CMP9135M — Computer Vision

1st George Davies
School of Computer Science
University of Lincoln
Lincoln, United Kingdom
27421138@students.lincoln.ac.uk

Task 1 — Image Processing

The aim of this task was to segment three different types of balls (a tennis ball, a football, and an american football) from a timeseries of images of them rolling across the frame. No deep learning solutions were allowed.

Task 1.a — Automated ball objects segmentation

1) Otsu thesholding: The first step was to binarise the image. By default, the matlab function imbinarize uses otsu to distinguish between foreground and background.
By ajusting the sensitivity, I found the value that removed the most of the background without removing the balls was 0.35. The imbinarize function only works on grayscale images so I tested parsing each colour channel separately and the red channel seemed to perform the best.
Examples of the output can be seen in Figure 1.

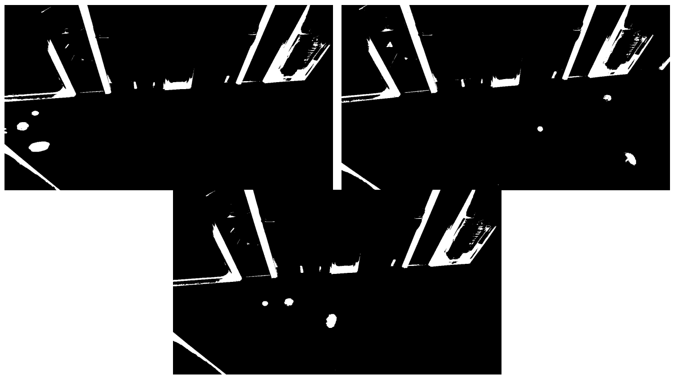
Fig. 1: Examples of the binary image output of the preliminarly segmentation using otsu

2) Converting to convex hulls: To removed the remaining background artifacts, I decided to convert all the connected components into convex hulls.
As the ball are supposed to be convex in the first place, they shouldn't change much (apart from the american football in the last few frames where the otsu segmentation was a bit aggressive).
This had the effect of massivly increasing the area of the background artifacts, making them easily distinguishable from the balls.
Examples of the output can be seen in Figure 2.

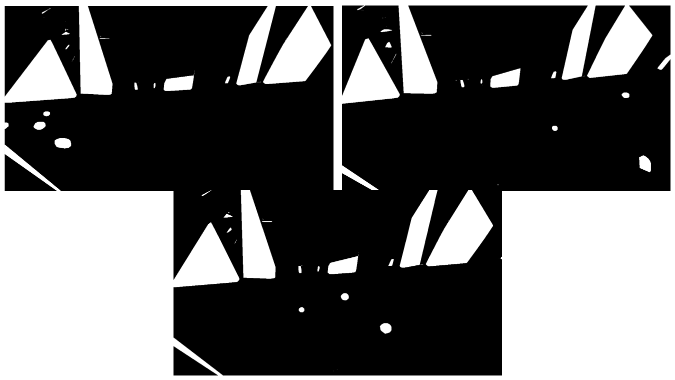
Fig. 2: Examples of the binary image output after transforming each connected component into a convex hull

3) Removing the remaining artifacts: Now, the background artifacts can be removed from the binarised images by removing every connected component that doesn't fit certain characteristics such as area and extent. After this, all that remains are the three balls.

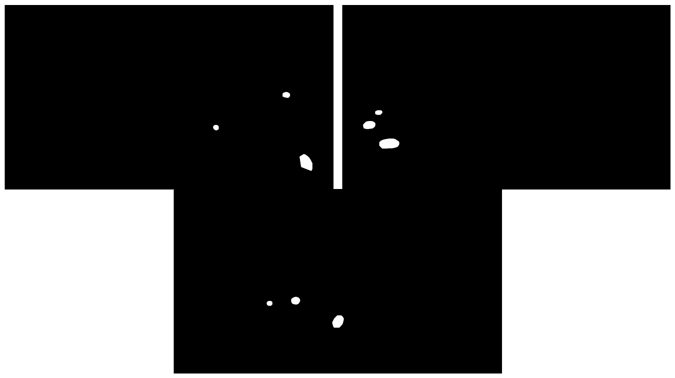
Fig. 3: Examples of the binary image output after transforming each connected component into a convex hull

More examples of the best and worst segmentations can be found in the Appendix Figures 9 and 10

Task 1.b — Segmentation evaluation

To evaluated the segmentation, the Dice Similarity score between the segmented images and the ground truth can be calculated. As each image is simply stored as a matrix of 1s and 0s, it is possible to do binary operations on the matrices.

