CS445_final_code

December 14, 2024

1 CS 445: Computational Photography Final Project

1.1 Algorithmic Basketball Detection via Segmentation

Inspired by (insert original paper link here)

1.1.1 Frame Extraction

[1]:
```
import cv2
import numpy as np
from pathlib import Path
import os
```

[2]:
```
def extract_frames(video_path, output_dir, frames_per_second=60):
    """
    Extract frames from video at specified temporal rate

    Args:
        video_path: Path to input video
        output_dir: Directory to save frames
        frames_per_second: Number of frames to extract per second of video,␣
 ↪should be geq the video's frame rate
    """
    Path(output_dir).mkdir(exist_ok=True)

    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        raise ValueError(f"Could not open video: {video_path}")

    # get video properties and calc sample rate
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    print(f"Video FPS: {fps}")
    sample_rate = int(fps / frames_per_second)

    frame_count = 0
    saved_count = 0

    while cap.isOpened():
```

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

        # save frame at the rate specified
        if frame_count % sample_rate == 0:
            output_path = os.path.join(output_dir, f"frame_{saved_count:04d}.
↪jpg")
            cv2.imwrite(output_path, frame)
            saved_count += 1

            # print progress
            if saved_count % 10 == 0:
                print(f"Saved {saved_count} frames...")

        frame_count += 1

    cap.release()
    print(f"Extraction complete. Processed {frame_count} frames, saved␣
↪{saved_count} frames")
```

[3]:
```
video_path = "../data/free_throws_urichmond.mp4"
output_dir = "../data/free_throws_urichmond_frames/"
extract_frames(video_path, output_dir, frames_per_second=20)
```

```
Video FPS: 23
Saved 10 frames…
Saved 20 frames…
Saved 30 frames…
Saved 40 frames…
Saved 50 frames…
Saved 60 frames…
Saved 70 frames…
Saved 80 frames…
Saved 90 frames…
Saved 100 frames…
Saved 110 frames…
Saved 120 frames…
Saved 130 frames…
Saved 140 frames…
Saved 150 frames…
Saved 160 frames…
Saved 170 frames…
Saved 180 frames…
Saved 190 frames…
Saved 200 frames…
Saved 210 frames…
Saved 220 frames…
```

```
Saved 230 frames…
Saved 240 frames…
Saved 250 frames…
Saved 260 frames…
Saved 270 frames…
Saved 280 frames…
Saved 290 frames…
Saved 300 frames…
Saved 310 frames…
Saved 320 frames…
Saved 330 frames…
Saved 340 frames…
Saved 350 frames…
Saved 360 frames…
Saved 370 frames…
Saved 380 frames…
Saved 390 frames…
Saved 400 frames…
Saved 410 frames…
Saved 420 frames…
Saved 430 frames…
Saved 440 frames…
Saved 450 frames…
Saved 460 frames…
Saved 470 frames…
Saved 480 frames…
Saved 490 frames…
Saved 500 frames…
Saved 510 frames…
Saved 520 frames…
Saved 530 frames…
Saved 540 frames…
Saved 550 frames…
Extraction complete. Processed 550 frames, saved 550 frames
```

[4]:
```
video_path = "../data/curry_kerr_free_throws.mp4"
output_dir = "../data/curry_kerr_free_throws_frames/"
extract_frames(video_path, output_dir, frames_per_second=11)
```

```
Video FPS: 11
Saved 10 frames…
Saved 20 frames…
Saved 30 frames…
Saved 40 frames…
Saved 50 frames…
Saved 60 frames…
Saved 70 frames…
Saved 80 frames…
```

```
Extraction complete. Processed 81 frames, saved 81 frames
```

[5]:
```
video_path = "../data/suns_warriors_kd3.mp4"
output_dir = "../data/suns_warriors_kd3_frames/"
extract_frames(video_path, output_dir, frames_per_second=30)
```

```
Video FPS: 60
Saved 10 frames…
Saved 20 frames…
Saved 30 frames…
Saved 40 frames…
Saved 50 frames…
Saved 60 frames…
Saved 70 frames…
Saved 80 frames…
Saved 90 frames…
Saved 100 frames…
Saved 110 frames…
Saved 120 frames…
Saved 130 frames…
Saved 140 frames…
Saved 150 frames…
Saved 160 frames…
Saved 170 frames…
Saved 180 frames…
Saved 190 frames…
Saved 200 frames…
Saved 210 frames…
Saved 220 frames…
Saved 230 frames…
Saved 240 frames…
Saved 250 frames…
Saved 260 frames…
Saved 270 frames…
Saved 280 frames…
Saved 290 frames…
Saved 300 frames…
Saved 310 frames…
Saved 320 frames…
Saved 330 frames…
Saved 340 frames…
Extraction complete. Processed 681 frames, saved 341 frames
```

[6]:
```
video_path = "../data/dame_logo_three.mp4"
output_dir = "../data/dame_logo_three_frames/"
extract_frames(video_path, output_dir, frames_per_second=25)
```

```
Video FPS: 25
Saved 10 frames…
```

```
Saved 20 frames…
Saved 30 frames…
Saved 40 frames…
Saved 50 frames…
Saved 60 frames…
Saved 70 frames…
Saved 80 frames…
Saved 90 frames…
Saved 100 frames…
Saved 110 frames…
Saved 120 frames…
Saved 130 frames…
Saved 140 frames…
Saved 150 frames…
Saved 160 frames…
Saved 170 frames…
Saved 180 frames…
Saved 190 frames…
Saved 200 frames…
Saved 210 frames…
Saved 220 frames…
Saved 230 frames…
Saved 240 frames…
Saved 250 frames…
Saved 260 frames…
Saved 270 frames…
Saved 280 frames…
Saved 290 frames…
Saved 300 frames…
Saved 310 frames…
Saved 320 frames…
Saved 330 frames…
Extraction complete. Processed 331 frames, saved 331 frames
```

[7]:
```
video_path = "../data/guangdong_beijing.mp4"
output_dir = "../data/guangdong_beijing_frames/"
extract_frames(video_path, output_dir, frames_per_second=25)
```

