CMP9135 Computer Vision Assessment 1

Tarteel Alkaraan
Department of Engineering and
Physical Sciences
University of Lincoln
Lincoln, United Kingdom
25847208@students.lincoln.ac.uk

1. First Task Image Processing

The aim of this task is to segment three type of balls football, American football and tennis ball from the background, producing a single binary mask for each picture. Use of learning-based approaches for example deep neural networks are not permitted. The goal of this assignment is to implement any image segmentation algorithm, apply it to the provided dataset, and evaluate its performance.

1.1 Automated Segmentation of Ball Objects

A) Otsu Thresholding: The first task is to convert the picture to black and white. In general, imbinarize MATLAB function utilises Otsu to differentiate among background and foreground. By adjusting sensitivity, I discovered the value that deleted majority of the background without deleting the balls was 0.35. imbinarize function just work on grey pictures hence I tested separating every RGB channel individually and the red channel seems to perform the best. Figure 1 below shows examples of the end result.

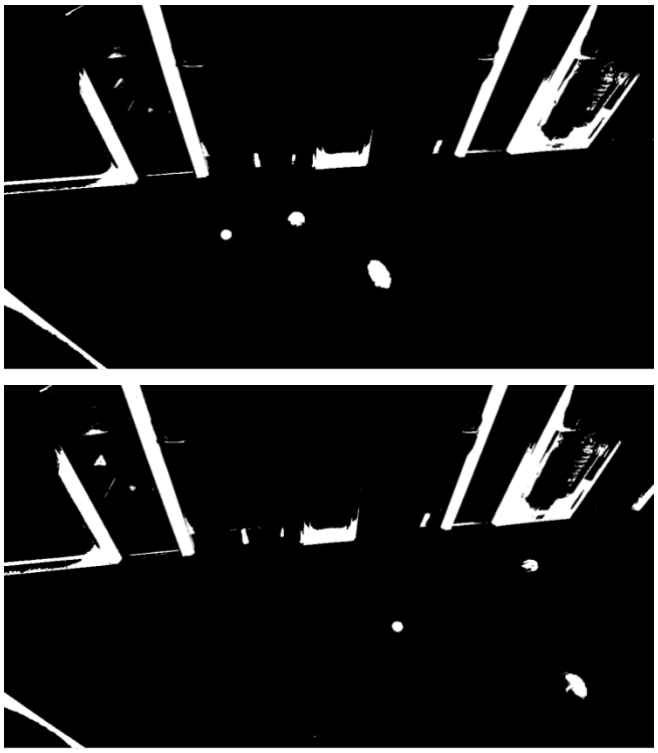

Figure 1 Examples of black and white picture end result of introductory segmentation utilising Otsu.

B) Changing to Convex Hulls: To delete remaining background objects, I choose to change all the linked components into convex hulls. As the balls are subject to be convex initially, they must not change mostly (without the American football in final couple of frames where Otsu segmentation was a little destructive). Which had the impact of hugely increasing the area of the background objects, in which they are clearly differentiable from the balls. End result examples could be seen in figure 2 below.

