for frame 174 with frame 175: 0.6897981762886047
Histogram intersection for frame 176 with frame 177: 0.5667871236801147
Histogram intersection for frame 178 with frame 179: 0.6899935007095337
Histogram intersection for frame 180 with frame 181: 0.6618717312812805
Histogram intersection for frame 182 with frame 183: 0.6182649731636047
Histogram intersection for frame 184 with frame 185: 0.6372233033180237
Histogram intersection for frame 186 with frame 187: 0.7213932275772095
Histogram intersection for frame 188 with frame 189: 0.6362630128860474
Histogram intersection for frame 190 with frame 191: 0.5940852761268616
Histogram intersection for frame 192 with frame 193: 0.6080761551856995
Histogram intersection for frame 194 with frame 195: 0.6543554663658142
Histogram intersection for frame 196 with frame 197: 0.6162728071212769
Histogram intersection for frame 198 with frame 199: 0.5363313555717468
Histogram intersection for frame 200 with frame 201: 0.5189746022224426
Histogram intersection for frame 202 with frame 203: 0.619544267654419
Histogram intersection for frame 204 with frame 205: 0.5979264378547668
Histogram intersection for frame 206 with frame 207: 0.6153352856636047
Histogram intersection for frame 208 with frame 209: 0.6304101347923279
Histogram intersection for frame 210 with frame 211: 0.6489355564117432
Histogram intersection for frame 212 with frame 213: 0.4980013072490692
Histogram intersection for frame 214 with frame 215: 0.6066113114356995
Histogram intersection for frame 216 with frame 217: 0.6394693851470947
Histogram intersection for frame 218 with frame 219: 0.5320931077003479
Histogram intersection for frame 220 with frame 221: 0.8085156083106995
Histogram intersection for frame 222 with frame 223: 0.5718066692352295
Histogram intersection for frame 224 with frame 225: 0.69486004114151
Histogram intersection for frame 226 with frame 227: 0.6866699457168579
Histogram intersection for frame 228 with frame 229: 0.6484765410423279
Histogram intersection for frame 230 with frame 231: 0.6738248467445374
Histogram intersection for frame 232 with frame 233: 0.8317122459411621
Histogram intersection for frame 234 with frame 235: 0.9251432418823242
Histogram intersection for frame 236 with frame 237: 0.8717448115348816
Histogram intersection for frame 238 with frame 239: 0.622265636920929
Histogram intersection for frame 240 with frame 241: 0.7212239503860474
Histogram intersection for frame 242 with frame 243: 0.678697943687439
Histogram intersection for frame 244 with frame 245: 0.6592415571212769
Histogram intersection for frame 246 with frame 247: 0.8093457221984863
Histogram intersection for frame 248 with frame 249: 0.759361982345581
Histogram intersection for frame 250 with frame 251: 0.9180273413658142
Histogram intersection for frame 252 with frame 253: 0.7258626222610474
Histogram intersection for frame 254 with frame 255: 0.7608203291893005
Histogram intersection for frame 256 with frame 257: 0.6548827886581421
Histogram intersection for frame 258 with frame 259: 0.7438867092132568
Histogram intersection for frame 260 with frame 261: 0.6309375166893005
Histogram intersection for frame 262 with frame 263: 0.7306868433952332
Histogram intersection for frame 264 with frame 265: 0.7588639259338379
Histogram intersection for frame 266 with frame 267: 0.8611230254173279
Histogram intersection for frame 268 with frame 269: 0.8682063817977905
Histogram intersection for frame 270 with frame 271: 0.8161914348602295
Histogram intersection for frame 272 with frame 273: 0.7608463764190674
Histogram intersection for frame 274 with frame 275: 0.8620052337646484
Histogram intersection for frame 276 with frame 277: 0.7752962112426758
Histogram intersection for frame 278 with frame 279: 0.8746061325073242
Histogram intersection for frame 280 with frame 281: 0.9357128739356995
Histogram intersection for frame 282 with frame 283: 0.7672883868217468
Histogram intersection for frame 284 with frame 285: 0.6617610454559326
Histogram intersection for frame 286 with frame 287: 0.8957454562187195
Histogram intersection for frame 288 with frame 289: 0.6767740845680237
Histogram intersection for frame 290 with frame 291: 0.8663508892059326
Histogram intersection for frame 292 with frame 293: 0.8912630081176758
Histogram intersection for frame 294 with frame 295: 0.7911132574081421
Histogram intersection for frame 296 with frame 297: 0.9186555743217468
Histogram intersection for frame 298 with frame 299: 0.8995475172996521
Histogram intersection for frame 300 with frame 301: 0.8785058856010437
Histogram intersection for frame 302 with frame 303: 0.8800944089889526
Histogram intersection for frame 304 with frame 305: 0.9066699147224426
Histogram intersection for frame 306 with frame 307: 0.8438150882720947
Histogram intersection for frame 308 with frame 309: 0.9352408647537231
Histogram intersection for frame 310 with frame 311: 0.9506087303161621
Histogram intersection for frame 312 with frame 313: 0.8919270634651184
Histogram intersection for frame 314 with frame 315: 0.8393489718437195
Histogram intersection for frame 316 with frame 317: 0.8624185919761658
Histogram intersection for frame 318 with frame 319: 0.8114811182022095
Histogram intersection for frame 320 with frame 321: 0.7571874856948853
Histogram intersection for frame 322 with frame 323: 0.826819658279419
Histogram intersection for frame 324 with frame 325: 0.8131282329559326
Histogram intersection for frame 326 with frame 327: 0.7441373467445374
Histogram intersection for frame 328 with frame 329: 0.840319037437439
Histogram intersection for frame 330 with frame 331: 0.8671289086341858
Histogram intersection for frame 332 with frame 333: 0.8188509345054626
Histogram intersection for frame 334 with frame 335: 0.6161230206489563
Histogram intersection for frame 336 with frame 337: 0.7816601395606995
Histogram intersection for frame 338 with frame 339: 0.8633886575698853
Histogram intersection for frame 340 with frame 341: 0.9848567843437195
Histogram intersection for frame 342 with frame 343: 0.8848567605018616
Histogram intersection for frame 344 with frame 345: 0.7834244966506958
Histogram intersection for frame 346 with frame 347: 0.8004785180091858
Histogram intersection for frame 348 with frame 349: 0.824378252029419
Histogram intersection for frame 350 with frame 351: 0.6523047089576721
Histogram intersection for frame 352 with frame 353: 0.7738476395606995
Histogram intersection for frame 354 with frame 355: 0.8814811110496521
Histogram intersection for frame 356 with frame 357: 0.7804557085037231
Histogram intersection for frame 358 with frame 359: 0.8291178345680237
Histogram intersection for frame 360 with frame 361: 0.7697721123695374
Histogram intersection for frame 362 with frame 363: 0.6605371236801147
Histogram intersection for frame 364 with frame 365: 0.89254230260849
Histogram intersection for frame 366 with frame 367: 0.8027181029319763
Histogram intersection for frame 368 with frame 369: 0.8885807394981384
Histogram intersection for frame 370 with frame 371: 0.8303027153015137
Histogram intersection for frame 372 with frame 373: 0.8887369632720947
Histogram intersection for frame 374 with frame 375: 0.9212564826011658
Histogram intersection for frame 376 with frame 377: 0.8598372340202332
Histogram intersection for frame 378 with frame 379: 0.9186848998069763
Histogram intersection for frame 380 with frame 381: 0.8952734470367432
Histogram intersection for frame 382 with frame 383: 0.8509733080863953
Histogram intersection for frame 384 with frame 385: 0.8855729103088379
Histogram intersection for frame 386 with frame 387: 0.8952669501304626
Histogram intersection for frame 388 with frame 389: 0.7937337160110474
Histogram intersection for frame 390 with frame 391: 0.9117578268051147
Histogram intersection for frame 392 with frame 393: 0.6645833253860474
Histogram intersection for frame 394 with frame 395: 0.8767317533493042
Histogram intersection for frame 396 with frame 397: 0.8811751008033752
Histogram intersection for frame 398 with frame 399: 0.7774381637573242
Histogram intersection for frame 400 with frame 401: 0.87479168176651
Histogram intersection for frame 402 with frame 403: 0.8839616179466248
Histogram intersection for frame 404 with frame 405: 0.8023632764816284
Histogram intersection for frame 406 with frame 407: 0.9085546731948853
Histogram intersection for frame 408 with frame 409: 0.8962662816047668
Histogram intersection for frame 410 with frame 411: 0.81002277135849
Histogram intersection for frame 412 with frame 413: 0.8159342408180237
Histogram intersection for frame 414 with frame 415: 0.7677701711654663
Histogram intersection for frame 416 with frame 417: 0.7321224212646484