```
Video FPS: 25
Saved 10 frames…
Saved 20 frames…
Saved 30 frames…
Saved 40 frames…
Saved 50 frames…
Saved 60 frames…
Saved 70 frames…
Saved 80 frames…
Saved 90 frames…
```

```
Saved 100 frames…
Saved 110 frames…
Saved 120 frames…
Saved 130 frames…
Saved 140 frames…
Saved 150 frames…
Saved 160 frames…
Saved 170 frames…
Saved 180 frames…
Saved 190 frames…
Saved 200 frames…
Saved 210 frames…
Saved 220 frames…
Saved 230 frames…
Saved 240 frames…
Saved 250 frames…
Saved 260 frames…
Saved 270 frames…
Extraction complete. Processed 277 frames, saved 277 frames
```

1.1.2 Ball Detection

[8]:
```python
import matplotlib.pyplot as plt

def detect_basketball_with_visualization(frame1, frame2, min_area=50,␣
↪circularity_threshold=0.25, visualize=False, intermediate_file=None):
    """
    Detects basketball in a frame using a combination of filtering and␣
↪morphological operations

    Args:
        frame1: frame we want to detect and visualize the detection on
        frame2: immediate subsequent frame after frame1, used for motion␣
↪detection and binary diff
        min_area: minimum area of connected component to keep in the binary␣
↪diff mask
        circularity_threshold (optional): threshold for circularity difference␣
↪to keep a contour
        visualize: whether to visualize intermediate steps
        intermediate_file: file to save the intermediate visualization to
    """
    # create binary difference image between frame1 and frame2
    diff = cv2.absdiff(frame1, frame2)
    gray_diff = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    _, binary_diff = cv2.threshold(gray_diff, 30, 255, cv2.THRESH_BINARY)

    # filter by HSV color (H values up to 60 as the paper section 2.3 suggests)
```

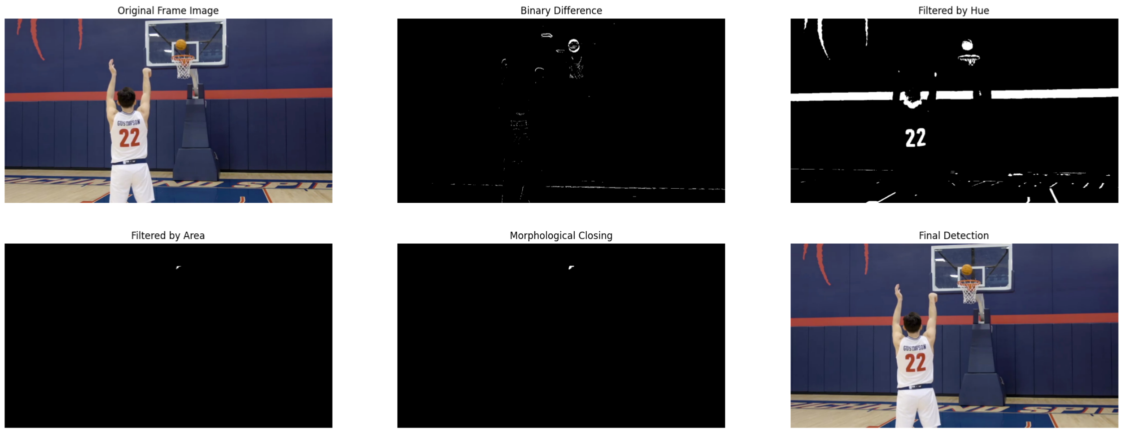
```
    hsv = cv2.cvtColor(frame1, cv2.COLOR_BGR2HSV)
    brown_mask = cv2.inRange(hsv, (0, 100, 20), (60, 255, 200))

    # combine with color mask
    combined_mask = cv2.bitwise_and(binary_diff, brown_mask)

    # filter by area, min_area is parameterized since the ball may be larger or␣
↪smaller based on camera angle (default 50px)
    area_filtered_mask = np.zeros_like(combined_mask)

    # find connected components (contours) of the combined mask
    contours, _ = cv2.findContours(combined_mask, cv2.RETR_EXTERNAL, cv2.
↪CHAIN_APPROX_SIMPLE)

    # if the component is large enough, keep it
    for contour in contours:
        area = cv2.contourArea(contour)
        if area >= min_area:
            cv2.drawContours(area_filtered_mask, [contour], -1, 255, -1)

    # use morphological closing to fill gaps in close-to-circle object (our␣
↪innovation since we noticed struggled without it)
    kernel = np.ones((3, 3), np.uint8)
    dilated_mask = cv2.dilate(area_filtered_mask, kernel)
    closed_mask = cv2.morphologyEx(dilated_mask, cv2.MORPH_CLOSE, kernel)
    combined_mask = closed_mask

    # filter by circularity, circularity_diff closer to 0 means closer to a␣
↪perfect circle
    circularity_filtered_mask = np.zeros_like(combined_mask)
    best_candidate = None
    best_circularity_diff = float('inf')

    # recompute contours after morphological closing
    contours, _ = cv2.findContours(combined_mask, cv2.RETR_EXTERNAL, cv2.
↪CHAIN_APPROX_SIMPLE)

    for contour in contours:
        area = cv2.contourArea(contour)
        perimeter = cv2.arcLength(contour, True)
        if perimeter == 0:
            continue

        circularity = (4 * np.pi * area) / (perimeter ** 2)
        circularity_diff = abs(circularity - 1)
```

