Computer Vision – Assignment 1

Omar Ali - 28587497

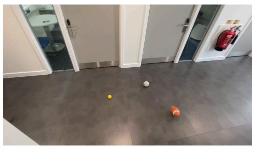
2024-05-07

Problem 1: Image Segmentation and Detection

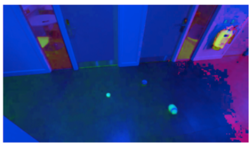
In this task, 3 balls from some images needed to be extracted from a set of 62 images and segmented into a mask. The balls were of three types: American Football, Football, and Tennis Ball. The images represent a sequence or frames where the balls are in motion. The task was to segment the balls from the background and calculate the Dice Similarity Score (DS) of the segmented balls against the ground truth mask of the balls.

1.1 Ball Segmentation

In this section, I will demonstrate how the balls were segmented from the background using a series of image-processing techniques. The steps are as follows:



Original
Step 1
Grab the original image from path

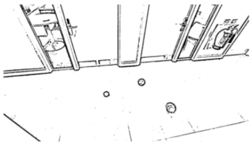


HSV
Step 2
Converted the RGB image into HSV colour space

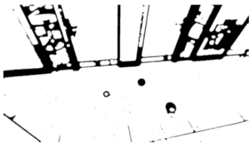
Intensity
Step 3
Extracted the Intensity Channel

GBlur
Step 4
Apply Gaussian Blur with k=[3x3] with 2 iterations

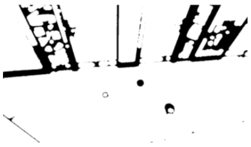
Median Blur
Step 5
Apply Median Blur with k=[3x3] with 2 iterations



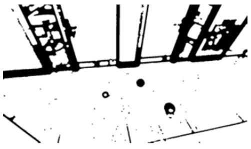
Adaptive Gaussian Threshold
Step 6
With Blocksize = 19 and c = 5



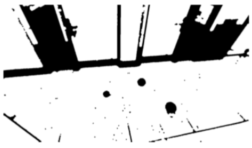
Opening
Step 7
to disconnecting white pixels k = (4,4) with 5 iterations



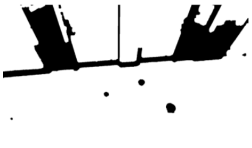
Dilate
Step 8
expanding white pixels to remove noisy black holes holes k = (3,3) with 4 iterations



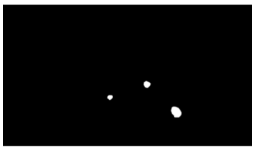
Erode
Step 9
re-shrinking the white pixels to let the balls begin to connect k = (3,3) with 2 iterations



Fill
Step 10
Fill the enclosed black areas



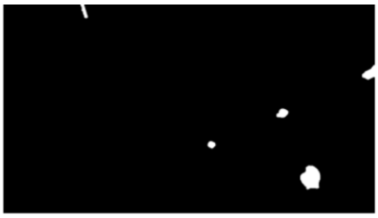
Dilate and Erode
Step 11
series of dilations and erosions to remove the noisy black pixels whilst maintaining *



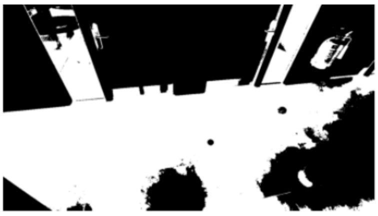
Contour
Step 12
find the contours of the (bitwise_not) image

At this stage, step 12, it can be seen that the balls have been segmented from the background. However, this is only the case in the particular sample, frame 85. In the images below, Frame 100 is re-displayed to show that the ball segmentation still had artifacts that needed to be removed. Two main culprits were the shadows of the American Football in the latter frames, as well as some sections of the background that were not yet caught by the segmentation process so far.

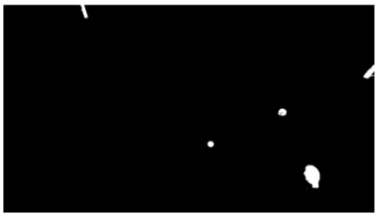
In order to remove the shadow, it was found that the intensity channel of the HSV image was quite effective and computationally cheap to apply.



Contour
Step 12
Grab the original image from path



Thresholing Intensity
Step 13.1
A threshold of 0.4 was applied to the intensity image from step 3 to remove shadows



Combine
Step 13.2
The contour and the thresholded intensity image were combined

The inverse of the thresholded intentity, in step 13.1, was combined with the contour using a 'bitwise and' operation.

```python
combo = cv2.bitwise_not(cv2.bitwise_or(thresh_intensity_image, contour_image))
```

Convex Hull
Step 14

Circularity
Step 15

Finally, a convex hull was applied to the contours to remove sharp edges and improve the circularity of the contours. The circularity of the contours was calculated and only contours with a circularity greater than 0.72 were kept.