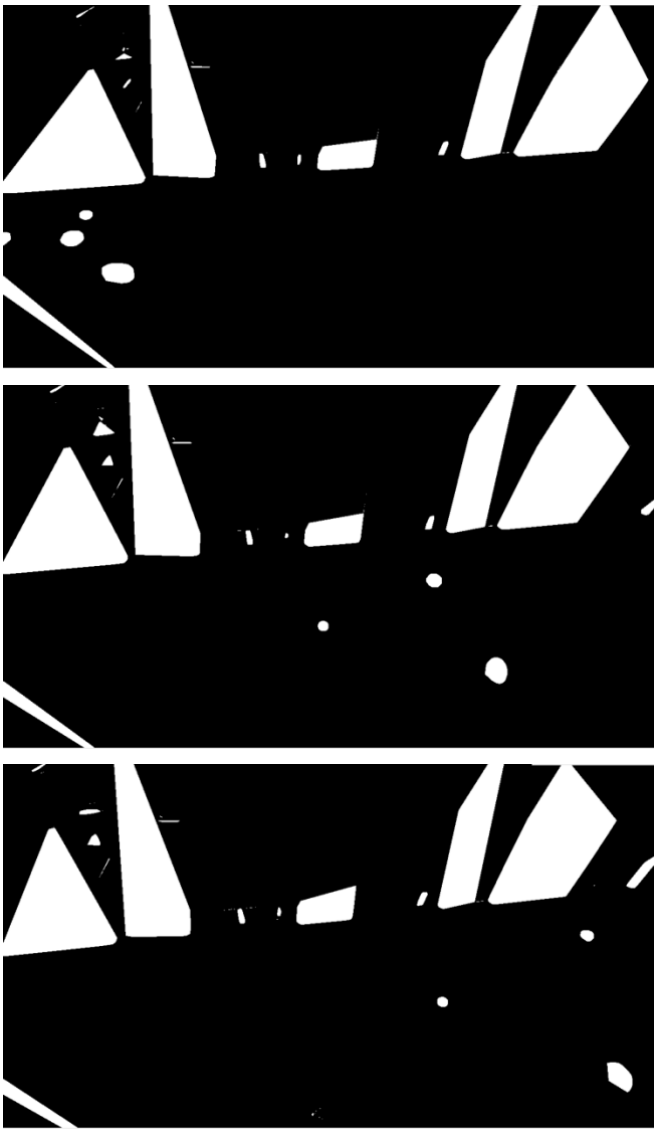

Figure 2 Examples of black and white picture end result once changing every linked component into a convex hull.

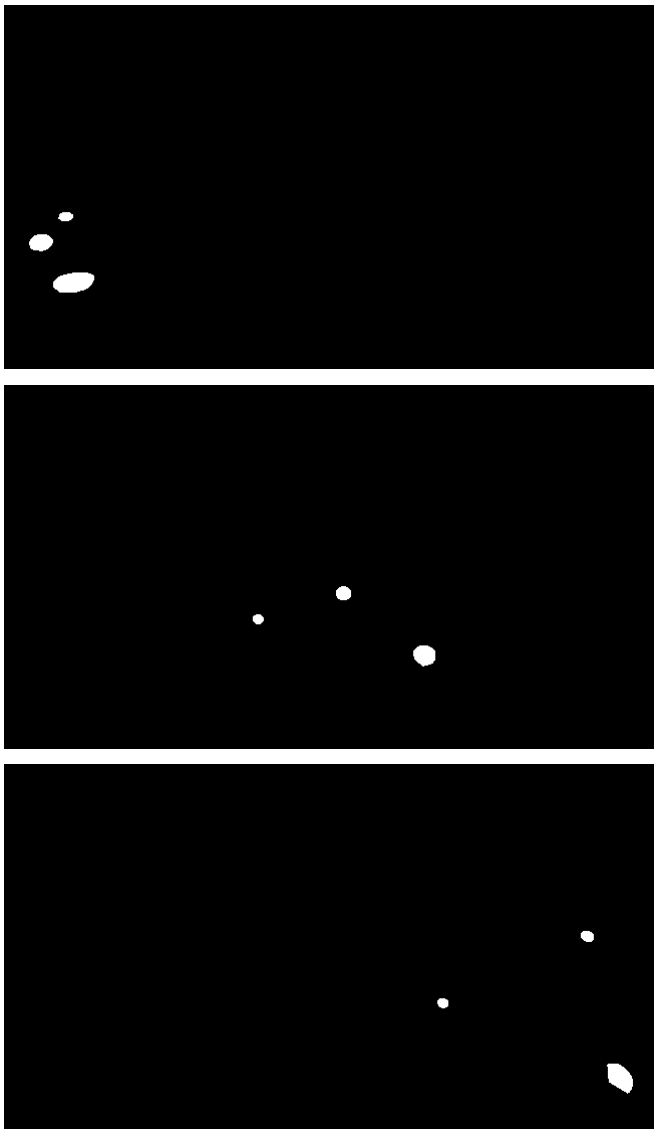

Figure 3 Examples of black and white picture end result once changing every linked component into a convex hull.