```
        if circularity_diff <= circularity_threshold:  # parameterized␣
↪threshold for how good the circle should be
            # only keep close to circle for visualizing the best candidates
            cv2.drawContours(circularity_filtered_mask, [contour], -1, 255, -1)
            if circularity_diff < best_circularity_diff:
                best_circularity_diff = circularity_diff
                best_candidate = contour

    # final detection result on top of frame1
    result = frame1.copy()
    if best_candidate is not None:
        x, y, w, h = cv2.boundingRect(best_candidate)
        padding = 2
        cv2.rectangle(result,
                      (max(0, x - padding), max(0, y - padding)),
                      (min(result.shape[1], x + w + padding),
                       min(result.shape[0], y + h + padding)),
                      (0, 255, 0), 2)
        # also print the circularity difference under the detection
        cv2.putText(result, f"Circularity: {best_circularity_diff:.2f}",
                    (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # if we want to visualize the intermediate steps, display in a grid␣
↪sequentially
    if visualize:
        fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(20, 8))
        ax[0, 0].imshow(cv2.cvtColor(frame1, cv2.COLOR_BGR2RGB))
        ax[0, 0].set_title('Original Frame Image')
        ax[0, 0].axis('off')

        ax[0, 1].imshow(binary_diff, cmap='gray')
        ax[0, 1].set_title('Binary Difference')
        ax[0, 1].axis('off')

        ax[0, 2].imshow(brown_mask, cmap='gray')
        ax[0, 2].set_title('Filtered by Hue')
        ax[0, 2].axis('off')

        ax[1, 0].imshow(area_filtered_mask, cmap='gray')
        ax[1, 0].set_title('Filtered by Area')
        ax[1, 0].axis('off')

        ax[1, 1].imshow(combined_mask, cmap='gray')
        ax[1, 1].set_title('Morphological Closing')
        ax[1, 1].axis('off')

        ax[1, 2].imshow(cv2.cvtColor(result, cv2.COLOR_BGR2RGB))
```

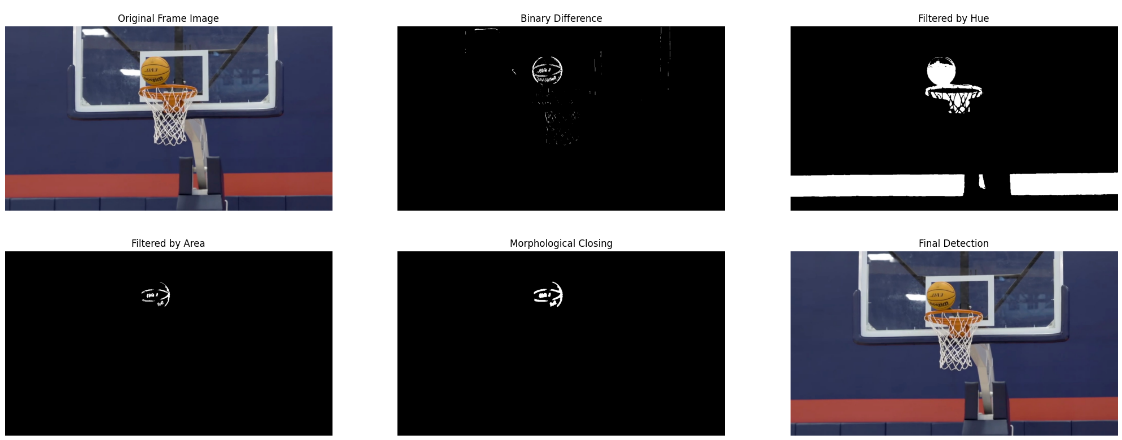
```
        ax[1, 2].set_title('Final Detection')
        ax[1, 2].axis('off')

        plt.tight_layout()
        plt.subplots_adjust(wspace=0.2, hspace=0.1)
        plt.savefig(intermediate_file)
        plt.show()

    return result
```

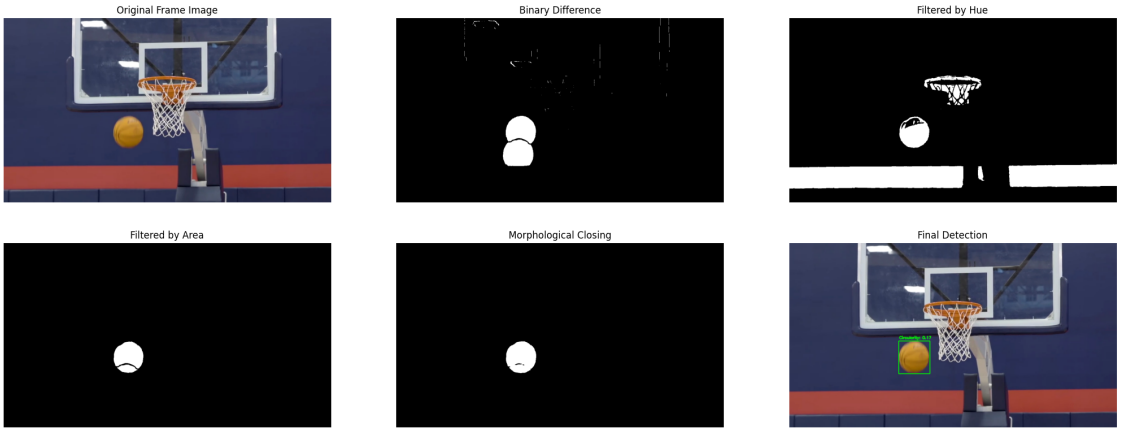
[9]:
```
# helper function to produce and save intermediate viz for a specific video,␣
 ↪frame number
def detect_basketball_in_frame(frames_dir, frame_number, save_name,␣
 ↪min_area=50, circularity_threshold=0.25):
    frame1_path = os.path.join(frames_dir, f"frame_{frame_number:04d}.jpg")
    frame2_path = os.path.join(frames_dir, f"frame_{frame_number + 1:04d}.jpg")
    # if frame one or two is missing, throw exception
    if not os.path.exists(frame1_path) or not os.path.exists(frame2_path):
        raise ValueError(f"Frame {frame_number} or {frame_number + 1} is␣
 ↪missing")
    frame1 = cv2.imread(frame1_path)
    frame2 = cv2.imread(frame2_path)