Histogram intersection for frame 418 with frame 419: 0.7830273509025574
Histogram intersection for frame 420 with frame 421: 0.8746484518051147
Histogram intersection for frame 422 with frame 423: 0.8436034917831421
Histogram intersection for frame 424 with frame 425: 0.8865852952003479
Histogram intersection for frame 426 with frame 427: 0.8092545866966248
Histogram intersection for frame 428 with frame 429: 0.8505598902702332
Histogram intersection for frame 430 with frame 431: 0.9018326997756958
Histogram intersection for frame 432 with frame 433: 0.8520475029945374
Histogram intersection for frame 434 with frame 435: 0.8061002492904663
Histogram intersection for frame 436 with frame 437: 0.8720638155937195
Histogram intersection for frame 438 with frame 439: 0.7469498515129089
Histogram intersection for frame 440 with frame 441: 0.4287630319595337
Histogram intersection for frame 442 with frame 443: 0.6673567891120911
Histogram intersection for frame 444 with frame 445: 0.6802441477775574
Histogram intersection for frame 446 with frame 447: 0.4714127480983734
Histogram intersection for frame 448 with frame 449: 0.44305989146232605
Histogram intersection for frame 450 with frame 451: 0.6601399779319763
Histogram intersection for frame 452 with frame 453: 0.6735091209411621
Histogram intersection for frame 454 with frame 455: 0.7594498991966248
Histogram intersection for frame 456 with frame 457: 0.8736523389816284
Histogram intersection for frame 458 with frame 459: 0.54153972864151
Histogram intersection for frame 460 with frame 461: 0.8049153685569763
Histogram intersection for frame 462 with frame 463: 0.8717219829559326
Histogram intersection for frame 464 with frame 465: 0.8605306148529053
Histogram intersection for frame 466 with frame 467: 0.83621746301651
Histogram intersection for frame 468 with frame 469: 0.7660872340202332
Histogram intersection for frame 470 with frame 471: 0.9144791960716248
Histogram intersection for frame 472 with frame 473: 0.8369433879852295
Histogram intersection for frame 474 with frame 475: 0.9127734303474426
Histogram intersection for frame 476 with frame 477: 0.8691112995147705
Histogram intersection for frame 478 with frame 479: 0.8869922161102295
Histogram intersection for frame 480 with frame 481: 0.8469694256782532
Histogram intersection for frame 482 with frame 483: 0.8275032639503479
Histogram intersection for frame 484 with frame 485: 0.9228580594062805
Histogram intersection for frame 486 with frame 487: 0.9111393094062805
Histogram intersection for frame 488 with frame 489: 0.7776399850845337
Histogram intersection for frame 490 with frame 491: 0.8431347608566284
Histogram intersection for frame 492 with frame 493: 0.8163802027702332
Histogram intersection for frame 494 with frame 495: 0.8302669525146484
Histogram intersection for frame 496 with frame 497: 0.8665299415588379
Histogram intersection for frame 498 with frame 499: 0.8494205474853516
Histogram intersection for frame 500 with frame 501: 0.8239225149154663
Histogram intersection for frame 502 with frame 503: 0.8503808379173279
Histogram intersection for frame 504 with frame 505: 0.7843424677848816
Histogram intersection for frame 506 with frame 507: 0.8039453029632568
Histogram intersection for frame 508 with frame 509: 0.8274934887886047
Histogram intersection for frame 510 with frame 511: 0.8607193827629089
Histogram intersection for frame 512 with frame 513: 0.8778222799301147
Histogram intersection for frame 514 with frame 515: 0.7585482001304626
Histogram intersection for frame 516 with frame 517: 0.792395830154419
Histogram intersection for frame 518 with frame 519: 0.643883466720581
Histogram intersection for frame 520 with frame 521: 0.730455756187439
Histogram intersection for frame 522 with frame 523: 0.7857226729393005
Histogram intersection for frame 524 with frame 525: 0.7960677146911621
Histogram intersection for frame 526 with frame 527: 0.8302050828933716
Histogram intersection for frame 528 with frame 529: 0.5820214748382568
Histogram intersection for frame 530 with frame 531: 0.6248567700386047
Histogram intersection for frame 532 with frame 533: 0.7172135710716248
Histogram intersection for frame 534 with frame 535: 0.7608919143676758
Histogram intersection for frame 536 with frame 537: 0.7906640768051147
Histogram intersection for frame 538 with frame 539: 0.7590950727462769
Histogram intersection for frame 540 with frame 541: 0.7639583349227905
Histogram intersection for frame 542 with frame 543: 0.760657548904419
Histogram intersection for frame 544 with frame 545: 0.7331575751304626
Histogram intersection for frame 546 with frame 547: 0.7977246046066284
Histogram intersection for frame 548 with frame 549: 0.755211591720581
Histogram intersection for frame 550 with frame 551: 0.7193619608879089
Histogram intersection for frame 552 with frame 553: 0.7794271111488342
Histogram intersection for frame 554 with frame 555: 0.7798307538032532
Histogram intersection for frame 556 with frame 557: 0.7101529836654663
Histogram intersection for frame 558 with frame 559: 0.7660351395606995
Histogram intersection for frame 560 with frame 561: 0.7537304759025574
Histogram intersection for frame 562 with frame 563: 0.6748274564743042
Histogram intersection for frame 564 with frame 565: 0.8981770873069763
Histogram intersection for frame 566 with frame 567: 0.8354719877243042
Histogram intersection for frame 568 with frame 569: 0.7606445550918579
Histogram intersection for frame 570 with frame 571: 0.7880794405937195
Histogram intersection for frame 572 with frame 573: 0.7982682585716248
Histogram intersection for frame 574 with frame 575: 0.7769270539283752
Histogram intersection for frame 576 with frame 577: 0.671959638595581
Histogram intersection for frame 578 with frame 579: 0.8200032711029053
Histogram intersection for frame 580 with frame 581: 0.8403939008712769
Histogram intersection for frame 582 with frame 583: 0.8751204609870911
Histogram intersection for frame 584 with frame 585: 0.8736230731010437
Histogram intersection for frame 586 with frame 587: 0.9253385663032532
Histogram intersection for frame 588 with frame 589: 0.5165820121765137
Histogram intersection for frame 590 with frame 591: 0.8257259130477905
Histogram intersection for frame 592 with frame 593: 0.6939518451690674
Histogram intersection for frame 594 with frame 595: 0.6895344853401184
Histogram intersection for frame 596 with frame 597: 0.6202669143676758
Histogram intersection for frame 598 with frame 599: 0.8329198956489563
Histogram intersection for frame 600 with frame 601: 0.855761706829071
Histogram intersection for frame 602 with frame 603: 0.8431249856948853
Histogram intersection for frame 604 with frame 605: 0.8605241179466248
Histogram intersection for frame 606 with frame 607: 0.8236881494522095
Histogram intersection for frame 608 with frame 609: 0.8199641704559326
Histogram intersection for frame 610 with frame 611: 0.9263118505477905
Histogram intersection for frame 612 with frame 613: 0.8258268237113953
Histogram intersection for frame 614 with frame 615: 0.8526172041893005
Histogram intersection for frame 616 with frame 617: 0.9415038824081421
Histogram intersection for frame 618 with frame 619: 0.8393521904945374
Histogram intersection for frame 620 with frame 621: 0.813916027545929
Histogram intersection for frame 622 with frame 623: 0.9341992139816284
Histogram intersection for frame 624 with frame 625: 0.7970963716506958
Histogram intersection for frame 626 with frame 627: 0.8700032830238342
Histogram intersection for frame 628 with frame 629: 0.9447656273841858
Histogram intersection for frame 630 with frame 631: 0.9122102856636047
Histogram intersection for frame 632 with frame 633: 0.7460481524467468
Histogram intersection for frame 634 with frame 635: 0.880358099937439
Histogram intersection for frame 636 with frame 637: 0.8902701735496521
Histogram intersection for frame 638 with frame 639: 0.8950423002243042
Histogram intersection for frame 640 with frame 641: 0.9124088287353516
Histogram intersection for frame 642 with frame 643: 0.8254719972610474
Histogram intersection for frame 644 with frame 645: 0.9060579538345337
Histogram intersection for frame 646 with frame 647: 0.9170442819595337
Histogram intersection for frame 648 with frame 649: 0.8514192700386047
Histogram intersection for frame 650 with frame 651: 0.8902343511581421
Histogram intersection for frame 652 with frame 653: 0.9065039157867432
Histogram intersection for frame 654 with frame 655: 0.8685709834098816
Histogram intersection for frame 656 with frame 657: 0.9378938674926758
Histogram intersection for frame 658 with frame 659: 0.8141927123069763
Histogram intersection for frame 660 with frame 661: 0.8824153542518616