C) Deleting Remaining Objects: Currently, the background objects could be deleted from the black and white pictures by deleting each linked component that does not fit specific characteristics like area and extent. Following this only the three balls remain. Additional examples of best and worst segmentations could be seen in the appendix figures 5 and 6. Below figure 3 shows examples of end result.

2. Second Task Segmentation Evaluation

For the segmentation to be evaluated, the Dice Score Similarity among the segmented pictures and ground truth could be computed. As every picture is clearly stored as a matrix of zeros and ones, it is feasible to perform binary operations on the matrices.

The Dice Score Similarity is computed by multiplying the value of 1s in the intersection of the two matrices (logical &) by 2 and divide that by whole value of 1s in the two matrices. If there is a complete overlay among the pictures, therefore the intersection will cover all the 1s and the Dice Score Similarity would be 1. However, if there is no overlay, then there is no intersection, and the Dice Score Similarity would be 0.

$$DSC = \frac{2|M \cap S|}{|M| + |S|}$$

Were M is the output of my segmentation algorithm, S is the corresponding ground truth mask. |M| and |S| are sizes of the M and S sets. Dice Score Similarity end result could be seen below in figure 4.

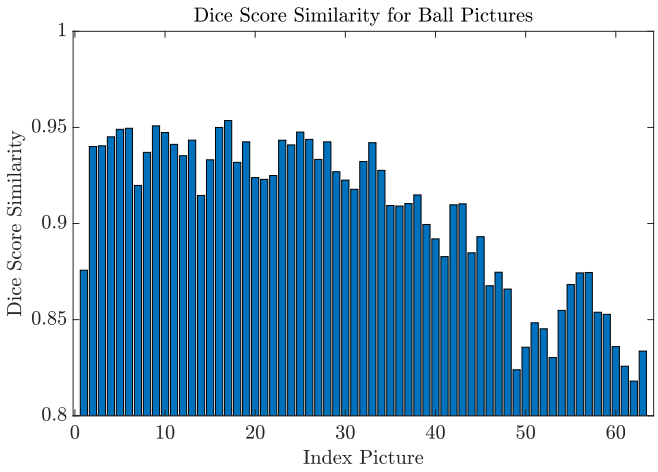

Figure 4 Dice Score Similarity for all 63 pictures. For more enhanced visualisation, y-axis limited among 0.8 and 1. By looking at figure 4, the segmentation performs better from pictures 2 to 34, after it begins to reduce in quality. This is because the Otsu segmentation of the American football becoming very strong close to the finish. The average for all ball pictures is 0.90464 and the standard deviation is 0.040538.

3. References

[1] A. Bora, "Computer_Vision_Assignment", https://github.com/arunabh-alt/Computer_Vision_Assignment/tree/main, 2024.

[2] O. Ali, "SegmentationsFeaturingAndTracking", https://github.com/omroali/SegmentationsFeaturingAndTracking, 2024.

[3] G. Davies, "Cmp_Vision_Lincoln", https://github.com/gdtdavies/Cmp_Vision_Lincoln/tree/master, 2024.

[4] O. Weekes, "computer_vision_assignment", https://github.com/oakleighw/computer_vision_assignment, 2023.