    intermediate_file = os.path.join("../results/intermediate", save_name)
    result = detect_basketball_with_visualization(frame1, frame2,
                                                  visualize=True,
                                                  min_area=min_area,
                                                 ␣
 ↪circularity_threshold=circularity_threshold,
                                                 ␣
 ↪intermediate_file=intermediate_file)

    result_file = os.path.join("../results/images", save_name)
    cv2.imwrite(result_file, result)
```

[10]:
```
# save frame results from free throws urichmond clip
detect_basketball_in_frame("../data/free_throws_urichmond_frames", 36,␣
 ↪"free_throws_urichmond_frame36.jpg") # bad detection
detect_basketball_in_frame("../data/free_throws_urichmond_frames", 310,␣
 ↪"free_throws_urichmond_frame310.jpg") # good detection
detect_basketball_in_frame("../data/free_throws_urichmond_frames", 167,␣
 ↪"free_throws_urichmond_frame167.jpg") # bad detection
detect_basketball_in_frame("../data/free_throws_urichmond_frames", 180,␣
 ↪"free_throws_urichmond_frame180.jpg") # good detection
```

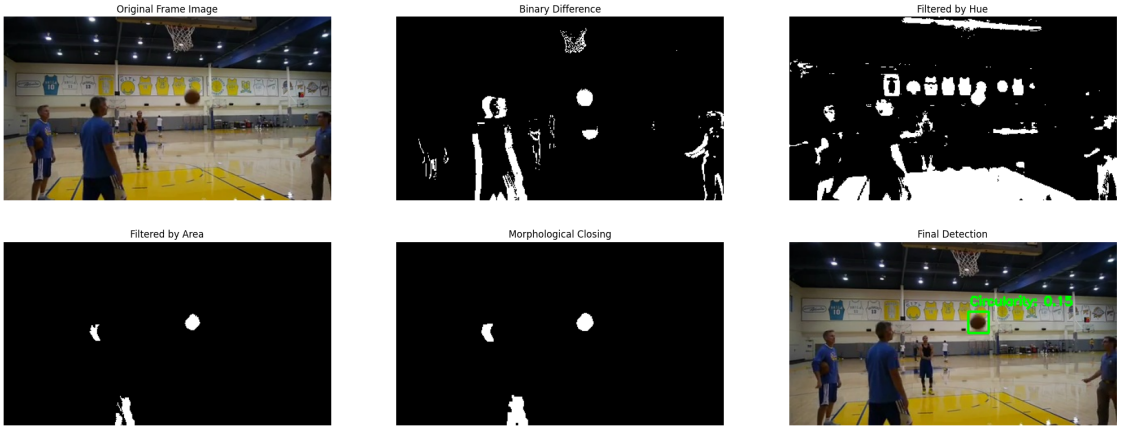

Original Frame Image
Binary Difference
Filtered by Hue
22