The DS score is calculated by multiplying the number of 1s in the intersection of the two matrices (logical &) by 2 and dividing that by the total number of 1s in both matrices. If there is a perfect overlap between the images, then the intersection will incompass all the 1s and the DS score will be 1. If there is no overlap, then there is no intersection and the DS score will be 0.

$$\text{Dice}(M,S) = \frac{2|M \cap S|}{|M| + |S|} \tag{1}$$

where:

- M and S are the to sets to compared
- |M| and |S| are sizes of the sets M and S

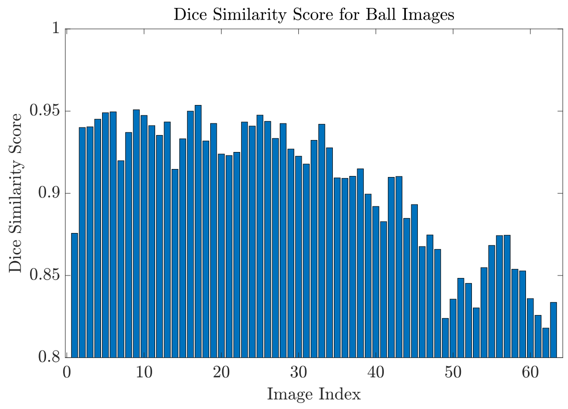

Fig. 4: Dice Similarity score for all 63 images. y-axis restricted between 0.8 and 1 for better visibility

As seen in Figure 4, the segmentation performs well from images 2 to 34, then it starts to lower in quality. This is due to the otsu segmentation of the american football being too strong near the end.

Task 2 — Feature Calculation

Task 2.a — Shape features

For this task, the following shape features were calculated:

1) Solidity: The proportion of the pixels in the convex hull that are also in the object.

$$\text{Solidity} = \frac{\text{Area of the object}}{\text{Area of its convex hull}}$$

2) Non-compactness: Compactness is the proportion of the region's pixels to all of the bounding box's pixels. So non-compactness is the inverse of this.

$$\text{Non-compactness} = 1 - \frac{\text{Area of the object}}{\text{Area of bounding box}}$$

3) Circularity: The roundness of the object.

$$\text{Circularity} = \frac{4\pi * \text{Area of the object}}{(\text{Perimeter of the object})^2} * \left(1 - \frac{0.5}{\text{r}}\right)$$

Where r = $\frac{\text{Perimeter of the object}}{2\pi} + 0.5$

4) Eccentricity: The eccentricity is the proportion of the distance between the foci of the ellipse and the length of its major axis. An eccentricity of 0 means its a perfect circle. An eccentricity of 1 means its a line.

$$\text{Eccentricity} = \frac{\text{Distance between foci of ellipse}}{\text{Length of major axis of ellipse}}$$

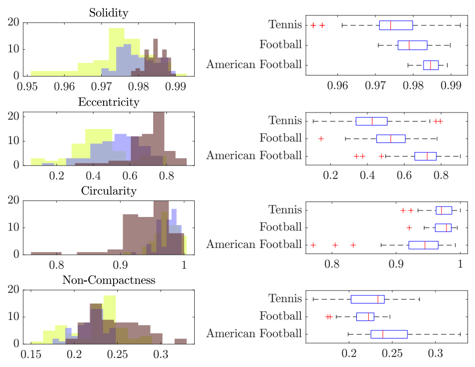

Fig. 5: Histograms and boxplots showing the Solidity, Non-Compactness, Circularity, and Eccentricity of the three ball types.
Legend:

- yellow: tennis
- blue: football
- brown: american football

Figure 5 displays histograms and boxplots representing the distribution of four different shape features (solidity, non-compactness, circularity, and eccentricity) for each ball type (tennis, football, and American football).
Going through each shape feature one by one.

There is a lot of overlap between the three ball types for solidity. The tennis ball has the lowest median solidity, followed by the football, and then the American football. The tennis ball has the highest range of solidity values, followed by the football, and then the american football.

For Eccentricity, the order of the median values is the same as for solidity but there is less overlap between the three ball types. This measure seems better at distinguishing the american football from the other two ball types as the american football is not a sphere like the other two balls.

Circularity like eccentricity is better at distinguishing the american football from the other two ball types, with the tennis ball and football having a similar Q1 and Q3 values. This is also due to the american football not being a sphere like the other two balls.