4. Appendix

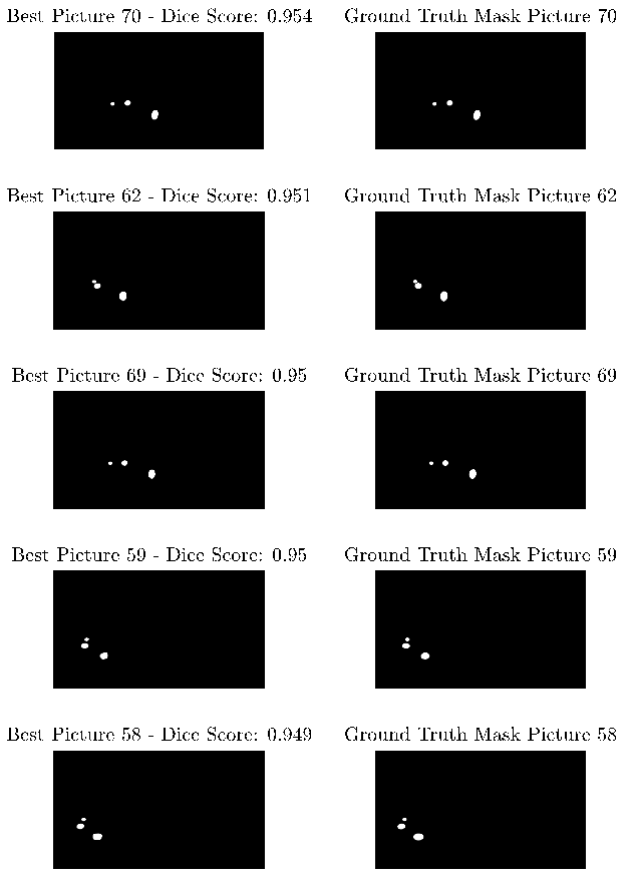

Figure 5 five best segmented ball pictures in comparison to the ground truth.

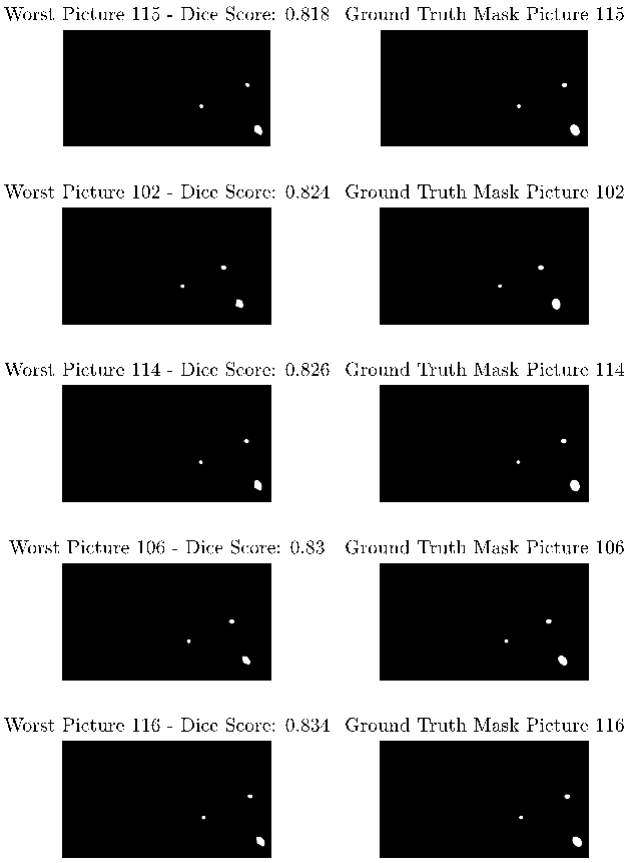

Figure 6 five worst segmented ball pictures in comparison to the ground truth.

Main.m

```
%Author: Tarteel Alkaraan (25847208)
%Last Updated: 28/02/2025
%This is an edited version of [3]
%Go Through All Pictures
Ground_Truth_Pictures = dir('BallFrames/GroundTruth/*.png');
Pictures = dir('BallFrames/Original/*.png');
Output = 'BallFrames/Mask/';

for I = 1:length(Pictures)
   %Read Picture
   Picture = imread(Pictures(I).folder + "/" + Pictures(I).name);

   %% Threshold Whole Picture and Look for Balls
   %Change Picture to Black and White using Otsu Thresholding
   MaskOtsu = imbinarize(Picture(:, :, 1), 'adaptive', 'Sensitivity', 0.35);

   %Look For Balls in Picture
   MaskBalls = LookForBalls(MaskOtsu);

   %% Save Mask
   imshow(MaskBalls)
   %imshow(MaskOtsu)
   %imwrite(MaskBalls, fullfile(Output, Pictures(I).name));
end
```

```
close("all")
```

