# PRN – 1132230471 FY MSC DS & BDA

**Digital Image Processing and Computer Vision (Assignment)**

# Question1 : Demonstrate thresholding while taking thresholding value differently 1. 100, 2. 200, take any 5 values

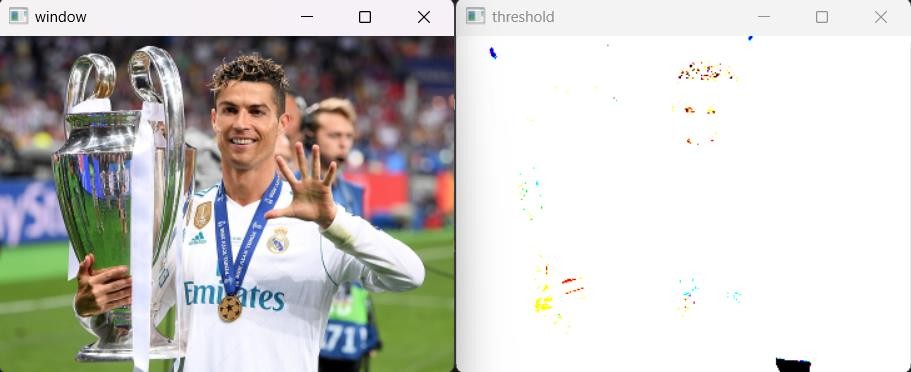
import cv2

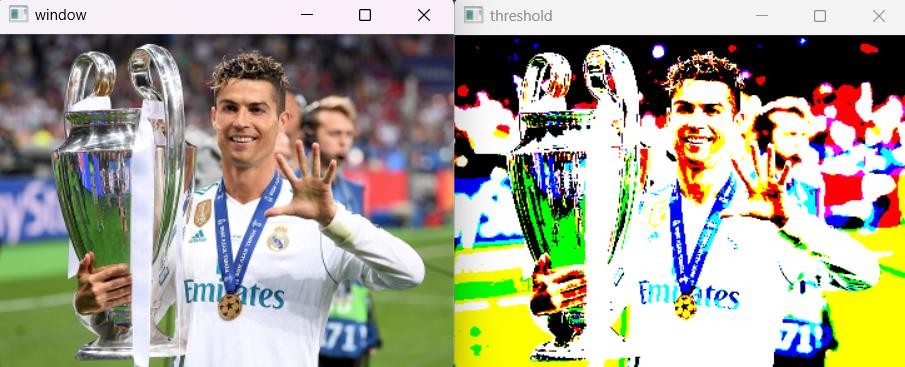
image = cv2.imread(r'D:\MSc DS\DIP\Images\CR7.jpg') image = cv2.resize(image, (0, 0), fx=0.5, fy=0.5)

(T, thresh) = cv2.threshold(image, 10, 255, cv2.THRESH\_BINARY) cv2.imshow('window', image)

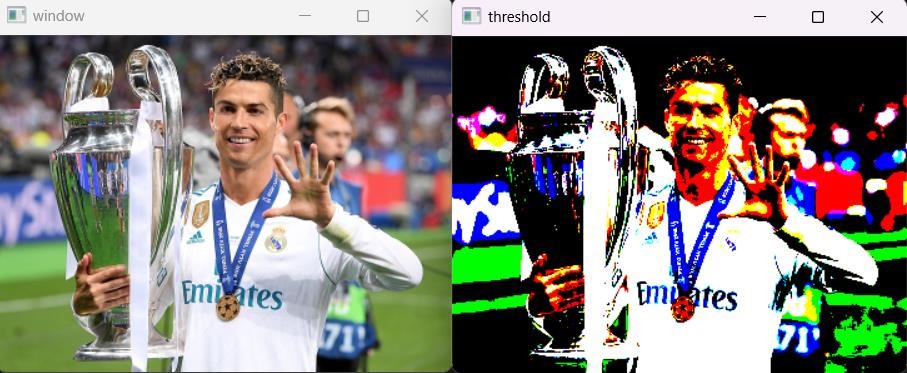
cv2.imshow('threshold', thresh)

cv2.waitKey(0) cv2.destroyAllWindows()