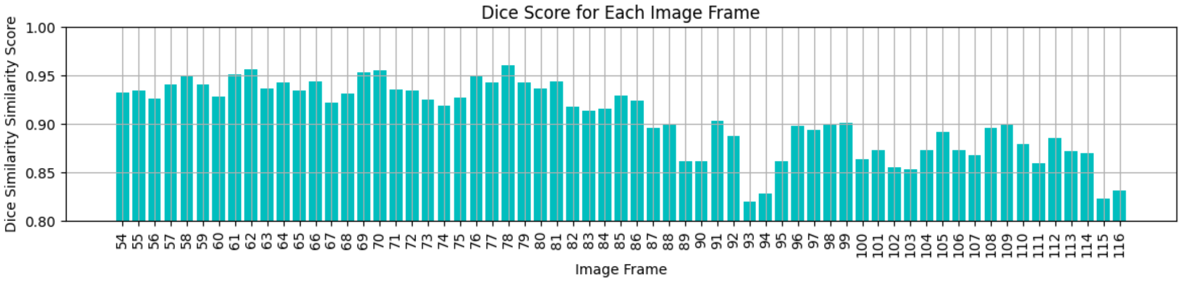
1.2 Dice Similarity Score

The Dice Similarity Score (DS) of every segmented ball image was compared against the ground truth mask of the ball image. The Dice Similarity Score (DS) is defined as:

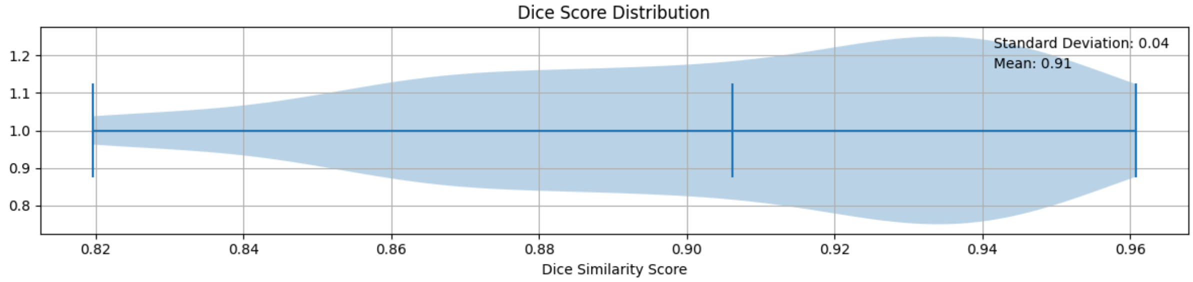
$$\text{DS} = \frac{2 * |M \cap S|}{|M| + |S|}$$

where M is the ground truth mask and S is the segmented mask. The DS score ranges from 0 to 1, where 1 indicates a perfect match between the two masks.

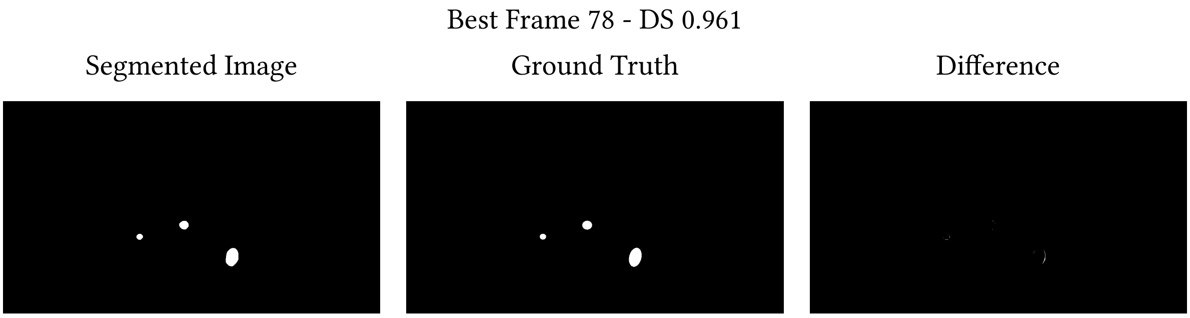
A bar chart of the DS has been plotted for every frame. The lowest recorded score is 0.82 for frame 93 and the highest is 0.96 for frame 78.



Additionally, a violin plot, was used to display the distribution of the DS score across all the frames, it can be seen that there is quite a skew of the data towards the higher range of the DS score, with the greatest proportion of the data sitting around the 0.93-0.94 range.



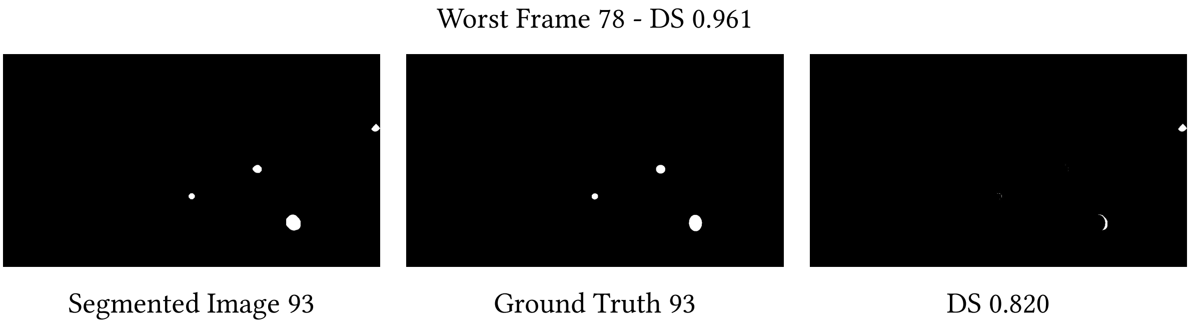
Below are the best and worst frames in terms of the Dice Similarity Score achieved by the segmentation process.