Finally, non-compactness is the worst at distinguishing between the three ball types as there is a lot of overlap between the three ball types. The mean of the non-compactness values for each ball lie within 0.025 of each other.

Task 2.b — Texture features

The texture features that were extracted were:

- Angular Second Moment:
- Contrast:
- Correlation:

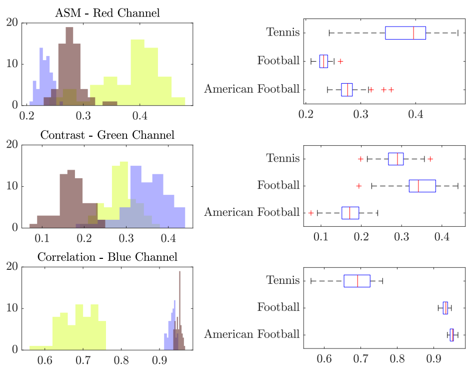

Fig. 6: Histograms and boxplots showing the average Angular Second Moment, Contrast, and Correlation associated with one colour channel for each ball type. Angular Second Moment on the red channel, Contrast on the green channel, Correlation on the blue channel. Same Legend as Figure 5.

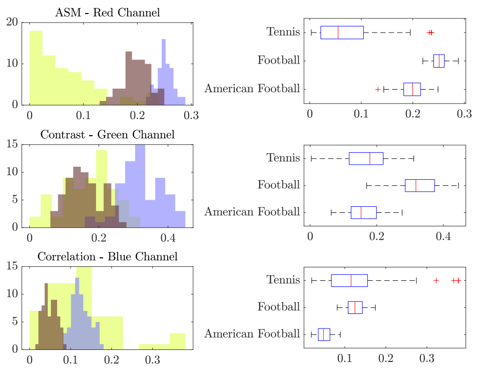

Fig. 7: Histograms and boxplots showing the ranges of Angular Second Moment, Contrast, and Correlation associated with one colour channel for each ball type. Angular Second Moment on the red channel, Contrast on the green channel, Correlation on the blue channel. Same Legend as Figure 5.

Task 2.c — Discriminative information

Task 3 — Object Tracking

The final task was to implement a kalman filter from scratch without using any methods from inbuilt libraries. Four trajectories were given, x.csv and y.csv contained the real coordinates and na.csv and nb.csv contained the noisy coordinates from which estimated coordinates could be calculated.

For this task, certain variables were set: F is should be a Constant Velocity motion model. i.e.

$$F = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where Δt is the time interval set to 0.5.
H is a Cartisian observation model. i.e.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

The covariance matices Q and R are

$$Q = \begin{bmatrix} 0.16 & 0 & 0 & 0 \\ 0 & 0.36 & 0 & 0 \\ 0 & 0 & 0.16 & 0 \\ 0 & 0 & 0 & 0.36 \end{bmatrix} R = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \tag{4}$$

Task 3.a — Kalman filter tracking

The kalman tracking algoritm iterates over every sample. For each sample, it predicts a state vector and a covariance using the constant velocity motion model and the motion noise, with which it then estimates the state and covariance with the Cartesian observation model, observation noise, and the sample.
Using the first sample and the initial state, Figure 8 is the result

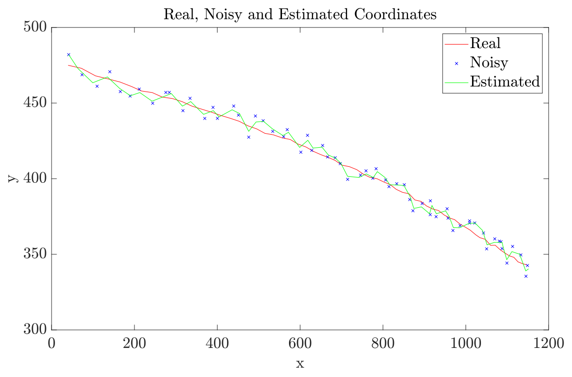

Fig. 8: plot of the estimated trajectory of coordinates $[x\_, y\_]$, together with the real $[x, y]$ and the noisy $[na, nb]$ for comparison

Task 3.b — Evaluation

The equation for Root Mean Squared Error (RMSE) is:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i))^2} \tag{5}$$

where:

- n is the total number of observations
- x_i and y_i are the i^{th} actual coordinates
- $\hat{x}_i$ and $\hat{y}_i$ is the i^{th} predicted coordinates