Histogram intersection for frame 662 with frame 663: 0.9042415618896484
Histogram intersection for frame 664 with frame 665: 0.8469401001930237
Histogram intersection for frame 666 with frame 667: 0.9104427099227905
Histogram intersection for frame 668 with frame 669: 0.8553873896598816
Histogram intersection for frame 670 with frame 671: 0.8787565231323242
Histogram intersection for frame 672 with frame 673: 0.8254166841506958
Histogram intersection for frame 674 with frame 675: 0.8797168135643005
Histogram intersection for frame 676 with frame 677: 0.9348372220993042
Histogram intersection for frame 678 with frame 679: 0.737500011920929
Histogram intersection for frame 680 with frame 681: 0.9227799773216248
Histogram intersection for frame 682 with frame 683: 0.7308658957481384
Histogram intersection for frame 684 with frame 685: 0.8968164324760437
Histogram intersection for frame 686 with frame 687: 0.9523177146911621
Histogram intersection for frame 688 with frame 689: 0.8031217455863953
Histogram intersection for frame 690 with frame 691: 0.8086230754852295
Histogram intersection for frame 692 with frame 693: 0.9548404812812805
Histogram intersection for frame 694 with frame 695: 0.8086556196212769
Histogram intersection for frame 696 with frame 697: 0.9061523675918579
Histogram intersection for frame 698 with frame 699: 0.7644661664962769
Histogram intersection for frame 700 with frame 701: 0.7072297930717468
Histogram intersection for frame 702 with frame 703: 0.901660144329071
Histogram intersection for frame 704 with frame 705: 0.8702213764190674
Histogram intersection for frame 706 with frame 707: 0.9411816596984863
Histogram intersection for frame 708 with frame 709: 0.8922753930091858
Histogram intersection for frame 710 with frame 711: 0.8599023222923279
Histogram intersection for frame 712 with frame 713: 0.8355924487113953
Histogram intersection for frame 714 with frame 715: 0.818128228187561
Histogram intersection for frame 716 with frame 717: 0.8036458492279053
Histogram intersection for frame 718 with frame 719: 0.8651204705238342
Histogram intersection for frame 720 with frame 721: 0.7964583039283752
Histogram intersection for frame 722 with frame 723: 0.9257616996765137
Histogram intersection for frame 724 with frame 725: 0.8976237177848816
Histogram intersection for frame 726 with frame 727: 0.7102636694908142
Histogram intersection for frame 728 with frame 729: 0.9051432013511658
Histogram intersection for frame 730 with frame 731: 0.9028353095054626
Histogram intersection for frame 732 with frame 733: 0.9348600506782532
Histogram intersection for frame 734 with frame 735: 0.7411133050918579
Histogram intersection for frame 736 with frame 737: 0.9220149517059326
Histogram intersection for frame 738 with frame 739: 0.9380306005477905
Histogram intersection for frame 740 with frame 741: 0.8425357937812805
Histogram intersection for frame 742 with frame 743: 0.783203125
Histogram intersection for frame 744 with frame 745: 0.8100488185882568
Histogram intersection for frame 746 with frame 747: 0.6732063889503479
Histogram intersection for frame 748 with frame 749: 0.8695377707481384
Histogram intersection for frame 750 with frame 751: 0.7981022000312805
Histogram intersection for frame 752 with frame 753: 0.8464811444282532
Histogram intersection for frame 754 with frame 755: 0.7370572686195374
Histogram intersection for frame 756 with frame 757: 0.8122656345367432
Histogram intersection for frame 758 with frame 759: 0.8235090970993042
Histogram intersection for frame 760 with frame 761: 0.8580371141433716
Histogram intersection for frame 762 with frame 763: 0.6631022095680237
Histogram intersection for frame 764 with frame 765: 0.8877278566360474
Histogram intersection for frame 766 with frame 767: 0.9483072757720947
Histogram intersection for frame 768 with frame 769: 0.7781282663345337
Histogram intersection for frame 770 with frame 771: 0.8483203053474426
Histogram intersection for frame 772 with frame 773: 0.8600260615348816
Histogram intersection for frame 774 with frame 775: 0.9029589891433716
Histogram intersection for frame 776 with frame 777: 0.9472005367279053
Histogram intersection for frame 778 with frame 779: 0.9007584452629089
Histogram intersection for frame 780 with frame 781: 0.8297363519668579
Histogram intersection for frame 782 with frame 783: 0.8071549534797668
Histogram intersection for frame 784 with frame 785: 0.8760156035423279
Histogram intersection for frame 786 with frame 787: 0.8753906488418579
Histogram intersection for frame 788 with frame 789: 0.8169173002243042
Histogram intersection for frame 790 with frame 791: 0.8680468797683716
Histogram intersection for frame 792 with frame 793: 0.9005468487739563
Histogram intersection for frame 794 with frame 795: 0.8194172978401184
Histogram intersection for frame 796 with frame 797: 0.8917708396911621
Histogram intersection for frame 798 with frame 799: 0.8724186420440674
Histogram intersection for frame 800 with frame 801: 0.8076269626617432
Histogram intersection for frame 802 with frame 803: 0.80712890625
Histogram intersection for frame 804 with frame 805: 0.8536295294761658
Histogram intersection for frame 806 with frame 807: 0.9225455522537231
Histogram intersection for frame 808 with frame 809: 0.7930143475532532
Histogram intersection for frame 810 with frame 811: 0.9013965129852295
Histogram intersection for frame 812 with frame 813: 0.803235650062561
Histogram intersection for frame 814 with frame 815: 0.942578136920929
Histogram intersection for frame 816 with frame 817: 0.9175227880477905
Histogram intersection for frame 818 with frame 819: 0.7860579490661621
Histogram intersection for frame 820 with frame 821: 0.8697070479393005
Histogram intersection for frame 822 with frame 823: 0.8169368505477905
Histogram intersection for frame 824 with frame 825: 0.8009700775146484
Histogram intersection for frame 826 with frame 827: 0.9572786688804626
Histogram intersection for frame 828 with frame 829: 0.8677962422370911
Histogram intersection for frame 830 with frame 831: 0.9721646904945374
Histogram intersection for frame 832 with frame 833: 0.91458660364151
Histogram intersection for frame 834 with frame 835: 0.9578710794448853
Histogram intersection for frame 836 with frame 837: 0.9246614575386047
Histogram intersection for frame 838 with frame 839: 0.8398502469062805
Histogram intersection for frame 840 with frame 841: 0.8974609375
Histogram intersection for frame 842 with frame 843: 0.732861340045929
Histogram intersection for frame 844 with frame 845: 0.9459244608879089
Histogram intersection for frame 846 with frame 847: 0.8166731595993042
Histogram intersection for frame 848 with frame 849: 0.7075488567352295
Histogram intersection for frame 850 with frame 851: 0.8691145777702332
Histogram intersection for frame 852 with frame 853: 0.9012174606323242
Histogram intersection for frame 854 with frame 855: 0.893151044845581
Histogram intersection for frame 856 with frame 857: 0.6899870038032532
Histogram intersection for frame 858 with frame 859: 0.7930631637573242
Histogram intersection for frame 860 with frame 861: 0.716637372970581
Histogram intersection for frame 862 with frame 863: 0.8335481882095337
Histogram intersection for frame 864 with frame 865: 0.613818347454071
Histogram intersection for frame 866 with frame 867: 0.7882161736488342
Histogram intersection for frame 868 with frame 869: 0.751451849937439
Histogram intersection for frame 870 with frame 871: 0.7602571845054626
Histogram intersection for frame 872 with frame 873: 0.7368620038032532
Histogram intersection for frame 874 with frame 875: 0.7240754961967468
Histogram intersection for frame 876 with frame 877: 0.7678157687187195
Histogram intersection for frame 878 with frame 879: 0.7479394674301147
Histogram intersection for frame 880 with frame 881: 0.020781250670552254
Applied histogram equalization to 883 frames and saved to equalized_frames
```

Visualization

In [5]:
```python
plt.figure(figsize=(20, 10))
plt.plot(intersections, marker='o')
plt.title('Histogram Intersection for Specific Pairs of Frames')
plt.xlabel('Frame Pair Index')
plt.ylabel('Intersection Value')
plt.grid(True)
plt.show()
```



Spatio-Temporal Segmentation:

Perform segmentation on each frame using a technique like color thresholding or edge detection.

Track the segmented objects across frames to observe changes in motion and shape.

Identify the regions that remain consistent over time (foreground vs. background segmentation).

In [7]:
```python
def apply_edge_detection(frame):
    """Applies edge detection using the Canny algorithm."""
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
    edges = cv2.Canny(gray, 100, 200)  # Perform edge detection
    return edges