Filtered by Area
Morphological Closing
Final Detection

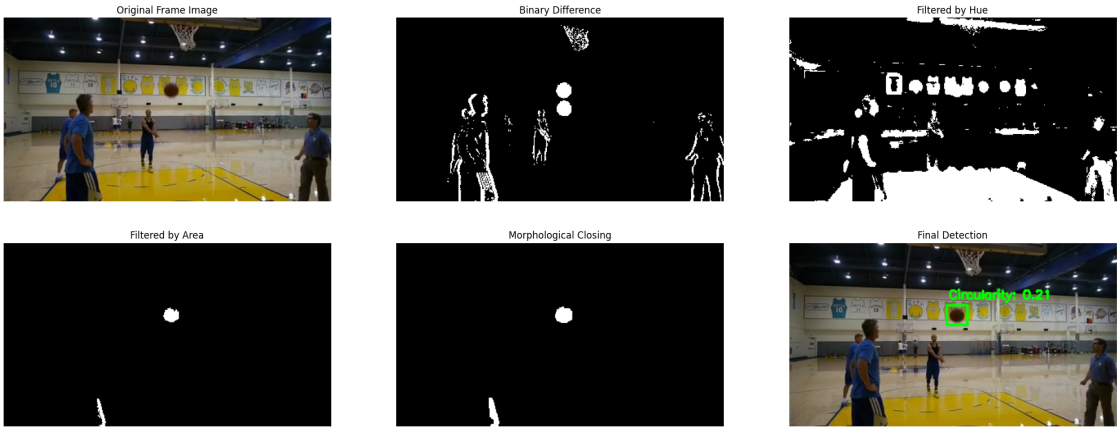

Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection

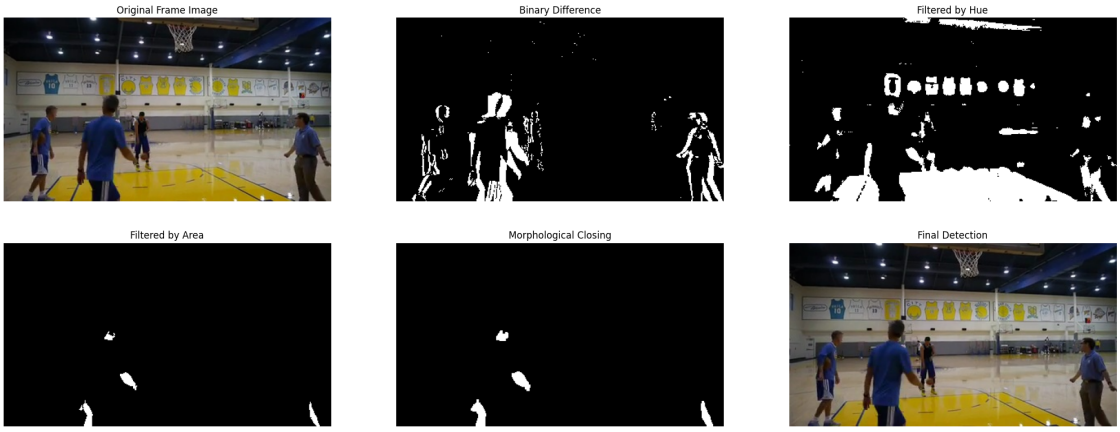

Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection

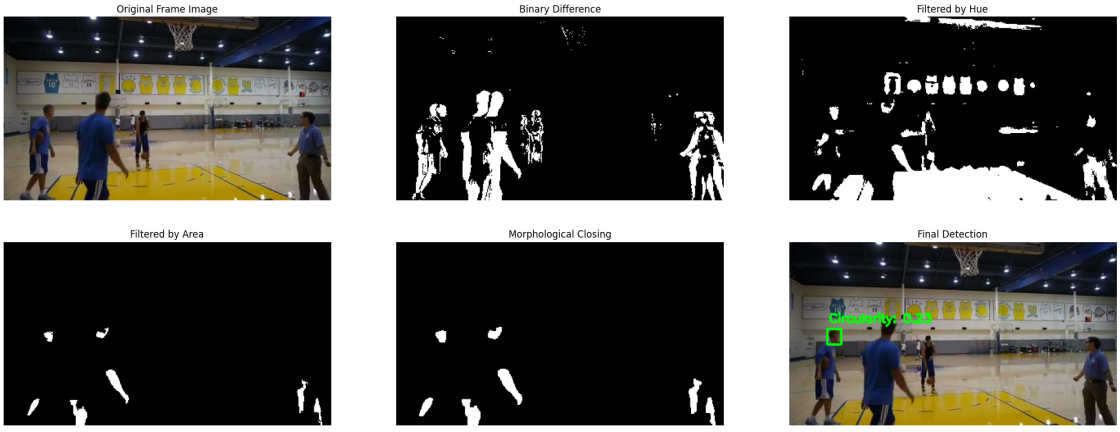


[11]:
```
# save frame results from curry kerr free throw clip
detect_basketball_in_frame("../data/curry_kerr_free_throws_frames", 30,␣
 ↪"curry_kerr_free_throws_frame30.jpg") # good detection
detect_basketball_in_frame("../data/curry_kerr_free_throws_frames", 76,␣
 ↪"curry_kerr_free_throws_frame76.jpg") # good detection
detect_basketball_in_frame("../data/curry_kerr_free_throws_frames", 45,␣
 ↪"curry_kerr_free_throws_frame45.jpg") # bad detection
detect_basketball_in_frame("../data/curry_kerr_free_throws_frames", 46,␣
 ↪"curry_kerr_free_throws_frame46.jpg") # bad detection (false pos.)
```

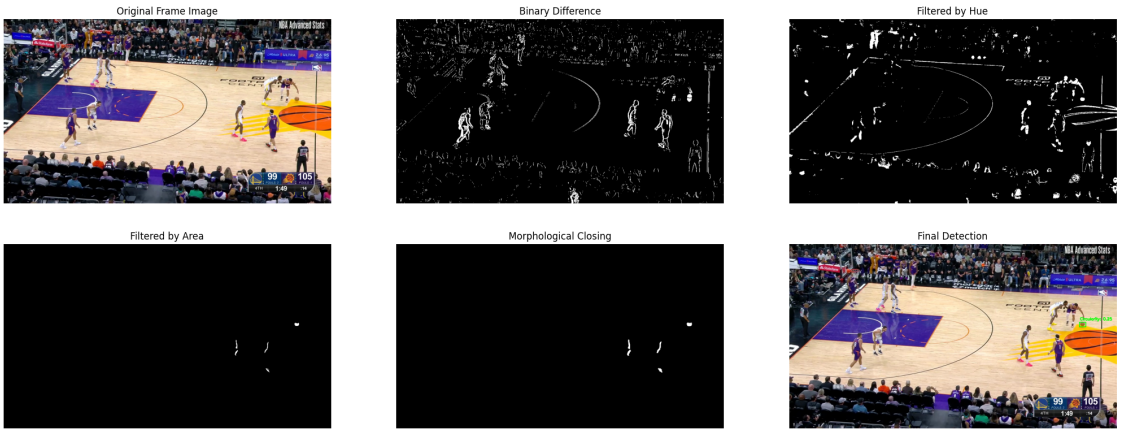


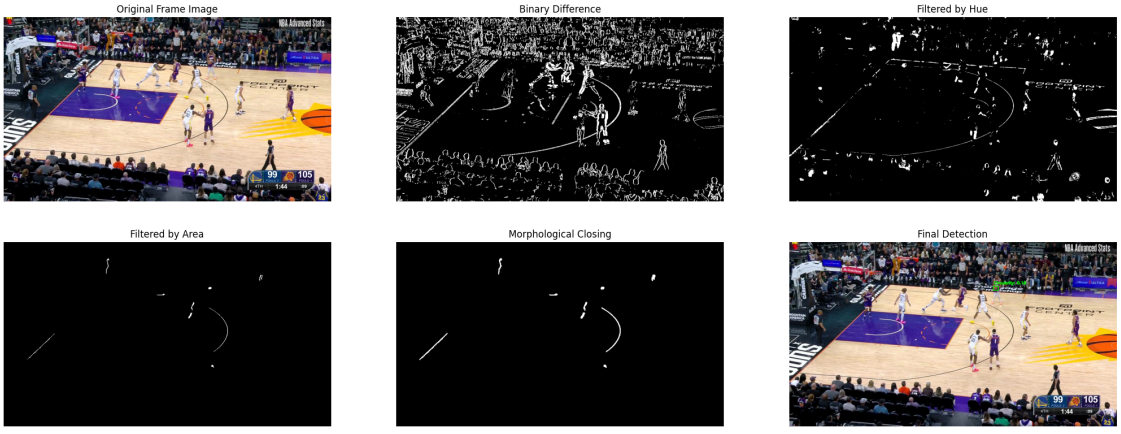

Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection
Circularity: 0.21


Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection


Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection

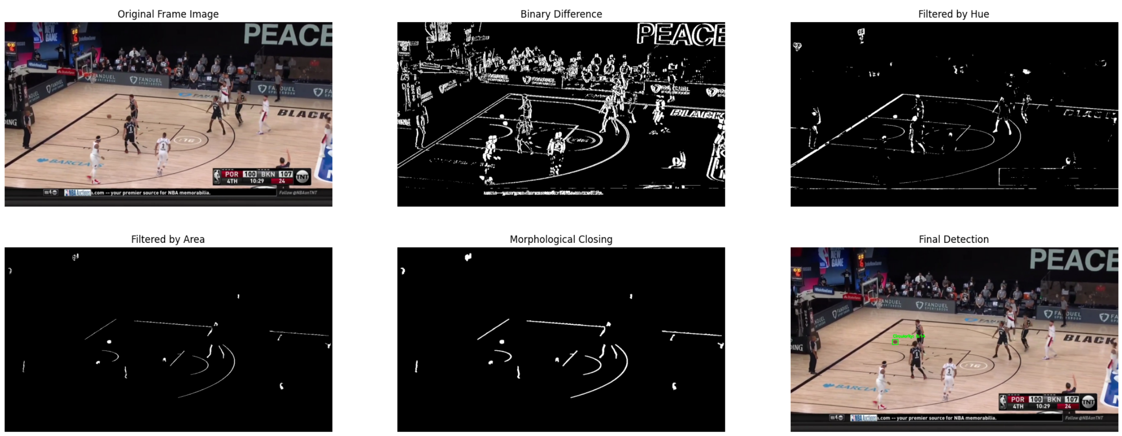
[12]:
```