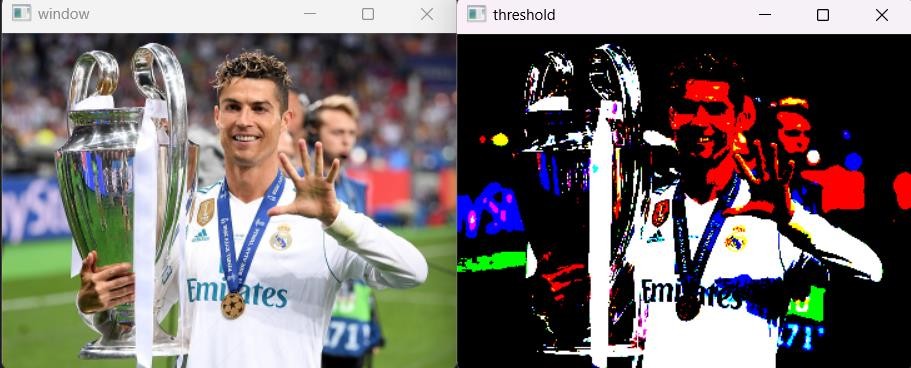
**Threshold value 150**



(T, thresh) = cv2.threshold(image, 150, 255, cv2.THRESH\_BINARY) cv2.imshow('window', image)

cv2.imshow('threshold', thresh)

cv2.waitKey(0) cv2.destroyAllWindows()

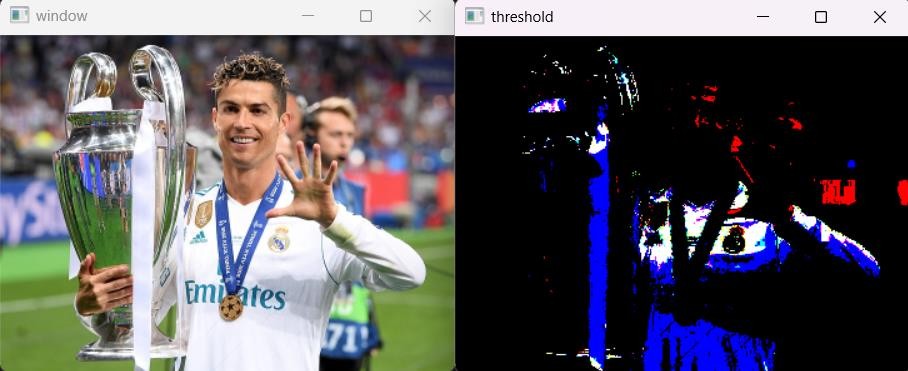


# Threshold value 250

(T, thresh) = cv2.threshold(image, 250, 255, cv2.THRESH\_BINARY) cv2.imshow('window', image)

cv2.imshow('threshold', thresh) cv2.waitKey(0)

cv2.destroyAllWindows()



import cv2

import numpy as np

img = cv2.imread(r'D:\MSc DS\DIP\Images\CR7.jpg', 0) img = cv2.resize(img, (0, 0), fx=0.5, fy=0.5)

# Find width and height of image row, column= img.shape

# Create an zeros array to store the sliced image img1 = np.zeros((row,column),dtype = 'uint8') #img1 = cv2.resize(img1, (0, 0), fx=0.5, fy=0.5) # Specify the min and max range

min\_range = 80

max\_range = 140

#Loop over the input image and if pixel value lies in desired range set it to 255 #otherwise set it to desired value

for i in range(row):

for j in range(column):