def save_edge_detection_video(input_video_path, output_video_path, display_size=(640, 480)):
    cap = cv2.VideoCapture(input_video_path)

    if not cap.isOpened():
        print(f"Error: Could not open video {input_video_path}")
        return

    # Define the codec and create VideoWriter object to save the output
    fourcc = cv2.VideoWriter_fourcc(*'mpv4')
    out = cv2.VideoWriter(output_video_path, fourcc, 20.0, display_size, isColor=False)

    while True:
        ret, frame = cap.read()
        if not ret:
            print("End of video reached or error reading video")
            break

        resized_frame = cv2.resize(frame, display_size)

        # Apply edge detection
        edges = apply_edge_detection(resized_frame)

        # Write the edge-detected frame to the video
        out.write(edges)

    cap.release()
    out.release()
    print(f"Edge detection video saved as {output_video_path}")

# Example usage
input_video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"
output_video_path = r"C:\Users\Abhineswari\Downloads\edge_detection_output.mp4"
save_edge_detection_video(input_video_path, output_video_path)
```

```
End of video reached or error reading video
Edge detection video saved as C:\Users\Abhineswari\Downloads\edge_detection_output.mp4
```

In [8]:
```python
def track_objects(frame, previous_centroids):
    """Tracks objects using contours and draws lines between consecutive centroids."""
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    edges = cv2.Canny(blurred, 50, 150)

    contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    centroids = []

    for contour in contours:
        if cv2.contourArea(contour) > 500:  # Filter small objects
            M = cv2.moments(contour)
            if M["m00"] != 0:
                cX = int(M["m10"] / M["m00"])
                cY = int(M["m01"] / M["m00"])
                centroids.append((cX, cY))
                cv2.circle(frame, (cX, cY), 5, (0, 255, 0), -1)  # Draw centroids

    # Draw lines between current and previous centroids
    if previous_centroids:
        for i in range(min(len(previous_centroids), len(centroids))):
            cv2.line(frame, previous_centroids[i], centroids[i], (255, 0, 0), 2)

    return centroids

def save_object_tracking_video(input_video_path, output_video_path, display_size=(640, 480)):
    cap = cv2.VideoCapture(input_video_path)

    if not cap.isOpened():
        print(f"Error: Could not open video {input_video_path}")
        return

    # Define the codec and create VideoWriter object
    fourcc = cv2.VideoWriter_fourcc(*'mpv4')
    out = cv2.VideoWriter(output_video_path, fourcc, 20.0, display_size)

    previous_centroids = []
    while True:
        ret, frame = cap.read()
        if not ret:
            print("End of video reached or error reading video")
            break

        resized_frame = cv2.resize(frame, display_size)

        # Track objects
        previous_centroids = track_objects(resized_frame, previous_centroids)

        # Write the tracking frame to the video
        out.write(resized_frame)

    cap.release()
    out.release()
    print(f"Object tracking video saved as {output_video_path}")

# Example usage
input_video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"
output_video_path = r"C:\Users\Abhineswari\Downloads\object_tracking_output.mp4"
save_object_tracking_video(input_video_path, output_video_path)
```

```
End of video reached or error reading video
Object tracking video saved as C:\Users\Abhineswari\Downloads\object_tracking_output.mp4
```

In [9]:
```python
import cv2
import numpy as np

background_subtractor = cv2.createBackgroundSubtractorMOG2(history=100, varThreshold=50, detectShadows=False)

def apply_background_subtraction(frame):
    """Applies background subtraction to distinguish foreground objects."""
    fg_mask = background_subtractor.apply(frame)

    # Clean up the mask using morphological operations
    kernel = np.ones((5, 5), np.uint8)
    fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN, kernel)

    return fg_mask

def save_foreground_segmentation_video(input_video_path, output_video_path, display_size=(640, 480)):
    cap = cv2.VideoCapture(input_video_path)

    if not cap.isOpened():
        print(f"Error: Could not open video {input_video_path}")
        return

    # Define the codec and create VideoWriter object
    fourcc = cv2.VideoWriter_fourcc(*'mpv4')
    out = cv2.VideoWriter(output_video_path, fourcc, 20.0, display_size, isColor=False)

    while True:
        ret, frame = cap.read()
        if not ret:
            print("End of video reached or error reading video")
            break

        resized_frame = cv2.resize(frame, display_size)

        # Apply background subtraction
        fg_mask = apply_background_subtraction(resized_frame)

        # Write the segmented frame to the video
        out.write(fg_mask)

    cap.release()
    out.release()
    print(f"Foreground segmentation video saved as {output_video_path}")

# Example usage
input_video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"
output_video_path = r"C:\Users\Abhineswari\Downloadsforeground_segmentation_output.mp4"
save_foreground_segmentation_video(input_video_path, output_video_path)
```