Best Frame 78 - DS 0.961

Segmented Image

Ground Truth

Difference


Worst Frame 78 - DS 0.961

Segmented Image 93

Ground Truth 93

DS 0.820

1.3 Discussion

The implementation was found to be quite good, where the DS scores had a maximum of 0.96, minimum of 0.82, mean of 0.91 and standard deviation of 0.04. However, the implementation is quite clunky and very tuned for this task.

Overall, the number of steps in order to reach the solution is quite large, at 15 steps. The additional processing was ultimately due to a whack-a-mole situation, where refinements in one area of the task cause another area to worsen. This makes this solution temperamental and not very robust to changes in the input data, however, for very refined results, this is likely to be a similar case for most image segmentation methodologies that do not more complex models such as deep learning.

In the initial processing, it was found that the Intensity value provided a very good initial starting point, as opposed to using grayscale, as the intensity of the balls has a very distinct value compared to the background. The Gaussian Blur and Median Blur were very effective at removing the nose from this image, whilst maintaining the edges of the balls. The Adaptive Gaussian Threshold (AGT) was effective in detecting the edges in the image, compared to a standard threshold, because it takes into consideration a normalised local area for its thresholding calculation.

The range of morphological filters was essential in getting a cleaner mask of the balls, with just the right amount of erosion and dilation to remove the noise from the background. Different ranges of kernel sizes were used for the erosion and dilation, however all used some scale of the `MORPH_ELLIPSE` structuring element. It is important to note that in the first applications of the morphological filters, the region being eroded or dilated was the background, not the balls. This was because the image had not yet been inverted and so the balls were considered the background.

Once the morphological filtering was complete, the balls were segmented using the `opencv.findContours` function. As mentioned, this was found to not be refined enough, so the intensity channel was used to remove the shadows from the American Football, a convex hull was applied to the contours to connect sharp edges, and the circularity of the contours where only contours with a circularity greater than 0.72 were kept.

There are certainly some improvements that could be made, and would have possibly decided to take a different approach had this task been reattempted. A possible alternative first step could be to make use of the colour channels of the image to segment the balls. This could be done by applying a threshold to the colour channels and then combining the results. This would have been a more robust approach where the main challenge would be the distinction of the white football from the walls of the room. Additionally, considering the constrained environment, it could have been possible to create a mask of the problematic parts of the room and have these be removed from the image before segmentation.

Problem 2: Feature Calculation

2.1 Shape Features

In this section, I will demonstrate how a range of shape features (Solidity, Non-compactness, Circularity, Eccentricity) can be calculated from the contours of the segmented balls.

In order to get the contours of the segmented balls, I made use of the openCV 'findContours' function which returns a list of contours and a hierarchy for all the contours in the image.

```
img = cv2.imread(image)
img = cv2.bitwise_and(cv2.cvtColor(msk, cv2.COLOR_GRAY2BGR), img)
contours, _ = cv2.findContours(msk, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

Each contour is split into the corresponding balls in order to combine the contours of the same ball into an array, this is done by exploiting the areas of each ball. The details of the code can be explored as needed, however the core logic is as follows:

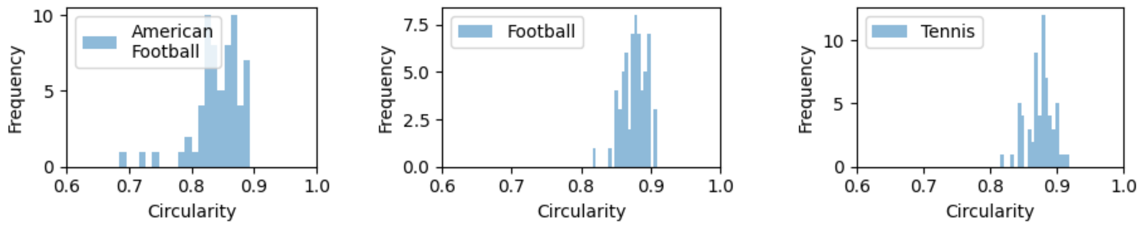
```python
area = cv2.contourArea(contour)
if area > 1300:  # football
    append_ball = BALL_LARGE
elif area > 500:  # soccer_ball
    append_ball = BALL_MEDIUM
else:  # tennis ball
    append_ball = BALL_SMALL
```

Using this, the features can be evaluated for every ball.

2.1.a Circularity

Circularity is defined as the ratio of the area of the circle with the same perimeter as the object to the area of the object. For a given contour C, the circularity is calculated by:

```python
area = cv2.contourArea(contour)
perimeter = cv2.arcLength(contour, closed=True)
circularity = (4 * math.pi * area) / (perimeter**2)
```

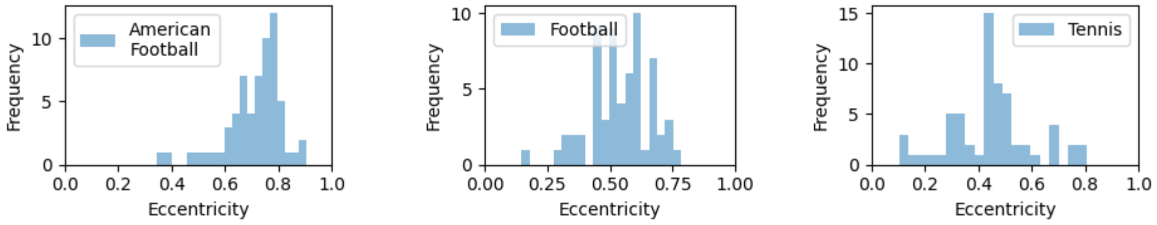


Both the Football and the Tennis ball have a higher circularity than the American Football. This is expected as the American Football has a more elongated shape compared to the other two balls. Visually, the football has a smaller variance in circularity compared to the tennis ball. This is likely due to the relative size of the football compared to the tennis ball, where the football is larger and has a more consistent shape from the perspective of an image and will not suffer from distortion and be impacted by smaller pixel ranges.

2.1.b Eccentricity

Eccentricity is the ratio of the distance between the foci of the ellipse to the major axis length. For a given contour C, the eccentricity is calculated by:

```python
ellipse = cv2.fitEllipse(contour)
a = max(ellipse[1])
b = min(ellipse[1])
eccentricity = (1 - (b**2) / (a**2)) ** 0.5
```

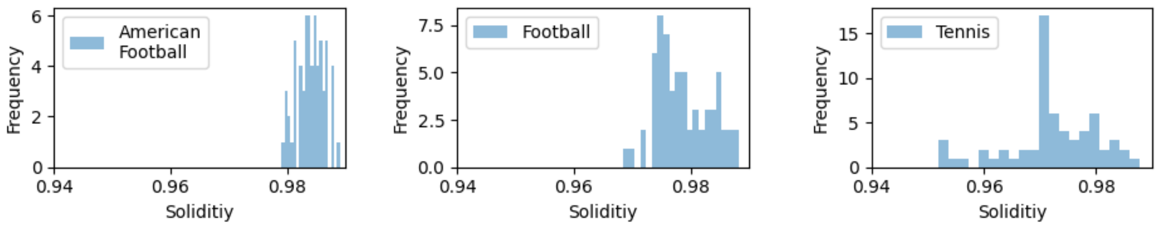


The American Football has the highest eccentricity of all the balls, which is expected as it has a more elongated shape compared to the other two balls. The football and the tennis ball both have very distributed eccentricity values.

2.1.c Solidity

Solidity is the ratio of the area of the object to the area of the convex hull of the object. For a given contour C, the solidity is calculated by:

```python
area = cv2.contourArea(contour)
convex_hull = cv2.convexHull(contour)
convex_area = cv2.contourArea(convex_hull)
solidity = area / convex_area
```

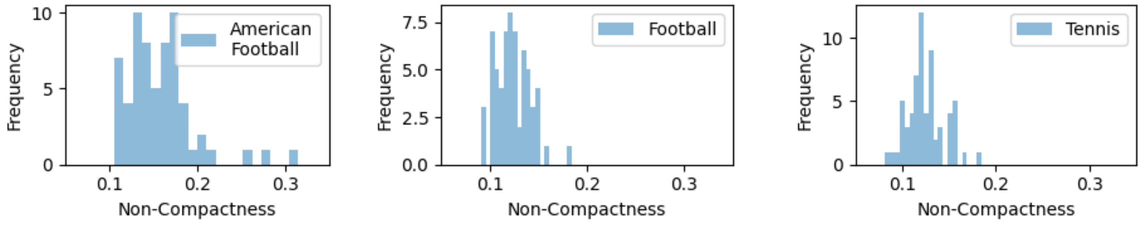


The solidity of the American Football is much higher and more consistent than the other two balls. This is likely due to the ball being larger in size and so a convex hull around the ball is likely to be more similar to the ball itself. This follows through as we see the football having a higher solidity than the tennis ball, and the tennis ball's solidity is much more distributed than the others.

2.1.d Non-compactness

Non-compactness is the ratio of the area of the object to the area of the circle with the same perimeter as the object. For a given contour C, the non-compactness is calculated by:

```python
area = cv2.contourArea(contour)
perimeter = cv2.arcLength(contour, closed=True)
non_compactness = 1 - (4 * math.pi * area) / (perimeter**2)
```



The Football has the most consistent non-compactness values, [[IDK add something about non compactness]]

2.2 Texture Features

In this section, texture features are calculated from the segmented balls. The texture features are evaluated by calculating the normalised Grey-Level Co-occurrence Matrix (GLCM) in four orientations (0°, 45°, 90°, 135°) for every individual ball. The GLCM is a matrix that describes how often different combinations of pixel intensity values occur in an image. The GLCM is calculated for each of the colour channels (red, green, blue) and for each of the four orientations then averaged across the orientations to determine the texture features of the ball. These features include Angular Second Moment, Contrast, Correlation, and Entropy.

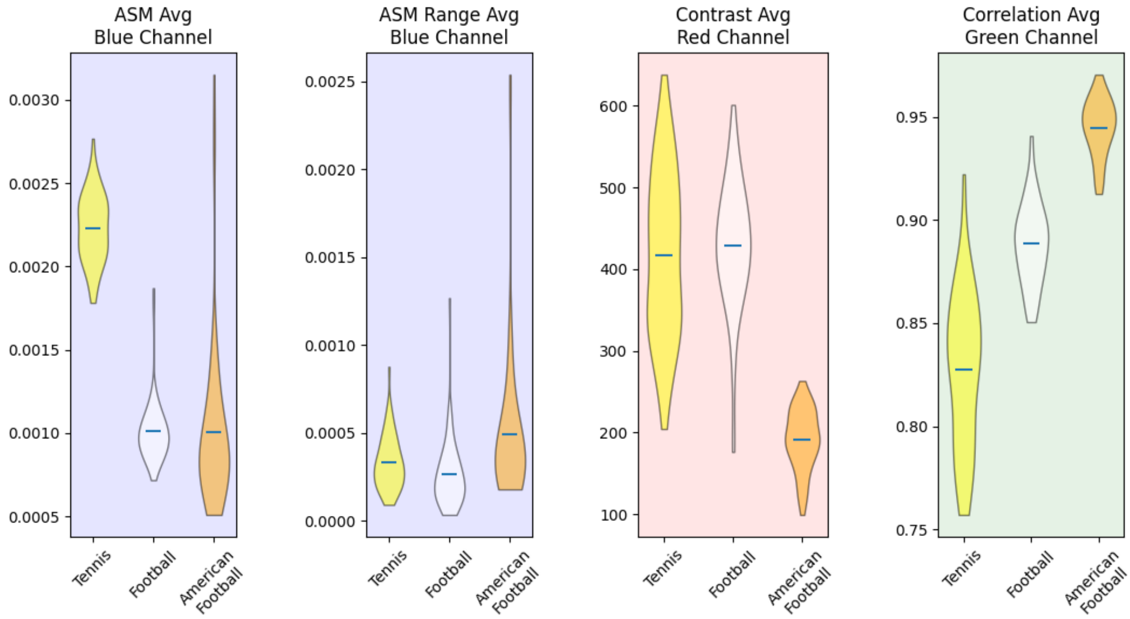
For each feature, one colour channel is selected to demonstrate the feature calculation. The blue channel was selected for the Angular Second Moment, the red channel for Contrast, and the green channel for Correlation.

To get the GLCM, the balls were segmented in a similar way to what was described previously in the shape features section. However, this time, the mask generated was overlayed onto the original image to get the pixel values of the balls.

The GLCM was calculated using the 'greycomatrix' function from the skimage library. The first row and column were both stripped away from the GLCM to remove the background noise. The GLCM was then normalised using 'greycoprops' and the texture features were evaluated for all four orientations. These were then averaged to get the final average texture value for the ball.

2.2.a Applied Texture Features

Below are a set of violin plot of the texture featuures for the three balls. The Angular Second Moment was calculated for the blue channel, the Contrast for the red channel, and the Correlation for the green channel, where the yellow plot represents the Tennis Ball, the white plot represents the Football, and the orange plot represents the American Football.



The **ASM**, is a measure of textural uniformity within an image. The ASM will be high when the image has constant or repetitive patterns, meaning the pixel intensities are similar or identical throughout the image. The yellow tennis ball has a high ASM mean in the blue channel, which suggests that the pixel intensities in the blue channel for the yellow tennis ball are very similar or identical throughout the image. This could be due to the yellow color having a low blue component in the RGB color model. The orange American football has a low ASM mean in the blue channel, which suggests that there is a lot of variation or randomness in pixel intensities in the blue channel for the orange American football. This could be due to the orange color having a low blue component in the RGB color model, or due to variations in lighting conditions or reflections on the ball. The large standard deviation indicates that the pixel intensities in the blue channel for the orange American football vary widely. This could be due to factors such as lighting conditions, shadows, or variations in the ball's color.

The **ASM range** in an image indicates the spread of textural uniformity across the image. A high range would indicate that there are areas of the image with very high textural uniformity. The yellow tennis ball has a low ASM range mean in the blue channel. This means that the pixel intensities in the blue channel for the yellow tennis ball are very similar or identical throughout the image.

The **contrast** of an image, specifically in a color channel, refers to the difference in color that makes an object distinguishable. In the red channel of an image, objects with a high amount of red will have a high intensity. The yellow tennis ball and the white football have the highest contrasts means in the red channel because yellow, and white have a combination of red and green in the RGB color model.

The **correlation** of an image, specifically in a color channel, refers to the degree to which neighboring pixel values are related. In the green channel of an image, objects with a high amount of green will have a high intensity. The orange American football has a high correlation mean in the green channel, which suggests that it has a strong relationship between neighboring pixel values in the green channel. This could be due to the orange color having less green component compared to yellow or due to different lighting conditions. The yellow tennis ball has a low correlation mean in the green channel because yellow is a combination of red and green in the RGB color model. This means that the yellow tennis ball will have a high intensity in the green channel, but the correlation is low.

In terms of using the features to classify the balls, the Shape features can be very useful in especially in classifying the American Football as it has the most distinctive shape of the three balls. In particular, the solidity of the American Football has a very high value, and low distribution, compared to the others, making it a particularly distringuishing feature. The texture features can also be useful in identifying the balls but this is colour dependant. The tennis ball tends to have the lowest correlation and the American Football the highest.

Problem 3: Object Tracking

3.1 Kalman Filter

A Kalman filter is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone.

It can be used in the context of object tracking to estimate the position of an object based on noisy measurements of its position. The Kalman filter uses a motion model to predict the next state of the object and an observation model to update the state based on the measurements.

The Kalman filter consists of two main steps: the prediction step and the update step.

These require a few set of parameters to be set up, such as the state vector x, the state covariance matrix P, the motion model F, the observation model H, the process noise covariance matrix Q, and the measurement noise covariance matrix R.