LookForBalls.m

```
%Author: Tarteel Alkaraan (25847208)
%Last Updated: 28/02/2025
%This is an edited version of [3]
function Output = LookForBalls(Picture)
   %% Close Shapes to Make Them Complete
   SE = strel('disk', 4);
   Picture = imclose(Picture, SE);
   Picture = imfill(Picture, 'holes');

   %% Make Shapes Convex to Delete of Holes in Balls
   ConvComp = bwconncomp(Picture);
   State = regionprops(ConvComp, 'ConvexHull');
   Picture_Convex = false(size(Picture));

   for I = 1:numel(State)

      %Receive Convex Hull for Existing Shape
      HullConvex = State(I).ConvexHull;

      %Change Convex Hull to Black and White Mask
      Mask = poly2mask(HullConvex(:,1), HullConvex(:,2), size(Picture, 1), size(Picture, 2));

      %Mix Mask with Current Black and White Picture
      Picture_Convex = Picture_Convex | Mask;

   end

   %% Delete Shapes that are very small or to big, are to long, and have a low area to bounding box area ratio
   ConvComp2 = bwconncomp(Picture_Convex);
   State2 = regionprops(ConvComp2, 'Area', 'Centroid', 'Extent');

   X = [State2.Centroid];
   X = X(2:2:end);

   %Make New Black and White Picture where just Elliptical Shapes are included
   Shapes_Elliptical = ismember(labelmatrix(ConvComp2), find((325 < [State2.Area] & [State2.Area] < 2600) &
(height(Picture) * 0.45 < X) & [State2.Extent] > 0.6));
   %imshow(Shapes_Elliptical)
   %imshow(Picture_Convex)
   Output = Shapes_Elliptical;
end
```

EvaluationSegmentation.m

```
%Author: Tarteel Alkaraan (25847208)
%Last Updated: 28/02/2025
%This is an edited version of [3]
PathGroundTruth = 'BallFrames/GroundTruth/';
PathMask = 'BallFrames/Mask/';

DiceScore = zeros(1, 63);

for I = 54:116
   %Load Ball Region Segmented and Ball Mask Ground-Truth in Balck and White
```

```
    A = imread([PathMask 'frame-' num2str(I) '.png']);
    B = imread([PathGroundTruth 'frame-' num2str(I) '_GT.png']);

    %Change Grayscale Pictures to Black and White
    A = imbinarize(uint8(A));
    B = imbinarize(uint8(B));

    %Compute Intersection between A and B
    Intersection = sum(sum(A & B));

    %Compute Size of A and B
    ASize = sum(sum(A));
    BSize = sum(sum(B));

    %Compute Dice Score Similarity (DS)
    DiceScore(I - 53) = 2 * Intersection ./ (ASize + BSize);
end

%Compute Average and Standard Deviation of Dice Score Similarity
DiceScoreAverage = mean(DiceScore);
DiceScoreStandardDeviation = std(DiceScore);
disp(['Dice Score Similarity Average for Ball Pictures is ' num2str(DiceScoreAverage)]);
disp(['Dice Score Similarity Standard Deviation for Ball Pictures is ' num2str(DiceScoreStandardDeviation)]);
DisplayDiceScore(DiceScore);

%Compute Mean and Standard Deviation of Dice Score
DiceScoreMean = mean(DiceScore);
DiceScoreStandardDeviation = std(DiceScore);

%Arange Dice Score Array in Ascending Order
[DiceScoreSorted, IndicesSorted] = sort(DiceScore);

%Plot 5 Worst Ball Segmented Pictures and their relating Ground-Truth Mask Picture
DisplayPictures(IndicesSorted, DiceScore, 'Worst');

%Plot 5 Best Ball Segmented Pictures and their relating Ground-Truth Mask Picture
DisplayPictures(IndicesSorted, DiceScore, 'Best');
```