```
End of video reached or error reading video
Foreground segmentation video saved as C:\Users\Abhineswari\Downloadsforeground_segmentation_output.mp4
```

Scene cut detection

Use pixel-based comparison or histogram differences between consecutive frames to detect abrupt changes (hard cuts).

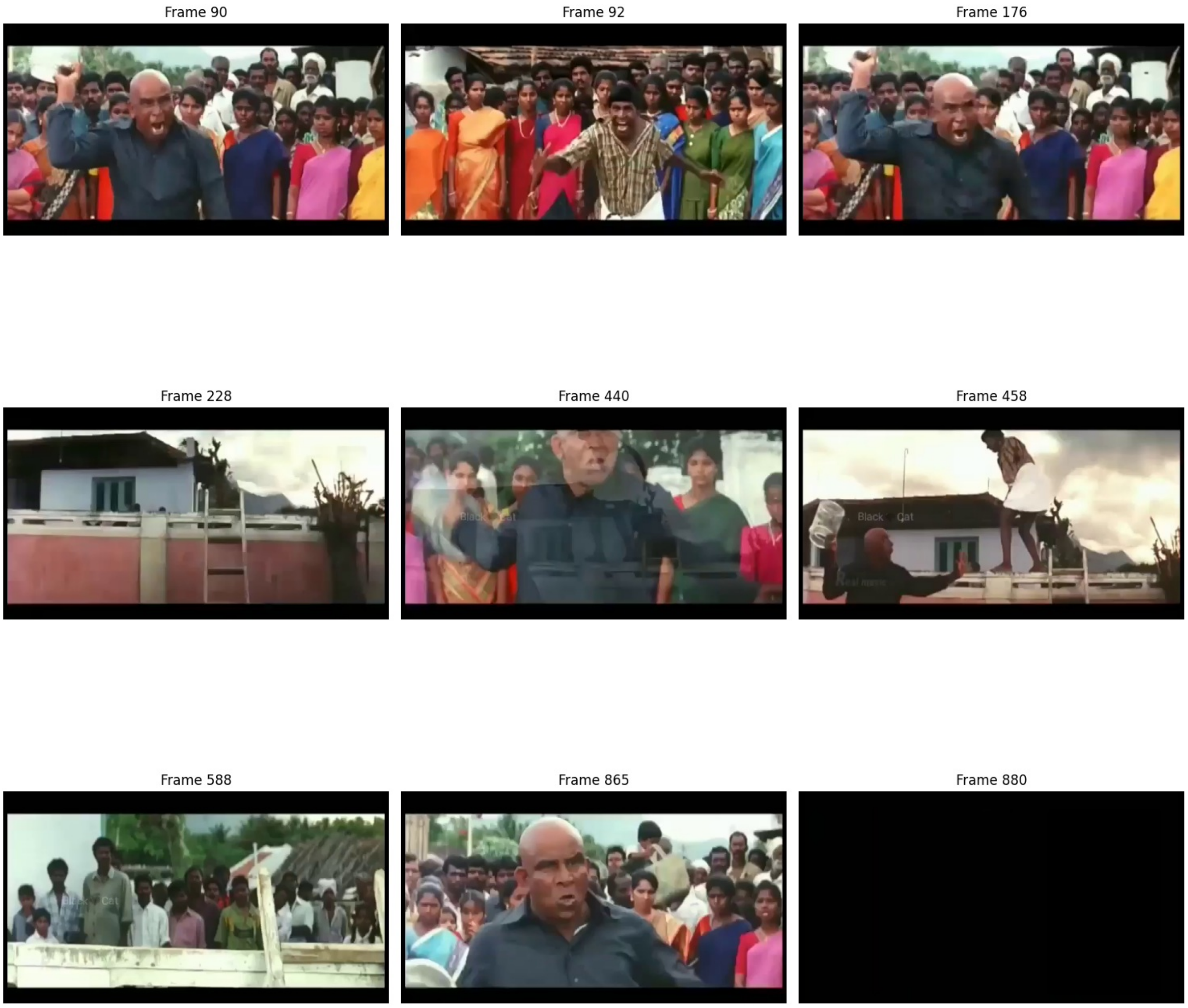
Detect gradual scene transitions (Soft cuts) by analyzing frame-to-frame intensity changes over time.

In [10]:
```python
def detect_hard_cuts(video_path, output_video_path, cuts_folder, threshold=30):
    # Create the output folder for detected cuts if it doesn't exist
    if not os.path.exists(cuts_folder):
        os.makedirs(cuts_folder)

    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        print(f"Error: Could not open video {video_path}")
        return

    # Define the codec and create VideoWriter object to save the output
    fourcc = cv2.VideoWriter_fourcc(*'mpv4')  # Codec for output video
    out = cv2.VideoWriter(output_video_path, fourcc, 20.0, (640, 480))

    ret, prev_frame = cap.read()
    if not ret:
        print("Error reading the first frame")
        return

    prev_frame = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
    hard_cut_indices = []

    frame_count = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Convert current frame to grayscale
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Calculate the absolute difference between the current and previous frame
        frame_diff = cv2.absdiff(gray_frame, prev_frame)

        # Calculate the mean pixel intensity of the difference image
        mean_diff = np.mean(frame_diff)

        # Check if the mean difference exceeds the threshold
        if mean_diff > threshold:
            hard_cut_indices.append(frame_count)
            cv2.putText(frame, "Hard Cut Detected", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

            # Save the detected hard cut frame as an image
            cut_frame_path = os.path.join(cuts_folder, f"hard_cut_frame_{frame_count:04d}.jpg")
            cv2.imwrite(cut_frame_path, frame)

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

        prev_frame = gray_frame
        frame_count += 1

    cap.release()
    out.release()
    print(f"Hard cut detection video saved as {output_video_path}")
    print(f"Detected hard cuts at frames: {hard_cut_indices}")
    print(f"Detected hard cut frames saved in: {cuts_folder}")

# Example usage
video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"
output_video_path = r"C:\Users\Abhineswari\Downloads\hard_cut_detection_output.mp4"
cuts_folder = 'detected_hard_cuts'  # Folder to save cut frames
detect_hard_cuts(video_path, output_video_path, cuts_folder)
```

```
Hard cut detection video saved as C:\Users\Abhineswari\Downloads\hard_cut_detection_output.mp4
Detected hard cuts at frames: [90, 92, 176, 228, 440, 458, 588, 865, 880]
Detected hard cut frames saved in: detected_hard_cuts
```

In [17]:
```python
def detect_hard_cuts(video_path, output_video_path, threshold=30, cols=3):
    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        print(f"Error: Could not open video {video_path}")
        return

    # Define the codec and create VideoWriter object to save the output
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Codec for output video
    out = cv2.VideoWriter(output_video_path, fourcc, 20.0, (640, 480))

    ret, prev_frame = cap.read()
    if not ret:
        print("Error reading the first frame")
        return

    prev_frame = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
    hard_cut_indices = []
    hard_cut_frames = []

    frame_count = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Convert current frame to grayscale
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Calculate the absolute difference between the current and previous frame
        frame_diff = cv2.absdiff(gray_frame, prev_frame)

        # Calculate the mean pixel intensity of the difference image
        mean_diff = np.mean(frame_diff)

        # Check if the mean difference exceeds the threshold (indicating a hard cut)
        if mean_diff > threshold:
            hard_cut_indices.append(frame_count)
            hard_cut_frames.append(frame.copy())  # Store the hard cut frame
            cv2.putText(frame, "Hard Cut Detected", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

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

        prev_frame = gray_frame
        frame_count += 1

    cap.release()
    out.release()

    # Print hard cut indices
    print(f"Detected hard cuts at frames: {hard_cut_indices}")

    # Plot the hard cut frames in a matrix
    if hard_cut_frames:
        rows = len(hard_cut_frames) // cols + 1
        fig, axs = plt.subplots(rows, cols, figsize=(15, 5 * rows))

        for i, frame in enumerate(hard_cut_frames):
            row, col = divmod(i, cols)
            if rows == 1:
                ax = axs[col]
            else:
                ax = axs[row, col]

            ax.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for display
            ax.set_title(f"Frame {hard_cut_indices[i]}")
            ax.axis('off')

        # Hide any remaining empty subplots
        for j in range(i + 1, rows * cols):
            row, col = divmod(j, cols)
            if rows == 1:
                axs[col].axis('off')
            else:
                axs[row, col].axis('off')

        plt.tight_layout()
        plt.show()