```
# nx=0.16, ny=0.36, nvx=0.16, nvy=0.36, nu=0.25, nv=0.25 x0=[0.0,0.0,0.0,0.0]
F = np.matrix([[1, dt, 0, 0], [0, 1, 0, 0], [0, 0, 1, dt], [0, 0, 0, 1]])
H = np.matrix([[1, 0, 0, 0], [0, 0, 1, 0]])
Q = np.matrix([[nx, 0, 0, 0], [0, nvx, 0, 0], [0, 0, ny, 0], [0, 0, 0, nvy]])
R = np.matrix([[nu, 0], [0, nv]])
x = np.matrix([x0]).T
P = Q
N = len(z[0])
s = np.zeros((4, N))
```

In the prediction step, the filter uses the motion model to predict the next state of the object based on the previous state.

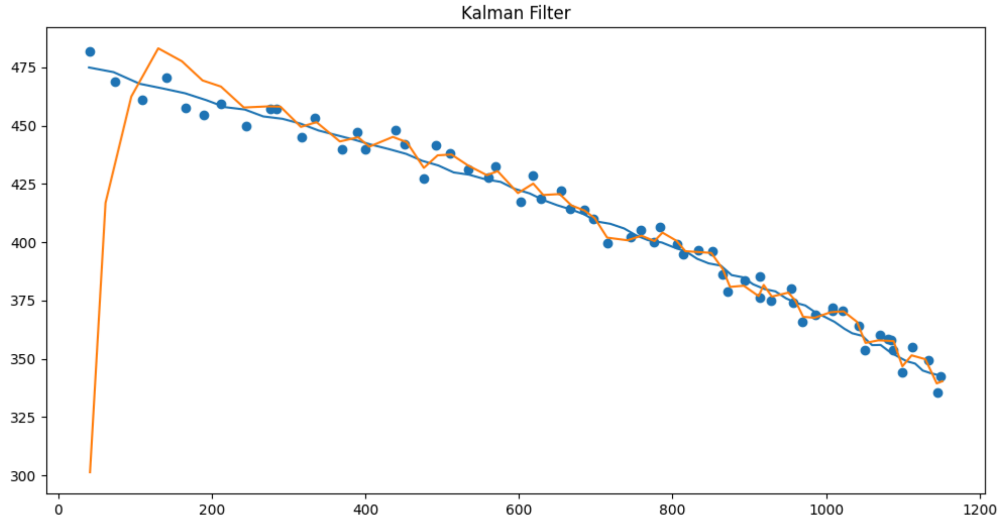
```
def kalman_predict(x, P, F, Q):
    xp = F * x
    Pp = F * P * F.T + Q
    return xp, P

def kalman_update(x, P, H, R, z):
    S = H * P * H.T + R
    K = P * H.T * np.linalg.inv(S)
    zp = H * x
    xe = x + K * (z - zp)
    Pe = P - K * H * P
    return xe, Pe
```

In the update step, the filter uses the observation model to update the state based on the measurements.

```
for i in range(N):
        xp, Pp = kalman_predict(x, P, F, Q)
        x, P = kalman_update(xp, Pp, H, R, z[:, i])
        val = np.array(x[:2, :2]).flatten()
        s[:, i] = val
    px = s[0, :]
    py = s[1, :]
```

The plotted graph of the initial noisy coordinates [na,nb] and the estimated coordinates [x*,y*] can be seen below.



In order to evaluate the performance of the Kalman filter, the root mean squared error (RMSE) can be evaluated to show how close the estimated trajectory is to the ground truth trajectory.

$$\mathrm{RMS} = \sum_{i=0}^{N} \sqrt{(x_i - \mathrm{px}_i)^2 + (y_i - \mathrm{py}_i)^2}$$