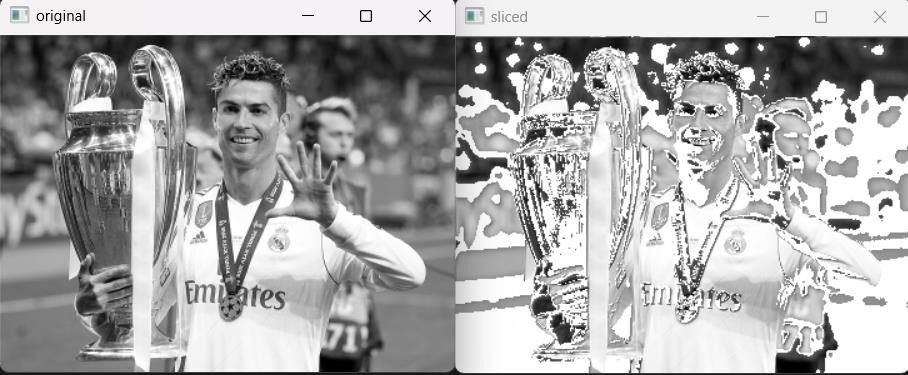
if img[i,j]>min\_range and img[i,j]<max\_range: img1[i,j] = 255

else:

img1[i,j] = img[i-1,j-1]

cv2.imshow('original', img) cv2.imshow('sliced', img1) cv2.waitKey(0)

cv2.destroyAllWindows()



def Contrast\_stretch(p, r1, s1, r2, s2): if (0 <= p and p <= r1):

equation = (s1 / r1) \* p elif (r1 < p and p <= r2):

equation = ((s2 - s1) / (r2 - r1)) \* (p - r1) + s1 else:

equation = ((255 - s2) / (255 - r2)) \* (p - r2) + s2 return equation

# Read original image

image = cv2.imread(r'D:\MSc DS\DIP\Images\CR7.jpg') image = cv2.resize(image, (0, 0), fx=0.5, fy=0.5)

# Convert image to floating point image\_float = image.astype(float)

# Initialize range r1 = 55

s1 = 40

r2 = 140

s2 = 200

# Vectorize the function for element-wise application pixelVal\_vec = np.vectorize(Contrast\_stretch)

# Contrast stretching

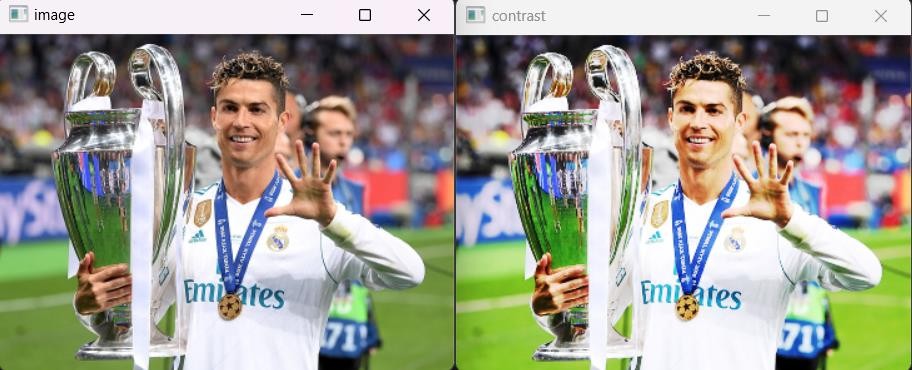
contrast\_float = pixelVal\_vec(image\_float, r1, s1, r2, s2)

# Convert back to uint8 data type

contrast = contrast\_float.astype(np.uint8) cv2.imwrite('contrast.jpg', contrast)

cv2.imshow('image', image) cv2.imshow('contrast', contrast)

cv2.waitKey(0) cv2.destroyAllWindows()



**Question4 : implement bit-plane slicing**

import cv2

import numpy as np

# Read the image in grayscale

image = cv2.imread(r"D:\MSc DS\DIP\Images\CR7.jpg") image = cv2.resize(image, (0, 0), fx=0.3, fy=0.3) # Check if image was loaded successfully

if image is None:

print("Error: Could not load image") exit()

# Perform bit plane slicing for 8 bits (0 to 7) for bit in range(8):

# Create a mask for the specific bit mask = 1 << bit

bit\_plane = image & mask

# Scale the bit plane to 0-255

bit\_plane = (bit\_plane \* 255).astype(np.uint8)

# Display the bit plane

cv2.imshow(f"Bit Plane {bit}", bit\_plane)

# Display the original image cv2.imshow("Original Image", image)

# Wait for a key press and then close all windows

cv2.waitKey(0) cv2.destroyAllWindows()