# Example usage
video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"
output_video_path = r"C:\Users\Abhineswari\Downloads\hard_cut_detection_output.mp4"

detect_hard_cuts(video_path, output_video_path)
```

```
Detected hard cuts at frames: [90, 92, 176, 228, 440, 458, 588, 865, 880]
```



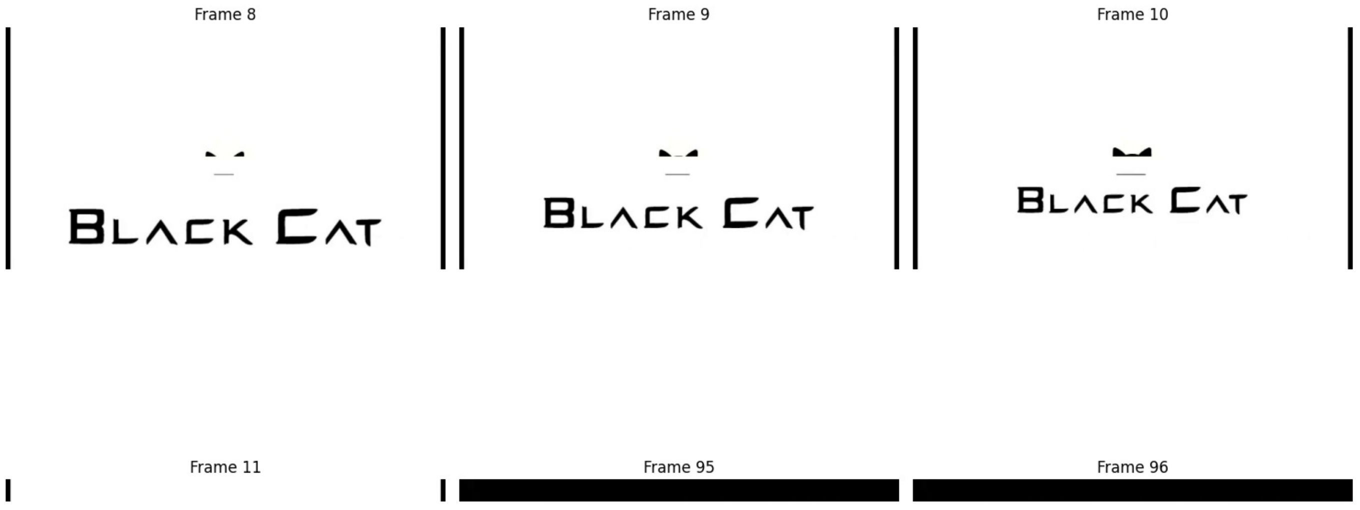
```
In [11]: def detect_soft_cuts(video_path, output_video_path, soft_cuts_folder, soft_cut_window=5, soft_cut_threshold=10)
             # Create the output folder for detected soft cuts if it doesn't exist
             if not os.path.exists(soft_cuts_folder):
                 os.makedirs(soft_cuts_folder)

             cap = cv2.VideoCapture(video_path)
```

```
    if not cap.isOpened():
        print(f"Error: Could not open video {video_path}")
        return

    # Define the codec and create VideoWriter object to save the output
    fourcc = cv2.VideoWriter_fourcc(*'mpv4')  # Codec for output video
    out = cv2.VideoWriter(output_video_path, fourcc, 20.0, (640, 480))

    # Initialize lists to hold previous frames for intensity analysis
    previous_frames = []
    soft_cut_indices = []

    frame_count = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Convert current frame to grayscale
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Add current frame to the list of previous frames (we'll analyze changes over the window size)
        previous_frames.append(gray_frame)
        if len(previous_frames) > soft_cut_window:
            previous_frames.pop(0)

        # Only proceed if we have enough frames to analyze
        if len(previous_frames) == soft_cut_window:
            # Calculate the average frame difference over the window
            intensity_changes = [np.mean(cv2.absdiff(previous_frames[i], previous_frames[i + 1])) for i in rang
            average_change = np.mean(intensity_changes)

            # Check if the average intensity change indicates a soft cut
            if soft_cut_threshold < average_change < 2 * soft_cut_threshold:  # The range is to ensure it's not
                soft_cut_indices.append(frame_count)
                cv2.putText(frame, "Soft Cut Detected", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

                # Save the detected soft cut frame as an image
                cut_frame_path = os.path.join(soft_cuts_folder, f"soft_cut_frame_{frame_count:04d}.jpg")
                cv2.imwrite(cut_frame_path, frame)

        # Write the frame to the output video
        out.write(frame)
        frame_count += 1

    cap.release()
    out.release()
    print(f"Soft cut detection video saved as {output_video_path}")
    print(f"Detected soft cuts at frames: {soft_cut_indices}")
    print(f"Detected soft cut frames saved in: {soft_cuts_folder}")

# Example usage
video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"
output_video_path = r"C:\Users\Abhineswari\Downloads\soft_cut_detection_output.mp4"
soft_cuts_folder = 'detected_soft_cuts'  # Folder to save soft cut frames
detect_soft_cuts(video_path, output_video_path, soft_cuts_folder)
```

```
Soft cut detection video saved as C:\Users\Abhineswari\Downloads\soft_cut_detection_output.mp4
Detected soft cuts at frames: [8, 9, 10, 11, 95, 96, 177, 178, 179, 180, 220, 221, 222, 223, 225, 230, 231, 232,
441, 442, 443, 459, 460, 461, 462, 562, 563, 564, 565, 589, 590, 591, 592, 865, 866, 867, 868, 869]
Detected soft cut frames saved in: detected_soft_cuts
```

In [18]:
```
def detect_soft_cuts(video_path, output_video_path, soft_cut_window=5, soft_cut_threshold=10, cols=3):
    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        print(f"Error: Could not open video {video_path}")
        return

    # Define the codec and create VideoWriter object to save the output
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Codec for output video
    out = cv2.VideoWriter(output_video_path, fourcc, 20.0, (640, 480))

    # Initialize lists to hold previous frames for intensity analysis
    previous_frames = []
    soft_cut_indices = []
    soft_cut_frames = []

    frame_count = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        # Convert current frame to grayscale
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Add current frame to the list of previous frames (for intensity analysis)
        previous_frames.append(gray_frame)
        if len(previous_frames) > soft_cut_window:
            previous_frames.pop(0)

        # Only proceed if we have enough frames for the analysis
        if len(previous_frames) == soft_cut_window:
            # Calculate the average frame difference over the window
            intensity_changes = [np.mean(cv2.absdiff(previous_frames[i], previous_frames[i + 1])) for i in rang
            average_change = np.mean(intensity_changes)

            # Check if the average intensity change indicates a soft cut
            if soft_cut_threshold < average_change < 2 * soft_cut_threshold:
                soft_cut_indices.append(frame_count)
                soft_cut_frames.append(frame.copy())  # Store the soft cut frame
                cv2.putText(frame, "Soft Cut Detected", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        # Write the frame to the output video
        out.write(frame)
        frame_count += 1

    cap.release()
    out.release()

    # Print soft cut indices
    print(f"Detected soft cuts at frames: {soft_cut_indices}")

    # Plot the soft cut frames in a matrix
    if soft_cut_frames:
        rows = len(soft_cut_frames) // cols + 1
        fig, axs = plt.subplots(rows, cols, figsize=(15, 5 * rows))

        for i, frame in enumerate(soft_cut_frames):
            row, col = divmod(i, cols)
            if rows == 1:
                ax = axs[col]
            else:
                ax = axs[row, col]

            ax.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for display
            ax.set_title(f"Frame {soft_cut_indices[i]}")
            ax.axis('off')

        # Hide any remaining empty subplots
        for j in range(i + 1, rows * cols):
            row, col = divmod(j, cols)
            if rows == 1:
                axs[col].axis('off')
            else:
                axs[row, col].axis('off')

        plt.tight_layout()
        plt.show()