```python
mse = []
for i in range(len(x)):
    mse.append(np.sqrt((x[i] - px[i]) ** 2 + (y[i] - py[i]) ** 2))
mean = err.mean()
rmse = np.sqrt(mean)
```

noise:

- mean = 6.962803604693961
- std = 2.5521595512429265
- rms = 2.638712489964369

prediction:

- mean = 9.24
- std =22.05
- rms = 3.03

Compare both noisy and estimated coordinates to the ground truth. Adjust the parameters associated with the Kalman filter, justify any choices of parameter(s) associated with Kalman Filter that can give you better estimation of the coordinates that are closer to the ground truth.

Discuss and justify your findings in the report.

Problem 4: Appendix.

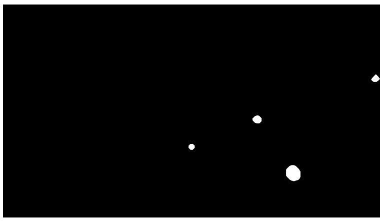
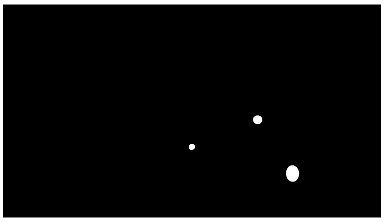
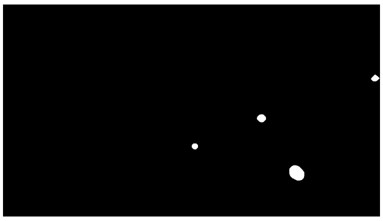
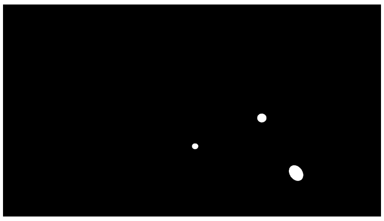
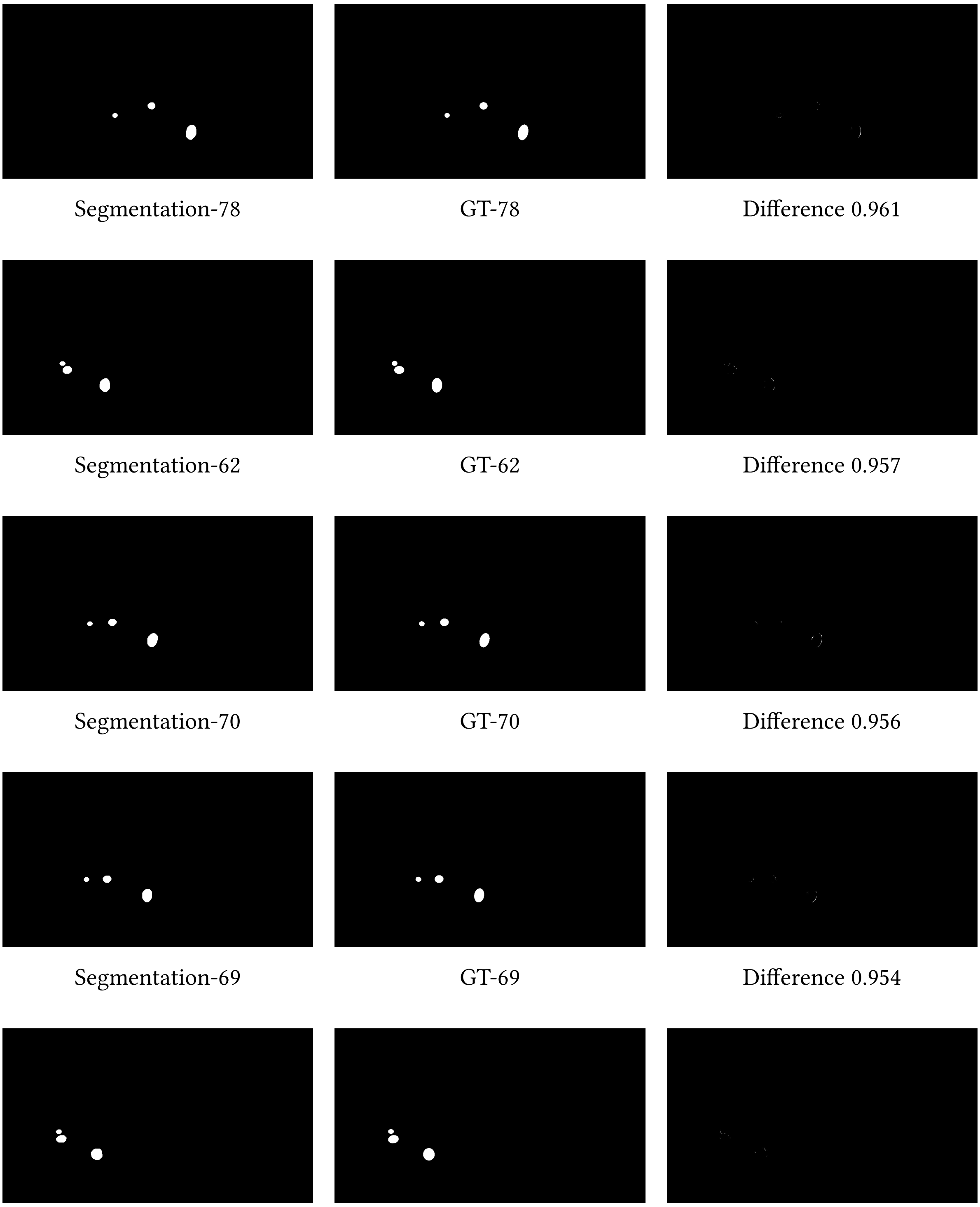
		
Segmentation-93	GT-93	Difference 0.820
Segmentation-115	GT-115	Difference 0.823
		
Segmentation-94	GT-94	Difference 0.829
Segmentation-116	GT-116	Difference 0.831