This equation in matlab looks like this:

```
% Root Mean Squared Error (RMSE) calculation
rmse_n = sqrt(mean((x - na).^2 + (y - nb).^2));
rmse = sqrt(mean((x - x_).^2 + (y - y_).^2));
```

$rsme\_n$ is the RMSE of the noisy coordinates relative to the ground truth.
$rsme$ is the RMSE of the estimated coordinates realtive to the ground truth.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{6}$$

where:

- σ is the standard deviation
- n is the total number of observations
- x_i is the i^{th} value
- μ is the mean of the values

This equation in matlab looks like this:

```
% Standard deviation calculation of the RMSE
std_n = std(sqrt((x - na).^2 + (y - nb).^2));
std = std(sqrt((x - x_).^2 + (y - y_).^2));
```

As the RMSE is a single number, there is no standard deviation. Therefore, I took the standard deviation of the root squared errors.

The results of these calualations are:

- RMSE for the noisy coordinates ≙ 7.416
- RMSE for the estimated coordinates ≙ 5.877
- STD for the noisy coordinates ≙ 2.573
- STD for the estimated coordinates ≙ 2.181

All these values are relative to the real coordinates and rounded to the 3^{rd} decimal. These results show that the kalman filter successfully removed some of the noise as it is closer to the real coordinates.
Better results may be found by testing different noise matrices and time intervals.

APPENDIX

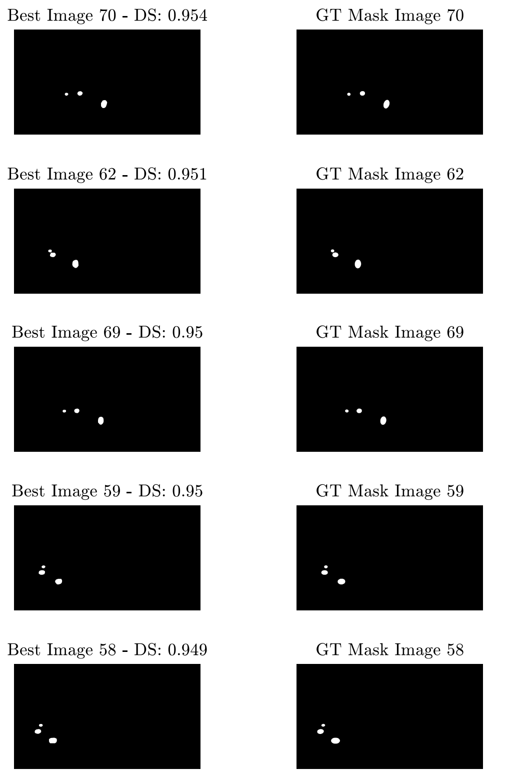

Fig. 9: Best 5 segmented ball images compared to the ground truth

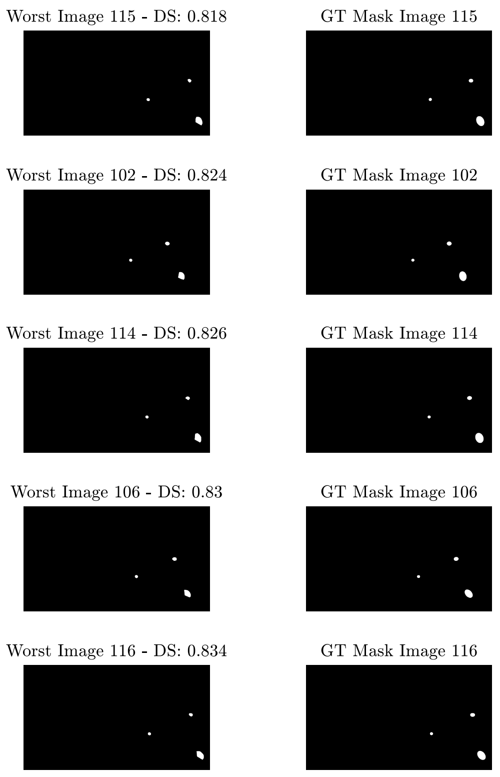
Worst Image 115 - DS: 0.818
GT Mask Image 115
Worst Image 102 - DS: 0.824
GT Mask Image 102
Worst Image 114 - DS: 0.826
GT Mask Image 114
Worst Image 106 - DS: 0.83
GT Mask Image 106
Worst Image 116 - DS: 0.834
GT Mask Image 116

Fig. 10: Worst 5 segmented ball images compared to the ground truth