# Example usage
video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"
output_video_path = r"C:\Users\Abhineswari\Downloads\soft_cut_detection_output.mp4"

detect_soft_cuts(video_path, output_video_path)
```

```
Detected soft cuts at frames: [8, 9, 10, 11, 95, 96, 177, 178, 179, 180, 220, 221, 222, 223, 225, 230, 231, 232,
441, 442, 443, 459, 460, 461, 462, 562, 563, 564, 565, 589, 590, 591, 592, 865, 866, 867, 868, 869]
```



BLACK CAT

Frame 177

Frame 178

Frame 179

Frame 180

Frame 220

Frame 221

Frame 222

Frame 223

Frame 225

Frame 230

Frame 231

Frame 232

Frame 441

Frame 442

Frame 443

Frame 459

Frame 460

Frame 461

Frame 462

Frame 562

Frame 563

Frame 564

Frame 565

Frame 589

Frame 590

Frame 591

Frame 592

Calculating the similarity score for the hard and soft cuts

In [12]:

```python
import cv2
import os
import numpy as np
from skimage.metrics import structural_similarity as ssim

def calculate_ssim(imageA, imageB):
    """Compute the Structural Similarity Index (SSIM) between two images."""
    grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
    grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)
    score, _ = ssim(grayA, grayB, full=True)
    return score

def calculate_similarity_for_cuts(cut_indices, folder, frame_prefix="frame_", display_cut_type="Hard Cut"):
    """Calculate SSIM between consecutive cut frames."""
    similarity_scores = {}
    for i in range(1, len(cut_indices)):
        frameA_path = os.path.join(folder, f"{frame_prefix}{cut_indices[i-1]:04d}.jpg")
        frameB_path = os.path.join(folder, f"{frame_prefix}{cut_indices[i]:04d}.jpg")

        # Read frames
        frameA = cv2.imread(frameA_path)
        frameB = cv2.imread(frameB_path)

        if frameA is None or frameB is None:
            print(f"Error reading frames {cut_indices[i-1]} or {cut_indices[i]}")
            continue

        # Calculate SSIM similarity score
        score = calculate_ssim(frameA, frameB)
        similarity_scores[cut_indices[i]] = score

        # Display the results
        print(f"{display_cut_type} Similarity between frame {cut_indices[i-1]} and {cut_indices[i]}: SSIM = {sc

    return similarity_scores

# Assuming we already have the hard cut and soft cut frame indices
hard_cut_indices = [90, 92, 176, 228, 440, 458, 588, 865, 880]
soft_cut_indices = [8, 9, 10, 11, 95, 96, 177, 178, 179, 180, 220, 221, 222, 223, 225, 230, 231, 232, 441, 442,

# Example paths
frame_folder = 'output_frames'
```

```python
# Calculate SSIM for hard cuts
print("Hard Cut SSIM Scores:")
hard_cut_ssim_scores = calculate_similarity_for_cuts(hard_cut_indices, frame_folder, display_cut_type="Hard Cut

# Calculate SSIM for soft cuts
print("\nSoft Cut SSIM Scores:")
soft_cut_ssim_scores = calculate_similarity_for_cuts(soft_cut_indices, frame_folder, display_cut_type="Soft Cut
```

```
Hard Cut SSIM Scores:
Hard Cut Similarity between frame 90 and 92: SSIM = 0.2582
Hard Cut Similarity between frame 92 and 176: SSIM = 0.3603
Hard Cut Similarity between frame 176 and 228: SSIM = 0.3869
Hard Cut Similarity between frame 228 and 440: SSIM = 0.4596
Hard Cut Similarity between frame 440 and 458: SSIM = 0.4928
Hard Cut Similarity between frame 458 and 588: SSIM = 0.5018
Hard Cut Similarity between frame 588 and 865: SSIM = 0.4423
Hard Cut Similarity between frame 865 and 880: SSIM = 0.5008

Soft Cut SSIM Scores:
Soft Cut Similarity between frame 8 and 9: SSIM = 0.9093
Soft Cut Similarity between frame 9 and 10: SSIM = 0.9271
Soft Cut Similarity between frame 10 and 11: SSIM = 0.9426
Soft Cut Similarity between frame 11 and 95: SSIM = 0.2002
Soft Cut Similarity between frame 95 and 96: SSIM = 0.9730
Soft Cut Similarity between frame 96 and 177: SSIM = 0.3463
Soft Cut Similarity between frame 177 and 178: SSIM = 0.9106
Soft Cut Similarity between frame 178 and 179: SSIM = 0.9349
Soft Cut Similarity between frame 179 and 180: SSIM = 0.9333
Soft Cut Similarity between frame 180 and 220: SSIM = 0.5978
Soft Cut Similarity between frame 220 and 221: SSIM = 0.9968
Soft Cut Similarity between frame 221 and 222: SSIM = 0.7556
Soft Cut Similarity between frame 222 and 223: SSIM = 0.7788
Soft Cut Similarity between frame 223 and 225: SSIM = 0.7932
Soft Cut Similarity between frame 225 and 230: SSIM = 0.5216
Soft Cut Similarity between frame 230 and 231: SSIM = 0.9923
Soft Cut Similarity between frame 231 and 232: SSIM = 0.9920
Soft Cut Similarity between frame 232 and 441: SSIM = 0.5398
Soft Cut Similarity between frame 441 and 442: SSIM = 0.8694
Soft Cut Similarity between frame 442 and 443: SSIM = 0.9207
Soft Cut Similarity between frame 443 and 459: SSIM = 0.4790
Soft Cut Similarity between frame 459 and 460: SSIM = 0.9108
Soft Cut Similarity between frame 460 and 461: SSIM = 0.9922
Soft Cut Similarity between frame 461 and 462: SSIM = 0.9395
Soft Cut Similarity between frame 462 and 562: SSIM = 0.4907
Soft Cut Similarity between frame 562 and 563: SSIM = 0.8136
Soft Cut Similarity between frame 563 and 564: SSIM = 0.9954
Soft Cut Similarity between frame 564 and 565: SSIM = 0.9942
Soft Cut Similarity between frame 565 and 589: SSIM = 0.4011
Soft Cut Similarity between frame 589 and 590: SSIM = 0.9824
Soft Cut Similarity between frame 590 and 591: SSIM = 0.9881
Soft Cut Similarity between frame 591 and 592: SSIM = 0.9879
Soft Cut Similarity between frame 592 and 865: SSIM = 0.5415
Soft Cut Similarity between frame 865 and 866: SSIM = 0.6030
Soft Cut Similarity between frame 866 and 867: SSIM = 0.9161
Soft Cut Similarity between frame 867 and 868: SSIM = 0.9638
Soft Cut Similarity between frame 868 and 869: SSIM = 0.9206
```

In [13]:
```python
def mark_scene_cut_frames(cut_indices, folder, output_folder, frame_prefix="frame_", cut_type="Hard Cut"):
    """
    Marks scene cuts on frames and saves them to an output folder.

    Args:
    - cut_indices: List of frame indices where cuts are detected.
    - folder: Path to the folder containing the original frames.
    - output_folder: Path to the folder where marked frames will be saved.
    - frame_prefix: Prefix for the frame filenames (e.g., "frame_").
    - cut_type: Type of cut being marked ("Hard Cut" or "Soft Cut").
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    for i, cut_index in enumerate(cut_indices):
        frame_path = os.path.join(folder, f"{frame_prefix}{cut_index:04d}.jpg")
        frame = cv2.imread(frame_path)

        if frame is None:
            print(f"Error reading frame {cut_index}")
            continue

        # Mark the scene cut by adding text and a rectangle
        cv2.putText(frame, f"{cut_type} Detected", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        cv2.rectangle(frame, (10, 10), (frame.shape[1] - 10, frame.shape[0] - 10), (0, 255, 0), 3)
        # Save the marked frame
        marked_frame_path = os.path.join(output_folder, f"{frame_prefix}{cut_index:04d}_cut.jpg")
        cv2.imwrite(marked_frame_path, frame)

        print(f"{cut_type} marked on frame {cut_index} and saved to {output_folder}")

def display_scene_summary(cut_indices, total_frames, cut_type="Hard Cut"):
    """
    Displays a summary of the detected scene boundaries.