DisplayPictures.m

```
%Author: Tarteel Alkaraan (25847208)
%Last Updated: 28/02/2025
%This is an edited version of [3]
function DisplayPictures(Pictures, DiceScore, Name)
    PathGroundTruth = 'BallFrames/GroundTruth/';
    PathMask = 'BallFrames/Mask/';
    Figure = figure;

    for I = 1:5
        if strcmp(Name, 'Best')
            subplot(5, 2, I + (I - 1));
            imshow([PathMask 'frame-' num2str(Pictures(end - I + 1) + 53) '.png']);
            title(['Best Picture ' num2str(Pictures(end - I + 1) + 53) ' - Dice Score: ' num2str(round(DiceScore(Pictures(end - I + 1)),
3))]);

            subplot(5, 2, I * 2);
            imshow([PathGroundTruth 'frame-' num2str(Pictures(end - I + 1) + 53) '_GT.png']);
            title(['Ground Truth Mask Picture ' num2str(Pictures(end - I + 1) + 53)]);
```

```
    elseif strcmp(Name, 'Worst')
        subplot(5, 2, I + (I - 1));
        imshow([PathMask 'frame-' num2str(Pictures(I) + 53) '.png']);
        title(['Worst Picture ' num2str(Pictures(I) + 53) ' - Dice Score: ' num2str(round(DiceScore(Pictures(I)), 3))]);

        subplot(5, 2, I * 2);
        imshow([PathGroundTruth 'frame-' num2str(Pictures(I) + 53) '_GT.png']);
        title(['Ground Truth Mask Picture ' num2str(Pictures(I) + 53)]);
    end
  end

  NameFile = strcat('BallFrames/Figures/', Name, '.pdf');
  WidthPicture = 30;
  RatioHeight = 1.4;
  set(findall(Figure, '-property', 'FontSize'), 'FontSize', 22)

  set(findall(Figure, '-property', 'Interpreter'), 'Interpreter', 'latex')
  set(findall(Figure, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
  set(Figure, 'Units', 'Centimeters', 'Position', [3 3 WidthPicture RatioHeight * WidthPicture])
  Position = get(Figure, 'Position');
  set(Figure, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', 'PaperSize', [Position(3), Position(4)])
  print(Figure, NameFile, '-dpdf', '-vector', '-fillpage')
end
```

DisplayDiceScore.m

```
%Author: Tarteel Alkaraan (25847208)
%Last Updated: 28/02/2025
%This is an edited version of [3]
function DisplayDiceScore(DiceScore)
  %Displaying Bar Graph
  Figure = figure;
  bar(DiceScore);
  xlabel('Index Picture');
  ylabel('Dice Score Similarity');
  title('Dice Score Similarity for Ball Pictures');
  ylim([0.8, 1]);

  NameFile = 'BallFrames/Figures/DiceScore.pdf';

  WidthPicture = 30;
  RatioHeight = 0.65;
  set(findall(Figure, '-property', 'FontSize'), 'FontSize', 22)

  set(findall(Figure, '-property', 'Interpreter'), 'Interpreter', 'latex')
  set(findall(Figure, '-property', 'TickLabelInterpreter'), 'TickLabelInterpreter', 'latex')
  set(Figure, 'Units', 'Centimeters', 'Position', [3 3 WidthPicture RatioHeight * WidthPicture])
  Position = get(Figure, 'Position');
  set(Figure, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', 'PaperSize', [Position(3), Position(4)])
  print(Figure, NameFile, '-dpdf', '-vector', '-fillpage')
end
```

DisplayVideo.m

```
%Author: Tarteel Alkaraan (25847208)
%Last Updated: 28/02/2025
%This is an edited version of [3]
Pictures = dir('BallFrames/Mask/*.png');
```

```
while true
   for I = 1:length(Pictures)
      Picture = imread(Pictures(I).folder + "/" + Pictures(I).name);
      imshow(Picture);
   end
end
```