Segmentation-103	GT-103	Difference 0.853

Table 11: Worst Frames



Segmentation-78 GT-78 Difference 0.961

Segmentation-62 GT-62 Difference 0.957

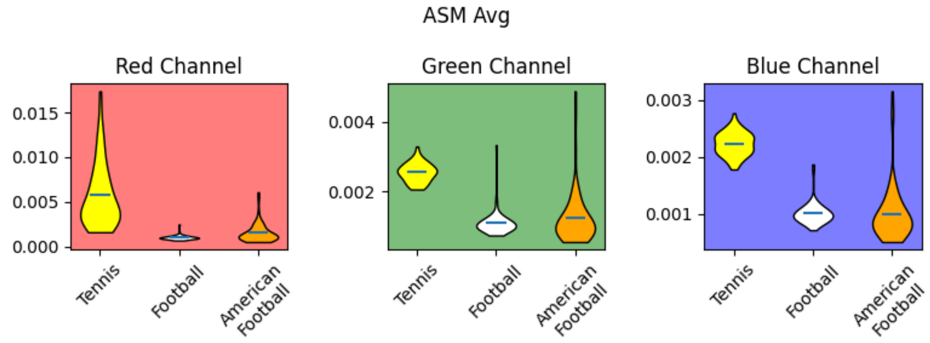
Segmentation-70 GT-70 Difference 0.956

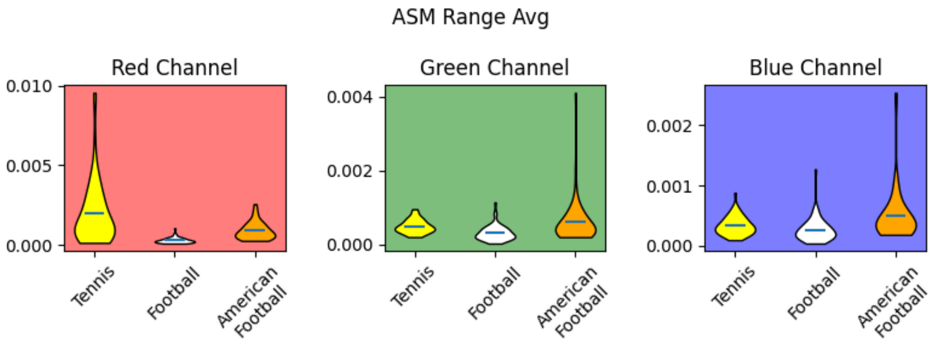
Segmentation-69 GT-69 Difference 0.954

Segmentation-61 GT-61 Difference 0.951

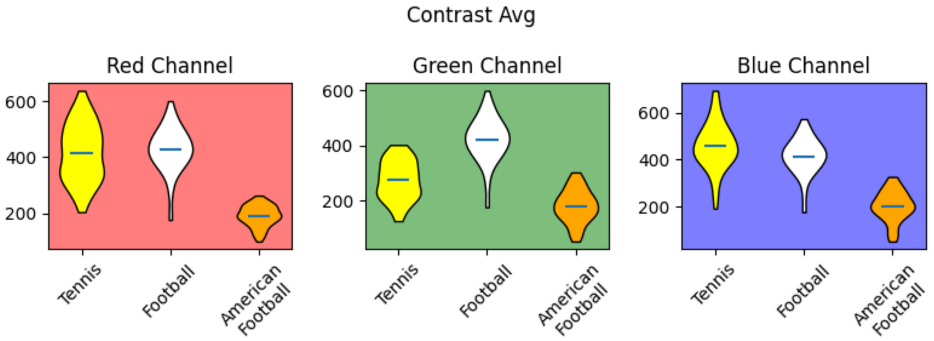
Table 12: Best Frames

4.1 All Features

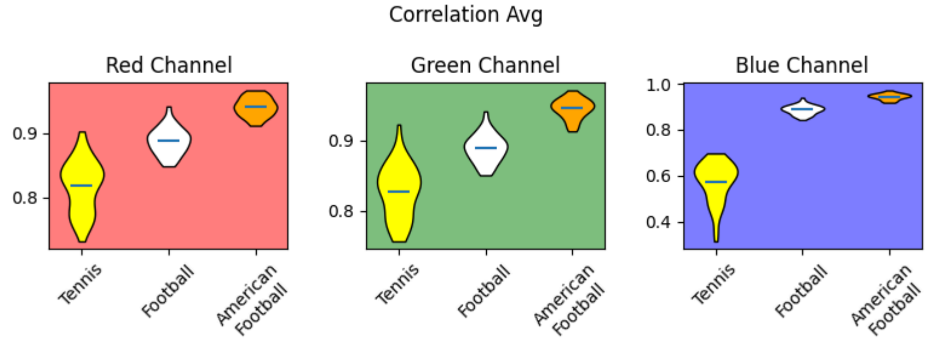

ASM Avg
Red Channel
Green Channel
Blue Channel
0.015
0.010
0.005
0.000
0.004
0.002
0.003
0.002
0.001
Tennis
Football
American Football


ASM Range Avg
Red Channel
Green Channel
Blue Channel
0.010
0.005
0.000
0.004
0.002
0.000
0.002
0.001
0.000
Tennis
Football
American Football

,


Contrast Avg
Red Channel
Green Channel
Blue Channel
600
400
200
600
400
200
600
400
200
Tennis
Football
American Football

,


Correlation Avg
Red Channel
Green Channel
Blue Channel
0.9
0.8
0.9
0.8
1.0
0.8
0.6
0.4
Tennis
Football
American Football

,