    Args:
    - cut_indices: List of frame indices where cuts are detected.
    - total_frames: Total number of frames in the video.
    - cut_type: Type of cut being summarized ("Hard Cut" or "Soft Cut").
    """
    print(f"\n{cut_type} Scene Boundaries:")
    cut_indices = [0] + cut_indices + [total_frames]  # Add first and last frame as boundaries
    for i in range(1, len(cut_indices)):
        print(f"Scene {i}: Frames {cut_indices[i-1]} to {cut_indices[i]-1}")

# Example frame cut indices (replace with your actual detected cuts)
hard_cut_indices = [90, 92, 176, 228, 440, 458, 588, 865, 880]
soft_cut_indices = [8, 9, 10, 11, 95, 96, 177, 178, 179, 180, 220, 221, 222, 223, 225, 230, 231, 232, 441, 442,

# Example paths
frame_folder = 'output_frames'
output_folder_hard_cuts = 'marked_hard_cut_frames'
output_folder_soft_cuts = 'marked_soft_cuts_frames'

# Total frames (replace with your actual total frames)
total_frames = 883
# Mark hard cuts on frames and save them
print("Marking Hard Cut Frames...")
mark_scene_cut_frames(hard_cut_indices, frame_folder, output_folder_hard_cuts, cut_type="Hard Cut")

# Mark soft cuts on frames and save them
print("\nMarking Soft Cut Frames...")
mark_scene_cut_frames(soft_cut_indices, frame_folder, output_folder_soft_cuts, cut_type="Soft Cut")

# Display scene summary for hard cuts
display_scene_summary(hard_cut_indices, total_frames, cut_type="Hard Cut")

# Display scene summary for soft cuts
display_scene_summary(soft_cut_indices, total_frames, cut_type="Soft Cut")
```

```
Marking Hard Cut Frames...
Hard Cut marked on frame 90 and saved to marked_hard_cut_frames
Hard Cut marked on frame 92 and saved to marked_hard_cut_frames
Hard Cut marked on frame 176 and saved to marked_hard_cut_frames
Hard Cut marked on frame 228 and saved to marked_hard_cut_frames
Hard Cut marked on frame 440 and saved to marked_hard_cut_frames
Hard Cut marked on frame 458 and saved to marked_hard_cut_frames
Hard Cut marked on frame 588 and saved to marked_hard_cut_frames
Hard Cut marked on frame 865 and saved to marked_hard_cut_frames
Hard Cut marked on frame 880 and saved to marked_hard_cut_frames

Marking Soft Cut Frames...
Soft Cut marked on frame 8 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 9 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 10 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 11 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 95 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 96 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 177 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 178 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 179 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 180 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 220 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 221 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 222 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 223 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 225 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 230 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 231 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 232 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 441 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 442 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 443 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 459 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 460 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 461 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 462 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 562 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 563 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 564 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 565 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 589 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 590 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 591 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 592 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 865 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 866 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 867 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 868 and saved to marked_soft_cuts_frames
Soft Cut marked on frame 869 and saved to marked_soft_cuts_frames

Hard Cut Scene Boundaries:
Scene 1: Frames 0 to 89
Scene 2: Frames 90 to 91
Scene 3: Frames 92 to 175
Scene 4: Frames 176 to 227
Scene 5: Frames 228 to 439
Scene 6: Frames 440 to 457
Scene 7: Frames 458 to 587
Scene 8: Frames 588 to 864
Scene 9: Frames 865 to 879
Scene 10: Frames 880 to 882

Soft Cut Scene Boundaries:
Scene 1: Frames 0 to 7
Scene 2: Frames 8 to 8
Scene 3: Frames 9 to 9
Scene 4: Frames 10 to 10
Scene 5: Frames 11 to 94
Scene 6: Frames 95 to 95
Scene 7: Frames 96 to 176
Scene 8: Frames 177 to 177
Scene 9: Frames 178 to 178
Scene 10: Frames 179 to 179
Scene 11: Frames 180 to 219
Scene 12: Frames 220 to 220
Scene 13: Frames 221 to 221
Scene 14: Frames 222 to 222
Scene 15: Frames 223 to 224
Scene 16: Frames 225 to 229
Scene 17: Frames 230 to 230
Scene 18: Frames 231 to 231
Scene 19: Frames 232 to 440
Scene 20: Frames 441 to 441
Scene 21: Frames 442 to 442
Scene 22: Frames 443 to 458
Scene 23: Frames 459 to 459
Scene 24: Frames 460 to 460
Scene 25: Frames 461 to 461
Scene 26: Frames 462 to 561
Scene 27: Frames 562 to 562
Scene 28: Frames 563 to 563
Scene 29: Frames 564 to 564
Scene 30: Frames 565 to 588
Scene 31: Frames 589 to 589
Scene 32: Frames 590 to 590
Scene 33: Frames 591 to 591
Scene 34: Frames 592 to 864
Scene 35: Frames 865 to 865
Scene 36: Frames 866 to 866
Scene 37: Frames 867 to 867
Scene 38: Frames 868 to 868
Scene 39: Frames 869 to 882
```

Hard and soft cut video visualization

In [14]:
```python
def mark_frame(frame, frame_index, hard_cut_indices, soft_cut_indices):
    """
    Marks the frame with blue for hard cut and green for soft cut based on the frame index.
    """
    if frame_index in hard_cut_indices:
        # Blue for hard cut
        cv2.putText(frame, "Hard Cut Detected", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
        cv2.rectangle(frame, (10, 10), (frame.shape[1] - 10, frame.shape[0] - 10), (255, 0, 0), 3)
    elif frame_index in soft_cut_indices:
        # Green for soft cut
        cv2.putText(frame, "Soft Cut Detected", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        cv2.rectangle(frame, (10, 10), (frame.shape[1] - 10, frame.shape[0] - 10), (0, 255, 0), 3)

    return frame
```

```python
def create_marked_video(video_path, output_video_path, hard_cut_indices, soft_cut_indices, frame_size=(640, 480
    """
    Creates a new video with marked hard (blue) and soft cuts (green) and saves it.
    """
    # Open the original video
    cap = cv2.VideoCapture(video_path)

    # Get the video properties
    original_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    original_fps = int(cap.get(cv2.CAP_PROP_FPS))

    # Resize if necessary
    if frame_size is None:
        frame_size = (original_width, original_height)

    # Define the codec and create VideoWriter object
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Use 'mp4v' for MP4 format
    out = cv2.VideoWriter(output_video_path, fourcc, fps, frame_size)

    frame_count = 0

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Resize the frame if necessary
        if frame_size != (original_width, original_height):
            frame = cv2.resize(frame, frame_size)

        # Mark hard/soft cuts on the current frame
        frame = mark_frame(frame, frame_count, hard_cut_indices, soft_cut_indices)

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

        # Optionally display the marked frame in real-time (remove if not needed)
        cv2.imshow('Marked Video', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

        frame_count += 1

    # Release the video capture and writer objects
    cap.release()
    out.release()
    cv2.destroyAllWindows()

    print(f"Marked video saved to {output_video_path}")

# Hard and soft cut frame indices
hard_cut_indices = [90, 92, 176, 228, 440, 458, 588, 865, 880]
soft_cut_indices = [8, 9, 10, 11, 95, 96, 177, 178, 179, 180, 220, 221, 222, 223, 225, 230, 231, 232, 441, 442,

# Example paths
video_path = r"C:\Users\Abhineswari\Downloads\scene_cut.mp4"  # Path to your original video
output_video_path = r"C:\Users\Abhineswari\Downloads\marked_video_hard_soft_cuts.mp4"  # Path to save the marke

# Create and save a marked video
create_marked_video(video_path, output_video_path, hard_cut_indices, soft_cut_indices, frame_size=(640, 480), f
```

```
Marked video saved to C:\Users\Abhineswari\Downloads\marked_video_hard_soft_cuts.mp4
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js