# save frame results from KD 3PT clip
detect_basketball_in_frame("../data/suns_warriors_kd3_frames", 14, "kd3_frame14.
 ↪jpg") # good detection
detect_basketball_in_frame("../data/suns_warriors_kd3_frames", 176,␣
 ↪"kd3_frame176.jpg") # good detection
detect_basketball_in_frame("../data/suns_warriors_kd3_frames", 1, "kd3_frame1.
 ↪jpg") # bad detection (nothing found)
detect_basketball_in_frame("../data/suns_warriors_kd3_frames", 102,␣
 ↪"kd3_frame102.jpg") # bad detection (false positive)
```



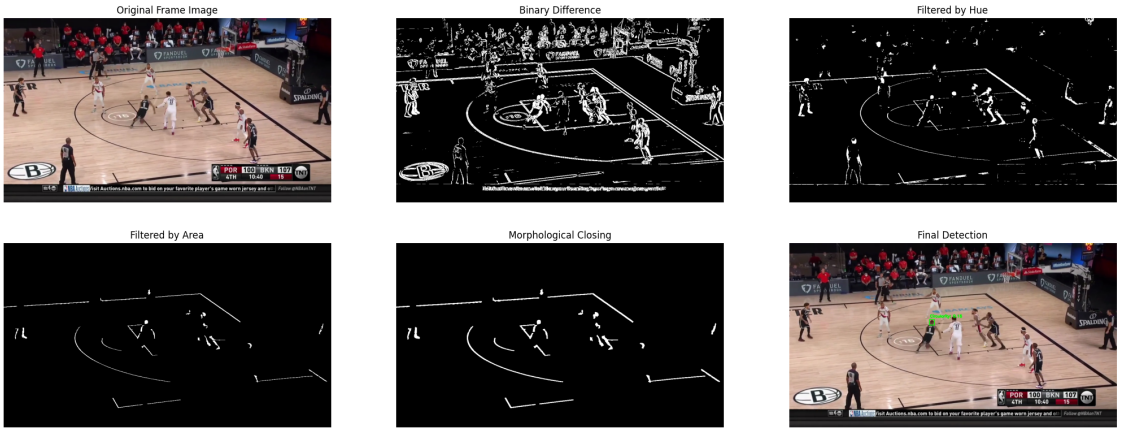


[13]:
```
# save frame results from dame logo three clip
detect_basketball_in_frame("../data/dame_logo_three_frames", 275,␣
 ↪"dame_logo_three_frame275.jpg") # good detection
detect_basketball_in_frame("../data/dame_logo_three_frames", 295,␣
 ↪"dame_logo_three_frame295.jpg") # good detection
detect_basketball_in_frame("../data/dame_logo_three_frames", 2,␣
 ↪"dame_logo_three_frame2.jpg") # bad detection (nothing found)
detect_basketball_in_frame("../data/dame_logo_three_frames", 37,␣
 ↪"dame_logo_three_frame37.jpg") # bad detection (false positive)
```

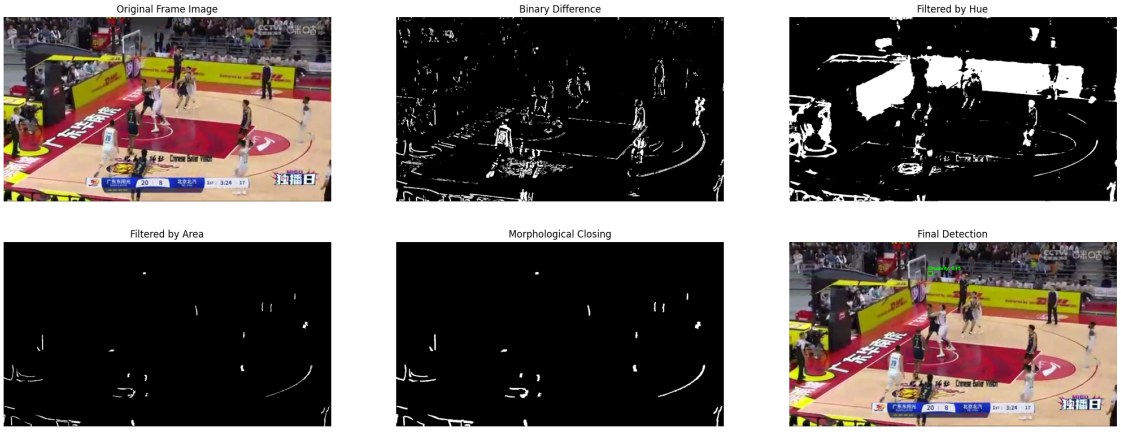

Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection

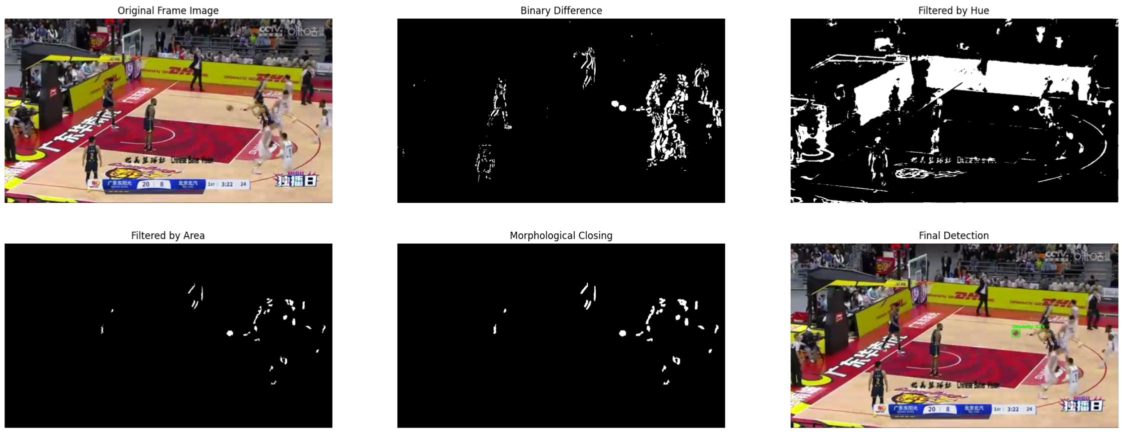

Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection

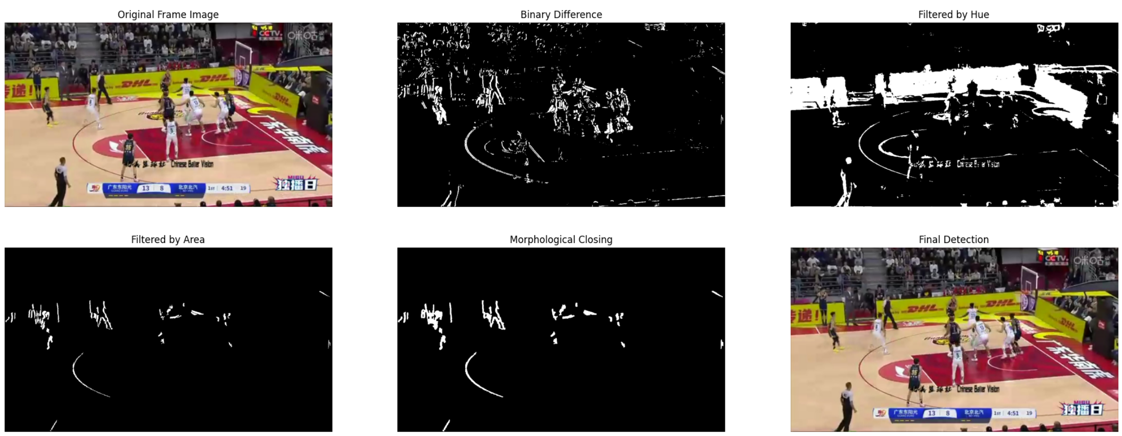

Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection

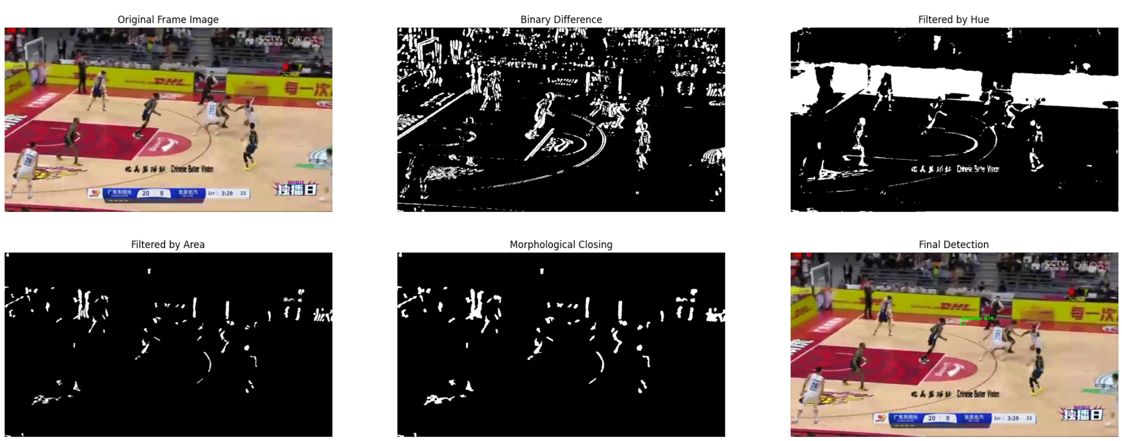


[14]:
```
# save frame results from guangdong beijing clip
detect_basketball_in_frame("../data/guangdong_beijing_frames", 220,␣
 ↪"guangdong_beijing_frame220.jpg") # good detection
detect_basketball_in_frame("../data/guangdong_beijing_frames", 272, "gu␣
 ↪angdong_beijing_frame272.jpg") # good detection
detect_basketball_in_frame("../data/guangdong_beijing_frames", 1,␣
 ↪"guangdong_beijing_frame1.jpg") # bad detection (false positive)
detect_basketball_in_frame("../data/guangdong_beijing_frames", 128,␣
 ↪"guangdong_beijing_frame128.jpg") # bad detection (false positive)
```




Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection


Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection


Original Frame Image
Binary Difference
Filtered by Hue
Filtered by Area
Morphological Closing
Final Detection

1.1.3 Frame Processing and Video Stitching

[15]:
```
def process_frames_to_video(frames_dir, output_dir, output_name, fps,␣
↪min_area=50, circularity_threshold=0.3):
    """
    Perform basketball detection on a full set of frames in a directory and␣
↪save to a video

    Args:
        frames_dir: directory containing frames extracted via extract_frames
        output_dir: directory to save output video
        output_name: name of output video file
        fps: frames per second of output video, should match fps used in␣
↪extract_frames
        circularity_threshold (optional): threshold for circularity difference␣
↪to keep a contour
    """
    # get list of frame paths
    frame_paths = sorted(Path(frames_dir).glob('*.jpg'))
    if len(frame_paths) == 0:
        raise ValueError(f"No frames found in directory: {frames_dir}")

    # load first frame to get dimensions
    frame = cv2.imread(str(frame_paths[0]))
    height, width, _ = frame.shape

    # initialize video writer (MP4 codec)
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    output_path = os.path.join(output_dir, output_name)
    out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

    for i in range(len(frame_paths) - 1):  # -1 because we need pairs of frames␣
↪for the function to work
        frame1 = cv2.imread(str(frame_paths[i]))
        frame2 = cv2.imread(str(frame_paths[i + 1]))
        result = detect_basketball_with_visualization(frame1, frame2,␣
↪min_area=min_area, circularity_threshold=circularity_threshold)
        out.write(result)

    out.release()
    print(f"Video saved to: {output_path}")
```

[16]:
```
# process urichmond free throw clip video, higher min_area because the ball is␣
↪larger in frame
ft_frames_dir = "../data/free_throws_urichmond_frames"
output_dir = "../results/videos"
output_name = "free_throws_urichmond_detections.mp4"
```

```
if os.path.exists(os.path.join(output_dir, output_name)):
    os.remove(os.path.join(output_dir, output_name))
process_frames_to_video(ft_frames_dir, output_dir, output_name, fps=20,␣
 ↪min_area=500)
```

```
Video saved to: ../results/videos/free_throws_urichmond_detections.mp4
```

[17]:
```
# process curry kerr free throw clip video
ft_frames_dir = "../data/curry_kerr_free_throws_frames"
output_dir = "../results/videos"
output_name = "curry_kerr_free_throws_detections.mp4"
if os.path.exists(os.path.join(output_dir, output_name)):
    os.remove(os.path.join(output_dir, output_name))
process_frames_to_video(ft_frames_dir, output_dir, output_name, fps=11)
```

```
Video saved to: ../results/videos/curry_kerr_free_throws_detections.mp4
```

[18]:
```
# process kd 3 pt clip
kd_frames_dir = "../data/suns_warriors_kd3_frames"
output_dir = "../results/videos"
output_name = "suns_warriors_kd3_detections.mp4"
if os.path.exists(os.path.join(output_dir, output_name)):
    os.remove(os.path.join(output_dir, output_name))
process_frames_to_video(kd_frames_dir, output_dir, output_name, fps=30,␣
 ↪min_area=75)
```

```
Video saved to: ../results/videos/suns_warriors_kd3_detections.mp4
```

[19]:
```
# process dame logo three clip
dame_frames_dir = "../data/dame_logo_three_frames"
output_dir = "../results/videos"
output_name = "dame_logo_three_detections.mp4"
if os.path.exists(os.path.join(output_dir, output_name)):
    os.remove(os.path.join(output_dir, output_name))
process_frames_to_video(dame_frames_dir, output_dir, output_name, fps=25)
```

```
Video saved to: ../results/videos/dame_logo_three_detections.mp4
```

[20]:
```
# process guangdong vs beijing clip, could experiment with lower circularity␣
 ↪threshold
gd_frames_dir = "../data/guangdong_beijing_frames"
output_dir = "../results/videos"
output_name = "guangdong_beijing_detections.mp4"
if os.path.exists(os.path.join(output_dir, output_name)):
    os.remove(os.path.join(output_dir, output_name))
process_frames_to_video(gd_frames_dir, output_dir, output_name, fps=25,␣
 ↪circularity_threshold=0.2)
```

```
Video saved to: ../results/videos/guangdong_beijing_detections.mp4
```

[21]:
```
# To obtain the final video (videos/compiled_detections.mp4), used a video␣
↪editor to combine all the individual videos together
```

[ ]:
```

```